Manuscript Number: EAAI-11-537

Title: Dynamic neural networks with hybrid structures for nonlinear system identification

Article Type: Contributed Paper

Keywords: recurrent neural networks, system identification, nonlinear systems

Corresponding Author: Dr Jiamei Deng,

Corresponding Author's Institution:

First Author: Jiamei Deng

Order of Authors: Jiamei Deng

Abstract: Dynamic neural networks are often used for nonlinear system
identification. This paper presents a novel series-parallel dynamic neural network structure which is suitable for nonlinear system identification. A theoretical proof is given showing that this type of dynamic neural network is able to approximate finite
trajectories of nonlinear dynamical systems. Also, this neural network is trained to identify a practical nonlinear 3D crane system, which could not be identified by the previous type of neural networks.

Dynamic neural networks with hybrid structures for nonlinear system identification

Jiamei Deng

Department of Aeronautical and Automotive Eng., Loughborough University, Loughborough LE11 3TU, UK

Abstract

Dynamic neural networks are often used for nonlinear system identification. This paper presents a novel series-parallel dynamic neural network structure which is suitable for nonlinear system identification. A theoretical proof is given showing that this type of dynamic neural network is able to approximate finite trajectories of nonlinear dynamical systems. Also, this neural network is trained to identify a practical nonlinear 3D crane system, which could not be identified by the previous type of neural networks.

Keywords: recurrent neural networks, system identification, nonlinear systems

1. Introduction

The introduction of artificial neural network methods for the identification and control of dynamical systems two decades ago has had a sgnificant impact in control systems research [14, 13]. A recurrent neural network is a closed loop system, with feedback paths introducing dynamics into the model. They can be trained to learn the system dynamics without assuming much knowledge about the structure of the system under consideration.

Dynamic neural networks (DNNs) have important properties that make them convenient to be used together with nonlinear control approaches based on state space models and differential geometry [6], such as feedback linearisation. However the mapping capability of DNNs are quite limited due to their fixed structure, that is, the number of layers and the number of hidden units [10] [12]. An example shown in this paper has demonstrate this limitation of DNNs. The development of novel DNN structures, which has good mapping capability, is a relevant challenge being addressed in this paper and previous work [3]. Although the structure is changed minorly only, the mapping capability of the new desined DNN in this paper has been improved dramatically. Deng *at al* [3] presents a new dynamic neural network structure which is suitable for the identification of highly nonlinear systems, which needs the outputs from the real system for training and operation. This paper presents a hybrid dynamic neural network structure which presents a similar idea of serial-parallel hybrid structure, but it uses an output from another neural network for training and operation classified as a series-parallel model [14]. This type of DNNs does not require the output of the plant to be used as an input to the model.This neural network has much better mapping capabilities and is more flexibile in traing complicated systems, compared to the DNNs in [6]. A theoretical proof showing how this hybrid dynamic neural network can approximate finite trajectories of general nonlinear dynamic systems is given. To illustrate the capabilities of the new structure, neural networks are trained to identify a real nonlinear 3D crane system.

The paper is organized as follows. Section 2 discusses the universal approximation property of static multilayer perceptrons. Section 3 introduces the class of dynamic neural networks of interest in this paper. Section 4 discusses theoretical results on the approximation ability of dynamic neural networks. Section 5 presents an example. Finally, Section 6 gives concluding remarks.

2. Different Types of Dynamic Neural Networks

Dynamic neural networks are made of interconnected dynamic neurons, also called units. The class of neuron of interest in this paper is described by the following differential equation:

$$\dot{x}_i = -\beta_i x_i + \sum_{j=1}^{N} \omega_{ij}\sigma(y_j) + \sum_{j=1}^{m} \gamma_{ij}u_j, \tag{1}$$

where β_i, ω_{ij} and γ_{ij} are adjustable weights, with $1/\beta_i$ a positive time constant and x_i the activation state of the ith unit, y_j the actual system output or the hidden state of the jth unit, $\sigma : \mathbb{R} \rightarrow \mathbb{R}$ a sigmoid function and $u_1, \ldots, u_m$ the input signals.

A dynamic neural network is formed by a single layer of N units. The first n units are taken as the output of the network, leaving $N - n$ units as hidden neurons. A type 1 DNN is defined by the following vectorised expression:

$$\begin{aligned} \dot{x} &= -\beta x + \omega\sigma(y) + \gamma u \\ y_n &= C_n x \qquad\qquad\quad , \end{aligned} \tag{2}$$

where x are coordinates on $\mathbb{R}^N$, $\beta \in \mathbb{R}^{N\times N}$ is a diagonal matrix with diagonal elements $\{\beta_1, \ldots, \beta_N\}$, $\omega \in \mathbb{R}^{N\times N}$, $\gamma \in \mathbb{R}^{N\times m}$ are weight matrices, $\sigma(x) = [\sigma(x_1), \ldots, \sigma(x_N)]^T$ is a vector sigmoid function, $u \in \mathbb{R}^m$ is the input vector, $y_n \in \mathbb{R}^n$ is the plant output vector, $y = [{y_n}^T, x_{n+1}, ..., x_N]^T$, $C_n = [I_{n\times N},\ 0_{n\times(N-n)}]$.

A type 1 DNN differs from the dynamic neural network described in Chapter 4 of the book [6], which in this paper is known as type 2 DNN, in the argument of the vector sigmoid function $\sigma(\cdot)$. A type 2 DNN is described by the following vectorised expression:

$$\begin{aligned} \dot{x} &= -\beta x + \omega\sigma(x) + \gamma u \\ y_n &= C_n x \qquad\qquad , \end{aligned} \tag{3}$$

Define the output state vector $x^p = [x_1^p, ..., x_n^p]^T = y_n$ as the internal state of the n output units. Define the hidden state vector $x^h = [x_1^h, ..., x_{N-n}^h]^T$ as the internal state of the $N - n$ hidden units. A type 1 DNN uses plant output and the hidden state in the argument of the vector sigmoid function $\sigma(\cdot)$, while a type 2 DNN uses the whole state vector of the network, which consists of the output states and the hidden states, in the argument of the vector sigmoid function. The difference is illustrated in Figure 1 and in Figure 2.

