J. Webster (ed.), *Wiley Encyclopedia of Electrical and Electronics Engineering*

NONLINEAR SYSTEM REPRESENTATION

It has been said that whereas linearity is a specification of a field of activity, nonlinearity is a "nonspecification" and its field is unbounded. In nature, nonlinearity is the rule rather than the exception, while linearity is a simplification adopted for analysis. Most practical systems used for control are essentially nonlinear, and in many applications, particular in the area of chaos, it is the nonlinear rather than the linear characteristics that are most used. Signals found in the physical world are also far from conforming to linear models. Indeed, the complex structure of dynamic systems makes it almost impossible to use linear models to represent them accurately. Nonlinear models are designed to provide a better mathematical way to characterize the inherent nonlinearity in real dynamic systems, although we may not be able to take all their physical properties into account. In this article, we will focus on other nonlinear techniques than modeling, which may provide readers with useful perspectives.

For most real-world practical applications, there are advantages to using nonlinear models to characterize the nonlinear relationships. Mathematical models may be expressed in the form of difference or differential equations. Depending on the given engineering problem and the circumstances, one mathematical model may be better suited than another. Figure 1 shows a typical system for processing and evaluating a system model.

In general, nonlinear representations can be classified into three types: (1) system input–output representation, (2) state-space representation, and (3) model-free representation. The first type considers the input–output behavior of a system without considering any internal variations. The second type focuses on both internal and external performance of the system, and the last type focuses on the representation of nonlinear systems that cannot be handled by the other two approaches.

Although the selection of the type of model is often quite subjective, it is usually the result of a compromise between the model selection and familiarity with the model itself. In this regard, a number of nonlinear techniques will be briefed in this article.

Input–Output System Representation

Nonlinear system representation means the characterization of nonlinear systems using nonlinear mathematical models. In fact, nonlinear models may be considered as a tool for explaining the nonlinear behavior patterns in terms of a set of easily understood elements. Figure 2 shows the input–output representation approach for describing a given nonlinear system

$$y(t) = f(u) \tag{1}$$

where $y(t)$ refers to the system output, $u(t)$ refers to the system input, the independent variable t is time, and the f denotes a mathematical relationship describing the nonlinear behavior: the system yields the output $y(t)$ when the system has a input $u(t)$.

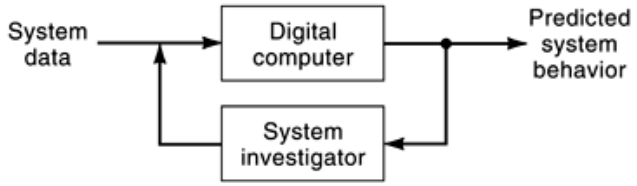

Fig. 1. Model for investigating systems.

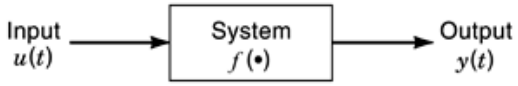

Fig. 2. Model of nonlinear input–output system.

Generally, the nonlinear mapping f is very complicated, and there is no single technique suitable for the analysis of all nonlinear behaviors. In order to appreciate the complexity associated with nonlinear systems, it is best first to review the relative simplicity associated with linear systems. The main reason is that dynamic system analysis relies heavily on linear models, due to their comprehensiveness and the availability of well-developed linear system theories.

A system (1) is called a *linear* system if it satisfies

$$f(au_1 + bu_2) = af(u_1) + bf(u_2) \tag{2}$$

We note two important features. One is that the sum of inputs results in the sum of the responses to the individual inputs, and the other is that a multiple of an input results in the same multiple of the corresponding output. An electric circuit containing a capacitor and a resistor is a common example used for explication. In this article we will mainly focus on the theoretical aspect.

A dynamic system is called *anticipatory* or *noncausal* if its outputs depend on the past, present, and future values of its inputs. A system is called *nonanticipatory* or *causal* if its outputs depend only on the past and present values of its inputs. We say that a system is *time-invariant* if its properties are invariant to a shift of time.

