Robust adaptive neural network control for a class of non-linear systems

S S Ge and **T H Lee**
Department of Electrical Engineering, National University of Singapore

Abstract: In this paper, a general framework for robust parallel adaptive neural network (NN) control design is presented for a class of non-linear systems motivated by the work in references (**14**) and (**15**). The controller is based on applying direct adaptive techniques to an additional parallel neural network to provide adaptive enhancements to a basic fixed controller and incorporating a sliding mode term for robustness. It is shown that if bounded basis function (BBF) networks are used for the additional parallel NN, uniformly stable adaptation is assured and asymptotic tracking of the reference signal is achieved. Because of the introduction of the GL (Ge–Lee) matrices and operator, the results presented here are more general than the existing results.

Keywords: neural network, non-linear systems, adaptive control

NOTATION

For clarity, some notations are introduced:

1. $(\overline{*})$ and $(\hat{*})$ denote the fixed estimate and the on-line estimate of $(*)$ respectively.
2. $(\tilde{*}) := (*) - (\hat{*})$.
3. $x_i^{(j)}$ is the jth derivative of x_i and $\boldsymbol{x}_i = [x_i \; x_i^{(1)} \cdots x_i^{(n_i-1)}]^{\mathrm{T}}$.
4. m_{ij} is the ijth element of matrix $\mathbf{M}$.
5. $\{*\}$ denotes a GL matrix or a GL vector, $\cdot$ is the corresponding GL operator and $\{\boldsymbol{W}_{i*}\}$ and $\{\boldsymbol{W}_{*i}\}$ are the corresponding GL row and GL column vectors introduced in Appendix 1.
6. $[*]$ represents a conventional matrix or a conventional vector.
7. $\boldsymbol{b}_i$ is the ith column vector of matrix $\mathbf{B}$, i.e. $\mathbf{B} = [\boldsymbol{b}_1 \; \boldsymbol{b}_2 \cdots \boldsymbol{b}_n]$.
8. I_0 is the set of integers.
9. $\mathbf{I}$ is the unit matrix of different dimensions.

1 INTRODUCTION

The fusion of mechanical, electrical and computer engineering has formed an important class of processes in modern control engineering. It has its own properties which distinguish itself from other processes, such as chemical and biochemical processes. The emergence of robotics is one good example. It is a multidisciplinary subject incorporating the three areas mentioned above. As a result of active research in robotics in the past two decades, a relatively good understanding of the control problems has been achieved for such systems. Some of the results can be seen being applied to other areas, such as spacecraft attitude control (**1**), weapon pointing control (**2**), vehicle suspension control (**3**) and multi-helicopter lifting control (**4**) problems.

Conventional linear feedback control techniques are unable to meet the ever-demanding control requirements of increasingly complex dynamical systems under significant uncertainties. Neural networks (NNs) distinguish themselves from other control techniques as a very attractive alternative because of their ability to learn, to approximate functions and their potential for parallel hardware implementation. Recently, it has been successfully shown in many cases (**5**–**8**) that NNs provide a suitable approach to control certain classes of non-linear dynamical systems.

In general, the NNs used in control system design are fixed feedforward multilayer networks. However, it does not have any built-in capability to handle changes in the system. By directly parametrizing a suitable NN control law, good closed-loop stability properties have been shown in references (**9**) to (**18**) by incorporating adaptive techniques. Parallel adaptive NN controllers were presented in references (**14**) and (**15**), each of which consists of a fixed structural non-linear controller and an on-line NN adaptive controller for performance enhancement. It was shown that if bounded basis functions, such as radial basis function (RBF) neural networks, are used for the additional parallel

The MS was received on 1 November 1995 and was accepted for publication on 28 March 1997.

NN, uniformly stable adaptation is assured and asymptotic tracking of the position reference signal is achieved under the very restrictive assumption that the system dynamics are in the functional range of the neural networks.

In this paper, a general framework for robust parallel NN adaptive controller design is presented for a class of non-linear systems, for which rigid-body robots, flexible joint robots, weapon positioning systems and various similar non-linear servo-mechanisms are special cases. The controller is developed under the relaxed and realistic assumption that the system dynamics are not in the functional range of the neural networks. It is shown that if bounded basis functions, such as RBF, B-splines or CMAC neural networks are used for the additional parallel NN, uniformly stable adaptation is assured and asymptotic tracking of the position reference signal is achieved.

In Section 2, the problems of NN approximations are briefly discussed, a matrix (vector) emulator is conveniently expressed as a GL product of two GL matrices (vectors) and a new stability lemma is introduced. The dynamics of the class of non-linear systems are presented in Section 3. The general robust neural network control framework is discussed in Section 4.

2 NEURAL NETWORK APPROXIMATIONS

In control engineering, an NN is usually used to generate an input/ouput map using the property that a multilayer neural network can approximate any function, under mild assumptions, with any desired accuracy. As a function emulator, firstly an approximating function $\hat{\boldsymbol{f}}(\boldsymbol{\theta}, \boldsymbol{x})$: $R^s \times R^n \rightarrow R^m$ for $\boldsymbol{f}(\boldsymbol{x})$: $R^n \rightarrow R^m$ is chosen; then an algorithm such as the back propagation algorithm can be used to adjust the neural network weight based on the output error by a training set.

A three-layer neural network is shown schematically in Fig. 1, where $\boldsymbol{x} \in R^n$, $\boldsymbol{y} \in R^m$, $\boldsymbol{a} \in R^k$ are the input, the output and the activation function vectors respectively, m_{il} the first-to-second layer interconnection weights and w_{lj} the second-to-third layer interconnection weights. The input/output map of the network is given by

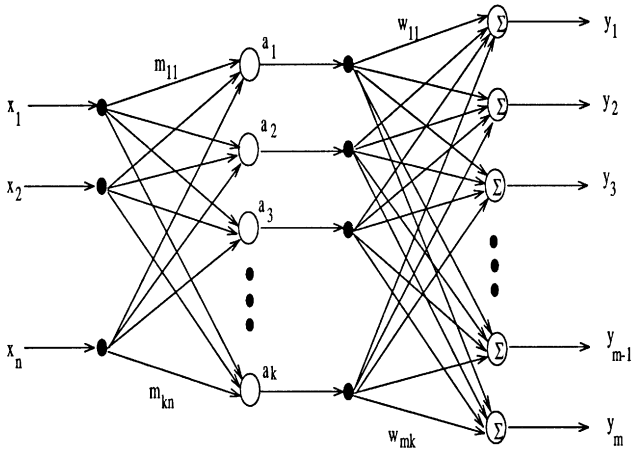

Fig. 1 Three-layer neural network

$$z_i = \sum_{j=1}^{n} m_{ij} x_j + m_{0i}, \qquad i = 1, 2, \ldots, k \tag{1}$$

$$y_j = \sum_{l=1}^{m} w_{jl} a_l(z_l) + w_{0j}, \qquad j = 1, 2, \ldots, m \tag{2}$$

where m_{0i} and w_{0j} are the threshold offsets. The NN equations (1) and (2) can be conveniently expressed in a matrix format as

$$\boldsymbol{z} = \mathbf{M}^{\mathrm{T}} \boldsymbol{x} + \boldsymbol{M}_0$$

$$\boldsymbol{y} = \mathbf{W}^{\mathrm{T}} \boldsymbol{a}(\boldsymbol{z}) + \boldsymbol{W}_0$$

where $\mathbf{W}^{\mathrm{T}} = [w_{ij}]$, $\mathbf{M}^{\mathrm{T}} = [m_{ij}]$ and

$$\boldsymbol{M}_0 = [m_{01} \quad m_{02} \cdots m_{0k}]^{\mathrm{T}}$$

$$\boldsymbol{W}_0 = [w_{01} \quad w_{02} \cdots w_{0m}]^{\mathrm{T}}$$

Without losing generality, $\boldsymbol{M}_0 = 0$, $\boldsymbol{W}_0 = 0$ is assumed in the following analysis for simplicity. Thus, the output of the network can be simply written as

$$\boldsymbol{y} = \mathbf{W}^{\mathrm{T}} \boldsymbol{a}(\mathbf{M}^{\mathrm{T}} \boldsymbol{x}) \tag{3}$$

A general function $\boldsymbol{f}(\boldsymbol{x}) \in R^m$ can then be described as

$$\boldsymbol{f}(\boldsymbol{x}) = \mathbf{W}^{\mathrm{T}} \boldsymbol{a}(\mathbf{M}^{\mathrm{T}} \boldsymbol{x}) + \boldsymbol{\epsilon}(\boldsymbol{x}) \tag{4}$$

where $\boldsymbol{\epsilon}(\boldsymbol{x})$ is an NN functional reconstruction error vector. If there exist an integer k and constants $\mathbf{W}$ and $\mathbf{M}$ such that $\boldsymbol{\epsilon} = 0$, then $\boldsymbol{f}(\boldsymbol{x})$ is said to be in the functional range of the NN. It is well known that any sufficiently smooth function can be approximated by a suitably large network using various activation functions, $\boldsymbol{a}(\cdot)$, based on the Stone–Weierstrass theorem. Typical choices for $\boldsymbol{a}(\cdot)$ include the sigmoid, hyperbolic tangent, radial basis functions, etc. (**19**, **20**).

It has been demonstrated (**21**) that a linear superposition of Gaussian RBFs results in an optimal mean square approximation to an unknown function which is infinitely differentiable and whose values are specified at a finite set of points in $\mathscr{R}^n$. Further, it has been proven (**22**, **23**) that any continuous functions, not necessarily infinitely smooth, can be uniformly approximated by linear combinations of Gaussians.

The Gaussian RBF neural network is a particular network architecture (**23**) utilizing k numbers of Gaussian radial basis functions with input variables $\boldsymbol{x} \in R^n$, variance $\sigma^2 \in R$ and the centre vector $\boldsymbol{c} = (c_1, \ldots, c_n)^{\mathrm{T}} \in R^n$, i.e.

$$\boldsymbol{y}(\mathbf{W}, \boldsymbol{x}) = \mathbf{W}^{\mathrm{T}}\boldsymbol{a}(\boldsymbol{x}) \tag{5}$$

$$a_i(\boldsymbol{x}) = \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{c}\|^2}{\sigma^2}\right) = \exp\left[-\frac{(\boldsymbol{x}-\boldsymbol{c})^{\mathrm{T}}(\boldsymbol{x}-\boldsymbol{c})}{\sigma^2}\right] \tag{6}$$

Comparing equations (3) and (5), it can be seen that $\mathbf{M} = \mathbf{I}$ in this case.