A type 3 DNN is defined by the following vectorised expression:

$$\begin{aligned} \dot{x} &= -\beta x + \omega\sigma(\hat{y}) + \gamma u \\ y_n &= C_n x \qquad\qquad , \end{aligned} \tag{4}$$

where x are coordinates on $\mathbb{R}^N$, $\beta \in \mathbb{R}^{N\times N}$ is a diagonal matrix with diagonal elements $\{\beta_1, \ldots, \beta_N\}$, $\omega \in \mathbb{R}^{N\times N}$, $\gamma \in \mathbb{R}^{N\times m}$ are weight matrices, $\sigma(x) = [\sigma(x_1), \ldots, \sigma(x_N)]^T$ is a vector sigmoid function, $u \in \mathbb{R}^m$ is the input vector, $x^e \in \mathbb{R}^n$ is the estimated output vector of another neural network, $\hat{y} = [x^{eT}, x_{n+1}, ..., x_N]^T$, $C_n = [I_{n\times N},\ 0_{n\times(N-n)}]$.

A type 3 DNN differs from a type 1 DNN, in the argument of the vector sigmoid function $\sigma(\cdot)$. A type 3 DNN is different from a type 1 DNN in that a type 3 DNN uses the outputs from another neural network and the hidden states in the argument of the vector sigmoid function $\sigma(\cdot)$, while a type 1 DNN uses the plant outputs and the hidden states of the network, which consists of the output states and the hidden states, in the argument of the vector sigmoid function. The type 3 DNN is illustrated in Figure 3

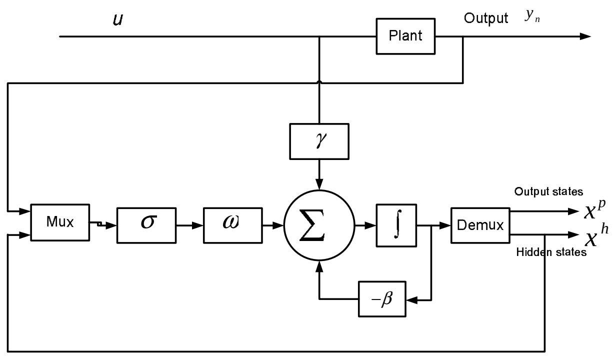

Figure 1: Block diagram of type 1 DNN

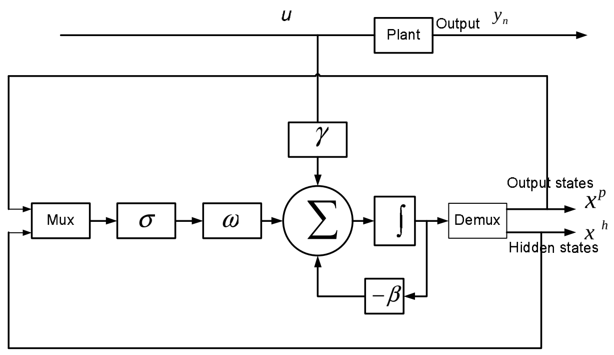

Figure 2: Block diagram of type 2 DNN

3. The Universal Approximation Property of Static Multilayer Networks

An important result of approximation theory states that a three-layer feedforward neural network with sigmoidal activation functions in the hidden layer and linear activation functions in the output layer, has the ability to approximate any continuous mapping $f : \mathbb{R}^n \to \mathbb{R}^q$ to arbitrary precision, provided that the number of units in the hidden layer is sufficiently large. This is stated by the following theorem, which is the theory basis of replacing the real outputs of the type 1 DNN presented in [3] with the outputs from another neural network in this new hybrid DNN, shown in Figure 3.

Theorem 1. *Let K be a compact set of $\mathbb{R}^n$ and $f : K \to \mathbb{R}^q$ be a continuous mapping. Then, for arbitrary $\epsilon > 0$ and the usual topology $\mathbb{R}^q$ induced by the netric m there exists a k-layer ($k \geq 3$) network of the input-output $\hat{f}$, such that $\max_{x \in K} ||f(x) - \hat{f}(x)|| \leq \epsilon$.*

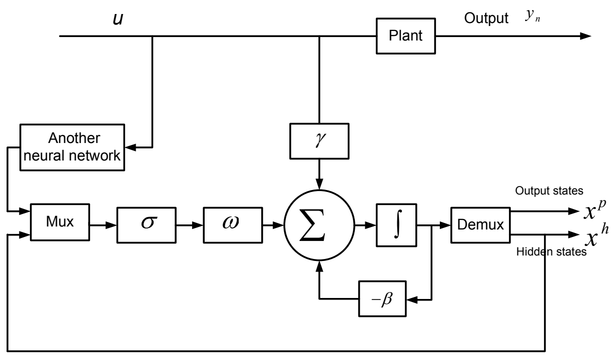

Figure 3: Block diagram of type 3 DNN

The following theorem is a version of the fundamental approximation theorem provided by Funahashi [4]. Similar results have been obtained by Cybenko [1] and others.

Theorem 2. *Let K be a compact set of $\mathbb{R}^n$ and $f : K \longrightarrow \mathbb{R}^q$ be a continuous mapping. Then, for arbitrary $\epsilon > 0$, there exists an integer N_h, a $q \times N_h$ matrix W_2, an $N_h \times n$ matrix W_1, and an N_h dimensional vector b such that:*

$$\max_{x \in K} ||f(x) - W_2 \sigma(W_1 x + b)|| \le \epsilon, \tag{5}$$

where $\sigma : \mathbb{R}^{N_h} \longrightarrow \mathbb{R}^{N_h}$ is a sigmoid mapping whose elements are defined as follows:

$$\sigma(z) = \begin{bmatrix} \sigma(z_1) \\ \vdots \\ \sigma(z_{N_h}), \end{bmatrix} \tag{6}$$

where $z = [z_1, \ldots, z_{N_h}]^T \in \mathbb{R}^{N_h}$.

For the proof of the above theorem, see [4].

4. Approximation Ability of Type 1 and Type 3 Dynamic Neural Networks

This section describes how any finite time trajectory of a given finite-dimensional non–autonomous dynamic system $\dot{x}(t) = f(x(t), u(t))$ can be approximated by type 1 and type 3 DNNs. The theory uses the fundamental approximation theorem of neural networks and shows that, under certain conditions, there exists a dynamic neural network with a sufficient number of hidden units such that the approximation error is bounded to a desired level. This theory is inspired by previous work on the approximation of finite trajectories of autonomous nonlinear systems [5, 9]. The book [6] presents a theorem that shows that a type 2 DNN can approximate general nonlinear systems.