First let us consider the following linear system, which can be described as single-input, single-output, time-invariant, and causal (1):

$$y(t) = \int_{-\infty}^{+\infty} h(s)u(t-s)\,ds \tag{3}$$

where $h(t)$ is called the *impulse response* of the system. Without loss of generality, this system can be assumed to satisfy the following assumptions: (1) $h(t)$ is a real-valued function defined for $t \in (-\infty, \infty)$ and piecewise continuous except possibly at $t = 0$; (2) the input signal is a real-valued function defined for $t \in (-\infty, \infty)$ and piecewise continuous. These conditions imply that the output signal is a continuous, real-valued function defined for $t \in (-\infty, \infty)$. In practical engineering, however, it is often assumed that $h(t)$ is a real-valued function defined for $t \in (0, \infty)$, corresponding to causality. Considering only input signals that are zero prior to $t = 0$, which allows the upper limit in Eq. (3) to be replaced by t, a change of the integration variable shows that Eq.

(3) can be rewritten as

$$y(t) = \int_{-\infty}^{+\infty} h(t-s)u(s)\,ds \tag{4}$$

In this form the one-sidedness assumption on $h(t)$ implies that the upper limit can be lowered to t, while a one-sided assumption on $u(t)$ enables the lower limit to be raised to 0. The representation (3) is usually favored for time-invariant systems.

If the assumption that the system is time-invariant is dropped, the following input–output representation applies. We replace $h(t)$ with a real-valued function $h(t, s)$ defined for $t \in (-\infty, \infty)$, $s \in (-\infty, \infty)$, with $h(t, s) = 0$ if $s > t$, that is,

$$y(t) = \int_{-\infty}^{+\infty} h(t,s)u(s)\,ds \tag{5}$$

It is straightforward to confirm that this represents a linear system in a general sense. In fact, the assumption on $h(t, s)$ implies causality; only the delay-invariance property has been dropped. Typically $h(t, s)$ is allowed to contain impulses for $s = t$, but otherwise is piecewise continuous for $t \geq s \geq 0$. The range of integration in Eq. (5) can be narrowed as discussed before.

Comparison of Eqs. (4) and (5) highlights the fact that time-invariant linear systems are, in fact, special cases of time-varying linear systems. Thus, the impulse function $h(t, s)$ in (5) is called time-invariant if there exists an impulse response $h'(t)$ such that

$$h'(t-s) = h(t,s) \tag{6}$$

An easy way to check for time invariance of $h(t, s)$ is to check the condition $h(0, s - t) = h(t, s)$. If this is satisfied, then setting $h'(t) = h(0, -t)$ verifies Eq. (6).

Nonlinear Differential Algebraic Representation

The specification of the modeling problem for a linear system is simplified by the fact that it is easy to parametrize the response via a defined coordinate system. This fact enables us to reduce the problem of constructing a standard model from an input–output relation to a linear-algebra expression. In the nonlinear case, there is no such global coordinate system. Usually we have to be cautious in defining what we mean by the problem data. We cannot simply assume the system response to be as simple as an infinite sequence of functions or an impulse function.

The first step in constructing a nonlinear model is the development of a differential representation for describing the system input–output behavior. Clearly, there are a number of ways in writing differential equations that describe the behavior of different dynamic systems. No single one of them is preferable in all circumstances. In general, the result required and the familiarity of the investigator with a particular method determine the form of the differential equations.

One particularly convenient method of characterizing the behavior of a nonlinear system is by (2)

$$\dot{y}(t) = f[y(t), u(t), t] \tag{7}$$

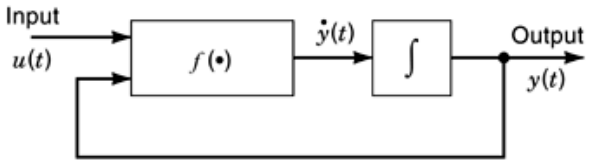

Fig. 3. A nonlinear system model in differential representation.

where $u(t)$ is the system input, $y(t)$ is the system output, and $f(\cdot)$ is an arbitrary nonlinear function. Figure 3 shows the corresponding input–output relationship. There are several reasons for the importance of this differential algebraic form of the system equations. Apart from the notational simplicity, one can deal with all systems by means of a compact notation instead of having to write a system of simultaneous differential equations. Also, this representation is the one that most modern literature in the theory of differential equations makes use of.

