Available online at www.sciencedirect.com
SCIENCE DIRECT®
Journal of Sound and Vibration 275 (2004) 693–718
JOURNAL OF SOUND AND VIBRATION
www.elsevier.com/locate/jsvi

Analysis and modification of Volterra/Wiener neural networks for the adaptive identification of non-linear hysteretic dynamic systems

J.-S. Pei[a], A.W. Smyth[b,*], E.B. Kosmatopoulos[c]

[a] *School of Civil Engineering & Environmental Science, University of Oklahoma, Norman, OK 73019-1024, USA*
[b] *School of Engineering & Applied Science, Columbia University, New York, NY 10027-6699, USA*
[c] *Department of Production Engineering & Management, Technical University of Crete, Chania, GR-73100, Greece*

Received 23 August 2002; accepted 30 June 2003

Abstract

This study attempts to demystify a powerful neural network approach for modelling non-linear hysteretic systems and in turn to streamline its architecture to achieve better computational efficiency. The recently developed neural network modelling approach, the Volterra/Wiener neural network (VWNN), demonstrated its usefulness in identifying the restoring forces for hysteretic systems in an off-line or even in an adaptive (on-line) mode, however, the mechanism of *how* and *why* it works has not been thoroughly explored especially in terms of a physical interpretation. Artificial neural network are often treated as "black box" modelling tools, in contrast, here the authors carry out a detailed analysis in terms of problem formulation and network architecture to explore the inner workings of this neural network. Based on the understanding of the dynamics of hysteretic systems, some simplifications and modifications are made to the original VWNN in predicting accelerations of hysteretic systems under arbitrary force excitations. Through further examination of the algorithm related to the VWNN applications, the efficiency of the previously published approach is improved by reducing the number of the hidden nodes without affecting the modelling accuracy of the network. One training example is presented to illustrate the application of the VWNN; and another is provided to demonstrate that the VWNN is able to yield a unique set of weights when the values of the controlling design parameters are fixed. The practical issue of how to choose the values of these important parameters is discussed to aid engineering applications.
© 2003 Elsevier Ltd. All rights reserved.

*Corresponding author. Fax: +1-212-854-6267.
E-mail address: smyth@civil.columbia.edu (A.W. Smyth).

0022-460X/$ - see front matter © 2003 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jsv.2003.06.005

1. Introduction and motivation

The modelling of non-linear physical phenomena is of great significance in many fields of engineering. In applied mechanics, the modelling of the dynamic response of hysteretic systems has a wide range of applications. Relevant problems include the behavior of building structures under earthquake excitation as well as many vibration problems with mechanical systems especially the joints of aerospace structures under working loads. In this paper, a study will be presented to predict accelerations of hysteretic systems with discrete degrees-of-freedom under known force excitations.

An artificial neural network (ANN) modelling approach will be adopted in this study. In contrast with the way ANNs are often used as a powerful "black box" modelling tool whose inner workings are not well understood, here the authors will explore in detail, fundamental network application issues (such as initial network design and non-uniqueness of solutions) by relating them to the fundamentals of mechanics as well as the mathematical foundation of neural networks.

This study builds on a recently published neural network approach [1]. In that study the dynamic response of non-linear hysteretic dynamic systems was modelled using a newly developed ANN system identification approach which used measurement data of the force excitation and the acceleration response at discrete locations. The Volterra/Wiener neural network (VWNN) as it was called, has demonstrated its power in modelling hysteretic systems in an off-line or even adaptive (on-line) mode.

The VWNN consists of a dynamic linear module in series with a static neural network module, where the former is a linear filter interconnected in series and the latter is a linear-in-the-weights neural network. An illustration of this architecture is shown in Fig. 1. Symbolically, the dynamic module can be represented by $\boldsymbol{\xi} = \mathbf{H}(s)\boldsymbol{\zeta}$, where $\boldsymbol{\zeta}$ is the input vector to the VWNN, $\boldsymbol{\xi}$ is the output of the linear module and $\mathbf{H}(s)$ is a stable transfer function matrix (here, s denotes the Laplace operator). The linear-in-the-weights neural network is described as $\boldsymbol{\eta} = \mathbf{W}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{\xi})$, where $\boldsymbol{\eta}$ is the output of the neural network, $\mathbf{W}$ denotes the matrix of the synaptic weights of the neural network, $\boldsymbol{\xi}$ is the output of the linear filter, and $\boldsymbol{\phi}$ is a vector of the non-linear activation functions of the neural network.

The adaptive capability of the VWNN modelling technique distinguishes the method from typical slow training often associated with neural networks. The feature of on-line prediction of the system response also is of great significance in the field of system identification as well as robust adaptive control. Unfortunately, the mechanism of *why* and *how* the VWNN works was not thoroughly explained especially in terms of a physical interpretation, and therefore the VWNN was basically treated as a "black box" in that study. This has prompted the authors to explore the reasons which make the VWNN work, especially those which can be categorized as physical explanations. As a byproduct of understanding the VWNN, the authors find that the computational efficiency of the published approach can also be improved by eliminating some

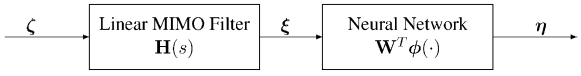

Fig. 1. Block diagram of the VWNN.

high order terms in the hidden layer of the VWNN without affecting the performance of the network. To aid engineering applications, a detailed analysis of the architecture and design parameters of the VWNN will be presented together with some training examples.

2. Demystifying Volterra/Wiener neural networks

2.1. Problem formulation

One of the main goals of this study is to predict the accelerations for hysteretic systems, and thus simulate the system response to known (or measured) force excitations. Such a predictive property is also an essential element for adaptive control applications. For this study it is assumed that no a priori knowledge is available on the dynamics of the system under consideration. Based on limited input–output measurements of the system, identification is carried out to model the system. Based on this model, the future outputs of the system can then be predicted for any given future input which is within the same range of the inputs used in the original modelling stage. The input–output pairs to be considered in this study are excitation forces and response accelerations.

In general, for a multi-degree-of-freedom (MDOF) system subjected only to force excitation, the equations of motion can be written as

$$\mathbf{M}_{11}\ddot{\mathbf{x}}_1(t) + \mathbf{r}(t) = \mathbf{f}_1(t), \tag{1}$$

where $\mathbf{M}_{11}$ has the same meaning as the more typical $\mathbf{M}$ notation of the mass matrix. The vectors $\ddot{\mathbf{x}}_1(t)$ and $\mathbf{f}_1(t)$ represent acceleration and excitation force, respectively; and they are the original input and output measurements available for the identification. The first subscript 1 denotes the active degrees of freedom, and a 0 subscript would refer to a support motion (this notation follows that used in Ref. [1]). Specifically, one has

$$\mathbf{x}_1(t) = [x_{11}(t), \ldots, x_{1n_1}(t)]^{\mathrm{T}}, \quad \dot{\mathbf{x}}_1(t) = [\dot{x}_{11}(t), \ldots, \dot{x}_{1n_1}(t)]^{\mathrm{T}}, \quad \ddot{\mathbf{x}}_1(t) = [\ddot{x}_{11}(t), \ldots, \ddot{x}_{1n_1}(t)]^{\mathrm{T}}$$

and

$$\mathbf{f}_1(t) = [f_{11}(t), \ldots, f_{1n_f}(t)]^{\mathrm{T}},$$

where n_1 denotes the number of active degrees of freedom, n_f the number of active degrees of freedom under force excitations and $n_f \leqslant n_1$. Displacements and velocities, $\mathbf{x}_1(t)$ and $\dot{\mathbf{x}}_1(t)$, do not appear in Eq. (1) because they are not directly available from the measurements. $\mathbf{r}(t)$ denotes the restoring force that is considered to be a function of many unknown quantities in an unknown format. Modelling this restoring force for hysteretic systems is the focus of this study.

Eq. (1) holds true at any time instance. The task of predicting the acceleration at the next time step given the acceleration at the current time step can be expressed in a generic format: to solve $\ddot{\mathbf{x}}_1(t_{n+1})$ given $\ddot{\mathbf{x}}_1(t_n)$ and the time history of $\mathbf{f}_1(t)$, where $n = 1, \ldots, N$ and N is the total number of the given input–output pairs minus 1.

Applying Eq. (1) at the time instance t_{n+1}, pre-multiplying the new equation by $\mathbf{M}_{11}^{-1}$ and rearranging it results in the following:

$$\ddot{\mathbf{x}}_1(t_{n+1}) = -\mathbf{M}_{11}^{-1}\mathbf{r}(t_{n+1}) + \mathbf{M}_{11}^{-1}\mathbf{f}_1(t_{n+1}). \tag{2}$$

It can be seen that the estimation of the restoring force at time instance t_{n+1} is the key in predicting the acceleration at t_{n+1}, assuming that the time history of the excitation force, $\mathbf{f}_1(t)$ is available and $\mathbf{M}_{11}$ are just coefficients to be identified.

Prior work on predicting restoring forces for hysteretic systems by the authors and others has noted that the idea of mapping restoring force in terms of displacement and velocity *alone* is often found to be insufficient to model hysteretic systems. A representation which better reflects, in particular, the hysteretic phenomenon for restoring forces is given in a non-linear differential equation of the form [2–4]

$$\dot{\mathbf{r}}(t) = \mathbf{Q}(\mathbf{r}(t), \mathbf{x}_1(t), \dot{\mathbf{x}}_1(t)), \tag{3}$$

where $\mathbf{Q}$ is an unknown non-linear function involving matrix operations. Note that this formula is not exhaustive; more terms could be added into the variable list to capture more complicated non-linearities (Ref. [3] for example). However, this general formula has been applied in various hysteretic systems and has provided a good modelling framework. Therefore, it is chosen as the theoretical foundation in this study to guide the identification architecture and detailed arrangement.

We apply Eq. (3) to the time instance t_n and write its left-hand side using the one-step forward difference $\dot{\mathbf{r}}(t_n) = (\mathbf{r}(t_{n+1}) - \mathbf{r}(t_n))/(t_{n+1} - t_n)$. Furthermore, it is assumed that the sampling time $t_{n+1} - t_n$ can be taken as a constant for a given data set or in the real sampling. After some rearranging and substitutions (see Ref. [5]), one has

$$\ddot{\mathbf{x}}_1(t_{n+1}) = \bar{\bar{\mathbf{Q}}}(\mathbf{x}_1(t_n), \dot{\mathbf{x}}_1(t_n), \ddot{\mathbf{x}}_1(t_n), \mathbf{f}_1(t_n), \mathbf{f}_1(t_{n+1})), \quad n = 1, \ldots, N, \tag{4}$$

where $\bar{\bar{\mathbf{Q}}}$ is another unknown non-linear function involving matrix operations.