Corollary 1 (Garces *et al.*, 2003). *Let K and U be compact subsets of $\mathbb{R}^n$ and $\mathbb{R}^m$, respectively, and $f : K \times U \to \mathbb{R}^n$ be a continuous mapping. Then, for arbitrary $\epsilon > 0$, there exists an integer N_h, an $n \times N_h$ matrix W_2, an $N_h \times n$ matrix W_1, an $N_h \times m$ matrix γ_1, and an N_h dimensional vector b such that:*

$$\max_{x \in K, u \in U} ||f(x,u) - W_2 \sigma(W_1 x + \gamma_1 u + b)|| \leq \epsilon, \tag{7}$$

where $\sigma : \mathbb{R}^{N_h} \to \mathbb{R}^{N_h}$ is a sigmoid mapping whose elements are defined as follows:

$$\sigma(z) = \begin{bmatrix} \sigma(z_1) \\ \vdots \\ \sigma(z_{N_h}), \end{bmatrix} \tag{8}$$

where $z = [z_1, \ldots, z_{N_h}]^T \in \mathbb{R}^{N_h}$.

Proof. The proof follows directly from Theorem 2, by making the following substitutions: $K \leftarrow K \times U$, $q \leftarrow (n+m)$, $x \leftarrow [x^T \, u^T]^T$, $W_1 \leftarrow [W_1 \, \gamma_1]$.

Theorem 3. *Let D be an open subset of $\mathbb{R}^n$, and U an open subset of $\mathbb{R}^m$. Let $f : D \times U \to \mathbb{R}^n$ be a C^1-mapping, $u : [0,T] \to U$ be a C^1 function, $\tilde{K}$ be a compact subset of D. Suppose that there exists a set $K \subset \tilde{K}$ so that any solution $x(t)$ with initial value $x(0) \in K$ of the non-autonomous system*

$$\dot{x}(t) = f(x(t), u(t)) \tag{9}$$

is defined on $I = [0,T]$ $(0 < T < \infty)$ for $u(t) \in U$ with $t \in I$, and is included in $\tilde{K}$ for any $t \in I$. Then, for an arbitrary $\varepsilon > 0$, there exists a non-autonomous dynamic neural network with n output units with states $x^o \in \mathbb{R}^n$ and N_h hidden units with states $x^h \in \mathbb{R}^{N_h}$, of the form:

$$\dot{z} = -\beta z + \omega \sigma(z) + \gamma \bar{u}, \tag{10}$$

where $z = [x^{oT} \, x^{h^T}]^T \in \mathbb{R}^{n+N_h}$, $\bar{u} = [u^T \, \dot{u}^T]^T \in \mathbb{R}^{2m}$, $\beta \in \mathbb{R}^{n+N_h \times n+N_h}$ is a diagonal matrix, $\omega \in \mathbb{R}^{n+N_h \times n+N_h}$ and $\gamma \in \mathbb{R}^{n+N_h \times 2m}$ are weight matrices, such *that for a solution $x(t)$ satisfying Equation (9), and an appropriate initial state, the states of the output units of the network, $x^o(t)$, approximate the solution of the non-autonomous system:*

$$\max_{t\in I} ||x(t) - x^o(t)|| < \varepsilon; I = [0,T] \quad (0 < T < \infty). \tag{11}$$

Proof. See the book [6].

Theorem 4. *Let D be an open subset of $\mathbb{R}^n$, and U and open subset of $\mathbb{R}^m$. Let $f : D \times U \to \mathbb{R}^n$ be a C^1-mapping, $u : [0,T] \to U$ be a C^1 function, $\tilde{K}$ be a compact subset of D. Suppose that there exists a set $K \subset \tilde{K}$ so that any solution $x(t)$ with initial value $x(0) \in K$ of the non-autonomous system*

$$\dot{x}(t) = f(x(t), u(t)), \tag{12}$$

is defined on $I = [0,T]$ $(0 < T < \infty)$ for $u(t) \in U$ with $t \in I$, and is included in $\tilde{K}$ for any $t \in I$. Then, for an arbitrary $\varepsilon_1 > 0$, there exists a non-autonomous dynamic neural network with n output units with states $x^p \in \mathbb{R}^n$ and N_h hidden units with states $x^h \in \mathbb{R}^{N_h}$, of the form:

$$\dot{z} = -\beta z + \omega\sigma(z_1) + \gamma\bar{u}, \tag{13}$$

where $z = [x^{pT}\ x^{h^T}]^T \in \mathbb{R}^{n+N_h}$, $z_1 = [x^T\ x^{h^T}]^T \in \mathbb{R}^{n+N_h}$, $\bar{u} = [u^T\ \dot{u}^T]^T \in \mathbb{R}^{2m}$, $\beta \in \mathbb{R}^{n+N_h \times n+N_h}$ is a diagonal matrix, $\omega \in \mathbb{R}^{n+N_h \times n+N_h}$ and $\gamma \in \mathbb{R}^{n+N_h \times 2m}$ are weight matrices, such that for a solution $x(t)$ satisfying Equation (12), and an appropriate initial state, the states of the output units of the network, $x^p(t)$, approximate the solution of the non-autonomous system:

$$\max_{t\in I} ||x(t) - x^p(t)|| < \varepsilon_1; I = [0,T] \quad (0 < T < \infty). \tag{14}$$

Proof. This proof uses Lemmas 1, 2 and 3, which are given in the appendix. For given $\varepsilon_1 > 0$, choose $\varepsilon > 0$, $\varepsilon_2 > 0$ and such that $\varepsilon+\varepsilon_2 \leq \varepsilon_1$, $\varepsilon_2 \leq \frac{\eta_1 l_G}{exp(l_G T-1)}$. Define now the mapping $F : \mathbb{R}^{n+N_h} \times \mathbb{R}^{2m} \to \mathbb{R}^{n+N_h}$ as follows:

$$F(z, \bar{\bar{u}}) = -\beta z + \omega\sigma(z) + \bar{\gamma}\bar{\bar{u}}. \tag{15}$$

Then the dynamic system defined by F is:

$$\dot{z} = -\beta z + \omega\sigma(z) + \bar{\gamma}\bar{\bar{u}}, \tag{16}$$

where $\bar{\bar{u}} = [\bar{u}\ \delta z]^T$, $\delta z = [\delta x\ 0_{N_h \times N_h}]^T$, $\delta x = [x - x^p]$, $\bar{\gamma} = [\gamma\ 0_{(n+N_h)\times(n+N_h)}]$. Equation (16) is equivalent to Equation (10).