It is natural to represent the output in terms of the input as a series expansion

$$y(t) = h_0(t) + \int_0^t h_1(t, s_1)u(s_1)\, ds_1 + \int_0^t \int_0^{s_1} h_2(t, s_1, s_2)u(s_2)u(s_1)\, ds_2\, ds_1 + \cdots \tag{8}$$

where the real-valued function of n variables $h_i(t_1,t_2,\ldots,t_n)$, $i = 0, 1, 2, \ldots$, is equal to zero if any $t_i < 0$, that is to say, the system is causal. Obviously the system is not linear, and it is a time-invariant system if $h_i(t_1,t_2,\ldots,t_n) = h'_i(t_1 - t_2 - \cdots - t_n)$. Formally the expansion (8) is a generalization of the linear-variation-of-constants formula (5). Clearly, this type of modeling problem for nonlinear systems may be expressed as follows: given a sequence of input–output pairs, find a canonical model (8) whose input–output behavior generates the series of impulse functions h_i, $i = 0, 1, 2, \ldots$.

The modeling process is rather straightforward if there are no further hypotheses on the analytic behavior of f in Eq. (7) and there is a suitable definition of the canonical model. In general the problem is unsolvable, but the following theorem (3) provides conditions under which the expansion (8) exists and is unique.

Theorem 1. If the nonlinear relationship f in Eq. (7) is an analytic vector field and Eq. (7) has a solution on $[0, T]$ with $y(0) = h_0(0)$, then the input–output behavior of Eq. (7) has a unique representation expressed by Eq. (8) on $[0, T]$ (3).

Now it is quite clear that the condition of analyticity of the defining vector field is essential. The reason is clear: analyticity forces a certain type of rigidity upon the system, namely, the system behavior is determined by its response in an arbitrarily small open set. Fortunately, it is a property possessed by all systems defined by sets of algebraic equations.

Volterra Representation

In this section, we introduce a special type of nonlinear series, *Volterra series*, which exhibit many important features for nonlinear system modeling, control, and signal processing. Put in simple words, a Volterra series contains symmetrical parameters, *Volterra kernels*, of order n, which are nth-order impulse responses, to

represent nonlinear dynamics. In most cases, we have to estimate the Volterra kernels. This is usually a computationally rather complex procedure and is best performed offline.

Based on a real-valued function of n variables $h_n(t_1, t_2, \ldots, t_n)$ defined for $t_i \in (-\infty, \infty)$, $i = 1, 2, \ldots, n$, and such that $h_n(t_1,t_2,\ldots,t_n) = 0$ if any $t_i < 0$, we consider the following system response:

$$y(t)=\int_0^\infty \cdots \int_0^\infty h_n(s_1, s_2, \ldots, s_n)u(t - s_1)u(t - s_2) \cdots u(t - s_n)\, ds_1\, ds_2 \cdots ds_n \tag{9}$$

Equation (9) is a nonlinear model of degree n, in which application of the input $au(t)$, where a is a scalar, yields the output $a^n\, y(t)$. The impulse function $h_n(t_1, t_2,\ldots,t_n)$ is called the *model kernel*.

If the input signal is one-sided, then all the upper limits can be replaced by t. A change of each variable of integration shows that Eq. (9) can be rewritten in the following form:

$$y(t) = \int_0^\infty \cdots \int_0^\infty h_n(t - s_1, t - s_2, \ldots, t - s_n)u(s_1)u(s_2) \cdots u(s_n)\, ds_1\, ds_2 \cdots ds_n \tag{10}$$

In most science and engineering applications the above model is usually obtained from physical systems that are structured in terms of interconnections of linear subsystems and simple nonlinearity, especially time-invariant linear subsystems and nonlinearities that can be represented in terms of multipliers. Simply by tracing the input signal through the system structure, it is easy to derive the system kernels from the subsystem kernels for the interconnection-structured systems. The model (10) can also arise with a state-equation description of a nonlinear system, which will be discussed in detail later on.

A model described by a finite sum of terms like those in Eq. (10),

$$y(t) = \sum_{n=1}^{N} \int_0^\infty \cdots \int_0^\infty h_n(s_1, s_2, \ldots, s_n)u(t - s_1)u(t - s_2) \cdots u(t - s_n)\, ds_1\, ds_2 \cdots ds_n \tag{11}$$

is called an Nth-order Volterra model (4). It is worth noting that, as special cases, static nonlinear models described by a polynomial or power series in the input,

$$y(t) = \sum_{i=1}^{N} a_i u^i(t) \tag{12}$$

or

$$y(t) = \sum_{i=1}^{\infty} a_i u^i(t) \tag{13}$$

are included in this Volterra model simply taking $h_n(t_1, t_2,\ldots, t_i) = a_i\delta(t_1)\delta(t_2)\cdots\delta(t_i)$.

