

Yasar Becerikli

## On three intelligent systems: dynamic neural, fuzzy, and wavelet networks for training trajectory

Received: 11 August 2003 / Accepted: 13 May 2004 / Published online: 18 September 2004  
© Springer-Verlag London Limited 2004

**Abstract** Intelligent systems cover a wide range of technologies related to hard sciences, such as modeling and control theory, and soft sciences, such as the artificial intelligence (AI). Intelligent systems, including neural networks (NNs), fuzzy logic (FL), and wavelet techniques, utilize the concepts of biological systems and human cognitive capabilities. These three systems have been recognized as a robust and attractive alternative to the some of the classical modeling and control methods. The application of classical NNs, FL, and wavelet technology to dynamic system modeling and control has been constrained by the non-dynamic nature of their popular architectures. The major drawbacks of these architectures are the curse of dimensionality, such as the requirement of too many parameters in NNs, the use of large rule bases in FL, the large number of wavelets, and the long training times, etc. These problems can be overcome with dynamic network structures, referred to as dynamic neural networks (DNNs), dynamic fuzzy networks (DFNs), and dynamic wavelet networks (DWNs), which have unconstrained connectivity and dynamic neural, fuzzy, and wavelet processing units, called “neurons”, “feurons”, and “wavelons”, respectively. The structure of dynamic networks are based on Hopfield networks. Here, we present a comparative study of DNNs, DFNs, and DWNs for non-linear dynamical system modeling. All three dynamic networks have a lag dynamic, an activation function, and inter-connection weights. The network weights are adjusted using fast training (optimization) algorithms (quasi-Newton methods). Also, it has been shown that all dynamic networks can be effectively used in non-linear system modeling, and that DWNs result in the best capacity. But all networks have non-linearity properties in non-linear systems. In this study, all dynamic

networks are considered as a non-linear optimization with dynamic equality constraints for non-linear system modeling. They encapsulate and generalize the target trajectories. The adjoint theory, whose computational complexity is significantly less than the direct method, has been used in the training of the networks. The updating of weights (identification of network parameters) is based on Broyden–Fletcher–Goldfarb–Shanno method. First, phase portrait examples are given. From this, it has been shown that they have oscillatory and chaotic properties. A dynamical system with discrete events is modeled using the above network structure. There is a localization property at discrete event instants for time and frequency in this example.

**Keywords** Dynamic networks · Neural network · Fuzzy systems · Wavelets · Training trajectory · Intelligent systems · Adjoint theory

### 1 Introduction

Intelligent systems including neural networks (NNs), fuzzy logic (FL), and wavelet techniques utilize the concepts of biological systems and human cognitive capabilities. They possess learning, adaptation, and classification capabilities that hold out the hope of improved modeling and control for today’s complex systems. In this study, we present improved model design through three sorts of intelligent modeler which will be used in intelligent controllers; those based on dynamic neural networks (DNNs), those based on dynamic fuzzy networks (DFNs), and those based on dynamic wavelet networks (DWNs). DNNs capture the dynamic parallel processing and learning capabilities of biological nervous systems; DFNs, in addition to those properties, capture the decision-making capabilities of human linguistic and cognitive systems; and DWNs give a better approximation to signals and other transient or localized phenomena, both in time and frequency, and also cap-

Y. Becerikli  
Department of Computer Engineering,  
Kocaeli University, Izmit, Turkey  
E-mail: becer@kou.edu.tr, ybecer@ieeee.org  
Tel.: +90-262-3351168  
Fax: +90-262-3351150

ture the dynamic parallel processing and learning capabilities.

This study brings DNNs, DFNs, and DWNs together with dynamical model and control systems. Intelligent systems modeling (or identification) and control achieve automation via the emulation of biological intelligence. Intelligent systems modeling and control contain a wide area of technologies, such as the proportional-integral-derivative (PID) control, optimal control theory, system identification, artificial intelligence (AI) such as NNs, FL, wavelets, etc., and heuristics. In many physical and engineering systems, non-linearity properties are enough to prevent the well known application of linear control theory. There are many methods to solve different kinds of non-linear optimal control problem [5, 6, 16, 25, 26, 57]. One of the difficult problems encountered is optimal control for non-linear systems. An important aspect of any control system is its implementation on actual industrial systems. The major complication introduced during the modeling of a non-linear dynamical system with intelligent systems (which are DNNs, DFNs, and DWNs in this study) is which principles should be considered to obtain the accurate “model equivalence” of a known model of a non-linear dynamical system? Neural, fuzzy, and wavelet modeling and control have emerged as the most important branches in the last decade. They achieved successes in their application to many engineering systems in the real world [21, 23, 41, 58, 73, 75]. One of the goals of AI is focused on developing computational approaches to intelligent behavior [14]. As a final analysis, the role model is the human brain and NNs, the human mind and fuzzy networks (FNs), and the localized signals and wavelet networks (WNs), which are three of the oversimplified models of it [23].

Recently, NNs, FNs, and WNs have been paid more attention in the identification and control of unknown non-linear systems, owing to their massive parallelism, fast adaptation properties, and locality capturing and learning capabilities. But, until now, the most widely used NNs, FNs, and WNs systems are algebraic systems, despite the immense popularity of the algebraic neural, fuzzy, and wavelet systems (or feedforward networks) that are usually implemented for the approximation of a non-linear function [13, 23, 50, 52, 69, 73, 75].

In this study, the modeling principles of a non-linear system with DNNs, DFNs, and DWNs with unconstrained connectivity and with dynamic neural, fuzzy, or wavelet processing units, called “neurons”, “feurons”, or “wavelons”, have been given. The dynamic networks modeling problem is considered as a non-linear optimization with dynamic equality constraints and DNNs, DFNs, and DWNs, as compared with each other, are used for modeling with learning, generalization, and encapsulating capabilities.

The application of NNs, FNs, and WNs to dynamic system modeling and control has been constrained by the non-dynamic nature of popular network architectures. All algebraic (feedforward) NNs, FNs, and

WNs suffer from some drawbacks. In non-linear system modeling, a taped-delay lines approach is required, resulting in the number of rules increasing exponentially, the number of parameters in the rules getting large (this is called as “the curse of dimensionality”), a long computational time, easily being affected by external noise, and difficulty in obtaining an independent system simulator [32, 45, 52, 54]. The major drawbacks in these architectures are the curse of dimensionality, such as the requirement of too many parameters in NNs, the use of large rule bases in FL, the large number of wavelets, and the long training times, etc. An important problem for neural and fuzzy system applications is how to deal with the neuron and layer number, and this rule explosion problem. The same problems also exist in algebraic (feedforward) wavelet networks. Many of the problems as stated above can be overcome with DNNs, DFNs, and DWNs [1–4, 17, 18, 21, 28, 30, 31, 33, 45].

In previous research, to overcome the drawbacks, some alternative approaches have been developed. The recurrent neural network (RNN) structure is developed for this purpose [33, 53, 56, 68]. The most important model is the fuzzy Takagi–Sugeno model. The original idea of the Takagi–Sugeno model comes from fuzzy identification. The linear dynamic fuzzy model is used for non-linear system modeling [63, 64]. The Takagi–Sugeno model incorporates an idea that local dynamics (linear dynamics) of a non-linear system can be represented by different linear dynamic models [8, 66, 67]. On the wavelet front, some important developments have been made in the last decade [2, 12, 40, 45, 65, 71, 73].

In this study, alternatively, we used dynamic networks (for DNNs, DFNs, and DWNs)—these have a quasi-linear dynamic nature—containing dynamic elements such as integrators (or delayers in discrete time) in their processing units, which promise to overcome those drawbacks and may also allow for the incorporation of both heuristics (this includes neuron number from test and experience, if-then rules from people experience, and wavelons number from test and experience) and hard knowledge to exploit the best characteristics of the dynamical systems [1–4, 28, 30, 31, 45, 52, 69, 72].

The most important complication when dynamics are incorporated into the networks (algebraic networks) model is related to supervised training algorithms. The training algorithms are used to obtain appropriate network weights, time constants, and membership and wavelet function parameters of the wavelet and fuzzy systems. In only algebraic/feedforward neural, fuzzy, and wavelet networks, identification of its parameters is easy to compute [13, 39, 45, 50, 52, 69]. In dynamic networks, the gradient calculation with respect to networks weights (or parameters) are more complicated [59]. The gradient calculation structure in dynamic systems has been developed in systems, control, identification, and optimal control theory [5, 16, 34, 38, 72]. These approaches have been successfully applied in identification, modeling, and control applications [1–4, 27–31, 45, 53].