Define a new mapping $\tilde{F} : \mathbb{R}^{n+N_h} \times \mathbb{R}^{2m} \to \mathbb{R}^{n+N_h}$ as follows:

$$\tilde{F}(\tilde{z}, \bar{\bar{u}}) = -\beta \tilde{z} + \omega\sigma(\tilde{z} + [0_{n\times n}\ I_{(n+N_h)\times(n+N_h)}]\bar{\bar{u}}) + \bar{\gamma}\bar{\bar{u}}. \tag{17}$$

Then the dynamic system defined by $\tilde{F}$ is:

$$\dot{\tilde{z}} = -\beta \tilde{z} + \omega\sigma(\tilde{z} + [0_{n\times n}\ I_{(n+N_h)\times(n+N_h)}]\bar{\bar{u}}) + \bar{\gamma}\bar{\bar{u}}. \tag{18}$$

Equation (18) is equivalent to Equation (13). Let l_G is the Lipschitz constant of F in z. It is not difficult to infer that $\tilde{F}$ is also Lipschitz, so that Lemma 2 is applicable to F and $\tilde{F}$.

Note that

$$||F(\tilde{z}, \bar{\bar{u}}) - \tilde{F}(\tilde{z}, \bar{\bar{u}})|| = ||\omega|| \cdot ||\sigma(z) - \sigma(z + \delta z)|| \tag{19}$$

Suppose that x_i is an element of z and that δx_i is an element of δz. Sigmoid function is a continuous and differentiable function. By using Taylor expansion to this sigmoid function:

$$\begin{gathered} ||\sigma(x^o{}_i) - \sigma(x^o{}_i + \delta x_i)|| = \\ || - \sigma'(x^o{}_i)\delta x_i - \tfrac{1}{2}\sigma''(x_o{}^i)\delta x_i{}^2 \\ - \cdots - O(\delta x_i{}^n)||, \end{gathered} \tag{20}$$

where

$$O(\delta x_i{}^n) = \int_z^{z+\delta z} f^{(n+1)}(t)\frac{(z-t)^n}{n!}dt, \tag{21}$$

by using Lemma 3

$$O(\delta x_i{}^n) = \sigma^{(n+1)}(\zeta)\frac{(z-\zeta)^n}{n!}\delta z, \tag{22}$$

for $\zeta \in [z,\ z + \delta z]$, therefore,

$$O(\delta x_i{}^n) \le \sigma^{(n+1)}(\zeta)\frac{\delta z^{n+1}}{n!}, \tag{23}$$

According to Equation (23), Equation (20) becomes

$$||\sigma(x^o{}_i) - \sigma(x^o{}_i + \delta x_i)|| \le \delta x_i d \le \varepsilon d, \tag{24}$$

where $d = || - \sigma'(x^o{}_i) - \frac{1}{2}\sigma''(x_o{}^i)\delta x_i - \cdots - \sigma^{(n+1)}(\zeta)\frac{\delta z^n}{n!}||$ is bounded. In conclusion, Equation (24) can be written as:

$$||\sigma(x^o{}_i) - \sigma(x^o{}_i + \delta x_i)|| \leq \varepsilon d, \tag{25}$$

According to Equation (25), Equation (19) can be written as:

$$||F(\tilde{z}, \bar{\bar{u}}) - \tilde{F}(\tilde{z}, \bar{\bar{u}})|| \leq ||\omega|| d\varepsilon, \tag{26}$$

Equation (26) can be written as:

$$||F(\tilde{z}, \bar{\bar{u}}) - \tilde{F}(\tilde{z}, \bar{\bar{u}})|| \leq \eta_1, \tag{27}$$

by using Lemma 2

$$||x^o(t) - x^p(t)|| \leq \frac{\eta_1}{l_G}(exp(l_G t) - 1), \tag{28}$$

$$\max_{t \in I} ||x^o(t) - x^p(t)|| < \varepsilon_2, \tag{29}$$

$$\begin{aligned} \max_{t \in I} ||x(t) - x^p(t)|| &\leq \\ \max_{t \in I}(||x(t) - x^o(t)|| + ||x^o(t) - x^p(t)||) &\leq \\ \max_{t \in I} ||x(t) - x^o(t)|| + \max_{t \in I} ||x^o(t) - x^p(t)|| &\leq \\ (\varepsilon_2 + \varepsilon) \leq \ \varepsilon_1, \end{aligned} \tag{30}$$

which completes the proof.

Theorem 5. *Let D be an open subset of $\mathbb{R}^n$, and U an open subset of $\mathbb{R}^m$. Let $f : D \times U \to \mathbb{R}^n$ be a C^1-mapping, $u : [0, T] \to U$ be a C^1 function, $\tilde{K}$ be a compact subset of D. Suppose that there exists a set $K \subset \tilde{K}$ so that any solution $x(t)$ with initial value $x(0) \in K$ of the non-autonomous system*

$$\dot{x}(t) = f(x(t), u(t)), \tag{31}$$

is defined on $I = [0, T]$ $(0 < T < \infty)$ for $u(t) \in U$ with $t \in I$, and is included in $\tilde{K}$ for any $t \in I$. Then, for an arbitrary $\varepsilon_3 > 0$, there exists a non-autonomous dynamic neural network with n output units with states $x^p \in \mathbb{R}^n$ and N_h hidden units with states $x^h \in \mathbb{R}^{N_h}$, of the form:

$$\dot{z} = -\beta z + \omega\sigma(z_2) + \gamma\bar{u}, \tag{32}$$

where o is the output vector from another neural network,$z = [x^{pT}\ x^{h^T}]^T \in \mathbb{R}^{n+N_h}$, $z_2 = [x^{eT}\ x^{h^T}]^T \in \mathbb{R}^{n+N_h}$, $\bar{u} = [u^T\ \dot{u}^T]^T \in \mathbb{R}^{2m}$, $\beta \in \mathbb{R}^{n+N_h \times n+N_h}$ *is a diagonal matrix,* $\omega \in \mathbb{R}^{n+N_h \times n+N_h}$ *and* $\gamma \in \mathbb{R}^{n+N_h \times 2m}$ *are weight matrices, such that for a solution* $x(t)$ *satisfying Equation (12), and an appropriate initial state, the states of the output units of the network,* $x^e(t)$*, approximate the solution of the non-autonomous system:*

$$\max_{t \in I} ||x(t) - x^e(t)|| < \varepsilon_3; I = [0, T] \quad (0 < T < \infty). \tag{33}$$

Proof. This proof is similar to Theorem 4 and uses Lemmas 1, 2 and 3, which are given in the appendix.