As a Volterra series is an infinite series, convergence conditions must be considered. Usually these conditions include a bound on the time interval and a bound for $u(t)$ on this interval, but the determination of the bound is often difficult.

With reference to the derivation of the Volterra representation for a nonlinear model, it is reasonable to consider the output $y(t)$ of a nonlinear model at a particular time t as depending on all values of the input at times prior to t, which implies that $y(t)$ depends on $u(t - i)$ for all $i \geq 0$. In other words, the present output depends on all past input values. This viewpoint leads to the following idea. If $u(t - i)$ for all $i \geq 0$ can be described by a set of quantities $u_1(t)$, $u_2(t)$,..., then the output $y(t)$ can be represented as a nonlinear function of these quantities,

$$y(t) = f[u_1(t), u_2(t), \cdots] \tag{14}$$

Suppose that t is fixed, and the input $u(t - i)$ is an element of the Hilbert space $L^2(0, \infty)$ of square-integrable functions, that is,

$$\int_0^\infty u^2(t - i)\, di < \infty \tag{15}$$

Also, suppose that $w_1(t)$, $w_2(t)$,... is an orthonormal basis for this space:

$$\int_0^\infty w_j(t) w_k(t)\, dt = \begin{cases} 1, & j = k \\ 0, & j \neq k \end{cases} \tag{16}$$

The value of the inputs at any time of the past is in the form

$$u(t - i) = \sum_{j=1}^{\infty} u_j(t) w_j(i) \tag{17}$$

where

$$u_j(t) = \int_0^\infty u(t - i) w_j(i)\, di \tag{18}$$

Equation (17) yields a characterization of the past of $u(t)$ in terms of $u_1(t)$, $u_2(t)$,..., regardless of t. Expand the function $f(u_1(t), u_2(t),...)$ into a power series, so that the output at any time t is

$$y(t) = a_0 + \sum_{i,j=1}^{\infty} a_{ij} u_i(t) u_j(t) + \cdots \tag{19}$$

Substituting Eq. (18) into Eq. (19), we obtain

$$\begin{aligned} y(t) &= a_0 + \int_0^\infty \sum_{i=1}^\infty a_i w(s_1) u(t-s_1)\, ds_1 \\ &\quad + \int_0^\infty \int_0^\infty \sum_{i,j=1}^\infty a_{ij} w_i(s_1) w_j(s_2) u(t-s_1) \\ &\quad \times u(t-s_2)\, ds_1\, ds_2 + \cdots \end{aligned} \tag{20}$$

In view of the definition of the kernel, this is the kind of representation that has been discussed.

The generalization of the Laplace transform to functions of variables is of importance for nonlinear system modeling 5. This representation is very handy for characterizing model properties and for describing the model input–output response. Given a time-invariant linear model, the transfer function of the model is the Laplace transform of $h(t)$,

$$H(s) = \int_0^\infty h(t) e^{-st} dt \tag{21}$$

For one-sided input signals, by using the convolution property of the Laplace transform, the input–output response can be expressed as

$$y(t) = \int_0^t h(s) u(t-s)\, ds \tag{22}$$

or

$$Y(s) = H(s) U(s) \tag{23}$$

where $Y(s)$ and $U(s)$ are the transforms of $y(t)$ and $u(t)$, respectively. If a model transfer function is given and the input signal of interest has a simple transform $U(s)$, then the utility of this representation for computing the corresponding output signal is quite clear. Moreover, many system properties can be expressed rather simply as properties of $H(s)$. Consider that an nth-order nonlinear model with one-sided inputs is represented by

$$y(t) = \int_0^t h(s_1, s_2, \ldots, s_n) u(t-s_1) u(t-s_2) \cdots u(t-s_n)\, ds_1\, ds_2 \cdots ds_n \tag{24}$$

Because of the properties of the multivariable Laplace transform, there is no direct way of updating in a form similar to Eq. (23). An indirect approach is therefore employed, writing Eq. (24) as the following pair of

equations:

$$y_n(t_1, t_2, \ldots, t_n) = \int_0^{t_1} \cdots \int_0^{t_n} h(s_1, s_2, \ldots, s_n) u(t_1 - s_1) \times u(t_2 - s_2) \cdots u(t_n - s_n)\, ds_1\, ds_2 \cdots ds_n \tag{25}$$

$$y(t) = y_n(t_1, t_2, \ldots, t_n)|_{t_1 = t_2 = \cdots = t_n} \tag{26}$$

Then, Eq. (25) can be written as a relationship between Laplace transforms in the form

$$Y_n(s_1, s_2, \ldots, s_n) = H(s_1, s_2, \ldots, s_n) U(s_1) U(s_2) \cdots U(s_n) \tag{27}$$