It is clear that a vector network emulator [$\boldsymbol{F}(\cdot)$: $R^n \rightarrow R^m$], accordingly a scalar network emulator [$f(\cdot)$: $R^n \rightarrow R$], can be easily expressed in the form of equation (3) or equation (5). The drawbacks of such expressions include:

1. They are not very flexible for cases when $y_i(\boldsymbol{x})$ requires more nodes than $y_j(\boldsymbol{x})$, $i \neq j$ (2). In such cases, $\mathbf{W}$ and $\mathbf{M}$ are sparse matrices, and a large amount of 'wasted' computation is involved in zero multiplications and summations.
2. It is difficult to extend the expressions to a matrix network emulator [$\mathbf{G}(\boldsymbol{x})$: $R^n \rightarrow R^{m_1 \times m_2}$] because of the limitations using the existing matrix and vector expressions.

To solve the drawbacks listed above, the GL products of GL matrices (vectors) are used in the following analysis (see Appendix 1 for detail). Let $\boldsymbol{W}_{ij}$, $\boldsymbol{N}_{ij}(\boldsymbol{x}) \in R^{n_{ij}}$, $n_{ij} \in I_0$, $i = 1, \ldots, n$, $j = 1, \ldots, k$. The product $\boldsymbol{W}_{ij}^{\mathrm{T}}\boldsymbol{N}_{ij}$ can be taken as a network emulator of the ijth element, $g_{ij}(\boldsymbol{x})$ of matrix $\mathbf{G}(\boldsymbol{x}) \in R^{n \times k}$, with $\boldsymbol{W}_{ij}$ and $\boldsymbol{N}_{ij}(\boldsymbol{x})$ being the weight vector and the basis function vector respectively.

The matrix network emulator $\mathbf{G}(\boldsymbol{x})$ can be conveniently expressed as a GL product of two GL matrices as:

$$\mathbf{G}(\boldsymbol{x}) = \{\boldsymbol{W}\}^{\mathrm{T}}\cdot\{\boldsymbol{N}\} := \begin{bmatrix} \{\boldsymbol{W}_{1*}\}^{\mathrm{T}}(\boldsymbol{N}_{1*}\} \\ \{\boldsymbol{W}_{2*}\}^{\mathrm{T}}\{\boldsymbol{N}_{2*}\} \\ \cdots \\ \{\boldsymbol{W}_{n*}\}^{\mathrm{T}}\{\boldsymbol{N}_{n*}\} \end{bmatrix}$$

$$:= \begin{bmatrix} \mathbf{W}_{11}^{\mathrm{T}}\mathbf{N}_{11} & \mathbf{W}_{12}^{\mathrm{T}}\mathbf{N}_{12} & \cdots & \mathbf{W}_{1k}^{\mathrm{T}}\mathbf{N}_{1k} \\ \mathbf{W}_{21}^{\mathrm{T}}\mathbf{N}_{21} & \mathbf{W}_{22}^{\mathrm{T}}\mathbf{N}_{22} & \cdots & \mathbf{W}_{2k}^{\mathrm{T}}\mathbf{N}_{2k} \\ \cdots & \cdots & \cdots & \cdots \\ \mathbf{W}_{n1}^{\mathrm{T}}\mathbf{N}_{n1} & \mathbf{W}_{n2}^{\mathrm{T}}\mathbf{N}_{n2} & \cdots & \mathbf{W}_{nk}^{\mathrm{T}}\mathbf{N}_{nk} \end{bmatrix} \in R^{n \times k}$$

where

$$\{\mathbf{W}\} := \begin{Bmatrix} \{\boldsymbol{W}_{1*}\} \\ \{\boldsymbol{W}_{2*}\} \\ \cdots \\ \{\boldsymbol{W}_{n*}\} \end{Bmatrix} = \begin{Bmatrix} \mathbf{W}_{11} & \mathbf{W}_{12} & \cdots & \mathbf{W}_{1k} \\ \mathbf{W}_{21} & \mathbf{W}_{22} & \cdots & \mathbf{W}_{2k} \\ \cdots & \cdots & \cdots & \cdots \\ \mathbf{W}_{n1} & \mathbf{W}_{n2} & \cdots & \mathbf{W}_{nk} \end{Bmatrix}$$

$$\{\mathbf{N}\} := \begin{Bmatrix} \{\boldsymbol{N}_{1*}\} \\ \{\boldsymbol{N}_{2*}\} \\ \cdots \\ \{\boldsymbol{N}_{n*}\} \end{Bmatrix} = \begin{Bmatrix} \mathbf{N}_{11} & \mathbf{N}_{12} & \cdots & \mathbf{N}_{1k} \\ \mathbf{N}_{21} & \mathbf{N}_{22} & \cdots & \mathbf{N}_{2k} \\ \cdots & \cdots & \cdots & \cdots \\ \mathbf{N}_{n1} & \mathbf{N}_{n2} & \cdots & \mathbf{N}_{nk} \end{Bmatrix}$$

with the following GL row vectors:

$$\{\boldsymbol{W}_{i*}\} = \{\boldsymbol{W}_{i1} \quad \boldsymbol{W}_{i2} \cdots \boldsymbol{W}_{ik}\}$$

$$\{\boldsymbol{N}_{i*}\} = \{\boldsymbol{N}_{i1} \quad \boldsymbol{N}_{i2} \cdots \boldsymbol{N}_{ik}\}$$

A vector emulator, $\boldsymbol{F}(\boldsymbol{x})$, can be expressed as a GL product of two GL vectors as follows:

$$\boldsymbol{F}(\boldsymbol{x}) = \{\boldsymbol{K}\}^{\mathrm{T}}\cdot\{\boldsymbol{H}\} := \begin{bmatrix} \boldsymbol{K}_1^{\mathrm{T}}\boldsymbol{H}_1 \\ \cdots \\ \boldsymbol{K}_n^{\mathrm{T}}\boldsymbol{H}_n \end{bmatrix} \tag{7}$$

where $\boldsymbol{K}_i$, $\boldsymbol{H}_i \in R^{n_i}$, $i = 1, \ldots, n$, and

$$\{\boldsymbol{K}\} := \begin{Bmatrix} \boldsymbol{K}_1 \\ \cdots \\ \boldsymbol{K}_n \end{Bmatrix}, \qquad \{\boldsymbol{H}\} := \begin{Bmatrix} \boldsymbol{H}_1 \\ \cdots \\ \boldsymbol{H}_n \end{Bmatrix} \tag{8}$$

The corresponding element-wise inner products, $\boldsymbol{K}_i^{\mathrm{T}}\boldsymbol{H}_i$ and $\boldsymbol{W}_{ij}^{\mathrm{T}}\boldsymbol{N}_{ij}$, give the approximations for $f_i(\boldsymbol{x})$ and $g_{ij}(\boldsymbol{x})$, i.e. the ith element of $\boldsymbol{F}(\boldsymbol{x})$ and the ijth element of $\boldsymbol{G}(\boldsymbol{x})$ respectively. Clearly, by letting $n = k = 1$, the conventional function emulator expression is obtained.

Let $\mathbf{A} \in R^{m \times m}$ be a stable matrix, $\mathbf{Q} \in \mathscr{R}^{m \times m}$ a given symmetric positive definite matrix and $\mathbf{P}$ a symmetric positive definite matrix that is the solution of the Lyapunov equation

$$\mathbf{A}^{\mathrm{T}}\mathbf{P} + \mathbf{P}\mathbf{A} = -\mathbf{Q} \tag{9}$$

Define $\boldsymbol{e} \in \mathscr{R}^m$, $\mathbf{B} = [\boldsymbol{b}_1 \ \boldsymbol{b}_2 \cdots \boldsymbol{b}_n] \in R^{m \times n}$, $\boldsymbol{v} \in \mathscr{R}^k$ and a conventional vector $\boldsymbol{W}_{i*} = [\boldsymbol{W}_{i1}^{\mathrm{T}} \ \boldsymbol{W}_{i2}^{\mathrm{T}} \cdots \boldsymbol{W}_{ik}^{\mathrm{T}}]^{\mathrm{T}} \in R^{m_i}$, $m_i = \sum_{j=1}^{k} n_{ij}$. The following lemma is then obtained, which will be used to show closed-loop stability of the proposed neural network adaptive controller.

Lemma 1

For an error system defined by

$$\dot{\boldsymbol{e}} = \mathbf{A}\boldsymbol{e} + \mathbf{B}\{\boldsymbol{W}\}^{\mathrm{T}}\cdot\{\boldsymbol{N}\}\boldsymbol{v} \tag{10}$$

the adaptive law that leads to uniform stability for $\boldsymbol{e}$ and $\{\boldsymbol{W}\}$ is given by

$$\dot{\boldsymbol{W}}_{i*} = -\boldsymbol{\Gamma}_i\cdot\{\boldsymbol{N}_{i*}\}\boldsymbol{v}\boldsymbol{e}^{\mathrm{T}}\mathbf{P}\boldsymbol{b}_i \tag{11}$$

where $\{\boldsymbol{N}_{i*}\}$ is a GL row vector of bounded basis functions and $\boldsymbol{\Gamma}_i \in R^{m_i} \times m_i$ is a symmetric positive-definite matrix. In addition, if $\boldsymbol{v}$ is a vector of bounded signals, then

$$\lim_{t \rightarrow \infty} \|\boldsymbol{e}\| = 0 \tag{12}$$

Proof. See Appendix 2.

This lemma is much more general than that introduced in references (**15**) and (**24**) because of the presence of the non-standard GL matrix operations. As a matter of fact, the results of the lemma also apply to other well-defined non-standard matrix operations.