For given $\varepsilon_3 > 0$, choose $\varepsilon > 0$, $\varepsilon_4 > 0$ and such that $\varepsilon + \varepsilon_4 \leq \varepsilon_3$, $\varepsilon_4 \leq \frac{\eta_1 l_G}{exp(l_G T - 1)}$. Define now the mapping $F : \mathbb{R}^{n+N_h} \times \mathbb{R}^{2m} \rightarrow \mathbb{R}^{n+N_h}$ as follows:

$$F(z, \bar{\bar{u}}) = -\beta z + \omega\sigma(z) + \bar{\gamma}\bar{\bar{u}}. \tag{34}$$

Then the dynamic system defined by F is:

$$\dot{z} = -\beta z + \omega\sigma(z) + \bar{\gamma}\bar{\bar{u}}, \tag{35}$$

where $\bar{\bar{u}} = [\bar{u}\ \delta z]^T$, $\delta z = [\delta x\ 0_{N_h \times N_h}]^T$, $\delta x = [x - x^e]$, $\bar{\gamma} = [\gamma\ 0_{(n+N_h)\times(n+N_h)}]$. Equation (35) is equivalent to Equation (10).

Define a new mapping $\tilde{F} : \mathbb{R}^{n+N_h} \times \mathbb{R}^{2m} \rightarrow \mathbb{R}^{n+N_h}$ as follows:

$$\tilde{F}(\tilde{z}, \bar{\bar{u}}) = -\beta\tilde{z} + \omega\sigma(\tilde{z} + [0_{n\times n}\ I_{(n+N_h)\times(n+N_h)}]\bar{\bar{u}}) + \bar{\gamma}\bar{\bar{u}}. \tag{36}$$

Then the dynamic system defined by $\tilde{F}$ is:

$$\dot{\tilde{z}} = -\beta\tilde{z} + \omega\sigma(\tilde{z} + [0_{n\times n}\ I_{(n+N_h)\times(n+N_h)}]\bar{\bar{u}}) + \bar{\gamma}\bar{\bar{u}}. \tag{37}$$

Equation (37) is equivalent to Equation (32). Let l_G is the Lipschitz constant of F in z. It is not difficult to infer that $\tilde{F}$ is also Lipschitz, so that Lemma 2 is applicable to F and $\tilde{F}$.

Note that

$$||F(\tilde{z}, \bar{\bar{u}}) - \tilde{F}(\tilde{z}, \bar{\bar{u}})|| = ||\omega|| \cdot ||\sigma(z) - \sigma(z + \delta z)|| \tag{38}$$

Using the similar method as in 4, the following equation could be obtained:

$$||F(\tilde{z}, \bar{u}) - \tilde{F}(\tilde{z}, \bar{u})|| \leq \eta_1, \tag{39}$$

by using Lemma 2

$$||x^o(t) - x^e(t)|| \leq \frac{\eta_1}{l_G}(exp(l_G t) - 1), \tag{40}$$

$$\max_{t \in I} ||x^o(t) - x^e(t)|| < \varepsilon_2, \tag{41}$$

$$\begin{aligned}
\max_{t \in I} ||x(t) - x^e(t)|| &\leq \\
\max_{t \in I}(||x(t) - x^o(t)|| + ||x^o(t) - x^e(t)||) &\leq \\
\max_{t \in I} ||x(t) - x^o(t)|| + \max_{t \in I} ||x^o(t) - x^e(t)|| &\leq \\
(\varepsilon_4 + \varepsilon) \leq \ \varepsilon_3. &
\end{aligned} \tag{42}$$

which completes the proof. ■

5. NLARX and its training procedure

The NLARX structure can take into account the dynamics of the system by feeding previous network outputs back into the input layer. It also enables the user to define how many previous output and input time steps are required for representing the systems dynamics best. In this paper, an NLARX model is applied. It represents a recurrent neural network, which fits the purpose of non-linearity of the problem. A typical structure of a NLARX model is illustrated in Figure 4. The inputs are represented by $u(n)$ and the outputs are described by $y(n)$. The formulation of this NLARX model can be described as:

$$\begin{aligned}
y(n) \ &= \ F(y(n-1), \ldots, y(n-ny), u(n-1), \ldots, \\
&\quad\ u(n-nu), \theta)
\end{aligned} \tag{43}$$

where ny is the number of past output terms used to predict the current output, nu is the number of input terms used to predict the current output.

This neural network model training problem can be cast as a non-linear unconstrained optimization problem:

$$\min_{\theta} F_M(\theta, Z_M) = \frac{1}{2M} \sum_{k=1}^{M} \|y(k) - \hat{y}(k|\theta)\|^2 \tag{44}$$

where $Z_M = [y(k), u(k)]_{k=1,\ldots,M}$ is a training data set, $y(k)$ represents the measured output, $\hat{y}(k|\theta)$ is the predicted output from the neural network NLARX output, $|| \cdot ||^2$ is 2-norm operation, and θ is a parameter vector.

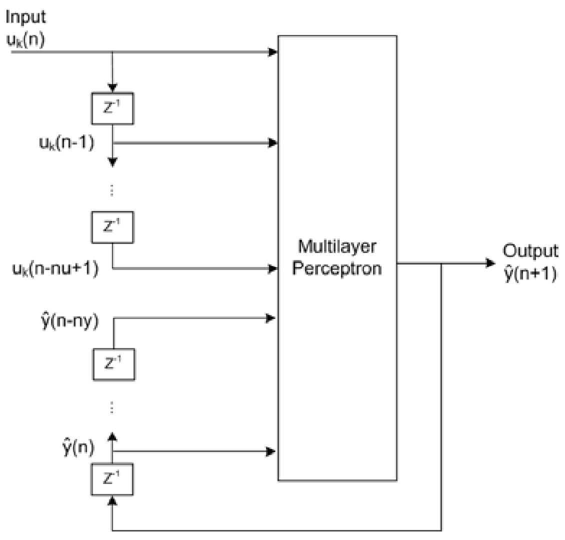

Figure 4: Structure of an NLARX

The optimization problem minimizes the averaged distance between the predicted outputs and the measured output of training samples. Predicted output of an NLARX model is a function of regressors which are transformations of past inputs and past outputs. Usually this function has a linear block and a nonlinear block. The predicted output of the model is the sum of the outputs of the two blocks. Typical regressors are simply delayed input or output variables. More advanced regressors are in the form of arbitrary user-defined functions of delayed input and output variables.