The symbolic representation shown in Eq. (4) indicates that the problem can be formulated to approximate the unknown function $\bar{\bar{\mathbf{Q}}}$ by fitting the neural network input–output pairs, $\mathbf{p}(t_n)$ and $\mathbf{g}(t_n)$, which are shown in Block 3 "final input–output used in identification" of Table 1, where $n = 1, \ldots, N$ covers all of the measurement data. In the table, the inputs and outputs are related back to the corresponding quantities in Ref. [1] whenever possible. Note that $\bar{\bar{\mathbf{Q}}}$ is targeted to be a

Table 1
Input–output as defined in different stages before identification, where $n = 1, \ldots, N$ and N is the total number of the given input–output pairs minus 1

1. *Original input–output measurements*	
Input $\ddot{\mathbf{x}}_1(t)$	$[\ddot{x}_{11}(t), \ldots, \ddot{x}_{1n_1}(t)]^{\mathrm{T}}$
Output $\mathbf{f}_1(t)$	$[f_{11}(t), \ldots, f_{1n_1}(t)]^{\mathrm{T}}$
2. *Reorganized input–output measurements* (*i.e.*, *input–output before pre-processing*)	
Input, denoted as $\boldsymbol{\zeta}$ in Ref. [1]	$[\ddot{\mathbf{x}}_1(t_n)^{\mathrm{T}}, \mathbf{f}_1(t_n)^{\mathrm{T}}, \mathbf{f}_1(t_{n+1})^{\mathrm{T}}]^{\mathrm{T}}$
Targeted output, i.e., $\mathbf{g}(t_n)$ or denoted as $\boldsymbol{\eta}$ in Ref. [1]	$\ddot{\mathbf{x}}_1(t_{n+1})$
3. *Final input–output used in identification* (*i.e.*, *input–output after pre-processing*)	
Input, i.e., $\hat{\mathbf{p}}(t_n)$ or denoted as $\boldsymbol{\xi}$ in Ref. [1]	$[\hat{\mathbf{x}}_1(t_n)^{\mathrm{T}}, \hat{\dot{\mathbf{x}}}_1(t_n)^{\mathrm{T}}, \ddot{\mathbf{x}}_1(t_n)^{\mathrm{T}}, \mathbf{f}_1(t_n)^{\mathrm{T}}, \mathbf{f}_1(t_{n+1})^{\mathrm{T}}]^{\mathrm{T}}$
Targeted output, i.e., $\mathbf{g}(t_n)$ or denoted as $\boldsymbol{\eta}$ in Ref. [1]	$\ddot{\mathbf{x}}_1(t_{n+1})$

single-valued function and it will be true if the use of Eq. (3) is accurate enough for a given hysteretic system.

2.2. Pre-processing and design parameter α

Table 1 summarizes how the original input–output pairs are modified to yield the final input–output pairs which are directly used in the identification algorithm. The inputs and outputs in the table are related to the corresponding quantities in Ref. [1] whenever possible. Such pre-processing is quite common in system identification applications, as long as critical signature information is not lost from any pre-filtering or manipulation.

The first step is to rearrange the original input–output pairs of $\mathbf{f}_1(t_n)$ and $\ddot{\mathbf{x}}_1(t_n)$. Then it can be seen clearly from the formulation of the problem, i.e., Eq. (4), that pre-processing is needed to prepare displacement and velocity, $\mathbf{x}_1(t_n)$ and $\dot{\mathbf{x}}_1(t_n)$, as part of the inputs used in the identification.

Some difficulties involved with the numerical integration of acceleration measurements are discussed in Refs. [5,6]. In a separate case study, it was recognized that there is a strong influence on the final parametrically identified results from the numerical integration scheme used in conjunction with band pass filtering for measurement noise removal [5]. In general therefore, the errors resulting from the numerical integration are expected to have an influence on the identified results. However, the identification approach which will be adopted in this study is a non-parametric rather than a parametric technique. Realizing this difference, the authors reckon that the identified model based on "imperfect" measurements of acceleration may still be used in the prediction of the system response.

In this study, a simple rule of integration is used to estimate displacement and velocity from acceleration measurements. It can be expressed as

$$\hat{\dot{\mathbf{x}}}_1(t_n) = \hat{\dot{\mathbf{x}}}_1(t_{n-1}) + \alpha\ddot{\mathbf{x}}_1(t_n), \quad \hat{\mathbf{x}}_1(t_n) = \hat{\mathbf{x}}_1(t_{n-1}) + \alpha\hat{\dot{\mathbf{x}}}_1(t_n), \tag{5}$$

where the notations $\hat{\dot{\mathbf{x}}}_1(t_n)$ and $\hat{\mathbf{x}}_1(t_n)$ represent the approximated velocity and displacement, respectively. It is assumed that the initial values of these two quantities are zero based on the fact that the system is at rest before the excitation starts, which is assumed to be the situation in all the cases under consideration. The scalar α in the equation is the size of the time step. Note that the same time step sizes are used in integrating both velocity and displacement, which is just for convenience at this stage of study. Issues can be further explored with regard to the details of the integration process and are discussed in Ref. [5].

The role played by this pre-processing stage is similar to that of the dynamic linear module (see Fig. 1) in the VWNN [1], however, there are important differences in details. Many filters can be used in the dynamic module of the VWNN and they are connected to each other in a cascaded sequence. One may think of the first filter as the analogue of the acceleration and excitation force at one step before the current time step, and the mth filter, the analogue of those two quantities at the mth step before the current time step. The number m needs to be determined separately beforehand. This design consideration differs from the idea of forming the current velocity and displacement, although Eq. (3) was also used as the theoretical basis for the VWNN. The authors feel that having more than two filters in the VWNN may not be consistent with Eq. (3). This adjustment from the previous work may be considered as one of the efforts to making the black-box type of identification procedure "translucent".

2.3. Overview of neural estimator

A neural network approach is adopted in this study to approximate the unknown function $\bar{\bar{\mathbf{Q}}}$ in Eq. (4). The mathematical foundation for applying neural networks in function approximation is proven theoretically and independently in Refs. [7,8]. It is said that a finite linear sum of continuous sigmoidal functions (with the limits of 0 and 1 when the variable approaches $-\infty$ and $+\infty$) can approximate any continuous scalar function to any desired degree of accuracy. Application of the theorem to neural networks is illustrated by an example in Fig. 2, where a feedforward network with one hidden layer (i.e., the architecture of "input–hidden–output") is a universal approximator provided that no constraints are placed on the number of nodes and size of the weights (i.e., n_h). All the weights and biases are obtained from training using the data set of the input $\mathbf{p}$ and output $G(\mathbf{p})$ (the approximated output is $Z(\mathbf{p})$). The outputs of the hidden layer can be considered as basis functions which are decided by the weights and biases in Layer 1, while the weights in Layer 2 are the coefficients of the linear combination. Having examined the dynamic module of the original VWNN in Ref. [1], it will be shown that the static module (see Fig. 1) of the VWNN is in fact linearly parameterized. The beauty of it is that the basis functions are formed corresponding to Volterra series expansion: It has been proven through function analysis in Ref. [1] that the linear parameterization of this set of basis functions can form a universal approximator.

The theorem in Refs. [7,8] ties in with the general structure of the so-called "black box" models of non-linear systems based on function analysis [9], where the concepts of functional space and regressors are used to generalize the modelling problems into a parameterized function expansion. Without getting into too many details of function analysis, it may be proper to express the

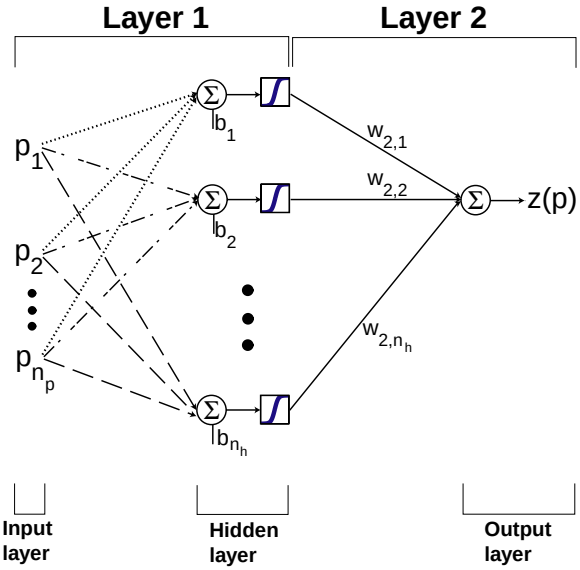

Fig. 2. Neural network architecture with one hidden layer, which is a universal approximator. In this figure, the dotted lines stand for the weighting vector $\mathbf{w}_{1,1}$, the dashed–dotted lines $\mathbf{w}_{1,2}$ and the dashed lines $\mathbf{w}_{1,n_h}$.

idea simply as

$$g(\varphi,\theta)=\sum_{l=1}^{n}\alpha_l h_l(\varphi),\quad \theta=[\alpha_1\ \cdots\ \alpha_n]^{\mathrm{T}}, \tag{6}$$

where the non-linear mapping $g(\varphi,\theta)$ is the non-linear mapping to be identified, the vector φ is the regressors, h_l's are the basis functions and θ is the parameterization. The format shown in Eq. (6) is not unfamiliar to researchers in system identification in the applied mechanics field. For example, the studies presented in Refs. [10,11] demonstrate the usefulness of linearly parametrized Bouc-Wen model [10] in on-line identification for hysteretic systems (in this case, the parameters to be identified can be related to some physical/or graphical meaning). The benefit of having such linearly parametrized estimators is to make it possible to apply recursive on-line linear estimation schemes that lead to fast identification, as indicated in Ref. [10]. There are many more examples but with different basis functions [12,3]. For the purpose of structural health monitoring and damage detection, the determination of parameters with physical or near physical meaning is crucial for real-world applications using model-based system identification approaches. To examine how the VWNN [1] works through some intuitive physical interpretations for a potential wide range of applications in health monitoring and damage detection, the architecture of its static module will be examined and modified based on the framework of this linear parameterization or equivalently, the universal approximator of a feedforward neural network with one hidden layer.

Fig. 3 presents the architectures of the VWNN in this study (left edge) and of that in Ref. [1] (right edge) by relating both of them to each other and to the universal approximator (center). The dynamic module is transformed into the pre-processing stage as discussed in Section 2.2, while the static module is "reorganized" and modified by reflecting more "transparent" features and physical interpretations. Compared with the universal approximator, in short, Layer 1 and the hidden layer of the VWNN are constructed in a totally different manner by introducing first and high order terms. Also, the weights of Layer 1 are pre-fixed and controlled by one design constant λ, while only the weights of Layer 2 are trained using the input–output data stream, thus the latter are the unique results from training. Based on some graphical explanations, it will be explored why a fixed value of λ may be sufficient to capture typical non-linearities through both first and high order terms and what the challenges are in real training. Details of the neural estimator in Fig. 3 are elaborated on here:

Input layer: The input vector, $\hat{\mathbf{p}}(t_n)=[\hat{\mathbf{x}}_1(t_n)^{\mathrm{T}},\hat{\dot{\mathbf{x}}}_1(t_n)^{\mathrm{T}},\ddot{\mathbf{x}}_1(t_n)^{\mathrm{T}},\mathbf{f}_1(t_n)^{\mathrm{T}},\mathbf{f}_1(t_{n+1})^{\mathrm{T}}]^{\mathrm{T}}$ is obtained from pre-processed signals and is denoted as $\boldsymbol{\xi}$ in the original VWNN (also see Table 1). The number of input nodes equals to $3n_1+2n_f$, where n_1 denotes the degrees of freedom of the system and n_f the degrees of freedom under force excitations.

Output layer: The targeted output vector is obtained directly from the original data, $\mathbf{g}(t_n)=\ddot{\mathbf{x}}_1(t_{n+1})$, and is denoted as $\boldsymbol{\eta}$ in the original VWNN (see Table 1). The number of output nodes equals to n_1.