3 DYNAMICS OF THE CLASS OF NON-LINEAR SYSTEMS

The systems that are most frequently encountered in the world of mechanical and electrical engineering can be described by n ordinary differential equations as

$$x_i^{(n_i)} = f_i(\boldsymbol{x}) + \sum_{j=1}^{n} g_{ij}(\boldsymbol{x})u_j, \qquad i = 1, \ldots, n \tag{13}$$

where

$$\boldsymbol{x} = [x_1, \ldots, x_1^{(n_1-2)}, x_2, \ldots, x_2^{(n_2-2)}, \ldots, x_n, \ldots, x_n^{(n_n-2)}, \boldsymbol{s}^{\mathrm{T}}]^{\mathrm{T}} \in R^m \tag{14}$$

$$\boldsymbol{s} = [s_1, s_2, \ldots, s_n]^{\mathrm{T}} = [x_1^{(n_1-1)}, x_2^{(n_2-1)}, \ldots, x_n^{(n_n-1)}]^{\mathrm{T}}, \qquad m = \sum_{i=1}^{n} n_i \tag{15}$$

and the matrix $\mathbf{G}(\boldsymbol{x}) = [g_{ij}(\boldsymbol{x})] \in R^{n\times n}$ is non-singular. The n ordinary differential equations are coupled and non-linear because of the presence of $f_i(\boldsymbol{x})$ and $g_{ij}(\boldsymbol{x})$. This is true for most mechanical servo systems, such as robotic manipulators and weapon positioning systems. The system (13) can be written in state space as

$$\dot{\boldsymbol{x}} = \mathbf{V}\boldsymbol{x} + \mathbf{B}(\boldsymbol{F}(x) + \mathbf{G}(\boldsymbol{x})u) \tag{16}$$

where

$$\mathbf{V} = \begin{bmatrix} \mathbf{U} \\ \mathbf{0}_{n\times m} \end{bmatrix}, \qquad \boldsymbol{F}(\boldsymbol{x}) = \begin{bmatrix} F_1 \\ F_2 \\ \ldots \\ F_n \end{bmatrix} \tag{17}$$

$$\mathbf{U} = \begin{bmatrix} \mathbf{U}_1 & 0 & \ldots & 0 & 0 & 0 & \ldots & 0 \\ 0 & 0 & \ldots & 0 & 1 & 0 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \mathbf{U}_n & 0 & 0 & \ldots & 0 \\ 0 & 0 & \ldots & 0 & 0 & 0 & \ldots & 1 \end{bmatrix}$$

$$\mathbf{U}_i = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix} \in \mathscr{R}^{(n_i-2)\times(n_i-1)} \tag{18}$$

$$\mathbf{B} = [\boldsymbol{b}_1 \quad \boldsymbol{b}_2 \cdots \boldsymbol{b}_n] = \begin{bmatrix} \mathbf{0}_{(m-n)\times n} \\ \mathbf{I}_{n\times n} \end{bmatrix} \tag{19}$$

In this paper, discussion will be restricted to systems having the following properties:

1. $\mathbf{G}(\boldsymbol{x})$ is positive definite.
2. $\mathbf{0} < \mathbf{G}(\boldsymbol{x}) \leqslant \alpha(\boldsymbol{x})\mathbf{I}$.

Therefore, $\mathbf{G}^{-1}(\boldsymbol{x})$ exists and it is also positive definite. It will be assumed that the upper bound, $\alpha(\boldsymbol{x})$, is known for controller design though $\mathbf{G}(\boldsymbol{x})$ may not be known exactly. With minor modifications, the proposed controller in the paper can be applied to those systems in which

1. $\boldsymbol{G}(x)$ is negative definite.
2. $-\beta(\boldsymbol{x})\mathbf{I} \leqslant \mathbf{G}(\boldsymbol{x}) < 0$.

4 ROBUST ADAPTIVE NN CONTROLLER DESIGN

For the dynamical system described in Section 3, if the functions $\boldsymbol{F}(\boldsymbol{x})$ and $\mathbf{G}(\boldsymbol{x})$ are known exactly, then for a suitable choice of $\mathbf{A}_{\mathrm{c}} \in \mathscr{R}^{n\times m}$, the control law defined by

$$\boldsymbol{u} = \mathbf{G}^{-1}(x)[\mathbf{A}_{\mathrm{c}}\boldsymbol{x} - \boldsymbol{F}(\boldsymbol{x}) + r] \tag{20}$$

will yield a stable closed-loop system for trajectory tracking. Substituting the control law (20) into (16) gives

$$\dot{\boldsymbol{x}} = \mathbf{V}\boldsymbol{x} + \mathbf{B}\mathbf{A}_{\mathrm{c}}\boldsymbol{x} + \mathbf{B}\boldsymbol{r} = \mathbf{A}\boldsymbol{x} + \mathbf{B}\boldsymbol{r}$$

where

$$\mathbf{A} = \begin{bmatrix} \mathbf{U} \\ \mathbf{A}_{\mathrm{c}} \end{bmatrix} \tag{21}$$

If $\mathbf{A}_{\mathrm{c}}$ is chosen as

$$\mathbf{A}_{\mathrm{c}} = [\mathbf{A}_{\mathrm{c}1} \quad \mathbf{A}_{\mathrm{c}2}] \tag{22}$$

with

$$\mathbf{A}_{\mathrm{c}1} = \mathrm{diag}\,[a_{i1}, a_{i2}, \ldots, a_{i(n_i-1)}] \tag{23}$$

$$\mathbf{A}_{\mathrm{c}2} = \mathrm{diag}\,[a_{in_i}] \tag{24}$$

The desired closed loop is decoupled, for $\boldsymbol{r} = [r_1, \ldots, r_n]^{\mathrm{T}} \in \mathscr{R}^n$, as

$$\dot{\boldsymbol{x}}_i = \mathbf{A}_i\boldsymbol{x}_i + r_i$$

where

$$\boldsymbol{x}_i = [x_i, x_i^{(1)}, \ldots, x_i^{(n_i-1)}]^{\mathrm{T}} \tag{25}$$

$$\mathbf{A}_i = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{i1} & a_{i2} & a_{i3} & \cdots & a_{in_i} \end{bmatrix} \in \mathscr{R}^{n_i \times n_i} \tag{26}$$

The system is then in the canonical controllable form (**25**).

In practice, however, the non-linear functions $\boldsymbol{F}(\boldsymbol{x})$ and $\mathbf{G}(\boldsymbol{x})$ are very often not known exactly except for their estimates $\overline{\boldsymbol{F}}(\boldsymbol{x})$ and $\overline{\mathbf{G}}(\boldsymbol{x})$ respectively. There are basically two methods used to obtain $\overline{\boldsymbol{F}}(\boldsymbol{x})$ and $\overline{\mathbf{G}}(\boldsymbol{x})$: (a) model-based and (b) neural-network-based methods. For the model-based method, the $\overline{\boldsymbol{F}}(\boldsymbol{x})$ and $\overline{\mathbf{G}}(\boldsymbol{x})$ are simplified versions of $\boldsymbol{F}(\boldsymbol{x})$ and $\mathbf{G}(\boldsymbol{x})$ from the viewpoint of model building (**26**, **27**), or nominal $\boldsymbol{F}(\boldsymbol{x})$ and $\mathbf{G}(\boldsymbol{x})$ with the parameters being replaced by the nominal parameter values. For NN-based, multilayer feedforward network emulators, $\overline{\boldsymbol{F}}(\boldsymbol{x})$ and $\overline{\mathbf{G}}(\boldsymbol{x})$ can be trained to approximate $\boldsymbol{F}(\boldsymbol{x})$ and $\mathbf{G}(\boldsymbol{x})$ (**14**).

To cope with the mis-matching dynamics, an adaptive segment of the parallel network is used on-line to improve the control performance. In terms of additive error description, there are the following differences:

$$\boldsymbol{D}_F(\boldsymbol{x}) := \boldsymbol{F}(\boldsymbol{x}) - \overline{\boldsymbol{F}}(\boldsymbol{x}) \tag{27}$$

$$\mathbf{D}_G(\boldsymbol{x}) := \mathbf{G}(\boldsymbol{x}) - \overline{\mathbf{G}}(\boldsymbol{x}) \tag{28}$$

Suppose that $\boldsymbol{D}_F(\mathbf{x})$ and $\mathbf{D}_G(\boldsymbol{x})$ can be approximated by the parameter and basis function pairs $(\{\boldsymbol{K}\}, \{\boldsymbol{H}(\boldsymbol{x})\})$ and $(\{\boldsymbol{W}\}, \{\boldsymbol{N}(\boldsymbol{x})\})$ as follows:

$$\boldsymbol{D}_F(\boldsymbol{x}) = \{\boldsymbol{K}\}^{\mathrm{T}} \cdot \{\boldsymbol{H}\} + \boldsymbol{E}_F \tag{29}$$

$$\mathbf{D}_G(\boldsymbol{x}) = \{\boldsymbol{W}\}^{\mathrm{T}} \cdot \{\boldsymbol{N}\} + \mathbf{E}_G \tag{30}$$

where

$$\{\boldsymbol{K}\}^{\mathrm{T}} := \begin{Bmatrix} \boldsymbol{K}_1^{\mathrm{T}} \\ \cdots \\ \boldsymbol{K}_n^{\mathrm{T}} \end{Bmatrix}, \qquad \{\boldsymbol{H}\}(\boldsymbol{x}) := \begin{Bmatrix} \boldsymbol{H}_1(\boldsymbol{x}) \\ \cdots \\ \boldsymbol{H}_n(\boldsymbol{x}) \end{Bmatrix} \tag{31}$$

$$\{\boldsymbol{W}\}^{\mathrm{T}} := \begin{Bmatrix} \boldsymbol{W}_{11}^{\mathrm{T}} & \cdots & \boldsymbol{W}_{1n}^{\mathrm{T}} \\ \cdots & \cdots & \cdots \\ \boldsymbol{W}_{n1}^{\mathrm{T}} & \cdots & \boldsymbol{W}_{nn}^{\mathrm{T}} \end{Bmatrix}$$

$$\{\mathbf{N}(\boldsymbol{x})\} := \begin{Bmatrix} \boldsymbol{N}_{11}(\boldsymbol{x}) & \cdots & \boldsymbol{N}_{1n}(\boldsymbol{x}) \\ \cdots & \cdots & \cdots \\ \boldsymbol{N}_{n1}(\boldsymbol{x}) & \cdots & \boldsymbol{N}_{nn}(\boldsymbol{x}) \end{Bmatrix} \tag{32}$$

and $\boldsymbol{E}_F$ is a vector of approximation errors $\epsilon_i(\boldsymbol{x})$ of $f_i(\boldsymbol{x}) - \overline{f}_i(\boldsymbol{x})$ and $\mathbf{E}_G$ is a matrix of approximation errors $\epsilon_{ij}(\boldsymbol{x})$ of $g_{ij}(\boldsymbol{x}) - \overline{g}_{ij}(\boldsymbol{x})$. Some of the existing model reference neural network controllers were developed under the assumptions that $\boldsymbol{E}_F = 0$ and $\mathbf{E}_G = 0$, i.e. $\boldsymbol{D}_F(\boldsymbol{x})$ and $\mathbf{D}_G(\boldsymbol{x})$ can be exactly emulated by the parameter and basis function pairs $(\{\boldsymbol{K}\}, \{\boldsymbol{H}(\boldsymbol{x})\})$ and $(\{\boldsymbol{W}\}, \{\boldsymbol{N}(\boldsymbol{x})\})$. However, this is not true in general. A sliding mode control part is introduced here for robust closed-loop stability in the presence of neural network approximations errors, $\boldsymbol{E}_F$ and $\mathbf{E}_G$.