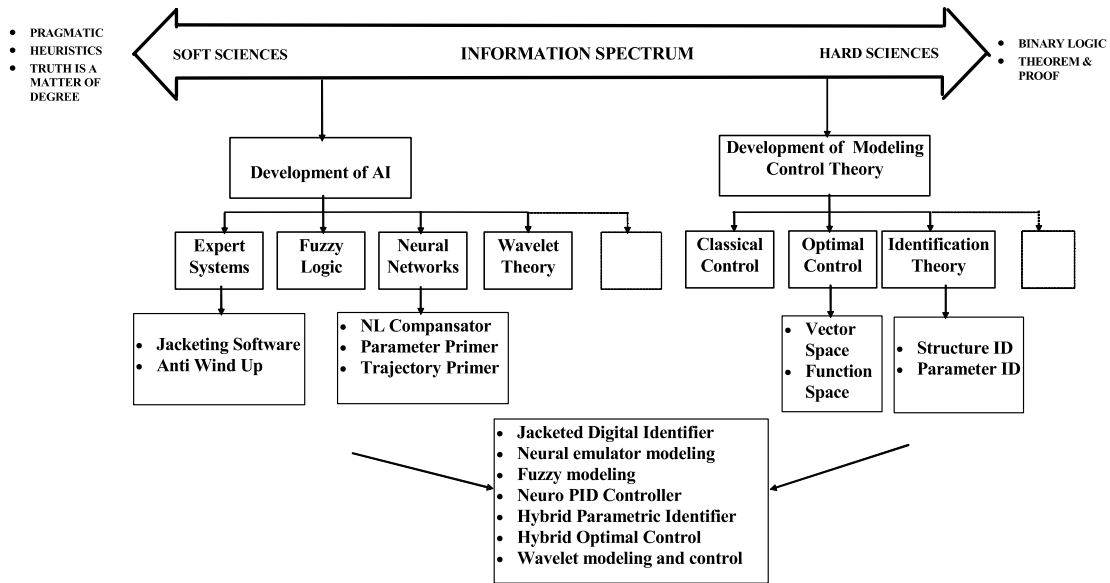


Fig. 1 Schematic diagram of intelligent modeling and control

Intelligent systems cover a wide range of technologies related to hard sciences, such as modeling and control theory, and soft sciences, such as AI. Figure 1 shows a general diagram of intelligent modeling and control history.

In Sect. 2, we present the structure of the DNN, DFN, and DWN we used, together with illustrative examples. The non-linear optimization problem based on the adjoint sensitivity approach is discussed in Sect. 3. Simulation results are given in Sect. 4 for modeling a system with a non-linear discrete event process using a fully connected neuron DNN, DFN, and DWN.

## 2 General dynamic network architecture

During the last few years, the non-linear dynamic system modeling of processes by neural and fuzzy networks has been extensively studied. NNs, FNs, and WNs have learning, approximation, and generalization properties. We present the dynamic type of networks. In fact, FNs and WNs are NNs with a special structure. NN and FN systems belong to a larger class of systems called “non-linear network structures” [37] that have some properties of extreme importance for feedback control systems. These networks are universal approximators [11, 19, 20, 51, 70], and WNs are also alternative universal approximators [12, 40, 74]. Non-linear dynamic models of processes with NNs, FNs with taped-delay lines, and recurrency have been often used [13, 32, 33, 39, 50, 52–54, 56, 68, 69], but WNs during the last few years have been even more widely used [45, 65, 71].

Dynamic network models has been used in the meaning of a network. The DNN, DFN, and DWN models we used have unconstrained connectivity and have dynamic elements in the neuro (neuron of DNN),

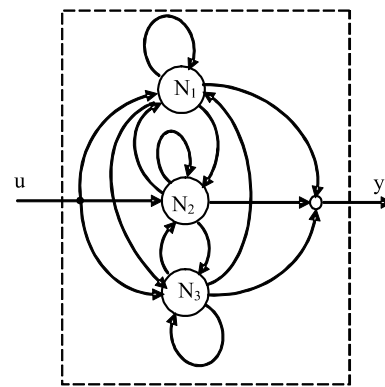


Fig. 2 Schematic diagram of a DNN, DFN, or DWN with three neurons/feurons/wavelons

feuro (neuron of DFN), and wavelo (neuron of DWN) processing units. A schematic diagram for the dynamic networks with three neurons is shown in Fig. 2.  $N_i$  can be a neuron in a DNN, a feuron in a DFN, or a wavelon in a DWN. In general, there are  $L$  input signals which can be time-varying,  $n$  dynamic units,  $n$  bias terms, and  $M$  output signals. The units have dynamics associated with them and they receive the input from themselves, the bias term, and from all other units. The output of a unit  $y_i$  is an activation function  $h(x_i)$  of a state variable  $x_i$  associated with the unit. The output of the overall network is a linear weighted sum of the unit outputs. The bias term  $b_i$  is added to the unit inputs.  $p_{ij}$  is the input connection weights from the  $j$ th input to the  $i$ th neuron (or feuron or wavelon),  $w_{ij}$  is the interconnection weight from the  $j$ th neuron (or feuron or wavelon) to the  $i$ th neuron (or feuron or wavelon) and  $q_{ij}$  is the output connection weight from the  $j$ th neuron (or feuron or wavelon) to the  $i$ th output.  $T_i$  is the dynamic constant of the  $i$ th neuron (or feuron or wavelon) and  $b_i$  is the bias (or polarization) term of the  $i$ th neuron (or feuron or wavelon).

The DNNs we describe here can be contrasted with the mathematical representations of neural systems found in the literature [1, 3, 4, 17, 18]. They take a popular form: standard algebraic neural network systems with external dynamics [15, 53]. In this study, a logarithmic sigmoid function is used as the activation function in the DNN:

$$h_i(x_i, \gamma_i, \beta_i) = \frac{1}{1 + \exp -(\gamma_i x_i + \beta_i)} \quad (1)$$

The processing unit in the DFN is the feuron [4, 46–48]. The feuron represents a single dynamic neuron with a fuzzy activation function. A DFN schematic diagram is as in Fig. 2. The dynamic feuron resembles the biological neuron model. This model fires if the inputs of the feurons are excited enough. The firing procedure is done through lag dynamics, such as Hopfield dynamics. The fuzzy activation function  $h$  behaves as biological neurons which have receptive field units in the visual cortex, in part of the cerebral cortex, and in the outer parts of the brain [17, 18, 52]. We have chosen the Gaussian function (this is known as the membership function in fuzzy logic literature) for the receptive field function (that is the part of fuzzy activation functions) as below:

$$R_{ij}(x_i) = \exp \left( -\frac{1}{2} \left( \frac{x_i - c_{ij}}{\sigma_{ij}} \right)^2 \right) \quad (2)$$

where  $c_{ij}$  is the center and  $\sigma_{ij}$  is the spread of the  $j$ th receptive field unit of the its  $i$ th feuron. The standard fuzzy system that has been used is the singleton fuzzifier, product inference engine with Gaussian membership function and center average defuzzifier. The  $i$ th activation function with the standard fuzzy system can be written as:

$$h_i(x_i) = \frac{\sum_{j=1}^{R_i} a_{ij} \mu_j(x_i)}{\sum_{j=1}^{R_i} \mu_j(x_i)} = \frac{\sum_{j=1}^{R_i} a_{ij} \exp \left( -\frac{1}{2} \left( \frac{x_i - c_{ij}}{\sigma_{ij}} \right)^2 \right)}{\sum_{j=1}^{R_i} \exp \left( -\frac{1}{2} \left( \frac{x_i - c_{ij}}{\sigma_{ij}} \right)^2 \right)} \quad (3)$$

The upper and lower membership functions of the universe of discourse can be considered by hard constraints ( $x_{iL}$  and  $x_{iU}$ ) as below:

$$\begin{aligned} \mu_1(x_i) &= 1 \quad \text{if } x_i \leq x_{iL} \text{ and} \\ \mu_{R_i}(x_i) &= 1 \quad \text{if } x_i \geq x_{iU} \end{aligned} \quad (4)$$

where  $R$  is number of fuzzy rules,  $a_{ij}$  are the output membership function centers and  $\mu_j(x_i)$  is the premise membership function of the  $j$ th rule. The feurons' fuzzification structure is a single input/single output (SISO) algebraic fuzzy system. The dynamic fuzzy networks we describe here can be contrasted with the mathematical representations of fuzzy and neural systems found in the literature. They take a popular form: standard algebraic neural network systems with external dynamics [15, 53,

69]. Another form is functional fuzzy systems, which are based on Takagi–Sugeno systems [49, 63, 64]. The standard algebraic and functional fuzzy systems necessitate the large number of rules that cause the important problem of “the curse of dimensionality.” On the contrary, the DFN has a fewer number of parameters and simpler units.