Hidden layer: One of the main features of the VWNN is the inclusion of high order nodes in the hidden layer. Hidden nodes are composed of first and high order terms, where the latter are derived from the first order nodes (see Section 2.4 for the advantage of including high order terms especially for hysteretic systems). The number of first order nodes is exactly the same as that of the

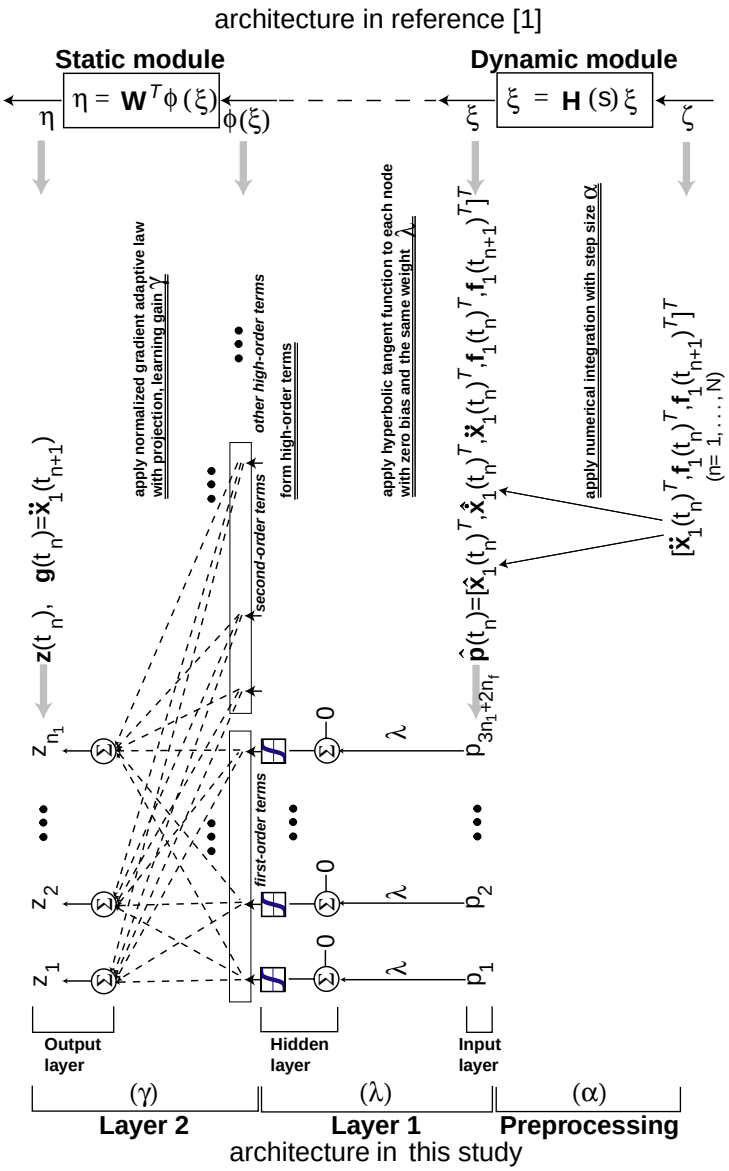

Fig. 3. Explanation of the identification procedure including pre-processing and neural network used in this study and comparison with the VWNN in Ref. [1]. Note that there are significant differences between the pre-processing stage and the dynamic module. In this figure, the dashed lines denote the weighting vector to be identified, $\mathbf{w}_j$, $j = 1, 2, \ldots, n_1$ in Section 2.4.

input nodes, $3n_1 + 2n_f$, because the first order nodes are obtained directly from passing the input nodes through a hyperbolic tangent sigmoidal function separately,

$$h(p_i) = \frac{2}{1 + e^{-2\lambda p_i}} - 1, \quad i = 1, \ldots, 3n_1 + 2n_f, \tag{7}$$

where n_1 denotes the degrees of freedom of the system and n_f the degrees of freedom under force excitations. In Eq. (7), also note that zero bias and a design constant λ as the weight are applied to all the connections; this simplicity and unification is another feature of the VWNN. It can be seen that a vector with all the hidden nodes (all first and high order terms) as its components, is the vector $\phi(\xi)$ in the VWNN.

Layer 1: Layer 1 spans from the input to the hidden layer. Direct connections are only made from the input nodes to the first order nodes. Also, the connections are one to one rather than "full" connection, where "full" connection refers to the condition where each input node is connected to all the hidden nodes and vice versa. Although no visible flow lines are shown between the input nodes and high order nodes, invisible connections do exist and the corresponding "invisible" weights are non-linear based on the way in which the high order nodes are derived from the first order nodes (see Section 2.4).

Layer 2: Layer 2 spans from the hidden to the output layer. The output nodes are fully connected to all the hidden nodes including the first and high order terms; the corresponding weights form a weighting matrix **W**. Based on the architecture of a universal approximator, all the hidden nodes can be considered to be non-linear basis functions and the linear weighting matrix **W**, the coefficients, in approximating the output vector. A normalized gradient adaptive law with projection [13] will be adopted to minimize the normalized output error where the weights **W** are updated using the training data, i.e., the input $\hat{\mathbf{p}}(t_n)$ and targeted output $\mathbf{g}(t_n)$, and n takes from 1 up to N. There are several parameters which control the minimization process, however, the most influential is the adaptive learning gain, γ. Details of the training process will be given in Section 2.5.

With a set of fixed values of the controlling design constants, α (see Section 2.2), λ (see Section 2.4) and γ (see Section 2.5), the trained linear weights **W** in Layer 2 are unique. This overcomes the problem of non-uniqueness. It should be noted that this non-uniqueness is, strictly speaking, conditional and depends on the chosen values of the controlling design constants. If these values vary, the identified weighting matrix **W** will also change accordingly. The choice of proper values for the design constants thus presents a significant challenge. While this could be done through a laborious trial and error approach, some guidance on their proper selection, through the minimization of certain performance indices, will be given in Section 2.5.

2.4. Design parameter λ and high order terms

A common problem for the universal approximator with the architecture of "input–hidden–output" is how to determine the number of hidden nodes. Normally, training can start with several numbers of hidden nodes either based on empirical experience or just an arbitrary guess. When the desired performance is not obtained, more nodes are added into the hidden layer. However, the way in which hidden nodes are added is totally different in the VWNN [1]. In the VWNN, the number of hidden nodes may start with the first order nodes only. When more nodes are added, they will not be added one by one. Instead, they will be inserted in batches including all the nodes belonging to a certain high order.

As defined in Ref. [14], high order terms can be considered as the power terms of the first order terms themselves as well as their cross terms. Several examples are provided in Ref. [5] to show what the corresponding first and high order terms should be. For the formulation considered here, the total number of nodes belonging to the mth order terms is the different combinations of m out

of $3n_1 + 2n_f$ different things, at a time, *with* repetitions, where $3n_1 + 2n_f$ is the dimension of the input vector to the neural network, which is the same as that of the first order hidden nodes. Forming these terms systematically requires an efficient numerical scheme especially for the cases where the degrees of freedom (n_1 or n_f) is high or the required power (m) is high, and this issue will be addressed later. Note that high order terms differ from sigma–pi units as commonly defined [15,16]; comparison has been made in Ref. [5].

Neural networks are considered to be non-parametric identification tools because the weights of both Layers 1 and 2 in the adopted architecture can hardly be related to the physical properties of the system modelled/identified by the neural networks. As presented above, however, the effects of the weights can be visualized based on a linear parameterization: the weights of Layer 1 define the non-linear basis functions and those of Layer 2 are related to the coefficients. In the VWNN, the weights in a not fully connected Layer 1 are unified to λ (see Fig. 3). It is then worth exploring why a unified λ value is sufficient and how the value of λ can influence the profile of the non-linear basis functions.

For the first order nodes, Fig. 4 indicates that a small value of the weight coefficient tends to flatten out the S shape while the large value tends to make a hyperbolic tangent function saturated. The bias defines the shift of the center of the antisymmetrical shape of the hyperbolic tangent function from (0, 0). It can be seen that a linear function can be formed by simply

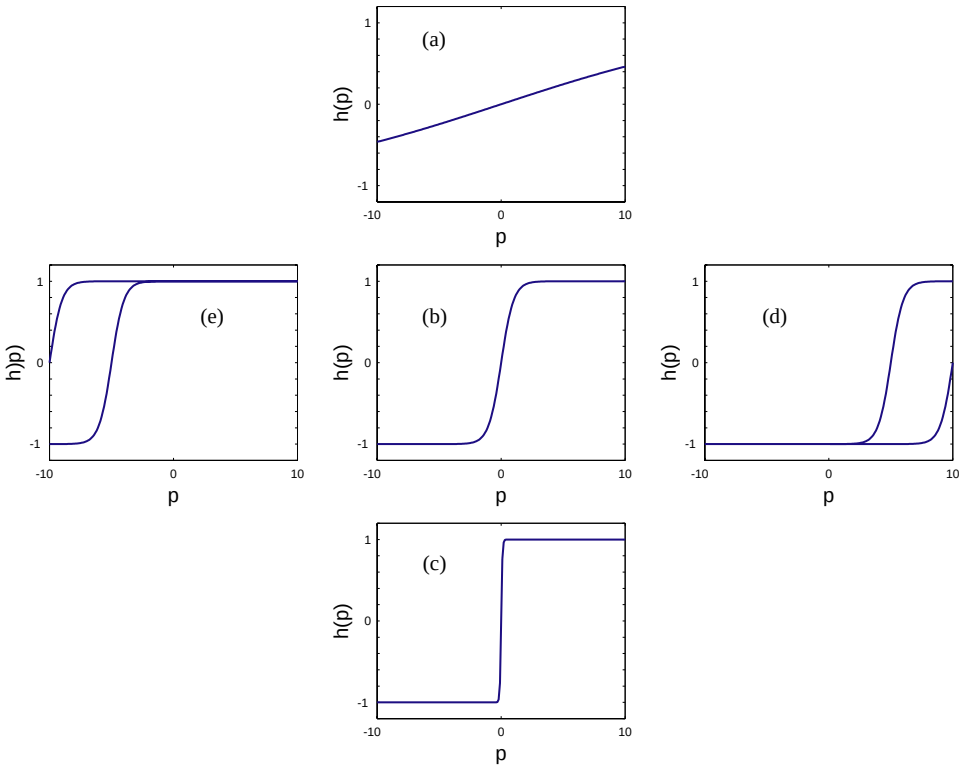

Fig. 4. Effects of varying weights and biases of a hyperbolic tangent sigmoidal function, one-variable case: (a) $w = 0.5,\ b = 0$; (b) $w = 1,\ b = 0$, (c) $w = 10,\ b = 0$; (d) $w = 1,\ b = 5$ and 10; (e) $w = 1,\ b = 5$ and -10.

choosing a very small weight and zero bias, while a sign function can be obtained as the result of a very large weight.

For the high order terms, the linear weights λ in the first order terms will propagate into the high order terms as non-linear weights. Serving as quick examples, Figs. 5 and 6 study the one- or two-input cases with the same input range. Fig. 5 shows the approximated and exact polynomial terms side by side. The comparison indicates how conveniently and efficiently the polynomial basis functions can be formed from the corresponding high order terms. As long as the first term is a rough approximation of a straight line (which can be done by choosing a small value for λ), the resulting high order terms will be good approximations of the polynomial terms. This is because the approximation error in the first order term, which is less than 1, decreases when the power is increased. Here, hyperbolic tangent sigmoidal functions are used to demonstrate polynomial approximation. For the capabilities of logistic sigmoidal functions in approximating polynomials, reference can be made to Ref. [5] for a detailed theoretical study. Since the polynomials play an important role in approximating typical non-linear restoring forces for hysteretic systems [17,3], the capability of approximating polynomials makes it possible for the VWNN to be an efficient approach in handling on-line prediction of accelerations of hysteretic systems.

The effectiveness of the VWNN is however not only limited to approximating polynomial-type non-linearities. It has been shown in Fig. 4 that the hard limiting basis function is readily available when λ is chosen to have a high value. As further indicated in Fig. 6, where three different values of λ and several different powers of high order terms are tried, the feature of dead-space non-linearity as often observed for hysteretic systems with slip can be captured by the hidden nodes of the VWNN. A localized peak or valley can also be mimicked when the power is taken to be an even number. A one-variable case is considered in this figure and the output range of interest is assumed to be $[-1,+1]$. The output range is emphasized here because the numerical errors can jeopardize good identification results when the hidden node outputs are too small.