Let the desired response of the system be given by the following reference model:

$$\dot{\boldsymbol{x}}_m = \mathbf{A}\boldsymbol{x}_m + \mathbf{B}\boldsymbol{r} \tag{33}$$

where $\boldsymbol{x}_m \in R^m$, $\boldsymbol{r} \in R^n$ and

$$\mathbf{A} = \begin{bmatrix} \mathbf{U} \\ \mathbf{A}_{\mathrm{c}} \end{bmatrix} \in R^{m \times m} \tag{34}$$

Suppose $\hat{\boldsymbol{K}}$ and $\hat{\boldsymbol{W}}$ are the estimates of $\boldsymbol{K}$ and $\boldsymbol{W}$ such that

$$\hat{\boldsymbol{D}}_F(\boldsymbol{x}) := \{\hat{\boldsymbol{K}}\}^{\mathrm{T}} \cdot \{\boldsymbol{H}\} \tag{35}$$

$$\hat{\mathbf{D}}_G(\boldsymbol{x}) := \{\hat{\boldsymbol{W}}\}^{\mathrm{T}} \cdot \{\boldsymbol{N}\} \tag{36}$$

From equations (29) and (30) and (35) and (36),

$$\tilde{\boldsymbol{D}}_F(\boldsymbol{x}) = \{\tilde{\boldsymbol{K}}\}^{\mathrm{T}} \cdot \{\boldsymbol{H}\} + \boldsymbol{E}_F \tag{37}$$

$$\tilde{\mathbf{D}}_G(\boldsymbol{x}) = \{\tilde{\boldsymbol{W}}\}^{\mathrm{T}} \cdot \{\boldsymbol{N}\} + \mathbf{E}_G \tag{38}$$

Assume that the control, $\boldsymbol{u}$, consists of two parts: a neural-network-based adaptive control part, $\boldsymbol{u}_{\mathrm{a}}$, and a robust sliding mode control part, $\boldsymbol{u}_{\mathrm{s}}$, to suppress the problems associated with the neural network modelling errors, i.e.

$$\boldsymbol{u} = \boldsymbol{u}_{\mathrm{a}} + \boldsymbol{u}_{\mathrm{s}} \tag{39}$$

which are defined as

$$\boldsymbol{u}_{\mathrm{a}} = (\overline{\mathbf{G}} + \hat{\mathbf{D}}_G)^{-1}[\mathbf{A}_{\mathrm{c}}\boldsymbol{x} - \overline{\boldsymbol{F}} - \hat{\boldsymbol{D}}_F(\boldsymbol{x}) + \boldsymbol{r}] \tag{40}$$

$$\boldsymbol{u}_{\mathrm{s}} = -K\,\mathrm{sgn}\,(\boldsymbol{e}^{\mathrm{T}}\mathbf{P}\mathbf{B}) \tag{41}$$

where $\overline{\boldsymbol{F}}$ and $\overline{\mathbf{G}}$ are the basic function approximations (either model based or feedforward neural network based), the square matrix $\hat{\mathbf{D}}_G(\boldsymbol{x})$ and the vector $\hat{\boldsymbol{D}}_F(\boldsymbol{x})$ are the function approximations arising from the additional parallel adaptive networks, K is chosen such that

$$K \geqslant \alpha(\boldsymbol{x})\|\boldsymbol{E}_F + \mathbf{E}_G\boldsymbol{u}_{\mathrm{a}})\| \tag{42}$$

and $\boldsymbol{e} = \boldsymbol{x} - \boldsymbol{x}_m$. The schematic implementation block diagram of the controller is shown in Fig. 2.

Define

$$\{\boldsymbol{S}\} := \{\{\boldsymbol{H}\} \quad \{\boldsymbol{N}\}\} = \begin{Bmatrix} \{\boldsymbol{S}_{1*}\} \\ \cdots \\ \{\boldsymbol{S}_{n*}\} \end{Bmatrix} \tag{43}$$

$$\{\boldsymbol{S}_{i*}\} = \{\boldsymbol{H}_i \quad \mathbf{N}_{i1} \cdots \mathbf{N}_{in}\}, \qquad i = 1, 2, \ldots, n \tag{44}$$

$$\boldsymbol{v} := [1, \boldsymbol{u}_{\mathrm{a}}^{\mathrm{T}}]^{\mathrm{T}} \in \mathscr{R}^{(n+1)} \tag{45}$$

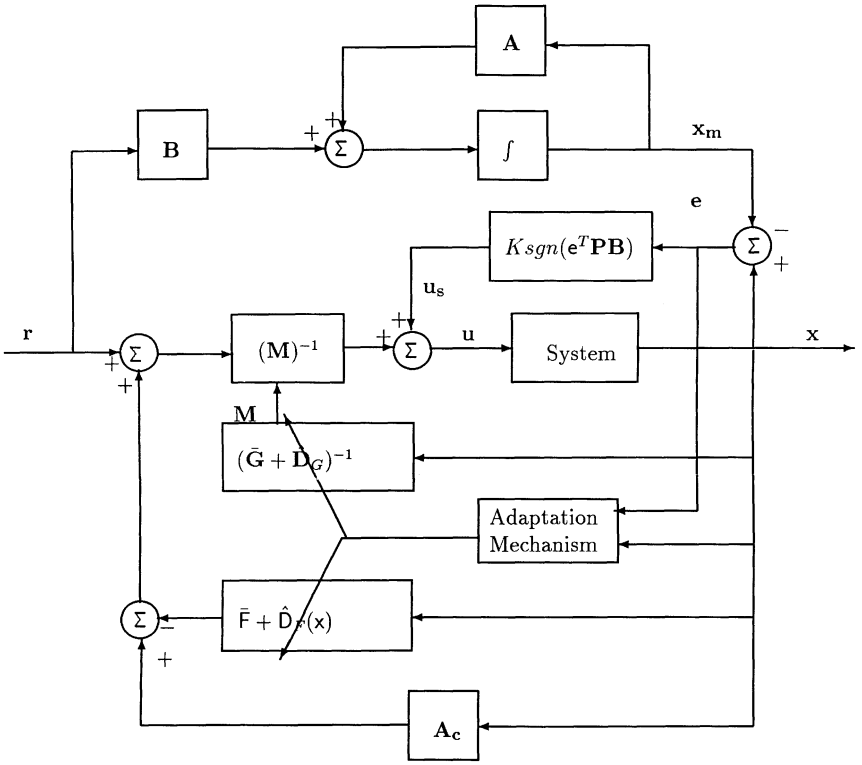

Fig. 2 Schematic implementation of the controller

The stability properties of the closed-loop system are stated in the following theorem.

Theorem 1

For the non-linear MIMO (multiple input–multiple output) system (16), consider the control law (39) which is based on a fixed controller (either model based or NN based) and an additional parallel adaptive NN for adaptive enhancement, and a sliding mode term for closed-loop robustness. If the adaptive laws for updating the parallel adaptive networks are given by

$$\begin{bmatrix} \dot{\hat{\boldsymbol{K}}}_i \\ \dot{\hat{\boldsymbol{W}}}_{i1} \\ \cdots \\ \dot{\hat{\boldsymbol{W}}}_{in} \end{bmatrix} = \{\boldsymbol{\Gamma}_i\}\cdot\{\boldsymbol{S}_{i*}\}v\boldsymbol{e}^{\mathrm{T}}\mathbf{P}\boldsymbol{b}_i \tag{46}$$

where $\boldsymbol{\Gamma}_i$ is a dimensional compatible positive definite matrix and its elements are grouped in accordance with $\mathbf{S}_i$, and $\mathbf{P}$ is the solution of the Lyapunov equation:

$$\mathbf{A}^{\mathrm{T}}\mathbf{P}+\mathbf{PA}=-\mathbf{Q}$$

where $\mathbf{Q}$ is symmetric positive definite as stated in Lemma 1. Then the direct parallel adaptively enhanced NN controller ensures that all signals ($\boldsymbol{x}$, $\boldsymbol{u}$, $\{\hat{\boldsymbol{K}}\}$ and $\{\hat{\boldsymbol{W}}\}$) in the system remain uniformly bounded for all initial conditions, and, in addition,

$$\lim_{t\to\infty}\boldsymbol{e}(t)=\lim_{t\to\infty}[\boldsymbol{x}(t)-\boldsymbol{x}_m(t)]=0$$

i.e. asymptotic tracking is achieved.

Proof. See Appendix 3.

Remarks

1. If the non-linear functions $\boldsymbol{D}_F$ and $\mathbf{D}_G$ are in the functional range of NNs, i.e. $\boldsymbol{E}_F=0$ and $\mathbf{E}_G=0$, the sliding mode control part can be dropped by letting $\boldsymbol{K}=0$ while the closed-loop system is still stable. If the non-linear functions $\boldsymbol{D}_F$ and $\mathbf{D}_G$ are not in the NN functional ranges, the existence of $\boldsymbol{K}>0$ will inevitably introduce chattering in the control signal. To solve this problem, a boundary layer can be introduced as shown in reference (**1**).
2. For the reference model

$$\dot{\boldsymbol{x}}_m=\mathbf{A}\boldsymbol{x}_m+\mathbf{B}\boldsymbol{r} \tag{47}$$

where

$$\mathbf{A}=\begin{bmatrix}\mathbf{U}\\ \mathbf{A_c}\end{bmatrix} \tag{48}$$

If $\mathbf{A}_c$ is chosen according to (22) to (24), then the model can be decoupled as

$$\dot{\boldsymbol{x}}_{im} = \mathbf{A}_i \boldsymbol{x}_m + r_i \tag{49}$$

with $\mathbf{A}_i$ defined by (26). Therefore, simple linear SISO (single input–single output) design specifications can be assigned for each degree of freedom (**15**).