The NLARX training process is as follows. Given a neural network described by Equation (43), there is an error metric that is referred to as performance index of Equation (44), which is to be minimized. This index is a representation of the approximation of the network to some given training patterns. The task will be to modify the network parameters θ to reduce the index $F_M(\theta, Z_M)$ over the complete trajectory to achieve the minimal value. In this paper the neural networks are trained using gradient descent algorithms while the initial value of θ is perturbed several times in order to avoid the local minimal solution. The gradient descent methods will calculate the vector $\nabla_\theta F_M$ whose elements are $\frac{\partial F_M}{\partial \theta_i}(i = 1, \cdots, i, \cdots, p)$. The training algorithm will find the parameters of the network for which the performance index has reached a desirable value. Given a vectorising trajectory for the network output and training patterns, the performance index is the Euclidean norm of the error matrix of the whole training batch for the output.

Fig 5 shows the training (the top figure) and validation (the bottom figure) trajectories for x_c of the crane. Fig 6 shows the training (the top figure) and validation trajectories (the bottom figure) for y_c of the crane. Fig 7 shows the training (the top figure)and validation (the bottom figure) for y_c of the crane. Fig 7 shows the training (the top figure) and validation (the bottom figure) for z_c of the crane. This NLARX is used to train a type 3 DNN. NLARX is used in a recursive way, which uses the past mode outputs.

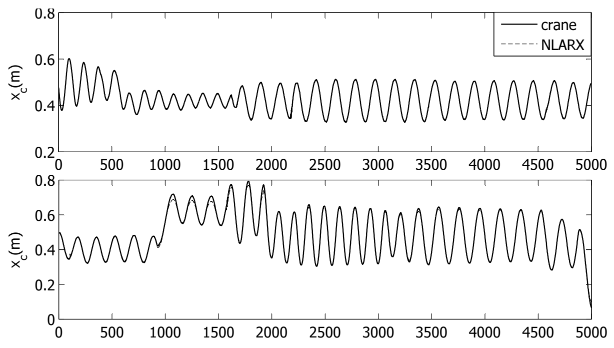

Figure 5: Training and validation trajectories of NLARX for x_c

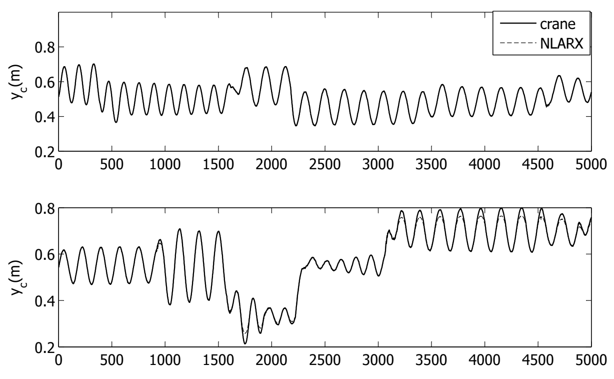

Figure 6: Training and validation trajectories of NLARX for y_c

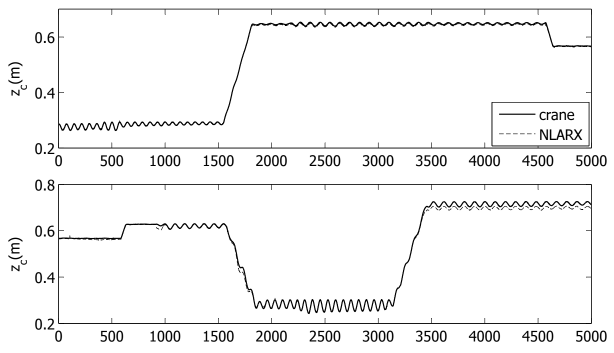

Figure 7: Training and validation trajectories of NLARX for z_c

6. Example

The 3D crane consists of a payload hanging on a pendulum-like lift-line wound by a motor mounted on a cart (Figure 8). The 3D crane system is multivariable, it exhibits highly nonlinear dynamics, and has oscillatory behaviour with different time scales, which makes it a challenging benchmark for nonlinear identification, particularly with recurrent model structures. The payload is lifted and lowered in the z direction. Both the rail and the cart are capable of horizontal motion in the x direction. The cart is capable of horizontal motion along the rail in the y direction. Therefore the payload attached to the end of the lift-line can move freely in 3 dimensions. The 3D crane is driven by the three DC motors and is fully interfaced to MATLAB and SIMULINK. The crane has three manipulated inputs, which are the references to PWM circuits that drive three DC motors, and five measurements obtained via optical encoders.

The schematic diagram of the 3D crane is given in Figure 9.

There are five measured quantities:

- x_w (not shown in Figure 9) denotes the distance of the rail with the cart from the centre of the construction frame;
- y_w (not shown in Figure 9) denotes the distance of the cart from the centre of the rail;
- R denotes the length of the lift-line;
- α denotes the angle between the y axis and the lift-line;

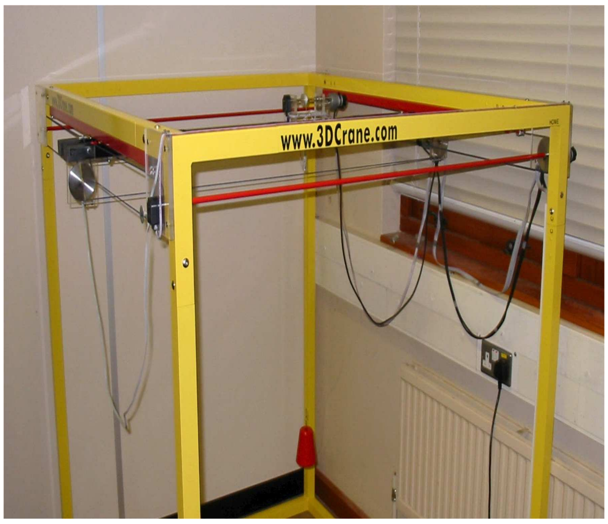

Figure 8: The 3D crane system setup.

- β denotes the angle between the negative direction on the z axis and the projection of the lift-line onto the xz plane.

The position in cartesian co-ordinates of the payload is denoted by x_c, y_c, z_c and can be found from the five measurements using kinematic equations given in [11]. A dynamic model of this crane is shown in [11].

According to Theorem 1, the dynamics of this crane could always be identified by a feedforwad neural network with a sufficient number of hidden states. However, due to personal computer's memory limits, a feedforward neural network is not found for the crane. Instead, A NLARX (non-linear autoregressive exogenous input) model is identified for the dynamics of the crane, which could be used to illustrate the idea of the type 3 DNN. The NLARX model could be describe as follows: The output of the NLARX is feed back into the input layer. In addition, the input allows consideration of previous inputs in order to incorporate dynamics within the systems behavior.