Now let us look at the Laplace transform representation of Volterra models. It basically involves the collection of submodel transfer functions in Eq. (25). Obviously, response calculations are performed by summing the responses of the submodels as calculated individually by association of variables:

$$Y(s_1, s_2, \ldots, s_n) = \sum_{i=1}^{\infty} H(s_1, s_2, \ldots, s_i) U(s_1) U(s_2) \cdots U(s_i) \tag{28}$$

This summation requires consideration of the convergence properties of an infinite series of time functions. Usually convergence is crucially dependent on properties of the input signals, for example, bounds on their amplitude. Using the growing exponential signals as inputs of Volterra models, it is easy to determine the transfer function of the model.

The nonlinear Volterra models discussed above can be developed for discrete-time cases. There are differences, of course, but these mostly are differences in technical details or interpretation of the results. The situation is similar to the linear case, where the continuous- and discrete-time theories look much the same.

Consider the following discrete-time system representation:

$$y(t) = \sum_{i_0=0}^{\infty} \sum_{i_1=0}^{\infty} \cdots \sum_{i_l=0}^{\infty} h(i_0, i_1, \ldots, i_l) \times \prod_{j=0}^{l} u(t - i_j), \qquad t = 0, 1, 2, \ldots \tag{29}$$

The input signal $u(t)$ and the output signal $y(t)$ are real sequences that are assumed to be zero for $t < 0$. The kernel $h(i_0, i_1, \ldots, i_l)$ is real, and assumed to be zero for any $i_0, i_1, \ldots, i_l < 0$. It is straightforward to verify that a system described by Eq. (29) is nth-order, time-invariant, and causal. It is noted that the upper limits on the summations can be lowered to t.

Since, for any t, $y(t)$ is given by a finite summation, problems like continuity and integrability in the continuous-time case do not arise with regard to the representation in Eq. (29). Notice that direct transmission terms are explicitly displayed in Eq. (29), and there is no need to consider impulsive kernels. The familiar sum-over-permutations argument shows that the kernel in Eq. (29) can be replaced by the symmetric kernel $h_{\text{sym}}(i_0, i_1, \ldots, i_n) = 1/n!\ \Sigma_{\pi(\cdot)}\, h(i_{\pi(1)}, i_{\pi(2)}, \ldots, i_{\pi(n)})$ without loss of generality. From the symmetric kernel representation,

a triangular-type kernel can be defined as $h_{\text{tri}}(i_0, i_1, \ldots, i_l) = h_{\text{sym}}(i_0, i_1, \ldots, i_l)\delta(i_1 - i_2, i_2 - i_3, \ldots, i_{l-1} - i_l)$ (6), where the special multivariable step function takes the form

$$\delta(i_1, i_2, \ldots, i_{n-1}) = \begin{cases} 0, & \text{if any } i_j < 0 \\ 1, & i_1 = i_2 = \cdots = i_{n-1} = 0 \\ \vdots & \\ \frac{n!}{m_1! \cdots m_j!}, & i_1 = i_2 = \cdots = i_{m_1 - 1} = 0, \ldots, \\ & i_{j+1} = i_{j+2} = \cdots = i_{j+m_j-1} = 0 \\ \vdots & \\ n!, & i_1, i_2, \ldots, i_{n-1} > 0 \end{cases} \tag{30}$$

It is easy to verify that when $n = 2$, this setup yields consistent results in going from the symmetric kernel to the triangular kernel and vice versa. The higher-degree cases are less easy but still straightforward.