3. If $\boldsymbol{\Gamma}_i$ is defined as

$$\boldsymbol{\Gamma}_i = \begin{bmatrix} \mathbf{B}_{i0} & 0 & 0 & 0 \\ 0 & \mathbf{B}_{i1} & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \mathbf{B}_{in} \end{bmatrix} \tag{50}$$

where $\mathbf{B}_{ij}$, $0 \leqslant j \leqslant n$ are dimensional compatible matrix blocks. Then, the parameter adaptation can be written as

$$\dot{\hat{\boldsymbol{K}}}_i = \mathbf{B}_{i0} \boldsymbol{H}_i \boldsymbol{e}^{\mathrm{T}} \mathbf{P} \boldsymbol{b}_i \tag{51}$$

$$\dot{\hat{\boldsymbol{W}}}_{ij} = \mathbf{B}_{ij} \boldsymbol{N}_{ij} u_j \boldsymbol{e}^{\mathrm{T}} \mathbf{P} \boldsymbol{b}_i \qquad \text{for } 1 \leqslant j \leqslant n \tag{52}$$

4. In order to implement the proposed controller, the existence of $(\overline{\mathbf{G}} + \hat{\mathbf{D}}_G)^{-1}$ is required. Its inverse does exist, provided that $(\overline{\mathbf{G}} + \hat{\mathbf{D}}_G)$ is diagonal dominant, which is directly related to the approximation accuracy of $\overline{\mathbf{G}}$. Since $\mathbf{W}_{ij}$ are bounded, the existence of $(\overline{\mathbf{G}} + \hat{\mathbf{D}}_G)^{-1}$ can be ensured by adjusting the accuracy of $\overline{\boldsymbol{F}}$ and $\overline{\mathbf{G}}$ and the adaptation gains. Therefore, the control law is well defined.
5. Intensive computer simulations have been carried out for robot control to verify its effectiveness in handling changing dynamics. It was found that the primary fixed controller alone cannot handle the changing dynamics of the systems, and the additional parallel adaptive neural network can effectively improve the system's performance (**15**) The introduction of the sliding mode term will guarantee the robustness of the closed loop but also introduce chattering in the control signals (**1**). Since all the computational results can be found in the literature, simulation results are not included here.
6. A general framework for neural network controller design has been proposed for a wide range of non-linear dynamical systems. Because of the introduction of the GL matrices and the GL operator, system analysis becomes easier and 'wasted' zero multiplications and summations in the neural networks can be avoided.

5 CONCLUSION

A general framework for robust parallel adaptive neural network controller design has been presented for a class of non-linear systems in the paper. The controller is based on applying direct adaptive techniques to an additional parallel NN to provide adaptive enhancements to a basic fixed controller which can be either NN based or model based and incorporating a sliding mode term for closed-loop robustness. It has been shown that if BBF networks are used for the additional parallel NN, uniformly stable adaptation is assured and asymptotic tracking of the position reference signal is achieved. In addition, the introduction of the GL matrices and the GL product operator was found to be very useful for system analysis.

REFERENCES

1 **Slotine, J. J. E.** and **Li, W.** *Applied Nonlinear Control*, 1991 (Prentice-Hall, Englewood Cliffs, New Jersey).
2 **Gu, Y.-L., Loh, R. N. K., Coleman, N.** and **Lin, C.-F.** Control of weapon pointing systems based on robotic formulation. In Proceedings of the American Control Conference, Chicago, Illinois, 24–26 June 1992, pp. 413–418.
3 **Blankenship, G. L., Ghanadan, R.** and **Polyakov, V.** Nonlinear adaptive control of active vehicle suspensions. Proceedings of the American Control Conference, San Francisco, California, June 1993.
4 **Mittal, M., Prasad, J. V. R.** and **Schrage, D. P.** Nonlinear adaptive control of a twin lift helicopter system. *IEEE Control Systems*, April 1991, **11**(3).
5 **Nguyen, D. H.** and **Widrow, B.** Neural networks for self-learning control systems. *IEEE Control System Mag.*, 1990, 18–23.
6 **Narendra, K. S.** and **Parthasarathy, K.** Identification and control of dynamical systems using neural networks. *IEEE Trans. on Neural Networks*, 1990, **1**, 4–27.
7 **Ozaki, T., Suzuki, T., Furuhashi, T., Okuma, S.** and **Uchikawa, Y.** Trajectory control of robotic manipulators using neural networks. *IEEE Trans. on Ind. Electronics*, 1991, **38**, 195–202.
8 **Lee, T. H., Hang, C. C., Lian, L. L.** and **Lim, B. C.** An approach to the inverse nonlinear control using neural networks. *Mechatronics*, 1992, **2**, 595–611.
9 **Sanner, R. M.** and **Slotine, J. J. E.** Gaussian networks for direct adaptive control. In Proceedings of the 1991 ACC, 1991, pp. 2153–2159.
10 **Sanner, R. M.** and **Slotine, J. J. E.** Stable adaptive control and recursive identification using radial Gaussian networks. In Proceedings of the Thirtieth IEEE CDC, 1991.
11 **Lewis, F. L., Yesildirek, A.** and **Liu, K.** Neural net robot controller: structure and stability proofs. In Proceedings of the Thirty-second Conference on *Decision and Control*, 1993.
12 **Lewis, F. L., Liu, K.** and **Yesildirek, A.** Neural net robot controller with guaranteed tracking performance. *IEEE Trans. on NN*, May 1995, **6**(3).
13 **Lightbody, G.** and **Irwin, G. W.** Direct neural model reference adaptive control. *IEE Proc.—Control Theory Applic.*, January 1995, **142**(1).
14 **Lee, T. H.** and **Tan, W. K.** Real-time parallel adaptive neural network control for nonlinear servomechanisms—an approach using direct adaptive technique. *Mechatronics*, 1993, **3**(6), 705–725.
15 **Ge, S. S.** and **Lee, T. H.** Parallel adaptive neural network

control of robots. *Proc. Instn Mech. Engrs, Part I*, 1994, **208**(I4), 231–237.
16 **Ge, S. S., Wang, Z.-L.** and **Chen Z.-J.** Adaptive static neural network control of robots. In IEEE International Conference on *Industrial Technology*, Guangzhou, P.R. China, 5–9 December 1994.
17 **Tzirkel-Hancock, E.** and **Fallside, F.** A direct control method for a class of nonlinear systems using neural networks. Cambridge University paper CUED/F-INFENG/TR65, 1991.
18 **Tzirkel-Hancock, E.** and **Fallside, F.** Stable control of nonlinear systems using neural networks. Cambridge University paper CUED/F-INFENG/TR81, 1991.
19 **Miller III, W. T., Glanz, F. H.** and **Kraft III, L. G.** CMAC: an associative neural network alternative to back propagation. *Proc. IEEE*, 1990, **78**, 1561–1567.
20 **Khanna, T.** *Foundations of Neural Networks*, 1990 (Addison-Wesley, Reading, Massachusetts).
21 **Poggio, T.** and **Girosi, F.** A theory of networks for approximation and learning. Artificial Intelligence Lab. Memo 1140, MIT, Cambridge, Massachusetts, July 1989.
22 **Girosi, F.** and **Poggio, T.** Networks and the best approximation property. Artificial Intelligence Lab. Memo 1164, MIT, Cambridge, Massachusetts, October 1989.
23 **Poggio, T.** and **Girosi, F.** Networks for approximation and learning. *Proc. IEEE*, 1990, **78**, 1481–1497.
24 **Narendra, K. S.** and **Annaswamy, A. M.** *Adaptive Control*, 1989 (Addison-Wesley, Reading, Massachusetts).
25 **Isidori, A.** *Nonlinear Control Systems*, 1985 (Springer-Verlag, Berlin).
26 **Paul, R. P.** *Robot Manipulators: Mathematics, Programming, and Control*, 1981 (MIT Press, Cambridge, Massachusetts).
27 **Craig, J.** *Introduction to Robotics: Mechanics and Control*, 1980 (Addison-Wesley, Reading, Massachusetts).

APPENDIX 1

'GL' matrix and operator

Let I_0 be the set of integers, and $\boldsymbol{W}_{ij}$, $\boldsymbol{N}_{ij}(\boldsymbol{x}) \in R^{n_{ij}}$, $n_{ij} \in I_0$, $i = 1, \ldots, n$, $j = 1, \ldots, k$. The product $\boldsymbol{W}_{ij}^{\mathrm{T}}\boldsymbol{N}_{ij}$ can be taken as a network emulator of the ijth element, $g_{ij}(\boldsymbol{x})$ of matrix $\mathbf{G}(\boldsymbol{x}) \in R^{n\times k}$, with $\boldsymbol{W}_{ij}$ and $\boldsymbol{N}_{ij}(\boldsymbol{x})$ the weight and basis function vectors respectively. Because functions $g_{ij}(\boldsymbol{x})$ are different in terms of complexity and smoothness, dimensions n_{ij} may be chosen differently. However, as long as n_{ij} are fixed, the matrices

$$\{\mathbf{W}\} = \begin{Bmatrix} \mathbf{W}_{11} & \mathbf{W}_{12} & \cdots & \mathbf{W}_{1k} \\ \mathbf{W}_{21} & \mathbf{W}_{22} & \cdots & \mathbf{W}_{2k} \\ \cdots & \cdots & \cdots & \cdots \\ \mathbf{W}_{n1} & \mathbf{W}_{n2} & \cdots & \mathbf{W}_{nk} \end{Bmatrix} \tag{53}$$

$$\{\mathbf{N}\} = \begin{Bmatrix} \mathbf{N}_{11} & \mathbf{N}_{12} & \cdots & \mathbf{N}_{1k} \\ \mathbf{N}_{21} & \mathbf{N}_{22} & \cdots & \mathbf{N}_{2k} \\ \cdots & \cdots & \cdots & \cdots \\ \mathbf{N}_{n1} & \mathbf{N}_{n2} & \cdots & \mathbf{N}_{nk} \end{Bmatrix} \tag{54}$$

are well defined. For clarity, they are referred to as the 'GL' matrices and denoted by $\{*\}$. The corresponding operator (the GL operator or product, say), denoted by $\cdot$, is defined as below:

$$\mathbf{G}(\boldsymbol{x}) = \{\boldsymbol{W}\}^{\mathrm{T}}\cdot\{\boldsymbol{N}\} := \begin{bmatrix} \mathbf{W}_{11}^{\mathrm{T}}\mathbf{N}_{11} & \mathbf{W}_{12}^{\mathrm{T}}\mathbf{N}_{12} & \cdots & \mathbf{W}_{1k}^{\mathrm{T}}\mathbf{N}_{1k} \\ \mathbf{W}_{21}^{\mathrm{T}}\mathbf{N}_{21} & \mathbf{W}_{22}^{\mathrm{T}}\mathbf{N}_{22} & \cdots & \mathbf{W}_{2k}^{\mathrm{T}}\mathbf{N}_{2k} \\ \cdots & \cdots & \cdots & \cdots \\ \mathbf{W}_{n1}^{\mathrm{T}}\mathbf{N}_{n1} & \mathbf{W}_{n2}^{\mathrm{T}}\mathbf{N}_{n2} & \cdots & \mathbf{W}_{nk}^{\mathrm{T}}\mathbf{N}_{nk} \end{bmatrix} \in R^{n\times k} \tag{55}$$

Therefore, a matrix network emulator can be conveniently expressed as a GL product of two GL matrices: a parameter matrix and a basis function matrix.