In DWNs, wavelet neurons (wavelons) input over a lag dynamic transport to output via a wavelet activation function. Wavelets are usually explained as basis functions which are compact (closed and bounded), orthogonal (or orthonormal), and have time–frequency localization properties. But, to provide all of those properties is very difficult. Basis functions are called “activation functions” in ANN literature, and can be a global or local feature in time. Global basis functions are active for the wide values of inputs and the receptive field of the basis function is approximately constant far from the center (i.e., logarithmic sigmoid function). But, the local basis functions are only active near the center; the value tends to zero far from the center.

If the global basis function is used in a network, all activation functions interact with each other and each node, and they cover a wide input interval. This causes the large number of parameters to adjust and necessitates a long computation time. In addition, for wide input intervals, much more extrapolation error occurs. The most important disadvantage of orthonormal compact basis functions is that they can not obtained in the closed analytical form.

To remove all those disadvantages, the local basis functions can be used. The local basis functions are only active for certain inputs. In addition, the generalization errors decrease [36]. In this study, only the local basis functions have been used. The most important local function is Gaussian:

$$\phi(x) = \exp \left( -\frac{x^2}{2} \right), \quad x \in R \quad (5)$$

where  $\phi \in L^2(R)$ . For the more general case:

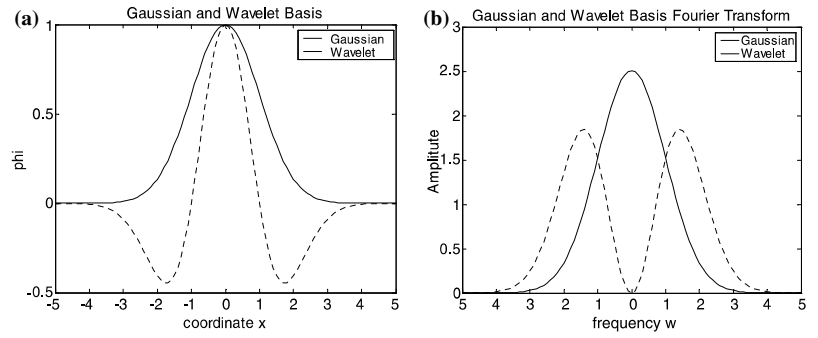
$$\phi \left( \frac{x - \mu}{\sigma} \right) = \exp \left( -\frac{1}{2} \left( \frac{x - \mu}{\sigma} \right)^2 \right), \quad x \in R \quad (6)$$

where  $\mu$  is the center or translation and  $\sigma$  is the standard deviation or dilation. The localization of the Gaussian function in time is shown in Fig. 3a. However, the Gaussian function is not local in frequency, as shown in Fig. 3b. The locality features in both time and frequency is a very important concept for the representation of the signals. Therefore, the mission of the wavelet functions is comprehensive.

The locality in time and frequency can be explained as follows:

- If a function is described in a bounded interval and has a very small value outside the boundary, then that function is local in time. The local function in time can be shifted by changing its center.

**Fig. 3** **a** Gaussian (solid line) and wavelet (dashed line) basis functions. **b** The Fourier transform of the Gaussian (solid line) and wavelet (dashed line) basis functions



- If the frequency spectrum of the local function in time is described in a bounded frequency interval and has very small value outside the boundary, and also can be shifted by changing its dilation, then that function is local in frequency.

A deficiency of Gaussian-based ANNs is that they do not have localization capabilities in frequency. As shown in Fig. 3b, the Gaussian function is not local in frequency. Therefore, it is very difficult to use Gaussian-based functions in some applications [60]. To overcome these problems, there is a very effective way to use wavelet functions with time–frequency localization properties [7]. The time and frequency envelope of the Mexican Hat function (second derivative of the Gaussian function) is shown in Fig. 3. In some studies, the first derivative of the Gaussian function has been used [40, 45]. However, the locality properties of the second derivative of the Gaussian function are clearer. A non-orthonormal Mexican Hat basis function can be easily written in the analytical form and its Fourier transform can be found [65], thus:

$$\phi(x_i) = (1 - x_i^2) \exp\left(-\frac{x_i^2}{2}\right), \quad x \in R \quad (7)$$

$$\phi(\omega) = \sqrt{2\pi}\omega^2 \exp\left(-\frac{\omega^2}{2}\right), \quad \omega \in R \quad (8)$$

where  $\omega$  is a real frequency. The last equation can be generalized as follows:

$$\phi\left(\frac{x_i - \mu_i}{\sigma_i}\right) = \left(1 - \left(\frac{x_i - \mu_i}{\sigma_i}\right)^2\right) \exp\left(-\frac{1}{2}\left(\frac{x_i - \mu_i}{\sigma_i}\right)^2\right) \quad (9)$$

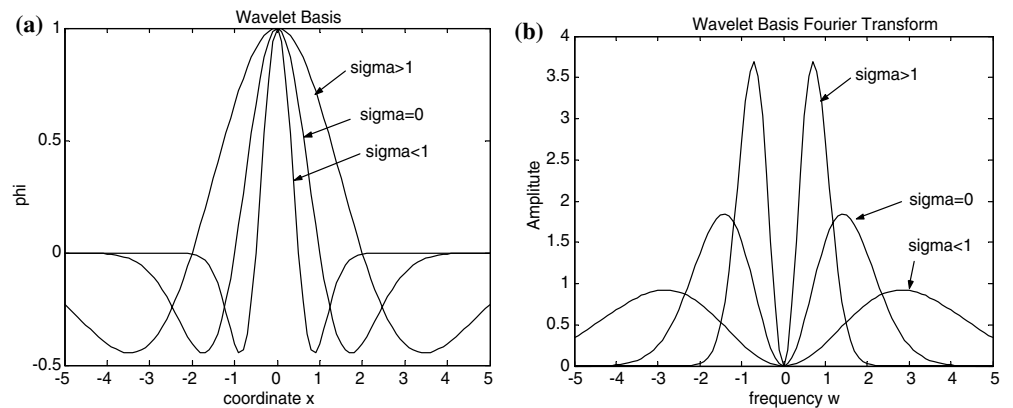
where  $\mu_i$  and  $\sigma_i$  are the translation (center) and dilation (standard deviation) parameters, respectively. Wavelet functions have efficient time–frequency localization properties, as shown from the frequency spectrum [40]. As shown in Fig. 4, if the dilation parameter is changed, the support region width of the wavelet function changes, but the number of cycles does not change. That is, the peak number does not change; however, when the dilation parameter decreases, the peak point of the spectrum shifts to a higher frequency. Therefore, all frequency spectrums can be obtained by changing the dilation. In this study, Eq. 7 has been used as a mother (main) wavelet [65]. An  $N$ -dimensional mother wavelet can be given in the separable structure with the product rule as follows [7, 40, 45, 74, 75]:

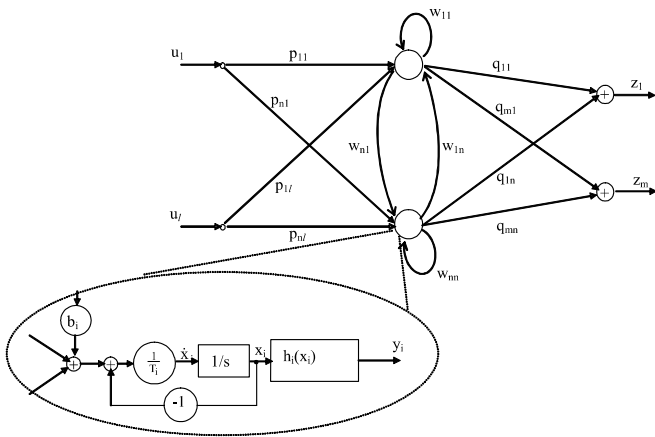
$$\Phi_i(x) = \prod_{j=1}^N \phi_j\left(\frac{x_j - \mu_{ij}}{\sigma_{ij}}\right) \quad (10)$$

where  $x_i \in R^N$  is the input and  $N$  is the input number. A function  $y = f(x)$  can be represented with wavelets obtained from the mother wavelet, [7, 40, 45] as below:

$$y_i = h_i(x) = \sum_{j=1}^{N_w} c_{ij} \Phi_j(x) + a_{i0} + \sum_{j=1}^N a_{ik} x_k \quad (11)$$

**Fig. 4a, b** Illustration of the dilation parameter effect. **a** Mexican Hat wavelet function. **b** Its Fourier transform





**Fig. 5** General mathematical computational model diagram of DNNs, DFNs, and DWNs

where  $c_{ij}$  are the coefficients of the mother wavelets,  $N_w$  is the number of wavelets,  $a_{i0}$  is a mean or bias term, and  $a_{ik}$  are the linear term coefficients of this approach.