The regular kernel representation is another special form of the system kernel. With the triangular kernel representation

$$y(t) = \sum_{i_0=0}^{\infty} \sum_{i_1=0}^{\infty} \cdots \sum_{i_l=0}^{\infty} h_{\text{tri}}(i_0, i_1, \ldots, i_l) \times \prod_{j=0}^{l} u(t - i_j), \qquad t = 0, 1, 2, \ldots \tag{31}$$

a simple change of variables argument gives

$$\begin{aligned} y(t) \;=\; & \sum_{i_0=0}^{\infty} \sum_{i_1=0}^{\infty} \cdots \sum_{i_l=0}^{\infty} h_{\text{reg}}(i_0, i_1, \ldots, i_l) \\ & \times u(t - i_1 - \cdots - i_l) u(t - i_2 - \cdots - i_l) \cdots u(t - i_l) \end{aligned} \tag{32}$$

where

$$\begin{aligned} h_{\text{reg}}(i_0, i_1, \ldots, i_l) \;&=\; h_{\text{tri}}(i_0 + i_1 + \cdots i_l, i_1 + \cdots i_l, \ldots, i_l) \\ &= h_{\text{sym}}(i_0 + i_1 + \cdots i_l, i_1 + \cdots i_l, \ldots, i_l)\delta(i_0, i_1, \ldots, i_l) \end{aligned} \tag{33}$$

Also it is noticed that the upper limits of the summation in Eqs. (31) and (32) can be replaced by finite quantities.

Although only time-invariant systems are usually considered, a general representation of the form

$$y(t) = \sum_{i_0=0}^{t} \sum_{i_1=0}^{t} \cdots \sum_{i_l=0}^{t} h(t, i_0, i_1, \ldots, i_l) \times \prod_{j=0}^{l} u(i_j) \tag{34}$$

is also very important. It is natural to follow the continuous-time case and call a kernel $h(t, i_0, i_1, \ldots, i_l)$ time-invariant if $h(0, i_0 - t, i_1 - t, \ldots, i_l - t) = h(t, i_0, i_1, \ldots, i_l)$. If this relationship holds, then setting $g(i_0, i_1,$

$\ldots, i_l) = h(0, -i_0, -i_1, \ldots, -i_l)$ yields the representation

$$y(t) = \sum_{i_0=0}^{t} \sum_{i_1=0}^{t} \cdots \sum_{i_l=0}^{t} g(t - i_0, t - i_1, \ldots, t - i_l) \times \prod_{j=0}^{l} u(i_j) \tag{35}$$

which is equivalent to Eq. (34). Actually the above equation can be described as

$$y(t) = \sum_{i_0=0}^{\infty} \sum_{i_1=0}^{\infty} \cdots \sum_{i_l=0}^{\infty} g(i_0, i_1, \ldots, i_l) \times \prod_{j=0}^{l} u(t - i_j), \qquad t = 0, 1, 2, \ldots \tag{36}$$

which is in the same form as Eq. (29).

With these basic representations, the description of Volterra models is simply a matter of finite sums of the terms in Eq. (29). In circuit theory, signal processing, and communication applications, the following two truncated Volterra models are usually utilized:

Quadratic Model.

$$y(t) = \sum_{i=0}^{q} h(i)u(t - i) + \sum_{i=0}^{q} \sum_{j=0}^{q} h(i, j)u(t - i)u(t - j) \tag{37}$$

Cubic Model.

$$\begin{aligned} y(t) = {} & \sum_{i=0}^{q} h(i)u(t - i) + \sum_{i=0}^{q} \sum_{j=0}^{q} h(i, j)u(t - i)u(t - j) \\ & + \sum_{i=0}^{q} \sum_{j=0}^{q} \sum_{k=0}^{q} h(i, j, k)u(t - i)u(t - j)u(t - k) \end{aligned} \tag{38}$$

Of course the convergence issue is important for Volterra models, but the basic approach to convergence in the continuous-time case carries over directly (4).

In the frequency domain, the z-transform representation is used for Volterra systems in just the same way that the Laplace transform is used in the continuous-time case (7). A transfer function for a nth-order discrete-time system [Eq. (29)] is defined as the z-transform of a kernel for the system. That is,

$$H(z_1, z_2, \ldots, z_l) = Z[h(i_0, i_1, \ldots, i_l)] \tag{39}$$

Unfortunately, it seems to be impossible to represent the input–output relationship in Eq. (29) directly in terms of $U(z)$, $Y(z)$, and $H(z_1, z_2, \ldots, z_l)$. The usual alternative is to rewrite Eq. (29) as a pair of equations

$$y_l(t_1, t_2, \ldots, t_l) = \sum_{i_0=0}^{\infty} \sum_{i_1=0}^{\infty} \cdots \sum_{i_l=0}^{\infty} h_{\text{sym}}(i_0, i_1, \ldots, i_l) \times u(t_1 - i_1)u(t_2 - i_2) \cdots u(t_l - i_l) \tag{40}$$