Figs. 5 and 6 give some clear indications on how the value of λ can affect the types of non-linear basis functions formed by the hidden nodes in the VWNN. Note that in both figures, the input ranges are the same for multi-inputs. This is designed only for convenience in a simplified preliminary trial. When using real data, the situation will be somewhat different. Normally, within the input vector $\hat{\mathbf{p}}(t_n) = [\hat{\mathbf{x}}_1(t_n)^{\mathrm{T}}, \hat{\dot{\mathbf{x}}}_1(t_n)^{\mathrm{T}}, \ddot{\mathbf{x}}_1(t_n)^{\mathrm{T}}, \mathbf{f}_1(t_n)^{\mathrm{T}}, \mathbf{f}_1(t_{n+1})^{\mathrm{T}}]^{\mathrm{T}}$, the magnitude of acceleration, velocity and displacement differ from each other in orders of magnitude. This is due to the different units they adopt and the frequency range of the dynamic response.

Fig. 7 demonstrates the effect of different input ranges on the output of the high order terms when the same value is chosen for λ. The first and second order terms of two different inputs (p_1 and p_2) having different orders of magnitudes, are compared with each other when two different values of λ are adopted. It can be seen that a smaller λ value can make the high order terms of p_1 approximate polynomials well, while the high order terms of p_2 are almost zero. But if the output range is still insensitive to numerical error, the high order terms of p_2 can be considered to be a good approximation of polynomials as shown in the localized view. When a larger λ value is chosen, the high order terms of p_2 can approximate polynomials well. However, those of p_1 lose their previous ability of mimicking polynomials.

Examples in Fig. 7 highlight the difficulties of handling the inputs with different input ranges especially when the small output resulting from small input is enough to cause numerical error. When the smallest output has not caused numerical error, it may be wise to choose a λ value

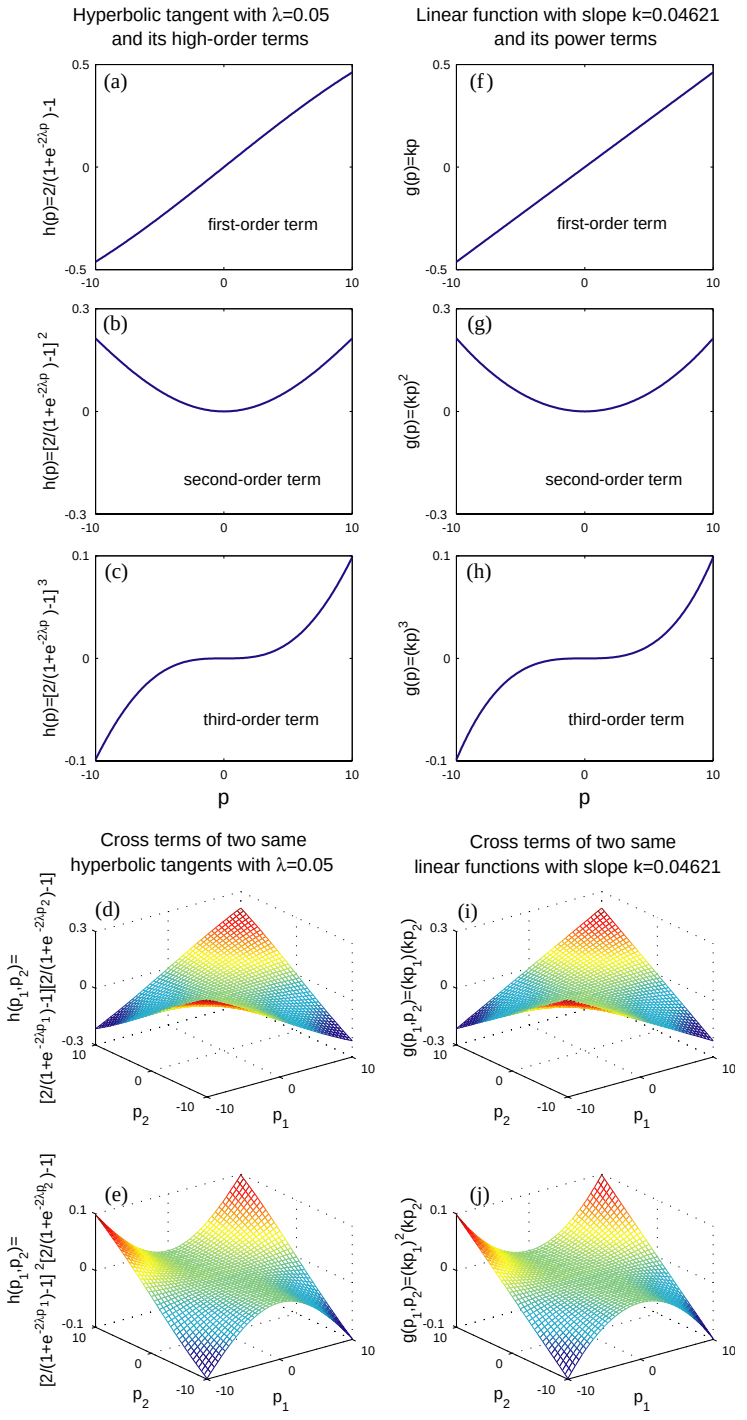

Fig. 5. Comparison of (a)–(e) high order hyperbolic tangents with (f)–(j) polynomial functions when the weights are small and biases are zero, one-input and two-input cases.

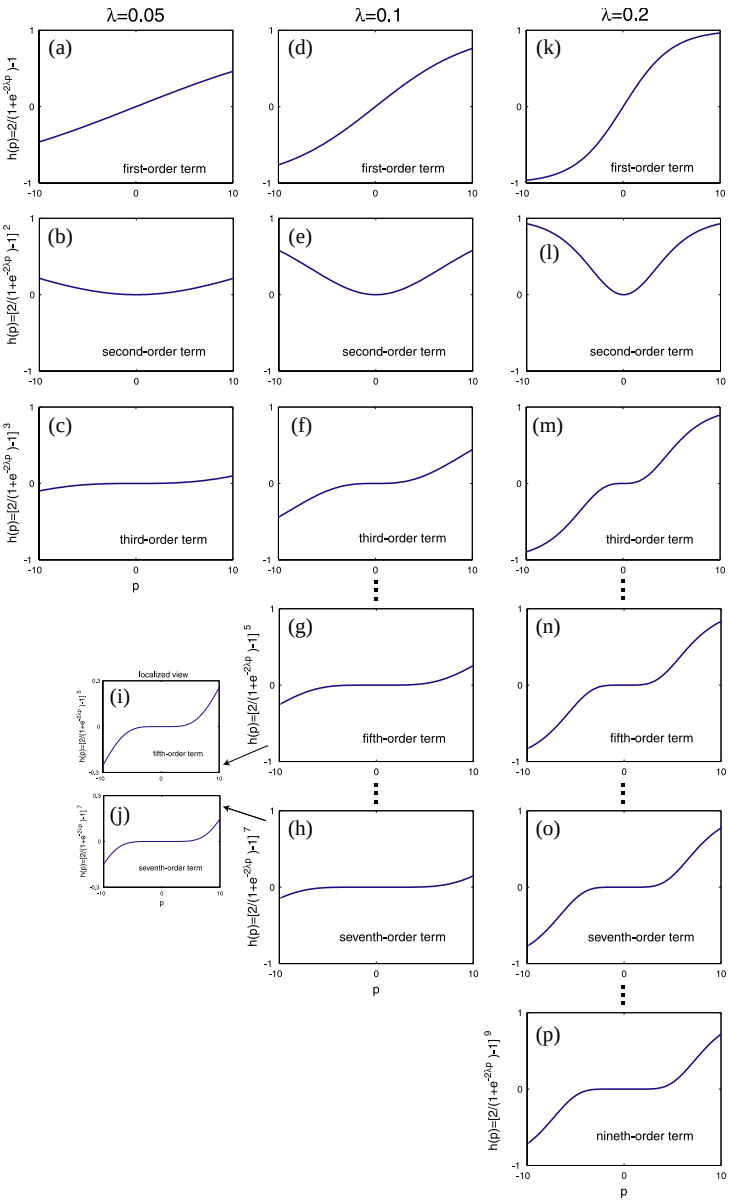

Fig. 6. Effects of varying weights on hyperbolic tangent sigmoidal function and its high order terms, one-input case: (a)–(c) $\lambda = 0.05$, (d)–(j) $\lambda = 0.1$ and (k)–(p) $\lambda = 0.2$. The output range of interest is assumed to be $[-1, +1]$.

mainly to make the input with the largest range behave most like the desired non-linear basis function. This comment is not exhaustive; there are many varying situations to be considered in real applications of the VWNN. It may be worth considering having more than one λ value for different input nodes to deal with cases with different input ranges. Adjusting the α values in the numerical integration in Eq. (5) might be another solution. By choosing proper α values, it might be possible to scale the integrated velocity and displacement into a proper range with respect to

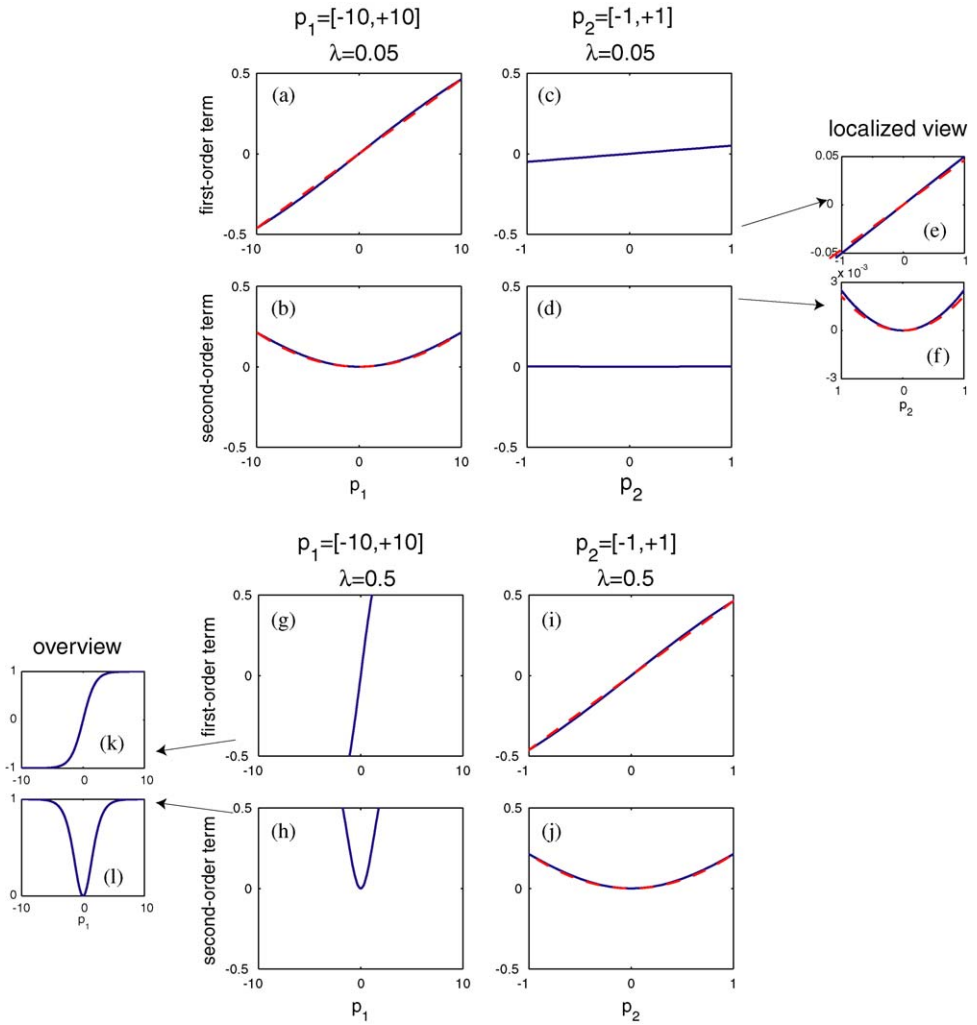

Fig. 7. Effects of different input range, one-input case: input $p_1 = [-10, +10]$ and input $p_2 = [-1, +1]$ when (a)–(f) $\lambda = 0.05$ and (g)–(l) $\lambda = 0.5$. Linear and quadratic terms are considered. The solid lines are for the hyperbolic tangent approximations, while the dashed lines are the exact polynomials. The output range of interest is assumed to be $[-0.5, +0.5]$.

the acceleration, which might in turn make adjusting λ an efficient way of producing the desired non-linear basis functions. This issue is worthy of further study.