Three neural networks of type 1, 2 and 3 were used to identify three-input three-output models, which had as inputs the three reference voltages to the PWM circuits and as outputs the three co-ordinates of the payload position. Training was performed using a genetic algorithm with real enconding [2]. In these three cases, a 6-state dynamic neural network structure was chosen. Figure 10 shows the training output and the model output using the type 1 neural network. Figure

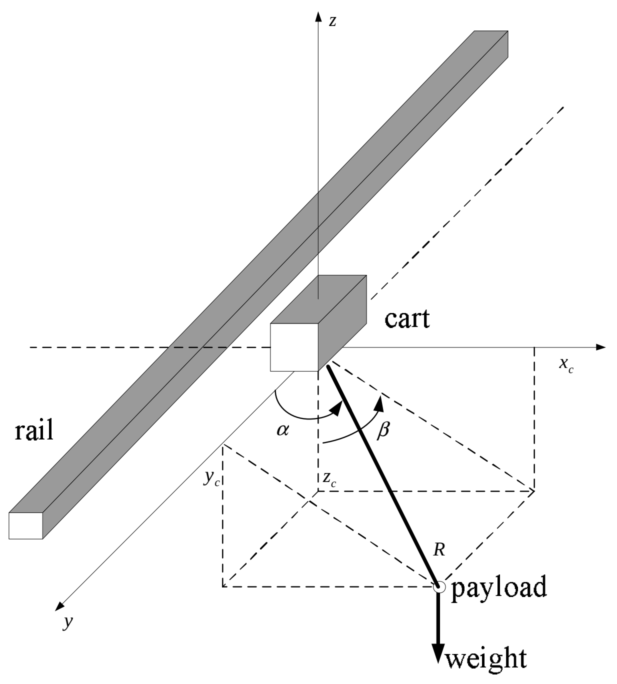

Figure 9: 3D crane system: coordinates and forces.

11 shows the validation output and model output for the same case. Figure 12 shows the training output and the model output for type 2 dynamic neural network. Figure 13 shows the validation data and model output for the same case. For both Figure 13 and Figure 12, there is no improvement on the results with the increase of iterations. Figure 14 shows the training output and the model output for type 3 dynamic neural network. Figure 15 shows the validation data and model output for the same case.

It is not difficult to see that a type 2 DNN had problems to approximate the dynamic behaviour of the system, whereas the type 1 and 3 DNN, which was easier to train, was able to approximate the system more accurately. The better approximation capability exhibited by the type 1 and 3 DNN can be attributed to the fact that this structure uses both output and input information, as it is a series-parallel model. But type 3 DNN does not require the plant output to operate.

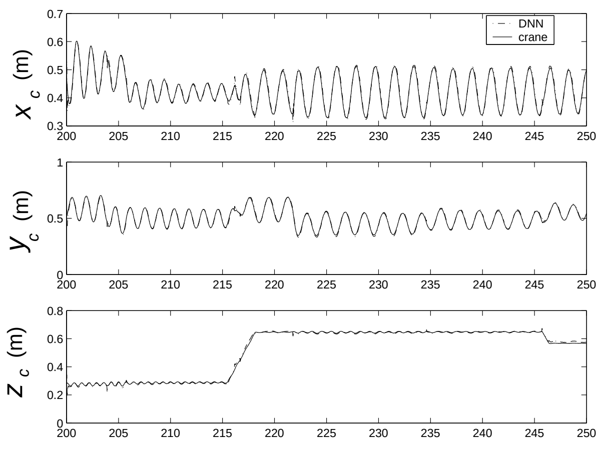

Figure 10: Training trajectories and model outputs using the type 1 DNN

7. Conclusions

This paper presented a novel hybrid dynamic neural network structure and it has been proved that the network has the ability to approximate finite trajectories of non-autonomous nonlinear dynamic systems and provide flexibility which does not depend on the outputs of the plant for operation. An example has been given to demonstrate the effectivity of the proposed structure in approximating complex nonlinear dynamics, and its performance has been favourably compared, in terms of training difficulty and approximation ability, with a previously proposed dynamic neural network structure.

8. Acknowledgement

This work has been supported by the UK EPSRC under Grant Reference Number GR/R64193, and the University of Reading Chinese Studentship Programme.

The following Lemmas are useful for the proof of Theorem 4.

Lemma 1 (Gronwall's inequality). *Let $v : [t_0, t_f] \to \mathbb{R}$ be continuous and nonnegative. Suppose that $C \geq 0$ and $L \geq 0$ are real numbers such that*

$$v(t) \leq C + \int_{t_0}^{t} Lv(\tau)d\tau \tag{.1}$$

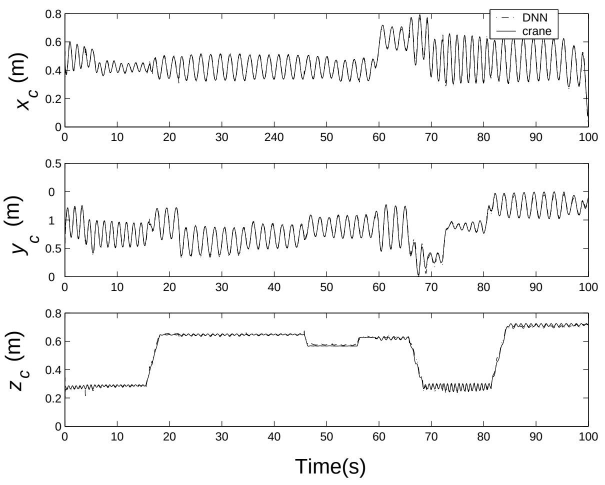

Figure 11: Validation trajectories and model outputs using the type 1 DNN

for all $t \in [t_0, t_f]$. *Then*

$$v(t) \leq C \exp(L|t - t_0|) \tag{.2}$$

for all $t \in [t_0, t_f]$

Proof. See Chapter 8 of [7].

Lemma 2. *Let* F, $\tilde{F} : S \times U \to \mathbb{R}^n$ *be Lipschitz continuous mappings and* L *be a Lipschitz constant of* $F(x, u)$ *in* x *on* $S \times U$. *Suppose that for all* $x \in S$ *and* $u \in U$*:*

$$||F(x, u) - \tilde{F}(x, u)|| < \varepsilon \tag{.3}$$

If $x(t)$ *and* $\tilde{x}(t)$*, are solutions to*

$$\begin{aligned} \dot{x} &= F(x, u) \\ \dot{\tilde{x}} &= \tilde{F}(\tilde{x}, u) \end{aligned} \tag{.4}$$

respectively, on some interval $I = \{t \in \mathbb{R} | t_0 \leq t \leq t_f\}$*, and* $x(t_0) = \tilde{x}(t_0)$*, then*

$$||x(t) - \tilde{x}(t)|| \leq \frac{\varepsilon}{L} \left(\exp(L|t - t_0|) - 1\right) \tag{.5}$$

holds for all $t \in I$.