$$y(t) = y_l(t_1, t_2, \ldots, t_l)|_{t_1=t_2=\cdots=t_l=t} \tag{41}$$

Then the first equation has the form

$$Y_n(z_1, z_2, \ldots, z_l) = H_{\text{sym}}(z_1, z_2, \ldots, z_l)U(z_1)U(z_2) \cdots U(z_l) \tag{42}$$

while the second equation is an association of variables that involves contour integrations of the following form:

$$Y(z) = \frac{1}{(2\pi i)^{l-1}} \int_{\Omega_1} \cdots \int_{\Omega_{l-1}} \frac{Y_l(z_1, z_2/z_1, \ldots, z/z_{l-1})}{z_1 z_2 \ldots z_{l-1}} \, dz_1 \, dz_2 \cdots dz_{l-1} \tag{43}$$

As far as the regular transfer function is concerned, the formulas in Eqs. (40) and (41) do not directly apply. However, by suitably restricting the class of input signals, a much more explicit formula can be obtained. By establishing a basic expression for the z-transform of the input–output relation (32), the regular transfer function is cast in the following form:

$$\begin{aligned} H_{\text{reg}}(z_1, z_2, \ldots, z_l) &= Z[h_{\text{reg}}(i_0, i_1, \ldots, i_l)] \\ &= \sum_{i_1=0}^{\infty} \cdots \sum_{i_{l-1}}^{\infty} H_{i_1 i_2 \cdots i_{l-1}}(z_l) z_1^{-i_1} z_2^{-i_2} \cdots z_{l-1}^{-i_{l-1}} \end{aligned} \tag{44}$$

where each $H_{i_1 i_2 \ldots i_{l-1}}(z_l)$ is defined according to

$$H_{i_1 i_2 \cdots i_{l-1}}(z_l) = \sum_{i_l=0}^{\infty} h_{\text{reg}}(i_1, i_2, \ldots, i_l) z_l^{-i_l}, \qquad i_1, i_2, \ldots, i_{l-1} = 0, 1, 2, \ldots \tag{45}$$

State-Space Representation

A state-space representation is usually used for describing physical systems. If the state of a system is known, then any output or quantity of interest with respect to certain performance indices can be achieved. To deter-

mine the state of a system as a function of time, we need a set of equations to relate the inputs state of the system. One approach for obtaining that set of equations is to consider each state variable as an "output," to be determined via an nth-order differential or difference equation. Using the state-space approach, we will be able to write n first-order differential or difference equations for the n state variables of the system. In general they will be coupled equations, that is, they will have to be solved simultaneously. For nonlinear systems, these first-order equations will be nonlinear ones. One advantage of the state-space representation is that once the first-order equations are solved, complete knowledge of the system behavior is obtained. All outputs are algebraic functions of the state variables. No further solution of a differential or difference equation is needed.

The input–output behavior of a nonlinear system can be characterized by first-order differential or difference equations

$$\begin{aligned} \dot{x}(t) &= f[x(t), u(t)] \\ y(t) &= g[x(t), u(t)] \end{aligned} \tag{46}$$

or

$$\begin{aligned} x(k+1) &= f[x(k), u(k)] \\ y(k) &= g[x(k), u(k)] \end{aligned} \tag{47}$$

where $x \in \Re^n$ are the internal states of the system, $u \in \Re^m$ are the inputs, $y \in \Re^q$ are the outputs and $f : \Re^{n+m} \rightarrow \Re^n$, $g : \Re^n \rightarrow \Re^q$. While the inputs and outputs of a system are generally the tangible physical data, it is the state variables that assume the dominant role in this formulation. It is possible that quantities that are not of interest will lead to an unnecessarily complicated problem.

A major difficulty in dealing with nonlinear equations is that the existence and the uniqueness of solutions, even in a local sense, cannot be taken for granted. As a matter of fact, there does not exist any general methodology to determine the nonlinear relations f and g. Instead, various simplified nonlinear models are widely used in practical engineering applications. The so-called bilinear model is among these. We will discuss the bilinear representation in detail later.