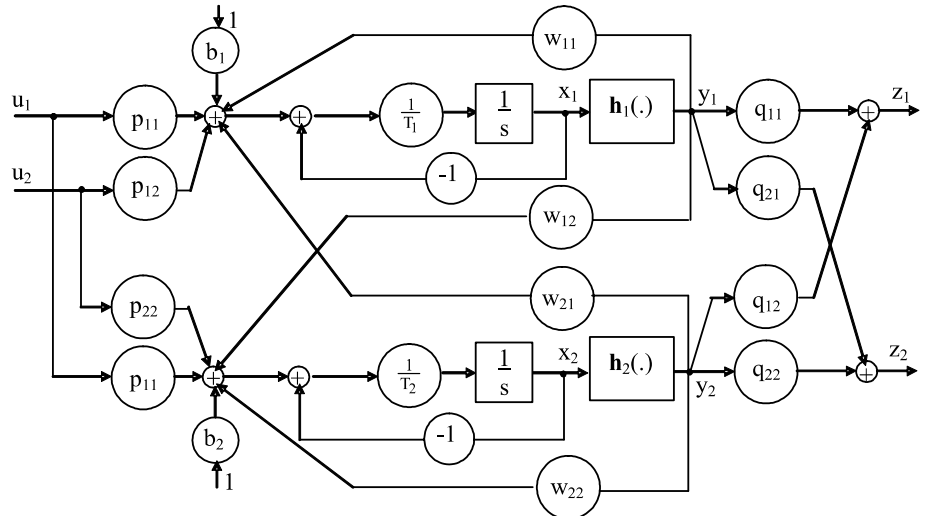
The wavelet function in this structure will be used in the DWN given in Fig. 2. The structure used in [1, 3, 4, 28, 30, 31, 46–48] has been adapted to this network. The wavelets in Eqs. 10 and 11 will be used as the activation functions in the network. Each activation function has a single input/single output (SISO), and can be re-expressed as:

$$\Phi_i(x_i) = \phi_i\left(\frac{x_i - \mu_{ij}}{\sigma_{ij}}\right) \quad (12)$$

$$y_i = h_i(x_i) = \sum_{j=1}^{N_w} c_{ij} \phi_i\left(\frac{x_i - \mu_{ij}}{\sigma_{ij}}\right) + a_{i0} + a_{i1}x_i \quad (13)$$

$$\phi_i\left(\frac{x_i - \mu_{ij}}{\sigma_{ij}}\right) = \left(1 - \left(\frac{x_i - \mu_{ij}}{\sigma_{ij}}\right)^2\right) \exp\left(-\frac{1}{2}\left(\frac{x_i - \mu_{ij}}{\sigma_{ij}}\right)^2\right) \quad (14)$$

**Fig. 6** The state diagram of DNN/DFN/DWN with two neurons/feurons/wavelons and two inputs/two outputs



The mathematical expression of the DWN can be written like that of the DNN and DFN [1–4, 28, 30, 31, 46–48].

In all of these theoretical aspects, the more general and open computational model of DNNs, DFNs, and DWNs is shown in Fig. 5.

The computational model of DNNs, DFNs, and DWNs is given in the following equations:

$$z_i = \sum_{j=1}^n q_{ij}y_j, \quad i = 1, 2, \dots, M \quad (15)$$

$$y_i = h(x_i, \pi_i), \quad i = 1, 2, \dots, n \quad (16)$$

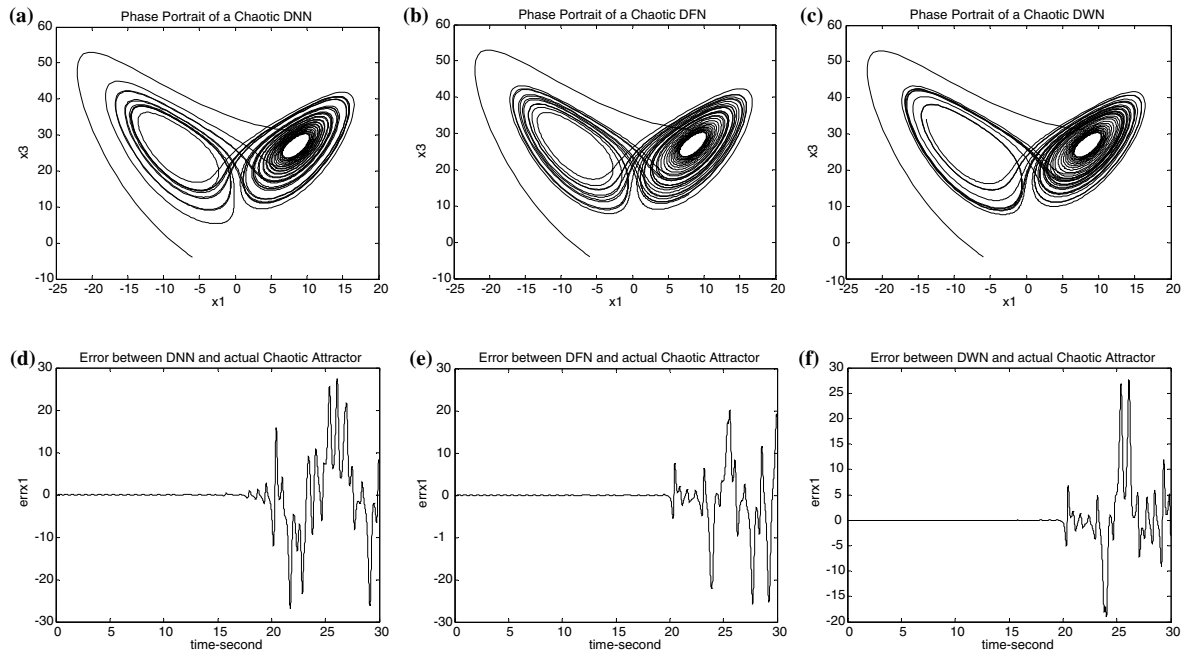
$$\dot{x}_i = f_i(x_i, p) = \frac{1}{T_i} \left[ -x_i + \sum_{j=1}^n w_{ij}y_j + \sum_{j=1}^L p_{ij}u_j + b_i \right]; \quad (17)$$

$$x_i(0) = x_{i0}, \quad i = 1, 2, \dots, n$$

where  $q_{ij}$  are the weights of the outputs of networks,  $w$ ,  $p$ ,  $q$ , and  $b$  are the interconnection parameters of the dynamic networks,  $T$  is the time constant, and  $\pi$  is the parameter of the activation function which are the neuron, feuron, or wavelon parameters, as given above. The initial conditions on the state variables  $x_i(0)$  must be specified. This model is similar to those in the literature [1–4, 18, 24, 43, 46–48, 56, 62].

### 2.1 Illustrative examples for the dynamical behavior of DNNs, DFNs, and DWNs

These models (DNN, DFN, DWN) approximate physical dynamic non-linear systems. In this section, some examples are given in which the DNNs, DFNs, and DWNs converge to an attractor or limit cycle, oscillate, or end in a chaotic fashion. The problem of training trajectories by means of continuous recurrent neural networks whose feedforward parts are as a multilayer perceptron has been studied [35]. The DNN, DFN, and DWN open diagram with two inputs/two outputs and two neurons, feurons, or wavelons is shown in Fig. 6.



**Fig. 7a–f** The state space trajectories of **a** DNN, **b** DFN, and **c** DWN as a chaotic system and error trajectories of state  $x_1$  for **d** DNN, **e** DFN, and **f** DWN

Given a set of parameters, initial conditions, and input trajectories, the output set of Eqs. 16 and 17 can be numerically integrated from  $t=0$  to the final time  $t_f$ . This will produce trajectories over time for the state variables  $x_i$ . We have used a 5-degree Runge–Kutta method [9, 44]. The integration step size has to be commensurate with the temporal scale of dynamics, determined by the time constants  $T_i$ . In our work, we have specified a lower bound on  $T_i$  and have used a fixed integration time step of some fraction (e.g., 1/10) of this bound.