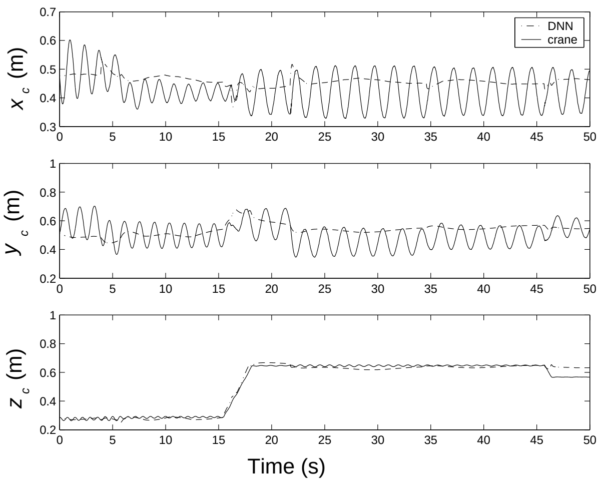

Figure 12: Training trajectories and model outputs using the type 2 DNN

Proof. Please see Chapter 15 of [7].

Lemma 3. *Let $f(x)$ be an integrable function in the interval (a, b). A point c can be found between a and b such that*

$$\int_a^b f(x)dx = f(c)(a-b) \tag{.6}$$

Proof. See Chapter XIII of [8].

References

[1] G. Cybenko, Approximation by superpositions of sigmoidal functions, Mathematics of Control, Signals and Systems 2 (1989) 303–314.

[2] J. Deng, V.M. Becerra, Modelling of a 3d crane system using parallel dynamic neural networks, in: Proceeding 2nd IEEE System, Man and Cybernetics UK and Republic of Ireland Chapter Conference: Cybernetic Intelligence, Chanllenges and Advances, Reading, UK, pp. 148–153.

[3] J. Deng, V.M. Becerra, S. Nasuto, The dynamic neural network of a hybrid structure for nonlinear system identification, in: 16th IFAC World Congress, Prague.

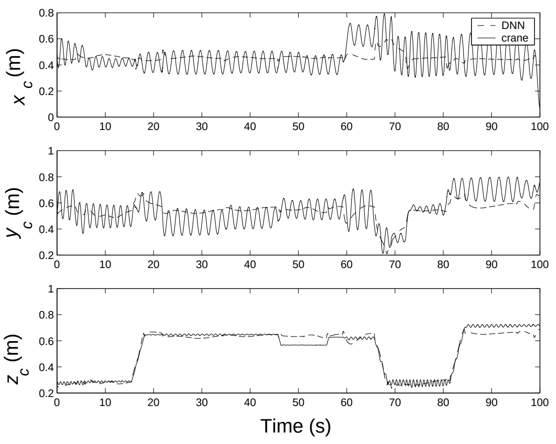

Figure 13: Validation trajectories and model outputs using the type 2 DNN

[4] K. Funahashi, On the approximate realization of continuous-mappings by neural networks, Neural Networks 2 (1989) 183–192.

[5] K. Funahashi, Y. Nakamura, Approximation of dynamical-systems by continuous-time recurrent neural networks, Neural Networks 6 (1993) 801–806.

[6] F. Garces, V. Becerra, C. Kambhampati, K. Warwick, Strategies for feedback linearisation: a dynamic neural network approach, Springer, London, 2003.

[7] M.W. Hirsch, S. Smale, Differential equations, dynamical systems and linear algebra, Academic Press, San Diego, CA, 1974.

[8] A. Khinchin, A course of mathematical analysis, Hindustan Publishing CORP., Delhi, 1960.

[9] M. Kimura, R. Nakano, Learning dynamical systems by recurrent neural networks from orbits, Neural Networks 11 (1998) 1589–1599.

[10] R.P. Lippman, An introduction to computing with neural nets, IEEE Acoustics, Speech and Signal Processing Magazine 4 (1987) 4–22.

[11] I. Ltd., 3DCrane Version 1.3 Getting Started, Krakov, Poland: Inteco, 2002.

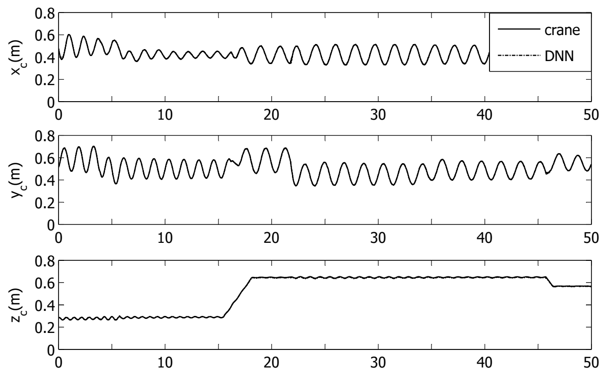

Figure 14: Training trajectories and model outputs using the type 3 DNN

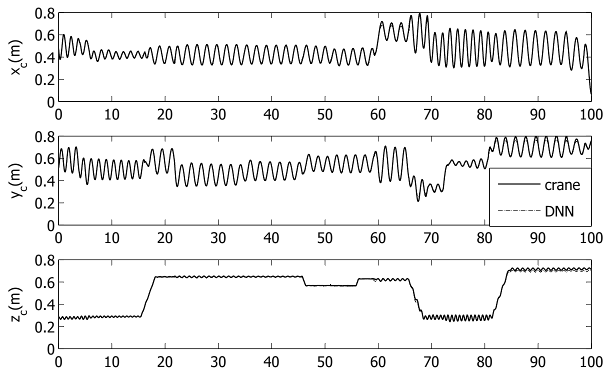

Figure 15: Validation trajectories and model outputs using the type 3 DNN

[12] T.M. Mabhan, A.Y. Zomaya, Toward generating neural network structures for function approximation, Neural Networks 7 (1994) 89–99.

[13] W.T. Miller, R. Sutton, P.J. Werbos, Neural networks for control, MIT Press, Cambridge, 1990.

[14] K.S. Narendra, K. Parthasarathy, Application and control of dynamical systems using neural networks, IEEE Transactions on Neural Networks 1 (1990) 4–27.

List of Figures