In the real implementation of the VWNN, it is important to consider the computational aspects of how to form high order terms. The procedure of constructing a complete list of the second and third order terms computationally without repetition is developed in this study as illustrated by a generic example in Fig. 8, where the analogous lattice structures are illustrated which help programming. This work is considered as a refinement and an improvement on the basic work in Ref. [1]. While having the same identification results, substantial computational time can be saved since a huge number of repetitive hidden nodes are removed.

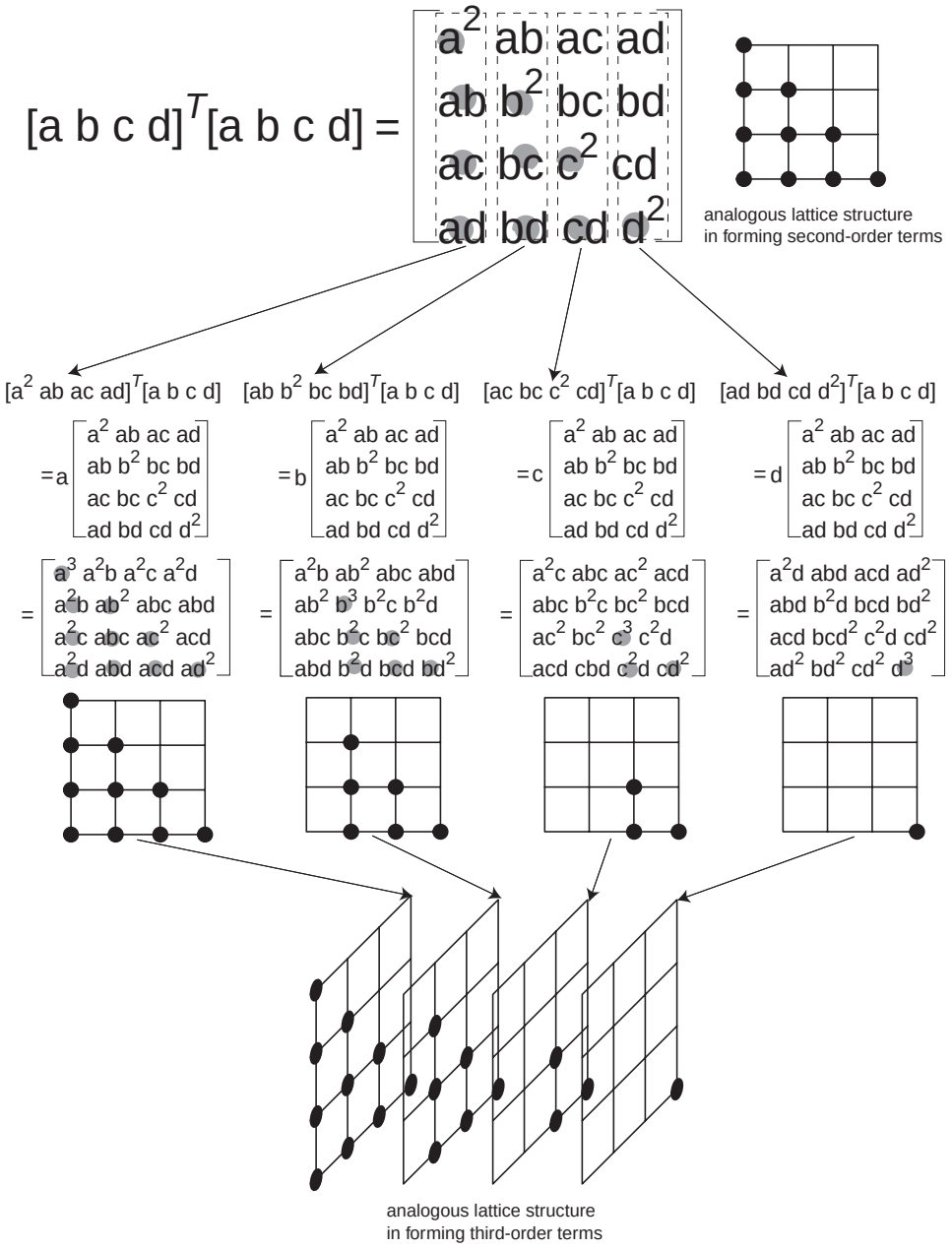

Fig. 8. Explanation of how to form second and third order terms efficiently based on vector products of first order terms. A generic example is shown here with four first order terms, *a*, *b*, *c* and *d*. Grey dots are used to highlight the final second order (by larger dots) and third order (by smaller dots) terms. The analogous 2-D and 3-D lattice structures are also demonstrated to help programming.

2.5. Adaptive law and design parameter γ

A normalized gradient adaptive law with projection [13] is used for the estimation of the weighting matrix $\mathbf{W}$. This adaptive law keeps the trained weights bounded regardless of the boundedness properties of the inputs and outputs. The original formula is designed to update a weighting vector. Based on the architecture of this neural network as shown in Fig. 3, each weighting vector in Layer 2 related to the jth output node, $\mathbf{w}_j$ (where $j = 1, 2, \ldots, n_1$) can be identified independently. With the hidden node output vector denoted as $\boldsymbol{\phi}$, the adaptive law can be expressed as

$$\mathbf{w}_j(t_{n+1}) = \mathbf{w}_j(t_n) + \Delta\mathbf{w}_j(t_n), \tag{8}$$

where

$$\Delta\mathbf{w}_j(t_n) = \begin{cases} \gamma\varepsilon_j(t_n)\boldsymbol{\phi}(t_n), & \text{if } \|\mathbf{w}_j(t_n)\| < M, \text{ or } \|\mathbf{w}_j(t_n)\| = M, \\ & \quad \text{and } (\gamma\varepsilon_j(t_n)\boldsymbol{\phi}(t_n))^{\mathrm{T}}\mathbf{w}_j(t_n) \leqslant 0, \\ \left(\mathbf{I} - \dfrac{\mathbf{w}_j(t_n)\mathbf{w}_j(t_n)^{\mathrm{T}}}{\mathbf{w}_j(t_n)^{\mathrm{T}}\mathbf{w}_j(t_n)}\right)\gamma\varepsilon_j(t_n)\boldsymbol{\phi}(t_n), & \text{otherwise,} \end{cases} \tag{9}$$

where the notation $\|\cdot\|$ represents vector Euclidean norm. The vectors $\mathbf{w}_j(t_n)$ and $\boldsymbol{\phi}_j(t_n)$ are column vectors. The identity matrix $\mathbf{I}$ has square dimensions which are of the same length as vector $\boldsymbol{\phi}$. The scalar $\varepsilon_j(t_n)$ is the jth normalized estimation error at time instance t_n, and it is defined as

$$\varepsilon_j(t_n) = \frac{g_j(t_n) - z_j(t_n)}{1 + \boldsymbol{\phi}(t_n)^{\mathrm{T}}\boldsymbol{\phi}(t_n)}, \tag{10}$$

where the scalar $g_j(t_n)$ is the jth targeted output component, $g_t(t_n) = \ddot{x}_j(t_{n+1})$, and $z_j(t_n)$ is the jth network output, $z_j(t_n) = \hat{\ddot{x}}_j(t_{n+1})$.

Scalars M and γ in Eq. (9) are design constants. The former is the bound in the projection method and has to be chosen to be a large positive value. The latter, γ, is the adaptive gain, which is an important parameter equivalent to the gradient in the gradient method. If this value is chosen to be too small, the convergence of $\mathbf{w}_j$'s will be very slow. But if it is chosen to be too large, the $\mathbf{w}_j$'s values may become unstable. There is a proper range for the value of γ to ensure the stability and fast rate of convergence of the identified values of $\mathbf{w}_j$'s.

The root-mean-square (RMS) output error is chosen to be the performance index. The values of the three design constants, α, λ and γ, need to be predetermined based on minimizing this performance index.

3. Application to a non-linear structure

The methodology presented above is applied to a simulated three-dimensional tripod structural system with nine degrees of freedom. This system is described in detail in Ref. [18]. Only three degrees of freedom are excited with random excitation. Two identification models are tested in this study; one considers high order terms up to second order and another up to third order. Two

situations are considered in each model, one with *training on* and the other with *training off*, where *training off* refers to the case when the final trained weights obtained in *training on* are frozen and used throughout the response for validation purposes. The value of M in Eq. (9) is chosen to be 10^5 throughout the training.

The values of α, λ and γ are determined beforehand. The value of α should be selected relative to (i.e., the same as, or some fraction thereof) the sampling rate used in the simulation (which in this case is 0.005 seconds), while the values of λ should cause no saturation of the maximum values of the acceleration, velocity and displacement, as a conclusion of the discussion of Fig. 7. Since the velocity and displacement data are integrated from the acceleration with small time step size, only the range of the acceleration is considered in choosing a proper λ value. Fig. 9 illustrates the profiles of the output of the first order hidden node of the acceleration when λ takes different values. It can be seen that $\lambda = 0.005$ may be a proper choice. The selection of the γ value can be based on trial and error. By adopting the RMS error as commonly defined, a parametric study is run to study the influence of varying α, λ and γ on the final RMS error. For example, the results of the second order model at the 2000th time step when the training is on are shown in Fig. 10. It can be seen that both λ and γ values have a significant impact on the performance.

When the parameters α, λ and γ take the values of 0.005, 0.005 and 1.2, respectively, the RMS errors at the 2000th time step of the second and third order models are presented in Table 2. Both cases of *training on* and *training off* are considered for each model. The corresponding time history of both the real and estimated acceleration are compared and presented from Figs. 11–14. The user-specified design parameters for this example include a relatively high learning rate, γ, which yields excellent output tracking performance. The cost for this performance is that given a relatively high gain, the model parameters will exhibit overly sensitive adaptation, and hence, will often not converge to stable values. This explains why here the *training off* case does not model the system well, because the fixed model parameters are those chosen from a time instant during a relatively unstable convergence of the model parameters. In contrast, a low learning rate would produce smoother parameter convergence but also poorer output tracking, however, the *training*

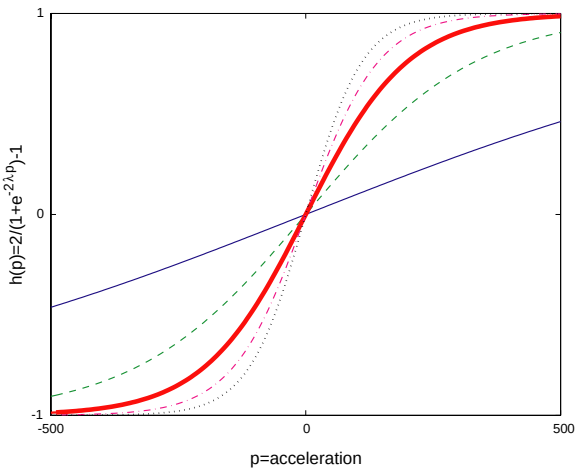

Fig. 9. Influence of varying λ on the profile of the first order hidden node of the acceleration data, where the thin solid, dashed, thick solid, dashed–dotted, dotted lines refer to the cases of $\lambda = 0.001$, 0.003, 0.005, 0.007, 0.009, respectively.