### 2.1.1 DNNs, DFNs, and DWNs as a chaotic system

Consider the Lorenz system [25, 32] for the training of DNNs, DFNs, and DWNs. The interconnection and some neurons', feurons' membership (here, we used five memberships in a feuron), and wavelons' (with three-mother wavelet) parameters of the networks found by the training algorithm are given below:

$$w = \begin{bmatrix} 1.5 & 3.5 & -3.5 \\ -3.5 & 1.2 & -5 \\ -3.5 & 5 & 1 \end{bmatrix}, p = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, q = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, b = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{ for DNN, DFN, DWN}$$

$$\text{additionally for DFN: } \sigma = \begin{bmatrix} 0.522 & 0.226 & 0.032 & 0.963 & 1.433 \\ 0.362 & 0.536 & 0.039 & 0.336 & 0.454 \\ 0.375 & 0.102 & 0.005 & 1.313 & 0.745 \end{bmatrix}, a = \begin{bmatrix} -0.548 & 2.232 & -2.322 & 0.983 & -0.926 \\ -2.243 & 1.541 & -1.145 & 0.879 & -1.135 \\ -1.432 & 1.123 & -1.029 & 1.125 & -1.155 \end{bmatrix}$$

$$\text{additionally for DWN: } c = \begin{bmatrix} -0.872 & -1.209 & -1.102 \\ -1.431 & -1.098 & -1.011 \\ -1.671 & -0.907 & -1.006 \end{bmatrix}, \mu = \begin{bmatrix} 0.059 & -1.113 & -2.537 \\ -0.287 & 0.098 & -1.786 \\ -0.265 & -1.203 & -2.976 \end{bmatrix}$$

The initial conditions were  $x_i(0) = -6, -10, -4$ , ( $i = 1, 2, 3$ ). All of the dynamic networks successfully realized a chaotic system, which only shows  $x_1-x_3$  as a state space combination of DNN, DFN, and DWN. Figure 7 also shows the state  $x_1$  error trajectories between DNN/DFN/DWN and the actual Lorenz attractor trajectories. The error is very small up to approximately 18 s. After that, the error increases and the overlap rate is high. Overall, the overlap rate is satisfactory. When these portraits are compared with the real Lorenz system, the DWN portrait is nearest to the Lorenz portrait with the DFN being next in terms of good performance and, lastly, is the DNN. All networks were trained with the same iterations. Trajectory tracking performance is excellent in this application for all networks.

### 2.1.2 DNN, DFN, and DWN as an oscillator example

In this application, an oscillator system in [25] is modeled with two neurons/feurons/wavelons in a DNN/DFN/DWN. The interconnection and some neurons', feurons' membership (we used five memberships in a

feuron), and wavelons' (with three-mother wavelet) parameters of networks are shown below:

$$w = \begin{bmatrix} 0 & -1 \\ 1 & 1 \end{bmatrix}, p = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, q = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, b = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, T = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \text{ for DNN, DFN, DWN}$$

additionally for DNN:  $\gamma = \begin{bmatrix} 0.786 \\ 0.812 \end{bmatrix}$

additionally for DFN:  $\sigma = \begin{bmatrix} 0.594 & 1.005 & 0.129 & 0.865 & 1.131 \\ 0.683 & 1.118 & 0.122 & 0.924 & 2.129 \end{bmatrix}$

additionally for DWN:  $c = \begin{bmatrix} -0.872 & -1.209 \\ -1.431 & -1.098 \end{bmatrix}, \mu = \begin{bmatrix} 0.059 & -1.113 \\ -0.287 & 0.098 \end{bmatrix}$

The DNN/DFN/DWN converge to an oscillation situation for several initial conditions ( $x_i(0), i=1, 2$ ) (see Fig. 8). As can be seen, all the dynamic networks capture the oscillator system's behavior adequately.

In all of the above illustrative examples, the DNN, DFN, and DWN successfully capture the behavior of a non-linear physical dynamic system.

### 3 Parameter identification based on adjoint sensitivity analysis for dynamic network training

The DNN, DFN, and DWN training is used to encapsulate a given set of trajectories by adjusting network parameters. In this section, adjusting the parameters of the dynamic networks is presented for trajectory tracking. This is done by minimizing the cost function (error function). The gradient-based algorithms have been used for this problem. The cost gradients with respect to network parameters are required for the algorithm. The dynamic networks' general schematic diagram is shown in Fig. 9. Our focus in this paper has been the adjoint sensitivity analysis for calculating the cost gradients with respect to all networks parameters. The common network parameters are  $w, p, q, b, T; \gamma$  and  $\beta$  for DNN also;  $c, \sigma$ , and  $a$  for DFN also; and  $c, \mu, \sigma$ , and  $a$  for DWN

also. Note that some DFN and DWN parameters are different but the same notation was used.

A performance index (PI) or cost structure is selected in the simple quadratic form as follows:

$$E = \frac{1}{2} \int_0^{t_f} [z(t) - z^d(t)]^T [z(t) - z^d(t)] dt \tag{18}$$

where  $e(t) = z(t) - z^d(t)$  is the error function.  $z(t)$  is the response of the DNN, DFN, and DWN models (output), and  $z^d(t)$  is the desired (target) system response. We want to compute the cost sensitivities with respect to the various parameters:

$$\frac{\partial E}{\partial w}, \frac{\partial E}{\partial p}, \frac{\partial E}{\partial q}, \frac{\partial E}{\partial T}, \frac{\partial E}{\partial b}, \frac{\partial E}{\partial c}, \frac{\partial E}{\partial \sigma}, \frac{\partial E}{\partial a}, \frac{\partial E}{\partial \mu} \tag{19}$$

The output weight gradients can be easily obtained by differentiating Eqs. 18 and 15:

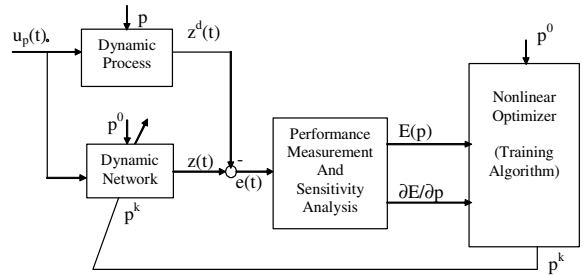
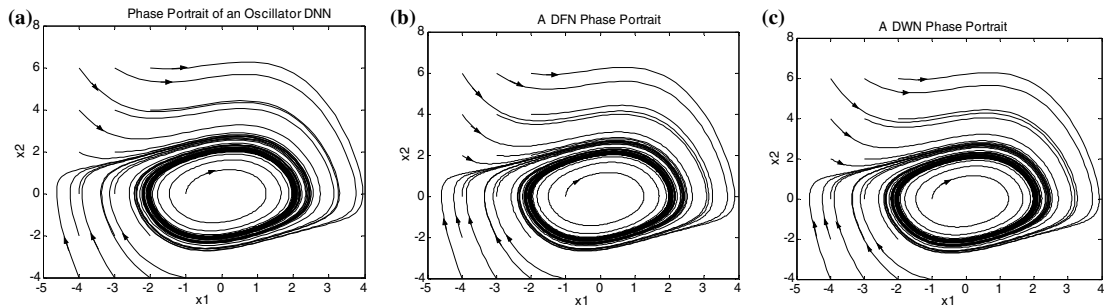


Fig. 8a–c The state space trajectories of a DNN, b DFN, and c DWN as an oscillatory system

Fig. 9 Dynamic network training block diagram





$$\frac{\partial E}{\partial q_{ij}} = \int_0^{t_f} [z_i(t) - z_i^d(t)] \frac{\partial z_i}{\partial q_{ij}} dt = \int_0^{t_f} e_i(t) y_j dt \quad (20)$$

One approach to solving the constrained dynamic optimization problem is based on the use of the calculus of variations, which is called the ‘‘adjoint’’ method for sensitivity computation [1, 3–5, 27–31, 34]. The number of differential equations to be solved only depends on the neuron/feuron/wavelon number, and does not depend on network parameters. A new dynamical system defined with adjoint state variables  $\lambda_i$  is obtained as follows:

$$-\dot{\lambda}_i = -\frac{1}{T_i} \lambda_i + \frac{1}{T_i} \sum_j w_{ij} y'_j \lambda_j + e_i(t) \sum_j q_{ij} y'_j, \quad \lambda_j(t_f) = 0 \quad (21)$$

$$y'_j = \frac{\partial h_j(x_j)}{\partial x_j} = \begin{cases} \gamma_j h_j(1-h_j) & \text{for DNN} \\ \frac{\sum_{k=1}^{R_j} (h_j - a_{jk}) \exp\left(-\frac{1}{2} \left(\frac{x_j - c_{jk}}{\sigma_{jk}}\right)^2\right) \left(\frac{x_j - c_{jk}}{\sigma_{jk}^2}\right)}{\sum_{k=1}^{R_j} \exp\left(-\frac{1}{2} \left(\frac{x_j - c_{jk}}{\sigma_{jk}}\right)^2\right)} & \text{for DFN} \\ -\sum_{k=1}^{N_w} c_{jk} \left(3\phi_j + 2 \left(\frac{x_j - \mu_j}{\sigma_{jk}^2}\right)\right) \left(\frac{x_j - \mu_j}{\sigma_{jk}^2}\right) + a_{j1} & \text{for DWN} \end{cases} \quad (22)$$

The size of the adjoint vector is  $n$  and is independent of the network parameters. There are  $n$  quadratures for computing the sensitivities. The integration of the differential equations must be performed backwards in time, from  $t_f$  to 0. We have used the 5th-order Runge–Kutta–Butcher integration rule [9, 44]. Let  $p$  be a vector containing all network parameters. Then, the cost gradients with respect to the parameters are given by the following quadratures:

$$\frac{\partial E}{\partial p} = \int_0^{t_f} \left(\frac{\partial f}{\partial p}\right)^T \lambda dt \quad (23)$$