Letting $k = 1$, a vector emulator can be conveniently expressed as a GL product of two GL vectors as follows:

$$\boldsymbol{F}(\boldsymbol{x}) = \{\boldsymbol{K}\}^{\mathrm{T}}\cdot\{\boldsymbol{H}\} := \begin{bmatrix} \boldsymbol{K}_1^{\mathrm{T}}\boldsymbol{H}_1 \\ \boldsymbol{K}_2^{\mathrm{T}}\boldsymbol{H}_2 \\ \cdots \\ \boldsymbol{K}_n^{\mathrm{T}}\boldsymbol{H}_n \end{bmatrix} \tag{56}$$

where

$$\{\boldsymbol{K}\} := \begin{Bmatrix} \boldsymbol{K}_1 \\ \boldsymbol{K}_2 \\ \cdots \\ \boldsymbol{K}_n \end{Bmatrix}, \qquad \{\boldsymbol{H}\} := \begin{Bmatrix} \boldsymbol{H}_1 \\ \boldsymbol{H}_2 \\ \cdots \\ \boldsymbol{H}_n \end{Bmatrix}$$

with $\boldsymbol{K}_i$, $\boldsymbol{H}_i \in R^{n_i}$, $i = 1, \ldots, n$. Therefore, the corresponding element-wise inner products, $\boldsymbol{K}_i^{\mathrm{T}}\boldsymbol{H}_i$ and $\boldsymbol{W}_{ij}^{\mathrm{T}}\boldsymbol{N}_{ij}$ give the approximations for $f_i(\boldsymbol{x})$ and $g_{ij}(\boldsymbol{x})$ respectively, i.e. the ith element of $\boldsymbol{F}(\boldsymbol{x})$ and ijth element of $\mathbf{G}(\boldsymbol{x})$ respectively. Clearly, by letting $n = k = 1$, the conventional function emulator expression is obtained.

In a rough analogy to the definitions of the row and the column vectors of a matrix, the GL row and the column vectors of a GL matrix can be defined. For the GL matrix $\{\mathbf{W}\}$ in equation (53), the corresponding GL row vector $\{\boldsymbol{W}_{i*}\}$ is defined as

$$\{\boldsymbol{W}_{i*}\} = \{\mathbf{W}_{i1} \quad \mathbf{W}_{i2} \cdots \mathbf{W}_{ik}\} \tag{57}$$

and the GL column vector $\{\boldsymbol{W}_{*j}\}$ is defined as

$$\{\boldsymbol{W}_{*j}\} := \begin{Bmatrix} \mathbf{W}_{1j} \\ \mathbf{W}_{2j} \\ \cdots \\ \mathbf{W}_{nj} \end{Bmatrix} \tag{58}$$

Though n_{ij}, $i = 1, \ldots, n$, $j = 1, \ldots, k$ may be different, the vector is uniquely defined as described for known n_{ij}.

The GL matrix $\{\mathbf{W}\}$ can also be expressed as

$$\{\mathbf{W}\} := \{\{\mathbf{W}_{*1}\} \quad \{\mathbf{W}_{*2}\} \cdots \{\mathbf{W}_{*k}\}\} = \begin{Bmatrix} \{\mathbf{W}_{1*}\} \\ \{\mathbf{W}_{2*}\} \\ \cdots \\ \{\mathbf{W}_{n*}\} \end{Bmatrix}$$

For completeness and ease of manipulations, their corresponding GL transposes $\{\mathbf{W}_{i*}\}^{\mathrm{T}}$, $\{\mathbf{W}_{*i}\}^{\mathrm{T}}$ and $\{\mathbf{W}\}^{\mathrm{T}}$ are defined as

$$\{\mathbf{W}_{i*}\}^{\mathrm{T}} := \{\mathbf{W}_{i1}^{\mathrm{T}} \quad \mathbf{W}_{i2}^{\mathrm{T}} \cdots \mathbf{W}_{ik}^{\mathrm{T}}\} \tag{59}$$

$$\{\mathbf{W}_{*i}\}^{\mathrm{T}} := \begin{Bmatrix} \mathbf{W}_{1i}^{\mathrm{T}} \\ \mathbf{W}_{2i}^{\mathrm{T}} \\ \cdots \\ \mathbf{W}_{ni}^{\mathrm{T}} \end{Bmatrix} \tag{60}$$

$$\{\mathbf{W}\}^{\mathrm{T}} := \begin{Bmatrix} \mathbf{W}_{11}^{\mathrm{t}} & \mathbf{W}_{12}^{\mathrm{T}} & \cdots & \mathbf{W}_{1k}^{\mathrm{T}} \\ \mathbf{W}_{21}^{\mathrm{T}} & \mathbf{W}_{22}^{\mathrm{T}} & \cdots & \mathbf{W}_{2k}^{\mathrm{T}} \\ \cdots & \cdots & \cdots & \cdots \\ \mathbf{W}_{n1}^{\mathrm{T}} & \mathbf{W}_{n2}^{\mathrm{T}} & \cdots & \mathbf{W}_{nk}^{\mathrm{T}} \end{Bmatrix} \tag{61}$$

As a matter of fact, $\{\mathbf{W}\}$ can also be expressed as

$$\{\mathbf{W}\}^{\mathrm{T}} := \{\{\mathbf{W}_{*1}\}^{\mathrm{T}} \cdots \{\mathbf{W}_{*k}\}^{\mathrm{T}}\} = \begin{Bmatrix} \{\mathbf{W}_{1*}\}^{\mathrm{T}} \\ \cdots \\ \{\mathbf{W}_{n*}\}^{\mathrm{T}} \end{Bmatrix}$$

It can be seen that the transposes of GL vectors and matrices transpose its elementary vectors locally. Define $\mathbf{W}_{i*}$ and $\mathbf{W}_{i*}^{\mathrm{T}}$ in the conventional way as

$$\mathbf{W}_{i*} = \begin{bmatrix} \mathbf{W}_{i1} \\ \mathbf{W}_{i2} \\ \cdots \\ \mathbf{W}_{ik} \end{bmatrix} \in R^{m_i}, \qquad \mathbf{W}_{i*}^{\mathrm{T}} = [\mathbf{W}_{i1}^{\mathrm{T}} \quad \mathbf{W}_{i2}^{\mathrm{T}} \cdots \mathbf{W}_{ik}^{\mathrm{T}}]$$

where $m_i = \sum_{j=1}^{k} n_{ij}$. Should any confusion arise in the text, $[*]$ is used to denote an ordinary matrix and $\{*\}$ for a GL matrix explicitly.

The GL matrix and its product have the following properties:

1. The definition of the transpose of a GL vector is different from that of a conventional vector. It only transposes its elementary vectors *locally*.
2. The transpose of a GL row vector equals the transpose of the corresponding conventional vector, e.g. $\{\boldsymbol{W}_{i*}\}^{\mathrm{T}} = \boldsymbol{W}_{i*}^{\mathrm{T}}$.
3. A GL column vector is a regular vector, e.g. $\{\boldsymbol{W}_{*i}\} = \boldsymbol{W}_{*i}$.
4. A conventional vector (matrix) can be taken as a GL vector (matrix) by grouping the elements of the vector (matrix) accordingly.
5. A GL product of two GL matrices (vectors) leads to a conventional matrix (vector) which is well defined, e.g. $[(\boldsymbol{W}\}^{\mathrm{T}}\cdot\{\boldsymbol{N}\}]$ and $[\{\boldsymbol{K}\}^{\mathrm{T}}\cdot\{\boldsymbol{H}\}]$.

The GL product of a square matrix and a GL row vector are defined as follows. Let a GL row vector $\{\mathbf{S}_{i*}\}$ be defined as $\{\boldsymbol{S}_{i*}\} = \{\boldsymbol{S}_{i1}\ \boldsymbol{S}_{i2} \cdots \mathbf{S}_{in}\}$, $\forall \mathbf{S}_{ij} \in \mathscr{R}^{n_{ij}}$, $j = 1, 2, \ldots, n$, and a matrix $\boldsymbol{\Gamma}_i = \boldsymbol{\Gamma}_i^{\mathrm{T}} = [\boldsymbol{\Gamma}_{i1}\ \boldsymbol{\Gamma}_{i2} \cdots \boldsymbol{\Gamma}_{in}]$, $\forall \boldsymbol{\Gamma}_{ij} \in \mathscr{R}^{m \times n_{ij}}$, $m = \sum_{j=1}^{n} n_{ij}$, $j = 1, 2, \ldots, n$. Then

$$\begin{aligned} \boldsymbol{\Gamma}_i\cdot\{\boldsymbol{S}_{i*}\} &= \{\boldsymbol{\Gamma}_i\}\cdot\{\boldsymbol{S}_{i*}\} \\ &:= [\boldsymbol{\Gamma}_{i1} \quad \boldsymbol{S}_{i1} \quad \boldsymbol{\Gamma}_{i2} \quad \boldsymbol{S}_{i2} \cdots \boldsymbol{\Gamma}_{in}\boldsymbol{S}_{in}] \in R^{m \times n} \end{aligned} \tag{62}$$

The GL product should be computed first in a mixed matrix product. For example, in $\{\mathbf{A}\}\cdot\{\mathbf{B}\}\mathbf{C}$, matrix $[\{\mathbf{A}\}\cdot\{\mathbf{B}\}]$ should be computed first to obtain a conventional matrix $[\{\mathbf{A}\}\cdot\{\mathbf{B}\}]$ and then $[\{\mathbf{A}\}\cdot\{\mathbf{B}\}]\mathbf{C}$ should be computed.