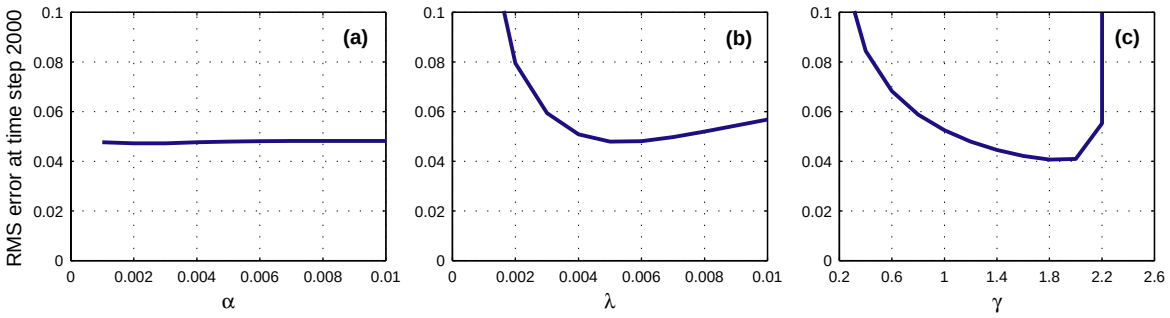

Fig. 10. Influence of varying α, λ and γ on RMS error, second order model with training on: (a) $\lambda = 0.005$, $\gamma = 1.2$, $M = 10^5$; (b) $\alpha = 0.005$, $\gamma = 1.2$, $M = 10^5$; (c) $\alpha = 0.005$, $\lambda = 0.005$, $M = 10^5$.

Table 2
RMS of estimation error, where $\alpha = 0.005$, $\lambda = 0.005$, $M = 10^5$

Training mode	RMS error in 2nd order model 594×9 hidden nodes in total		RMS error in 3rd order model 6578×9 hidden nodes in total	
	$\gamma = 1.2$	$\gamma = 0.3$	$\gamma = 1.2$	$\gamma = 0.3$
Training on	0.048	0.098	0.049	0.086
Training off	0.356	0.263	0.234	0.150

off case is able to model the system better. This can be verified by choosing $\gamma = 0.3$ and the results are shown in Table 2. It can also be observed from these time history plots that the mean of the estimate errors tends to be non-zero for the second order model, while that for the third order model tends to be zero. This indicates that the third order model better captures the system dynamics. The problem dealt with is not the force-state mapping problem but could be related to it. This result is not surprising, given that typical restoring force phenomena exhibits *odd* function behavior. Therefore, *even* modelling terms from the second order powers will not fit the system response well, and will hence produce a biased error. Another advantage of including third order high order terms is a better *training off* estimate, however, the computational time is increased significantly because many more nodes have to be included.

4. Discussion

Three simulation experiments are performed to further explore the convergence nature of the weights. For convenience of presentation, an imaginary SDOF system is adopted with up to second order terms. According to Table 3, there are 20 nodes in the network model. Therefore, there are 20 weights to be trained.

Case 1: Using a wideband random signal with zero mean and standard deviation of one as the excitation force (denoted as $\mathbf{f}_1^{\star}$) and the values of the weighting vector shown as "exact" values in Table 3, the acceleration "measurement" (denoted as $\ddot{\mathbf{x}}_1^{\star}$) can be calculated. The values of the parameters involved are $\alpha = 0.005$ and $\lambda = 0.001$. The excitation force, $\mathbf{f}_1^{\star}$, and acceleration, $\ddot{\mathbf{x}}_1^{\star}$,

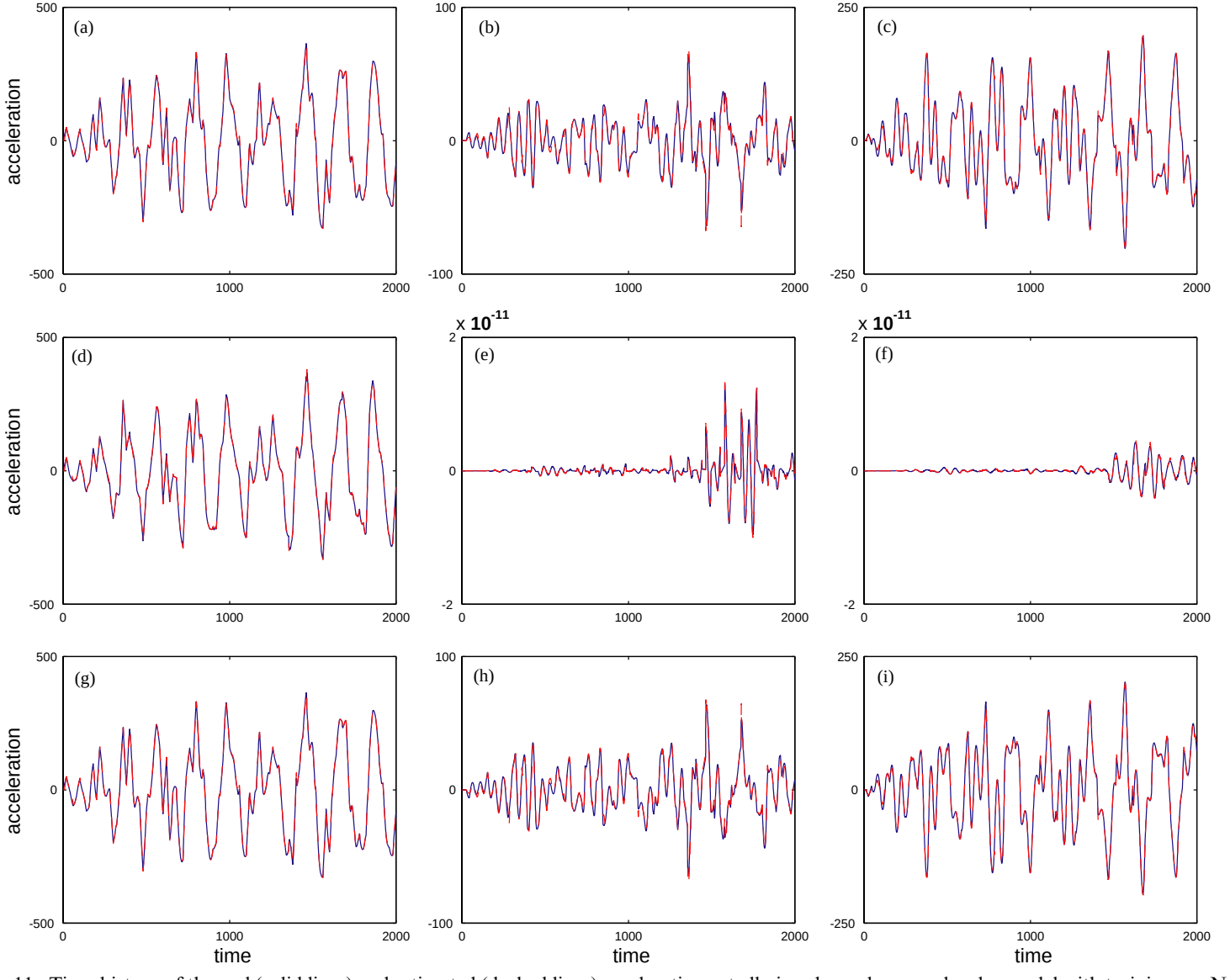

Fig. 11. Time-history of the real (solid lines) and estimated (dashed lines) accelerations at all nine channels, second order model with training on. Note the different output range at each channel. (a)–(i) correspond to channels 1–9.

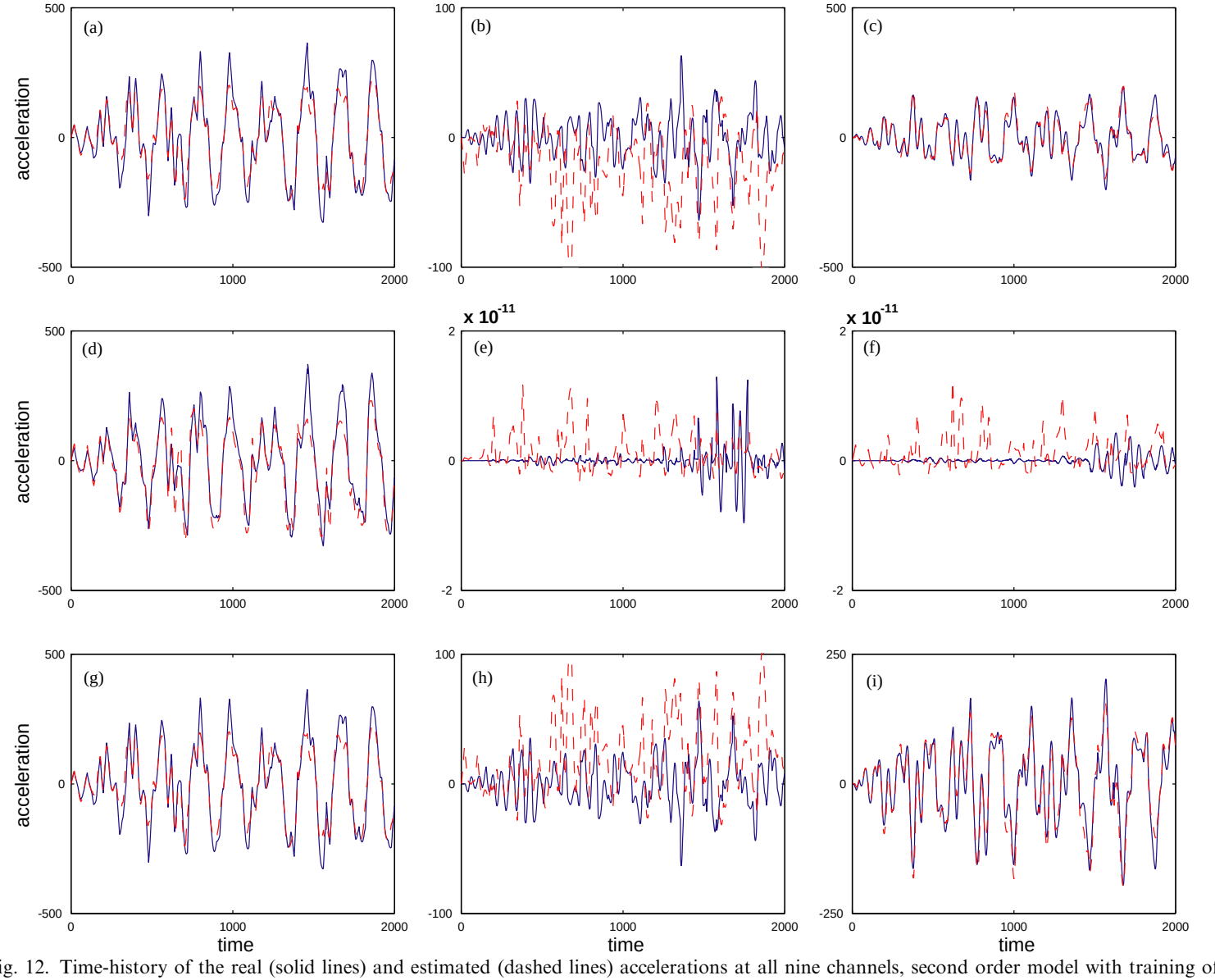

Fig. 12. Time-history of the real (solid lines) and estimated (dashed lines) accelerations at all nine channels, second order model with training off. Note the different output range at each channel. (a)–(i) correspond to channels 1–9.