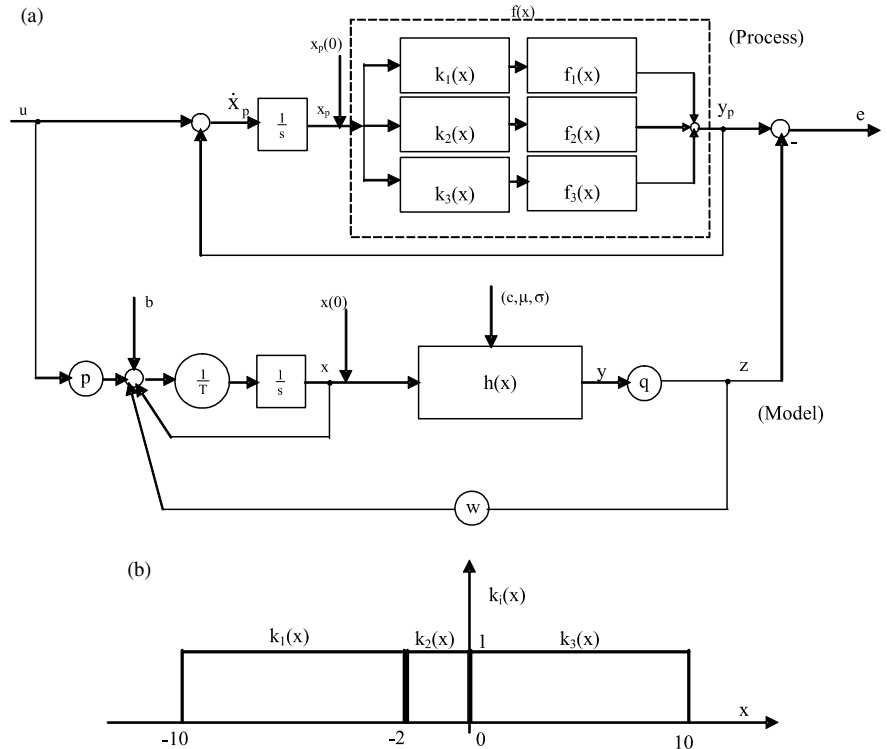
Some of the cost gradients as in [1–4, 30, 46–48] are as follows:

$$\frac{\partial E}{\partial w_{ij}} = \int_0^{t_f} \frac{\lambda_i y_j}{T_i} dt, \quad \frac{\partial E}{\partial b_i} = \int_0^{t_f} \frac{\lambda_i}{T_i} dt \quad (24)$$

$$\frac{\partial E}{\partial T_i} = \int_0^{t_f} \frac{\lambda_i}{T_i^2} \left[-x_i + \sum_{j=1}^n w_{ij} y_j + \sum_{j=1}^l p_{ij} u_j + b_i\right] dt \quad (25)$$

$$\frac{\partial E}{\partial \gamma_i} = \int_0^{t_f} \left(\sum_i \frac{\lambda_i}{T_i} w_{ij}\right) x_j h_j(1-h_j) dt \quad (26)$$

**Fig. 10 a** Modeling diagram of discrete event system. **b** The unit step gain functions



$$\frac{\partial E}{\partial c_{ik}} = \int_0^{t_f} \left( \sum_i \frac{\lambda_i}{T_i} w_{ij} \frac{a_{ik} - f_i}{\sum_{k=1}^R \exp\left(-\frac{1}{2} \left(\frac{x_i - c_{ik}}{\sigma_{ik}}\right)^2\right)} \right) \times \exp\left(-\frac{1}{2} \left(\frac{x_i - c_{ik}}{\sigma_{ik}}\right)^2\right) \left(\frac{x_i - c_{ik}}{\sigma_{ik}^2}\right) dt$$

$$k = 1, 2, \dots, R_i \text{ for DFN} \quad (27)$$

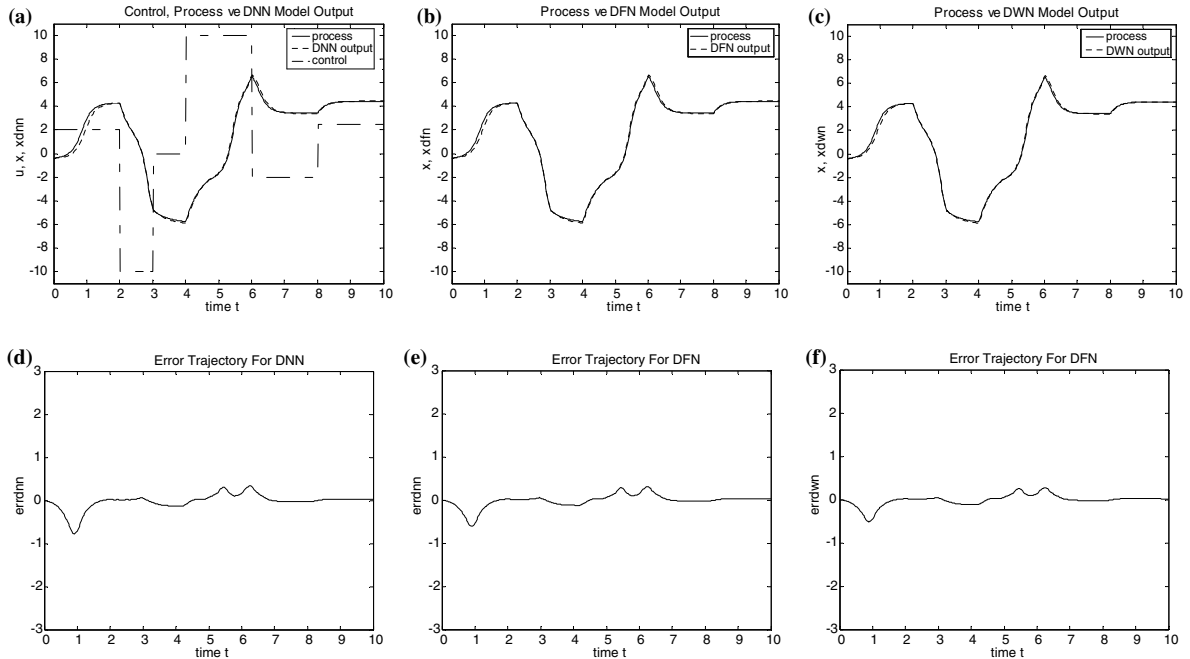
$$\frac{\partial J}{\partial c_{ij}} = \int_0^{t_f} \left( \sum_k \frac{\lambda_k}{T_k} w_{ki} \phi_i \left(\frac{x_i - \mu_{ij}}{\sigma_{ij}}\right) \right) dt \text{ for DWN} \quad (28)$$

All other gradients can be easily derived. Detailed results can be found in the literature [1, 3, 46, 47]. We assume that, at each iteration, the gradients of the performance index with respect to all networks parameters,  $g = \frac{\partial E}{\partial p}$ , is computed. Here, we describe an algorithm we have used for updating parameter values based on this gradient information:

$$p^{k+1} = p^k + \tau^k d^k, \quad d^k = -H^k g_p^k \quad (29)$$

where  $d$  is the search direction,  $\tau$  is the optimal step size along the search direction,  $g$  is the cost gradient with respect to the parameter, and  $H \cong (\nabla_{pp} J)^{-1}$  is the inverse of the approximate Hessian matrix. The Broyden-Fletcher-Golfarb-Shanno gradient method has been used for updating network weights. This method is faster than the simple gradient method and

**Fig. 11a–f** The modeled process with DNN, DFN, and DWN. **a** Control input (dashed-dotted line), DNN output (dashed line), and process output (solid line). **b** DFN output (dashed line) and process output (solid line). **c** DWN output (dashed line) and process output (solid line), Error trajectories for **d** DNN, **e** DFN, and **f** DWN



more robust than the simple conjugate gradient approach [1, 3, 4, 10, 16, 30, 46, 47, 55, 61]. This method provides the history of the parameter and gradient changes, yielding approximately second-order information.

The adjoint way of computing performance index sensitivities is efficient in the number of differential equations that need to be solved, but the intermediate computations within the time interval do not produce information that is meaningful in the original networks (DNN, DFN, DWN). Whereas the forward sensitivity method produces trajectories of the state and response sensitivities, the adjoint method produces trajectories of adjoint variables.

For algorithms requiring the exact Hessian, a computationally efficient approach is available using both the adjoint and forward response sensitivities [27, 29]. Thus, by performing both the forward and adjoint sensitivity analyses, an exact Newton method in the function space can be implemented at a substantially lower cost than that involved in the “forward” computation of exact second-order sensitivities.