APPENDIX 2

Proof of Lemma 1

For the stable matrix $\mathbf{A}$ and a given symmetric positive definite matrix $\mathbf{Q}$, let $\mathbf{P}$ be the symmetric positive definite matrix solution to

$$\mathbf{A}^{\mathrm{T}}\mathbf{P} + \mathbf{P}\mathbf{A} = -\mathbf{Q} \tag{63}$$

Define the non-negative function $\mathbf{V}$ as

$$V(\boldsymbol{e}, \mathbf{W}) = \boldsymbol{e}^{\mathrm{T}}\mathbf{P}\boldsymbol{e} + \sum_{i=1}^{m} \mathbf{W}_{i*}^{\mathrm{T}}\boldsymbol{\Gamma}_i^{-1}\mathbf{W}_{i*} \tag{64}$$

where $\boldsymbol{\Gamma}_i$ is a dimensional compatible symmetric positive-definite matrix. Its time derivative along equation (10) is given by

$$\begin{aligned} \dot{V} &= -\boldsymbol{e}^{\mathrm{T}}\mathbf{Q}\boldsymbol{e} + 2\boldsymbol{e}^{\mathrm{T}}\mathbf{P}\mathbf{B}\{\mathbf{W}\}^{\mathrm{T}}\cdot\{\mathbf{N}\}\boldsymbol{v} \\ &\quad + 2\sum_{i=1}^{n} \mathbf{W}_{i*}^{\mathrm{T}}\boldsymbol{\Gamma}_i^{-1}\dot{\mathbf{W}}_{i*} \\ &= -\boldsymbol{e}^{\mathrm{T}}\mathbf{Q}\boldsymbol{e} + 2\boldsymbol{e}^{\mathrm{T}}\mathbf{P}[b_1 \quad b_2 \cdots b_n]\{\mathbf{W}\}^{\mathrm{T}}\cdot\{\mathbf{N}\}\boldsymbol{v} \\ &\quad + 2\sum_{i=1}^{n} \mathbf{W}_{i*}^{\mathrm{T}}\boldsymbol{\Gamma}_i^{-1}\dot{\mathbf{W}}_{i*} \\ &= -\boldsymbol{e}^{\mathrm{T}}\mathbf{Q}\boldsymbol{e} + 2[\boldsymbol{e}^{\mathrm{T}}\mathbf{P}\boldsymbol{b}_1 \quad \boldsymbol{e}^{\mathrm{T}}\mathbf{P}\boldsymbol{b}_2 \cdots \boldsymbol{e}^{\mathrm{T}}\mathbf{P}\boldsymbol{b}_n] \\ &\qquad \begin{bmatrix} \{\mathbf{W}_1\}^{\mathrm{T}}\cdot\{\mathbf{N}_1\}\boldsymbol{v} \\ \{\mathbf{W}_2\}^{\mathrm{T}}\cdot\{\mathbf{N}_2\}\boldsymbol{v} \\ \cdots \\ \{\mathbf{W}_n\}^{\mathrm{T}}\cdot\{\mathbf{N}_n\}\boldsymbol{v} \end{bmatrix} + 2\sum_{i=1}^{n} \mathbf{W}_{i*}^{\mathrm{T}}\boldsymbol{\Gamma}_i^{-1}\dot{\mathbf{W}}_{i*} \\ &= -\boldsymbol{e}^{\mathrm{T}}\mathbf{Q}\boldsymbol{e} + 2\sum_{i=1}^{n} \boldsymbol{e}^{\mathrm{T}}\mathbf{P}\boldsymbol{b}_i\{\boldsymbol{W}_{i*}\}^{\mathrm{T}}\cdot\{\boldsymbol{N}_{i*}\}\boldsymbol{v} \\ &\quad + 2\sum_{i=1}^{n} \mathbf{W}_{i*}^{\mathrm{T}}\boldsymbol{\Gamma}_i^{-1}\dot{\mathbf{W}}_{i*} \end{aligned} \tag{65}$$

Because $\{\boldsymbol{W}_{i*}\}$ is a GL row vector, $\mathbf{W}_{i*}^{\mathrm{T}} = \{\boldsymbol{W}_{i*}\}^{\mathrm{T}}$. Therefore,

$$\dot{V} = -\boldsymbol{e}^{\mathrm{T}}\mathbf{Q}\boldsymbol{e} + 2\sum_{i=1}^{n}\boldsymbol{e}^{\mathrm{T}}\mathbf{P}\boldsymbol{b}_i\{\boldsymbol{W}_{i*}\}^{\mathrm{T}}\cdot\{\boldsymbol{N}_{i*}\}\boldsymbol{v} + 2\sum_{i=1}^{n}\{\boldsymbol{W}_{i*}\}^{\mathrm{T}}\boldsymbol{\Gamma}_i^{-1}\dot{\mathbf{W}}_{i*} \tag{66}$$

Substituting equation (11), namely

$$\dot{\boldsymbol{W}}_{i*} = -\{\boldsymbol{\Gamma}_i\}\cdot\{\boldsymbol{N}_{i*}\}\boldsymbol{v}\boldsymbol{e}^{\mathrm{T}}\mathbf{P}\boldsymbol{b}_i \tag{67}$$

gives

$$\dot{V} = -\boldsymbol{e}^{\mathrm{T}}\mathbf{Q}\boldsymbol{e} \tag{68}$$

Hence, $V(\boldsymbol{e}, \mathbf{W})$ is a Lyapunov function and the following can be concluded:

1. $\boldsymbol{e} \in L^{\infty}$ and $\mathbf{W}_{i*} \in L^{\infty}$.
2. Since $\int_{t_0}^{\infty}\dot{V}\,\mathrm{d}t < \infty$, then $\boldsymbol{e} \in L^2$.
3. If $\boldsymbol{v} \in L^{\infty}$, $\dot{\boldsymbol{e}}$ is bounded (since $\{\mathbf{N}\}$ is a bounded basis function).

Therefore it can be concluded that

$$\lim_{t\to\infty}\|\boldsymbol{e}\| = 0 \tag{69}$$

APPENDIX 3

Proof of asymptotic stability

The control given by equation (40) can be written as

$$(\overline{\mathbf{G}} + \hat{\mathbf{D}}_G)\boldsymbol{u}_{\mathrm{a}} = \mathbf{A}_{\mathrm{c}}\boldsymbol{x} - \overline{\boldsymbol{F}} - \hat{\boldsymbol{D}}_{\mathrm{F}} + \boldsymbol{r} \tag{70}$$

Therefore, equation (16) becomes

$$\begin{aligned}\dot{\boldsymbol{x}} &= \mathbf{V}\boldsymbol{x} + \mathbf{B}\boldsymbol{F}(\boldsymbol{x}) + \mathbf{B}\mathbf{G}\boldsymbol{u}_{\mathrm{a}} + \mathbf{B}\mathbf{G}\boldsymbol{u}_{\mathrm{s}} - \mathbf{B}(\overline{\mathbf{G}} + \hat{\mathbf{D}}_G)\boldsymbol{u}_{\mathrm{a}} + \mathbf{B}(\overline{\mathbf{G}} + \hat{\mathbf{D}}_G)\boldsymbol{u}_{\mathrm{a}}\\ &= \mathbf{V}\boldsymbol{x} + \mathbf{B}\boldsymbol{F}(\boldsymbol{x}) + \mathbf{B}\mathbf{G}\boldsymbol{u}_{\mathrm{a}} - \mathbf{B}(\overline{\mathbf{G}} + \hat{\mathbf{D}}_G)\boldsymbol{u}_{\mathrm{a}} + \mathbf{B}\mathbf{A}_{\mathrm{c}}\boldsymbol{x} - \mathbf{B}(\overline{\boldsymbol{F}} + \hat{\boldsymbol{D}}_F) + \mathbf{B}\boldsymbol{r} + \mathbf{B}\mathbf{G}\boldsymbol{u}_{\mathrm{s}}\\ &= \mathbf{A}\boldsymbol{x} + \mathbf{B}[\boldsymbol{F} - (\overline{\boldsymbol{F}} + \hat{\boldsymbol{D}}_F)] + \mathbf{B}[\mathbf{G} - \overline{\mathbf{G}} - \hat{\mathbf{D}}_G]\boldsymbol{u}_{\mathrm{a}} + \mathbf{B}\boldsymbol{r} + \mathbf{B}\mathbf{G}\boldsymbol{u}_{\mathrm{s}}\end{aligned} \tag{71}$$

where

$$\mathbf{A} = \begin{bmatrix}\mathbf{U}\\ \mathbf{A}_{\mathrm{c}}\end{bmatrix} \tag{72}$$

By combining equations (33) and (71), the error equations are obtained:

$$\begin{aligned}\dot{\boldsymbol{e}} &= \mathbf{A}\boldsymbol{e} + \mathbf{B}[\boldsymbol{F} - \overline{\boldsymbol{F}} - \hat{\boldsymbol{D}}_F] + \mathbf{B}[\mathbf{G} - \overline{\mathbf{G}} - \hat{\mathbf{D}}_G]\boldsymbol{u}_{\mathrm{a}} + \mathbf{B}\mathbf{G}\boldsymbol{u}_{\mathrm{s}}\\ &= \mathbf{A}\boldsymbol{e} + \mathbf{B}[\boldsymbol{D}_F - \hat{\mathbf{D}}_F] + \mathbf{B}[\mathbf{D}_G - \hat{\mathbf{D}}_G]\boldsymbol{u}_{\mathrm{a}} + \mathbf{B}\mathbf{G}\boldsymbol{u}_{\mathrm{s}}\\ &= \mathbf{A}\boldsymbol{e} + \mathbf{B}[\tilde{\mathbf{D}}_F] + \mathbf{B}[\tilde{\mathbf{D}}_G]\boldsymbol{u}_{\mathrm{a}} + \mathbf{B}\mathbf{G}\boldsymbol{u}_{\mathrm{s}}\end{aligned} \tag{73}$$

Substituting equations (37) and (38), i.e.