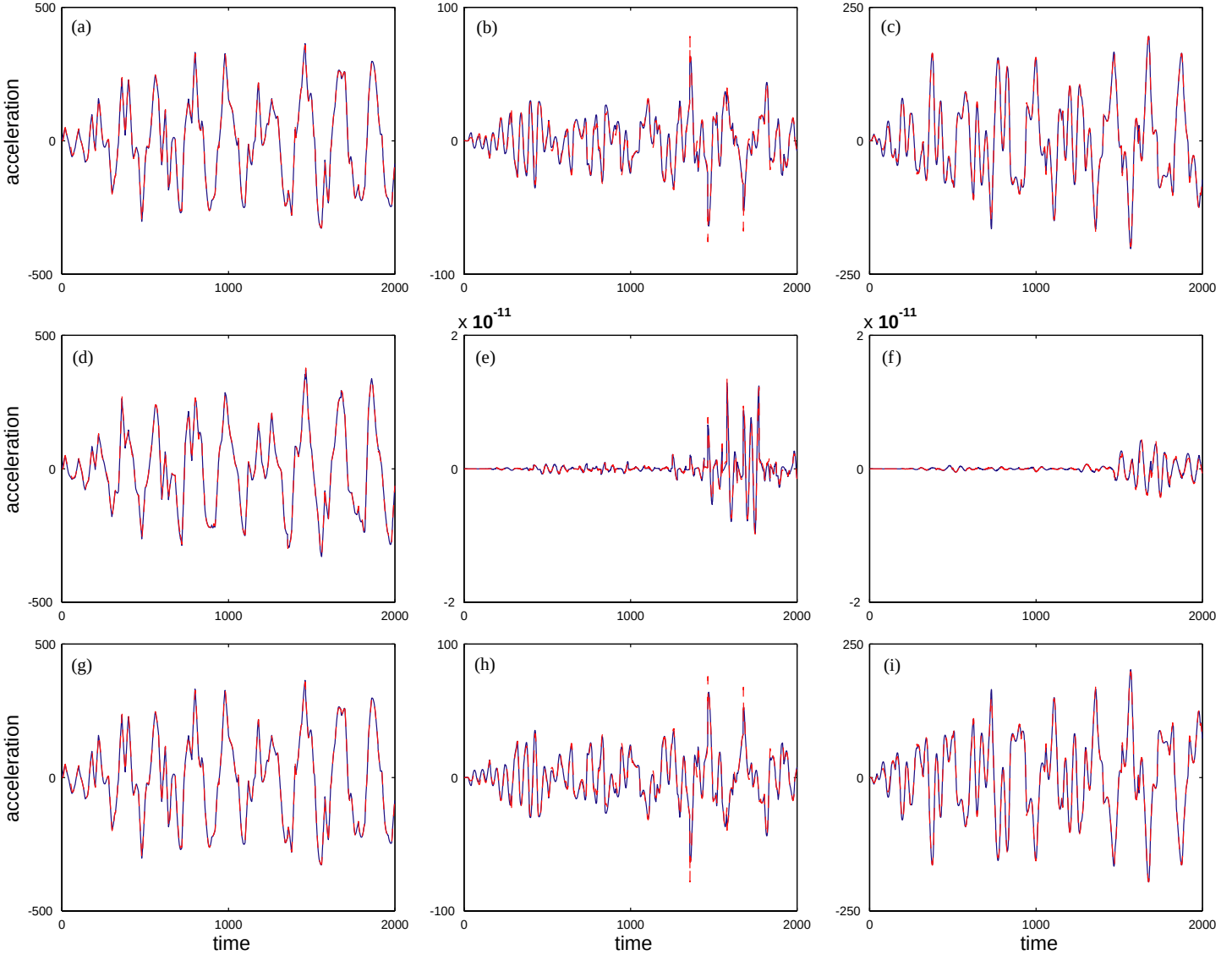

Fig. 13. Time-history of the real (solid lines) and estimated (dashed lines) accelerations at all nine channels, third order model with training on. Note the different output range at each channel. (a)–(i) correspond to channels 1–9.

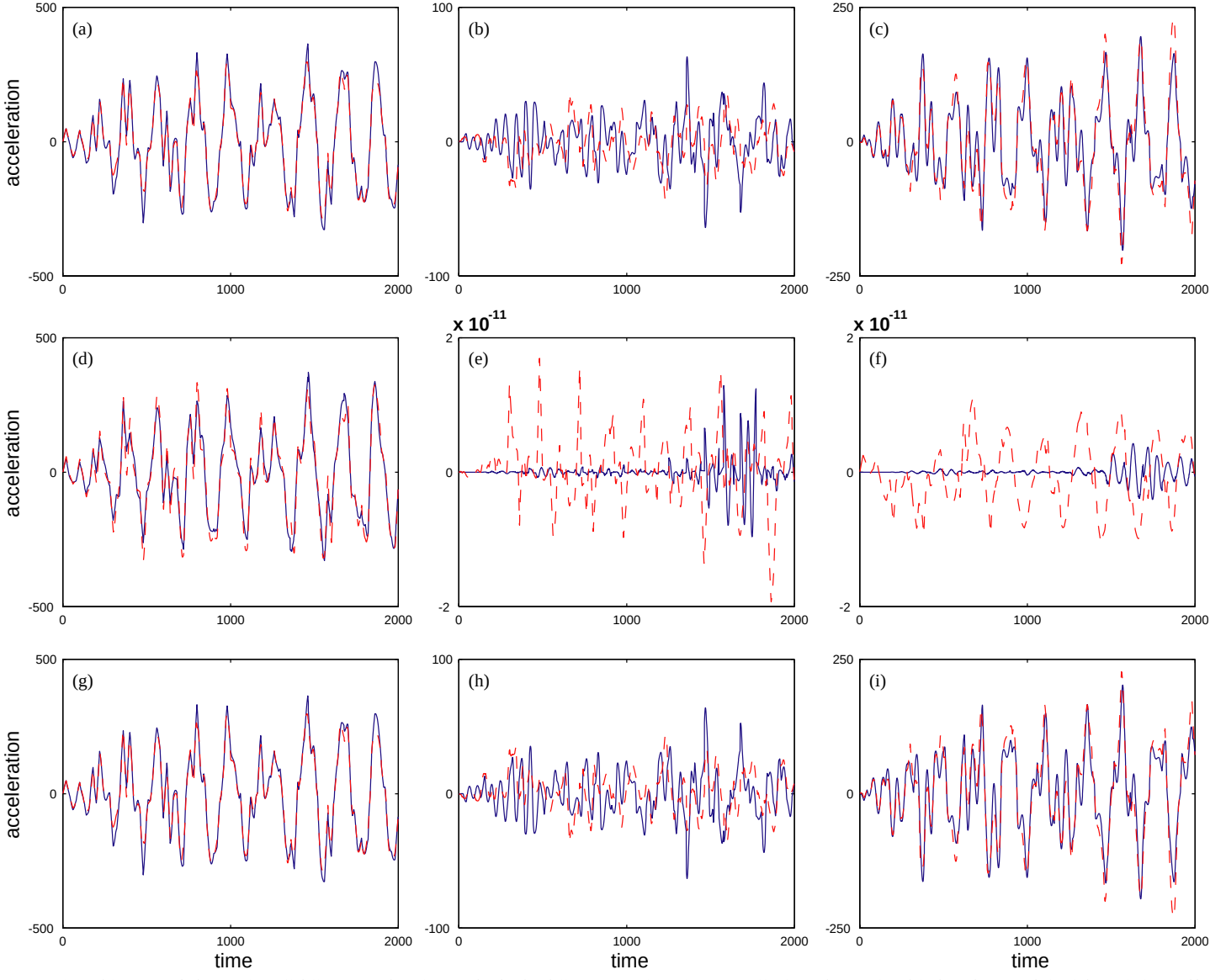

Fig. 14. Time-history of the real (solid lines) and estimated (dashed lines) accelerations at all nine channels, third order model with training off. Note the different output range at each channel. (a)–(i) correspond to channels 1–9.

Table 3
Summary of the values of the weights for the imaginary SDOF system

Serial number of weights	Exact value in Case 1 $\alpha = 0.005,\ \lambda = 0.001$	Learned value in Case 2 RMS error = 0.0741 $\alpha = 0.005,\ \lambda = 0.001$ $\gamma = 3 \times 10^5$	Learned value in Case 3 RMS error = 0.0836 $\alpha = 0.05,\ \lambda = 0.05$ $\gamma = 80$
1	4.7481×10^{-6}	5.8162×10^{-6}	1.0376×10^{-7}
2	-9.6241×10^{-2}	-9.6241×10^{-2}	-1.9261×10^{-3}
3	-1.5458×10^{-2}	-1.5458×10^{-2}	-3.0929×10^{-4}
4	-4.4409×10^{-8}	5.1862×10^{-8}	-2.5951×10^{-8}
5	-8.3404×10^{-9}	1.2711×10^{-9}	-3.6726×10^{-7}
6	-1.4190×10^{-12}	-7.5765×10^{-13}	-3.3230×10^{-13}
7	1.0486×10^{-8}	5.9206×10^{-9}	6.6442×10^{-9}
8	-1.1669×10^{-4}	-8.1795×10^{-5}	-1.1239×10^{-5}
9	-3.9996×10^{-9}	5.1521×10^{-9}	5.9356×10^{-9}
10	-8.6292×10^{-5}	-8.2892×10^{-5}	-5.1505×10^{-5}
11	-7.3933×10^{-5}	-1.4600×10^{-4}	-1.5017×10^{-5}
12	-2.5090×10^{-14}	-6.3401×10^{-15}	-3.5611×10^{-14}
13	1.1157×10^{-10}	5.0896×10^{-11}	9.9564×10^{-10}
14	-1.8323×10^{-10}	-7.1022×10^{-11}	2.9314×10^{-10}
15	2.6984×10^{-16}	-9.1984×10^{-17}	1.9276×10^{-13}
16	-6.7156×10^{-16}	-2.2546×10^{-17}	-2.0204×10^{-13}
17	-3.8225×10^{-13}	2.4335×10^{-12}	-2.2762×10^{-10}
18	-6.7676×10^{-12}	-5.1323×10^{-12}	-5.0012×10^{-10}
19	4.8487×10^{-17}	-1.4650×10^{-18}	2.0710×10^{-12}
20	7.9820×10^{-18}	2.5570×10^{-18}	3.3372×10^{-11}

of 1000 time steps each are plotted in Fig. 15. The weight vector in this case, in fact, is obtained from freezing the final trained values of the weights of another set of random wideband excited signal with zero mean and standard deviation of one and of 100 time steps. The parameter values used there are: $\alpha = 0.005,\ \lambda = 0.001$, and $\gamma = 2 \times 10^5$.