## 4 Simulation results

As an application, a non-linear piecewise-continuous scalar function (discrete event system) [74] has been considered in the dynamic structure (passing through  $1 \text{ s}^{-1}$ ) by adding a control function and this function is the one to be modeled with DNN, DFN, and DWN. For this, the  $f(x)$  function is substituted into the  $\dot{x} = f(x, u)$ ,  $x(t_0) = x_0$ ,  $0 \leq t \leq t_f$  expression as below:

$$\dot{x} = f(x, u) = \begin{cases} -2.186x - 12.864 + u & -10 \leq x < -2 \\ 4.246x + u & -2 \leq x < 0 \\ 10 \exp(-0.05x - 0.5) \times \sin((0.03x + 0.7)x) + u & 0 \leq x < 10 \end{cases} \quad (30)$$

The modeling structure is shown in Fig. 10a. The unit step gain functions  $k_i(x)$  ( $i=1, 2, 3$ ) used are given in Fig. 10b.

This process has been trained by a DWN with a wavelon in the time interval  $t \in [0, 10]$ . The control function to be applied to the system input that was selected so that there was an adequate amount of oscillation of the system is shown in Fig. 11a. The initial condition was taken to be  $x_0 = -0.4$ . At the beginning of the training, some modeling parameters were set to  $p_{ij} = 1$  and  $T_i = 1$ , but the others were started randomly. After the training, the output of the DNN, DFN, and DWN are as in Fig. 11a–c, respectively. The right-hand side of Eq. 17 for  $u(t) = 0$  (that is,  $\hat{f}(x)$ , the static side of DNN, DFN, and DWN) has been successfully fitted to the real function  $f(x)$  given by right-hand side of Eq. 30 for  $u(t) = 0$  (see Fig. 12a–c). As can be seen, the joint point at  $x = -2$  was successfully modeled with DNN, DFN, and DWN. When one carefully looks at Figs. 11 and 12, the DWN approximation is better than the others, but DNN and DFN also are successful approximators.

## 5 Conclusions and future works

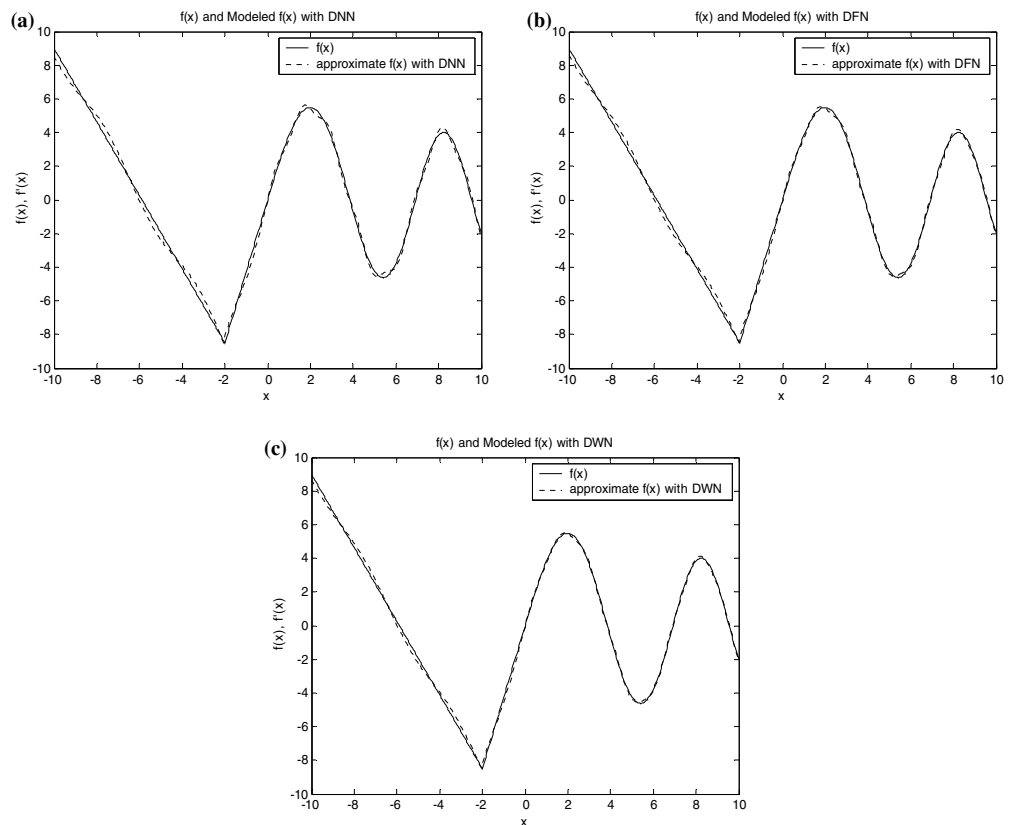
In this work, we presented three intelligent methods to be used in modeling, control, and the other applications. Any non-linear physical dynamic system can be captured by dynamic neural networks (DNNs), dynamic fuzzy networks (DFNs), and dynamic wavelet networks (DWNs). Simulation results show that the dynamic network structure can grow more accurately by neuro/fuzzy/wavelet approximators.

All of the results presented here were obtained with the help of a trained DNN, DFN, and DWN, which generated the model response policy close to the target process. In the illustrative examples, the dynamic networks used have some non-linear dynamic system behavior, such as chaotic, oscillator, etc.

In the simulations presented, we used a non-linear system with a discrete event system. All three networks were successfully used for modeling the target process. According to the modeling and training speed performance, better results have been obtained from DWNs, but DFNs and DNNs have also produced satisfactory

**Fig. 12a–c** Static function approximation performance of DNN, DFN, and DWN.

**a** DNN approximation  $\hat{f}(x)$  (dashed line) and real function  $f(x)$  (solid line). **b** DFN approximation  $\hat{f}(x)$  (dashed line) and real function  $f(x)$  (solid line). **c** DWN approximation  $\hat{f}(x)$  (dashed line) and real function  $f(x)$  (solid line)



results. The exact Hessian-based optimization algorithm for application to DNN, DFN, and DWN is a valuable approximation to speed up training time. In addition, the local and orthogonal wavelet usage in these areas can increase the training speed for DWNs.