$$\tilde{\boldsymbol{D}}_F = \{\tilde{\boldsymbol{K}}\}^{\mathrm{T}}\cdot\{\boldsymbol{H}\} + \boldsymbol{E}_F \tag{74}$$

$$\tilde{\mathbf{E}}_G = \{\tilde{\boldsymbol{W}}\}^{\mathrm{T}}\cdot\{\mathbf{N}\} + \mathbf{E}_G \tag{75}$$

into equation (73), gives

$$\dot{\boldsymbol{e}} = \mathbf{A}\boldsymbol{e} + \mathbf{B}\{\tilde{\boldsymbol{K}}\}^{\mathrm{T}}\cdot\{\boldsymbol{H}\} + \mathbf{B}\{\tilde{\boldsymbol{W}}\}^{\mathrm{T}}\cdot\{\boldsymbol{N}\}\boldsymbol{u}_{\mathrm{a}} + \mathbf{B}[\boldsymbol{E}_F + \mathbf{E}_G\boldsymbol{u}_{\mathrm{a}}] + \mathbf{B}\mathbf{G}u_{\mathrm{s}} \tag{76}$$

which can be further written as

$$\dot{\boldsymbol{e}} = \mathbf{A}\boldsymbol{e} + \mathbf{B}\{\boldsymbol{X}\}^{\mathrm{T}}\cdot\{\boldsymbol{S}\}\mathbf{v} + \mathbf{B}[\boldsymbol{E}_F + \mathbf{E}_G\boldsymbol{u}_{\mathrm{a}}] + \mathbf{B}\mathbf{G}\boldsymbol{u}_{\mathrm{s}} \tag{77}$$

where

$$\{\boldsymbol{X}\}^{\mathrm{T}} := \{\{\tilde{\boldsymbol{K}}\}^{\mathrm{T}} \quad \{\tilde{\boldsymbol{W}}\}^{\mathrm{T}}\} = \begin{Bmatrix}\tilde{\boldsymbol{K}}_1^{\mathrm{T}} & \tilde{\boldsymbol{W}}_{11}^{\mathrm{T}} & \tilde{\boldsymbol{W}}_{12}^{\mathrm{T}} & \cdots & \tilde{\boldsymbol{W}}_{1n}^{\mathrm{T}}\\ \cdots & \cdots & \cdots & \cdots & \cdots\\ \tilde{\boldsymbol{K}}_n^{\mathrm{T}} & \tilde{\boldsymbol{W}}_{n1}^{\mathrm{T}} & \tilde{\boldsymbol{W}}_{n2}^{\mathrm{T}} & \cdots & \tilde{\boldsymbol{W}}_{nn}^{\mathrm{T}}\end{Bmatrix}$$

$$\{\boldsymbol{S}\} := \{\{\boldsymbol{H}\} \quad \{\boldsymbol{N}\}\} = \begin{Bmatrix}\boldsymbol{H}_1 & \boldsymbol{N}_{11} & \boldsymbol{N}_{12} & \cdots & \boldsymbol{N}_{1n}\\ \cdots & \cdots & \cdots & \cdots & \cdots\\ \boldsymbol{H}_n & \boldsymbol{N}_{n1} & \boldsymbol{N}_{n2} & \cdots & \boldsymbol{N}_{nn}\end{Bmatrix}$$

$$\boldsymbol{v} := [1, \boldsymbol{u}_{\mathrm{a}}^{\mathrm{T}}]^{\mathrm{T}} \in \mathscr{R}^{(n+1)}$$

Note that $\{\boldsymbol{X}\}^{\mathrm{T}}$ can also be written as

$$\{\boldsymbol{X}\}^{\mathrm{T}} = \begin{Bmatrix}\{\boldsymbol{X}_{1*}\}^{\mathrm{T}}\\ \{\boldsymbol{X}_{2*}\}^{\mathrm{T}}\\ \cdots\\ \{\boldsymbol{X}_{n*}\}^{\mathrm{T}}\end{Bmatrix} \tag{78}$$

with $\{\boldsymbol{X}_{i*}\}$ defined as

$$\{\boldsymbol{X}_{i*}\} = \{\tilde{\boldsymbol{K}}_i \quad \tilde{\boldsymbol{W}}_{i1} \quad \tilde{\boldsymbol{W}}_{i2} \cdots \tilde{\boldsymbol{W}}_{in}\} \tag{79}$$

which is a GL row vector and its GL transpose $\{\boldsymbol{X}_{i*}\}^{\mathrm{T}}$ is the transpose of the corresponding regular vector:

$$\boldsymbol{X}_{i*}^{\mathrm{T}} = [\tilde{\boldsymbol{K}}_i^{\mathrm{T}} \quad \tilde{\boldsymbol{W}}_{i1}^{\mathrm{T}} \quad \tilde{\boldsymbol{W}}_{i2}^{\mathrm{T}} \cdots \tilde{\boldsymbol{W}}_{in}^{\mathrm{T}}] \tag{80}$$

Now the system has been successfully formulated into a form similar to (10), and is given by

$$\dot{\boldsymbol{e}} = \mathbf{A}\boldsymbol{e} + \mathbf{B}\{\boldsymbol{X}\}^{\mathrm{T}}\cdot\{\boldsymbol{S}\}\boldsymbol{v} + \mathbf{B}[\boldsymbol{E}_F + \mathbf{E}_G\boldsymbol{u}_{\mathrm{a}} + \mathbf{G}\boldsymbol{u}_{\mathrm{s}}] \quad (81)$$

For the stable matrix $\mathbf{A}$ and a given symmetric positive definite matrix $\mathbf{Q}$, let $\mathbf{P}$ be the symmetric positive definite matrix solution to

$$\mathbf{A}^{\mathrm{T}}\mathbf{P} + \mathbf{P}\mathbf{A} = -\mathbf{Q} \quad (82)$$

Define the non-negative function $\mathbf{V}$ as

$$\mathbf{V}(\boldsymbol{e}, \mathbf{X}) = \boldsymbol{e}^{\mathrm{T}}\mathbf{P}\boldsymbol{e} + \sum_{i=1}^{m}\mathbf{X}_{i*}^{\mathrm{T}}\boldsymbol{\Gamma}_i^{-1}\mathbf{X}_{i*} \quad (83)$$

where $\boldsymbol{\Gamma}_i$ is a dimensional compatible symmetric positive-definite matrix. Following the proof of Lemma 1, it is known that if

$$\dot{\mathbf{X}}_{i*} = -\{\boldsymbol{\Gamma}_i\}\cdot\{\boldsymbol{S}_i\}\boldsymbol{v}\boldsymbol{e}^{\mathrm{T}}\mathbf{P}\boldsymbol{b}_i \quad (84)$$

then

$$\begin{aligned}\dot{V}(\boldsymbol{e}, \mathbf{X}) &= -\mathbf{e}^{\mathrm{T}}\boldsymbol{Q}\mathbf{e} + 2\boldsymbol{e}^{\mathrm{T}}\mathbf{P}\mathbf{B}[\boldsymbol{E}_F + \mathbf{E}_G\boldsymbol{u}_{\mathrm{a}} + \mathbf{G}\boldsymbol{u}_{\mathrm{s}}] \\ &= -\boldsymbol{e}^{\mathrm{T}}\mathbf{Q}\boldsymbol{e} + 2\boldsymbol{e}^{\mathrm{T}}\mathbf{P}\mathbf{B}\mathbf{G}^{-1}[\mathbf{G}(\boldsymbol{E}_F + \mathbf{E}_G\boldsymbol{u}_{\mathrm{a}}) + \boldsymbol{u}_{\mathrm{s}}]\end{aligned}$$

Since the robust control $\boldsymbol{u}_{\mathrm{s}}$ (41) is defined as

$$\boldsymbol{u}_{\mathrm{s}} = -K\,\mathrm{sgn}\,(\boldsymbol{e}^{\mathrm{T}}\mathbf{P}\mathbf{B}) \quad (85)$$

and K (42) is chosen such that

$$\begin{aligned}K &\geqslant \alpha\|\boldsymbol{E}_F + \mathbf{E}_G\boldsymbol{u}_{\mathrm{a}})\| \\ &\geqslant \|\mathbf{G}(\boldsymbol{E}_F + \mathbf{E}_G\boldsymbol{u}_{\mathrm{a}})\|\end{aligned}$$

then

$$\dot{V} \leqslant -\boldsymbol{e}^{\mathrm{T}}\mathbf{Q}\boldsymbol{e} \leqslant 0 \quad (86)$$

which results in uniform boundedness of $\boldsymbol{e}$ and subsequently the uniform boundedness of $\mathbf{X}_{i*}$. Since $\dot{\tilde{\boldsymbol{H}}}_i = -\dot{\hat{\boldsymbol{H}}}_i$ and $\dot{\tilde{\mathbf{W}}}_{ij} = -\dot{\hat{\mathbf{W}}}_{ij}$, $j = 1, \ldots, n$, the update laws for the parameters of the parallel adaptive neural network are obtained as

$$\begin{bmatrix}\dot{\hat{\boldsymbol{K}}}_i \\ \dot{\hat{\mathbf{W}}}_{i1} \\ \ldots \\ \dot{\hat{\mathbf{W}}}_{in}\end{bmatrix} = \{\boldsymbol{\Gamma}_i\}\cdot\{\boldsymbol{S}_i\}\boldsymbol{v}\boldsymbol{e}^{\mathrm{T}}\mathbf{P}\boldsymbol{b}_i, \qquad i = 1, 2, \ldots, n \quad (87)$$

Since $\boldsymbol{x}_m$ reference signals which must be bounded signals by design, and since $\boldsymbol{K}_i$ and $\mathbf{W}_{ij}$, $i = 1, 2, \ldots, n$, $j = 1, \ldots, n$, must also be bounded and desired exact emulation parameters, the uniform boundedness of $\boldsymbol{e}$ and $\mathbf{X}_{i*}$ imply the uniform boundedness of $\boldsymbol{x}$, $\hat{\boldsymbol{K}}_i$ and $\hat{\mathbf{W}}_{ij}$, which proves the statement in the theorem that $\boldsymbol{x}$ and $\boldsymbol{u}$ [under the assumption that $(\overline{\mathbf{G}} + \hat{\mathbf{D}}_G)^{-1}$ exists] remain uniformly bounded. Then, since $\boldsymbol{H}_i$ and $\mathbf{N}_{ij}$ are bounded basis functions which are uniformly bounded by definition, and since $\boldsymbol{u}$ has already been shown to be bounded, it further follows from the results of Lemma 1 that

$$\lim_{t\to\infty}\|\boldsymbol{e}\| = 0 \quad (88)$$

which in turn implies that

$$\lim_{t\to\infty}[\boldsymbol{x}(t) - \boldsymbol{x}_m(t)] = 0$$

i.e. asymptotic tracking is achieved.