Case 2: The wideband excitation force, $\mathbf{f}_1^{\star}$ and the corresponding acceleration, $\ddot{\mathbf{x}}_1^{\star}$, derived in Case 1 are used in the training. The weights to be trained are all initialized to zero. When $\alpha = 0.005,\ \lambda = 0.001$, and $\gamma = 3 \times 10^5$, the RMS error at the 1000th time step is considered acceptable. The final trained weights are shown in Table 3. Note that the same values of α and λ are used in both Cases 1 and 2. Since the value of α and λ control the input layer and Layer 1 as shown in Fig. 3; the similarity of the weights in Cases 1 and 2 indicates that a unique solution can be obtained in this case. The time history of the trained weights is illustrated in Fig. 16. It can be seen that the trained weights converge to the exact values for the dominating weights in terms of magnitude. Almost all the weights converge very fast; however, two weights out of the twenty become unstable. These two weights, however, correspond with terms which yield a negligible contribution to the network output. It also seems that the two weights may be related to the numerical integration scheme in Eq. (5); further work being undertaken to study these unstable weights.

Case 3: While the same input and output to the neural network are used as in Case 2, the values of the three design constants are different. All the weights are initialized to zero. It can be seen

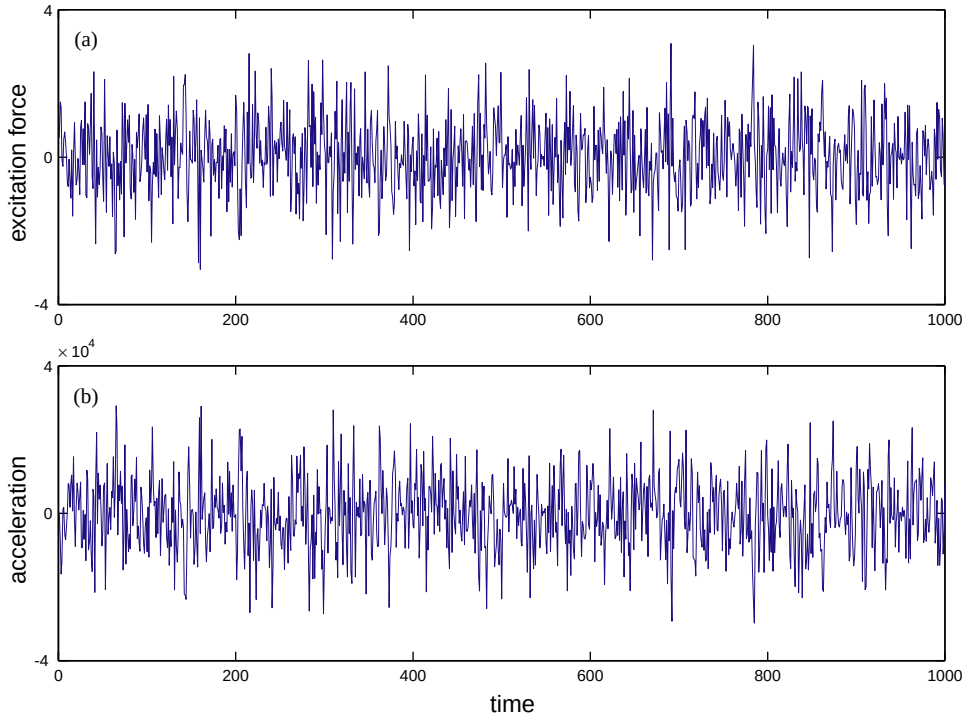

Fig. 15. Time-history of (a) the wideband excitation force and (b) the calculated acceleration of an imaginary SDOF system.

that the final trained weights are very different from those in Case 2, although the RMS errors in both cases are comparable. This indicates the non-uniqueness of the neural network solution, when varying the values of α and λ.

5. Conclusions

This study makes an attempt to demystify the inner-workings of a powerful VWNN in terms of problem formulation and network architecture and also improves its computational efficiency. A training example using generic data demonstrates that the VWNN is able to yield a unique set of weights when the values of the controlling parameters α and λ are fixed. Meanwhile, the design parameters α and λ, as well as the network architecture, can be related to the non-linearities of the dynamics system to be modelled. A training example using simulation data shows the efficiency of this neural network in predicting accelerations. The advantages of the VWNN indicate the potential of applying such highly flexible non-parametric identification techniques (namely, neural networks) in a parametric fashion to structural health monitoring and damage detections. Further work is needed to better control this VWNN in order to better perform on-line identification of non-linear systems.

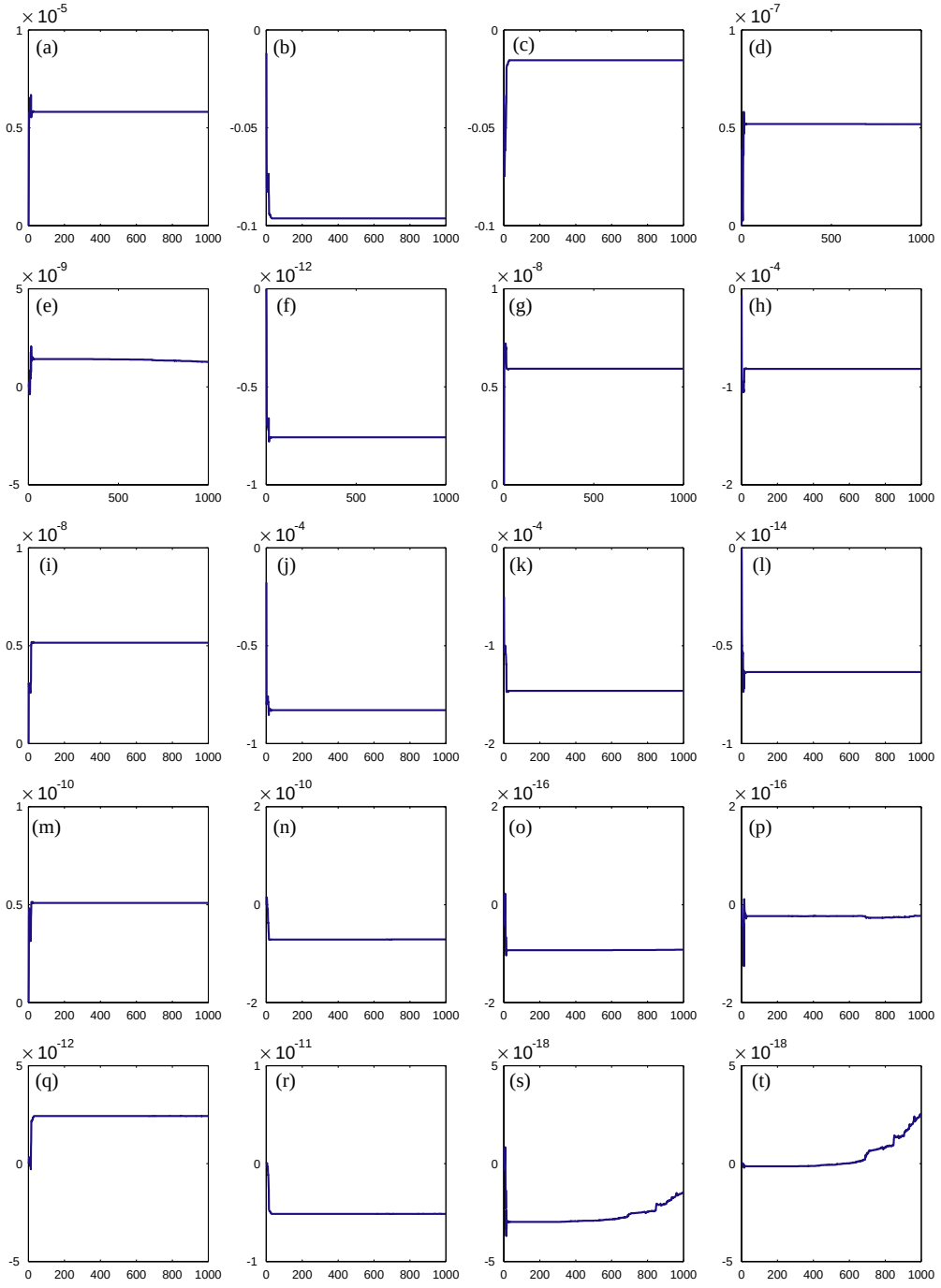

Fig. 16. Time-history of the trained weights in Case 2 of Table 3. Note the different orders of magnitude of these weights. (a)–(t) correspond to w_1–w_{20}.

Acknowledgements

This study was supported in part by the National Science Foundation under CAREER Award CMS-0134333.

References

[1] E.B. Kosmatopoulos, A.W. Smyth, S.F. Masri, A.G. Chassiakos, Robust adaptive neural estimation of restoring forces in nonlinear structures, *American Society of Mechanical Engineers Journal of Applied Mechanics* 68 (2001) 880–893.
[2] Y.K. Wen, Methods of random vibration for inelastic structures, *Applied Mechanical Review of the American Society of Mechanical Engineers* 42 (2) (1989) 39–52.
[3] F. Benedettini, D. Capecchi, F. Vestroni, Identification of hysteretic oscillators under earthquake loading by nonparametric models, *American Society of Civil Engineers Journal of Engineering Mechanics* 121 (5) (1995) 606–612.
[4] A.W. Smyth, E.B. Kosmatopoulos, S.F. Masri, A.G. Chassiakos, T.K. Caughey, Development of adaptive modeling techniques for nonlinear hysteretic systems, *International Journal of Nonlinear Mechanics* 37 (2002) 1435–1451.
[5] J.S. Pei, Parametric and Nonparametric Identification of Nonlinear Systems, PhD Dissertation, Columbia University, 2001.
[6] K. Worden, Data processing and experiment design for the restoring force surface method, Part I: integration and differentiation of measured time data, *Mechanical Systems and Signal Processing* 4 (4) (1990) 295–319.
[7] G. Cybenko, Approximation by superpositions of sigmoidal function, *Mathematics of Control, Signals, and Systems* 2 (1989) 303–314.
[8] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, *Neural Networks* 2 (1989) 359–366.
[9] L. Ljung, *System Identification Theory for the User*, 2nd Edition, Prentice-Hall PTR, Englewood Cliffs, NJ, 1999.
[10] A.G. Chassiakos, S.F. Masri, A.W. Smyth, T.K. Caughey, On-line identification of hysteretic systems, *American Society of Civil Engineers Journal of Engineering Mechanics* 65 (3) (1998) 194–203.
[11] A.W. Smyth, Analytical and Experimental Studies in Nonlinear System Identification and Modeling for Structural control, PhD Dissertation, University of Southern California, 1998.
[12] S.F. Masri, T.K. Caughey, A nonparametric identification technique for nonlinear dynamic problems, *American Society of Mechanical Engineers Journal of Applied Mechanics* 46 (1979) 433–447.
[13] K.S. Narendra, A.M. Annaswamy, *Stable Adaptive Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1989.
[14] E.B. Kosmatopoulos, M.M. Polycarpou, M.A. Christodoulou, P.A. Ioannou, High-order neural network structures for identification of dynamical systems, *IEEE Transactions on Neural Networks* 6 (2) (1995) 422–431.
[15] D.E. Rumelhart, J.L. McClelland, the PDP Research Group, *Parallel Distributed Processing Explorations in the Microstructure of Cognition*, *Volume 1*: *Foundations*, MIT Press, Cambridge, MA, 1986.
[16] K. Gurney, *An Introduction to Neural Networks*, UCL Press, London, 1997.
[17] M.A. Al-Hadid, J.R. Wright, Developments in the force-state mapping technique for non-linear systems and the extension to the location of non-linear elements in a lumped-parameter system, *Mechanical Systems and Signal Processing* 3 (3) (1989) 269–290.
[18] S.F. Masri, R.K. Miller, H. Sassi, T.K. Caughey, A method for reducing the order of nonlinear dynamic-systems, *American Society of Mechanical Engineers Journal of Applied Mechanics* 51 (2) (1984) 391–398.