## References

1. Becerikli Y (1998) Neuro-optimal control. PhD thesis, Sakarya University, Sakarya, Turkey
2. Becerikli Y, Oysal Y, Konar AF (2002) Modeling of nonlinear systems with dynamic wavelet networks. In: Proceedings of the Turkish symposium on automatic control (TOK 2002), Ankara, Turkey, 9–11 September 2002, pp 71–79
3. Becerikli Y, Konar AF, Samad T (2003) Intelligent optimal control with dynamic neural networks. *Neural Netw* 16(2):251–259
4. Becerikli Y, Oysal Y, Konar AF (2004) Trajectory priming with dynamic fuzzy networks in nonlinear optimal control. *IEEE Trans Neural Net* 15(2):383–394
5. Bryson AE, Ho YC (1975) Applied optimal control. Hemisphere, New York
6. Bullock TE (1966) Computation of optimal controls by a method based on second variations. Department of Aeronautics and Astronautics, Stanford University, SUDAAR 297, December 1966
7. Cannon M, Slotine J-JE (1995) Space-frequency localized basis function networks for nonlinear system estimation and control. *Neurocomputing* 9(3):293–342
8. Cao SG, Rees NW (1995) Identification of dynamic fuzzy model. *Fuzzy Set Syst* 74(3):307–320
9. Chapra SC, Canale RP (1989) Numerical methods for engineers. McGraw-Hill, New York
10. Chong EKP, Zyak SH (1996) An introduction to optimization. Wiley, New York
11. Cybenko G (1989) Approximation by superpositions of a sigmoidal function. *Math Control Signal* 2:303–314
12. Erdol N, Basbuğ F (1996) Wavelet transform based adaptive filters: analysis and new results. *IEEE Trans Signal Process* 44(9):2163–2169
13. Geva S, Sitte J (1992) A constructive method for multivariate function approximation by multilayer perceptrons. *IEEE Trans Neural Netw* 3(4):621–624
14. Gevarter WB (1983) An overview of artificial intelligence and robotics, vol I. NASA technical memorandum 85836
15. Gherrity M (1989) A learning algorithm for analog, fully recurrent neural networks. In: Proceedings of the international joint conference on neural networks (IJCNN'89), Washington, DC, June 1989, vol. 1, pp 643–644
16. Hasdorff L (1976) Gradient optimization and nonlinear control. Wiley, New York
17. Hopfield JJ (1982) Neural networks and physical systems with emergent collective computational abilities. *Proc Nat Acad Sci USA* 79:2554–2558
18. Hopfield JJ (1984) Neurons with graded response have collective computational properties like those of two-state neurons. *Proc Nat Acad Sci USA* 81:3088–3092
19. Hornik K, Stinchcombe M, White H (1985) Multilayer feedforward networks are universal approximators. *Neural Netw* 2:359–366
20. Hornik K, Stinchcombe M, White H, Auer P (1994) Degree of approximation results for feedforward networks approximating unknown mappings and their derivatives. *Neural Comput* 6(6):1262–1275
21. Hunt KJ, Sbarbaro D, Zbikowski R, Gawthrop PJ (1992) Neural networks for control systems—a survey. *Automatica* 28(6):1083–1112
22. Iannou PA, Datta A (1991) Robust adaptive control: a unified approach. *Proc IEEE* 79(12):1736–1768
23. Jang JSR, Sun CT, Mizutani E (1997) Neuro-fuzzy and soft computing: a computational approach to learning and machine intelligence. Prentice Hall, Englewood Cliffs, New Jersey
24. Jordan MI (1986) Attractor dynamics and parallelism in connectionist sequential machine. In: Proceedings of the 8th annual conference of the cognitive science society, Amherst, Massachusetts. Lawrence Erlbaum, Hillsdale, New Jersey
25. Khalil HK (1996) Nonlinear systems. Prentice-Hall, Englewood Cliffs, New Jersey
26. Kirk DE (1970) Optimal control theory: an introduction. Prentice-Hall, Englewood Cliffs, New Jersey
27. Konar AF (1991) Gradient and curvature in nonlinear identification. In: Proceedings of the Honeywell advanced control workshop, Minneapolis, Minnesota, January 1991
28. Konar AF (1997) Historical and philosophical perspectives on intelligent and neural control. In: Proceedings of the IEEE international symposium on intelligent control (ISIC'97), Istanbul, Turkey, July 1997
29. Konar AF, Samad T (1991) Hybrid neural network/algorithmic system identification. Technical report SSDC-91-I 4051-1, Honeywell Technology Center, 3660 Technology Drive, Minneapolis, MN 55418
30. Konar AF, Samad T (1992) Dynamic neural networks. Technical report SSDC-92-I 4152-2, Honeywell Technology Center, 3660 Technology Drive, Minneapolis, MN 55418
31. Konar AF, Becerikli Y, Samad T (1997) Trajectory tracking with dynamic neural networks. In: Proceedings of the IEEE international symposium on intelligent control (ISIC'97), Istanbul, Turkey, July 1997
32. Kosko B (1997) Fuzzy engineering. Prentice-Hall, Upper Saddle River, New Jersey
33. Kosmatopoulos EB, Polycarpas MM, Christodoulou MA, Iannou PA (1995) High-order neural network structures for identification of dynamical systems. *IEEE Trans Neural Netw* 6(2):431–442
34. Lapidus L, Luus R (1996) Optimal control of engineering processes. Blaisdell, Waltham, Massachusetts
35. Leistriz L, Galicki M, Witte H, Kochs E (2002) Training trajectories by continuous recurrent multilayer networks. *IEEE Trans Neural Netw* 13(2):283–291
36. Leonard J, Kramer M (1991) Radial basis function networks for classifying process faults. *IEEE Control Syst* 11:31–38
37. Lewis F (1999) Nonlinear network structures for feedback control. *Asian J Control* 1:205–228
38. Lewis FL (1992) Applied optimal control and estimation. Prentice-Hall, Englewood Cliffs, New Jersey
39. Lippmann RP (1987) An introduction to computing with neural nets. *IEEE ASSP Mag* 4(2):4–22
40. Mallat SG (1987) Multifrequency channel decompositions of images and wavelet models. *IEEE Trans on ASSP* 37(12):2091–2109
41. Mauer GF (1995) A fuzzy logic controller for an ABS braking system. *IEEE Trans on Fuzzy Systems* 3(4):381–388
42. Mitter SK (1996) Successive approximation methods for the solution of optimal control problems. *Automatica* 3:135–149
43. Morris AJ (1996) Artificial neural networks in process engineering. *IEE Proc D* 138(3):256–266
44. Nakamura S (1991) Applied numerical methods with software. Prentice-Hall, Englewood Cliffs, New Jersey
45. Oussar Y, Rivals I, Personnaz L, Dreyfus G (1998) Training wavelet networks for nonlinear dynamic input-output modeling. *Neurocomputing* 20:173–188
46. Oysal Y (2002) Feuro modeling and optimal fuzzy control. PhD thesis, Sakarya University, Sakarya, Turkey
47. Oysal Y, Becerikli Y, Konar AF (2003) Generalized modeling principles of a nonlinear system with a dynamic fuzzy network. *Comput Chem Eng* 27(11):1657–1664
48. Oysal Y, Becerikli Y, Konar AF (2004) Phase portrait modeling of a nonlinear system with a dynamic fuzzy network. *Lecture Notes in Computer Science (LNCS)* (accepted 2004)

49. Palm R, Driankow D, Hellendorn H (1997) Model based fuzzy control. Springer, Berlin Heidelberg New York
50. Parisi R, DiClaudid ED, Orlandi G, Rao BD (1996) A generalized learning paradigm exploiting the structure of feedforward neural networks. *IEEE Trans Neural Networks* 7(6):1450–1460
51. Park J, Sandberg I-W (1991) Universal approximation using radial-basis function networks. *Neural Comput* 3:246–257
52. Passino KM, Yurkovich S (1998) Fuzzy control. Addison-Wesley, Menlo Park, California
53. Pearlmutter B (1989) Learning state space trajectories in recurrent neural networks. *Neural Comput* 1:263–269
54. Pham DT, Liu X (1990) Dynamic system identification using partially recurrent neural networks. *J Syst Eng* 2(2):4–27
55. Pierre DA (1986) Optimization theory with applications. Dover Publication, New York
56. Pineda F (1987) Generalization of backpropagation to recurrent and higher order neural networks. In: Proceedings of the IEEE international joint conference on neural networks (IJCNN'87), San Diego, California, June 1987
57. Ray WH (1981) New approaches to the dynamics of nonlinear systems with implications for process and control system design. In: Seabor E (ed) *Chemical process control II*, pp 245–267
58. Rubio FR, Berenguel M, Camacho EF (1995) Fuzzy logic control for a solar power plant. *IEEE Trans on Fuzzy Syst* 3(4):459–468
59. Rumelhart DE, Hinton GE, Williams RJ (1986) Learning internal representations by back-propagating errors. *Nature* 323:533–536
60. Sanner R, Slotine J-JE (1992) Gaussian networks for direct adaptive control. *IEEE Trans Neural Netw* 13(6):837–863
61. Scales LE (1985) Introduction to non-linear optimization. Springer, Berlin Heidelberg New York
62. Servan-Schreiber D, Cleeremans A, McClellans JL (1988) Encoding sequential structure in simple recurrent networks. Technical report CMU-CS-88-183, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania
63. Sugeno M, Kang GT (1998) Structure identification of fuzzy model. *Fuzzy Sets Syst* 28:15–33
64. Takagi T, Sugeno M (1985) Fuzzy identification of systems and its application to modeling and control. *IEEE Trans on SMC* 15(1):116–132
65. Tan Y, Dang X, Liang F, Su CY (2000) Dynamic wavelet neural network for nonlinear dynamic system identification. In: Proceedings of the IEEE international conference on control applications (CCA 2001), Anchorage, Alaska, September 2000
66. Tanaka K, Sugeno M (1992) Stability analysis and design of fuzzy control systems. *Fuzzy Sets Syst* 45(2):135–156
67. Tanaka K, Sugeno M (1993) Fuzzy stability criterion of a class of nonlinear systems. *Inform Sci* 71(1–2):3–26
68. Tsung F-S (1990) Learning in recurrent finite difference networks. In: Touretzky DS, Elman JL, Sejnowski TJ, Hinton GE (eds) Proceedings of the connectionist models summer school. San Diego, California. Morgan Kaufmann, San Mateo, California
69. Wang LX (1997) A course in fuzzy systems and control. Prentice-Hall, Upper Saddle River, New Jersey
70. Wang LX, Mendel JM (1992) Fuzzy basis function, universal approximation, and orthogonal least squares learning. *IEEE Trans Neural Netw* 3:807–814
71. Wang D, Romagnoli JA, Safavi AA (2000) Wavelet-based adaptive robust M-estimator for nonlinear system identification. *AIChE J* 6(8):1607–1615
72. Zadeh LA, Desoer CA (1963) Linear system theory. McGraw-Hill, New York
73. Zhang Q (1997) Using wavelet network in nonparametric estimation. *IEEE Trans on Neural Netw* 8(2):227–236
74. Zhang Q, Benveniste A (1992) Wavelet networks. *IEEE Trans Neural Netw* 3(6):889–898
75. Zhang J, Walter GG et al (1995) Wavelet neural networks for function learning. *IEEE Trans Signal Process* 43(6):1485–1497