Now let us look at the discrete-time nonlinear model (47). It is assumed that the initial state is $x(0) = 0$, and that $f(0, 0) = 0$ and $g(0, 0) = 0$. Then the functions $f(x, u)$ and $g(x, u)$ can be represented using a Taylor series about $x = u = 0$ of order sufficient to permit calculating the polynomial input–output representation to the degree desired:

$$\begin{aligned} x(k+1) &= \sum_{i,j=0}^{N} F_{ij} x^{(i)}(k) u^{(j)}(k) \\ y(k) &= \sum_{i,j=0}^{N} G_{ij} x^{(i)}(k) u^{(j)}(k) \end{aligned} \tag{48}$$

where $x^{(i)} = x \otimes x \otimes \cdots \otimes x$ (i factors) and F_{ij}, G_{ij} are the standard Kronecker products. Just as in the continuous-time case, the crucial requirement is that the kernels through order N corresponding to Eq. (48) should be identical to the kernels through order N corresponding to Eq. (47).

Bilinear Representation

In the class of nonlinear dynamic systems, bilinear models are probably the simplest ones. They take the form (8)

$$\begin{aligned} \dot{x}(t) &= Fx(t) + Gu(t) + \left(\sum_{i=1}^{m} N_i u_i\right) x(t) \\ y(t) &= Cx(t) + Du(t) \end{aligned} \tag{49}$$

or

$$\begin{aligned} x(k-1) &= Fx(k) + Gu(k) + \left(\sum_{i=1}^{m} N_i u_i\right) x(k) \\ y(k) &= Cx(k) + Du(k) \end{aligned} \tag{50}$$

where F, N_i, and C are $n \times n$ real matrices and G, D are $n \times m$ real matrices. Bilinear models are widely used in control systems, signal processing, and communications, because they can be parametrized by matrix operations and lead to linear models of familiar type.

The solution of Eq. (49) for $x(t)$ in terms of a series expansion is given by (9)

$$x(t) = x_l(t) + x_b(t) \tag{51}$$

$$x_l(t) = e^{Ft}x(0) + \int_0^t e^{F(t-s)}Gu(s)\,ds \tag{52}$$

$$\begin{aligned} x_b(t) = \sum_{i=1}^{\infty} \int_0^t \int_0^{t_1} \cdots \int_0^{t_{i-1}} & e^{F(t-t_1)}N[\cdots[e^{F(t_{i-1}-t_i)}N[x_l(t_i)]u(t_i)]\cdots] \\ & \times u(t_1)\,dt_i \cdots dt_1 \end{aligned} \tag{53}$$

where $Nx(t)u(t) = (\Sigma^m{}_{i=1} N_i u_i)x(t)$. The first term $x_l(t)$ in Eq. (51) is the solution, for the state vector $x(t)$, of the linear system described by the equation $\bar{x}(t) = Fx(t) + Gu(t)$. The second term $x_b(t)$ is due to the bilinear term $(\Sigma^m{}_{i=1} N_i u_i)x(t)$ appearing in Eq. (49).

The term $x_l(t)$ may be approximated via orthogonal series as follows:

$$x_l(t) = A_o f_r(t) \tag{54}$$

where $f_r(t) = [f_0(t), f_1(t), \ldots, f_{r-1}(t)]^{\mathrm{T}}$ is the orthogonal basis vector and

$$A_o = [x(0) \vdots Fx(0) \vdots \cdots \vdots F^{k-1}x(0)][e^{\mathrm{T}} \quad e^{\mathrm{T}}P_r \cdots e^{\mathrm{T}}P_r^{k-1}]^{\mathrm{T}}$$
$$+[G \vdots FG \vdots \cdots \vdots F^{k-1}G][UP_r\ UP_r^2 \cdots UP_r^k]^{\mathrm{T}} \tag{55}$$

where e is a constant $r \times 1$ vector, whose form depends on the particular orthogonal series; P_r is the $r \times r$ operational matrix of integration; and U is the $m \times r$ coefficient matrix of the input $u(t)$, defined by

$$u(t) = Uf_r(t) \tag{56}$$

An approximation of Eq. (54) is obtained by keeping the first k power-series terms of e^{Ft} and $e^{F(t-t_1)}$ appearing in $x_l(t)$.

Similarly to $x_l(t)$ in Eq. (54), the second term $x_b(t)$ can be expressed by the following equations:

$$x_b(t) = \sum_{i=1}^{q} A_i f_r(t) \tag{57}$$

where

$$A_i = \sum_{j=1}^{m}[N_jG_{i-1} \vdots FN_jG_{i-1} \vdots \cdots \vdots F^{k-1}N_jG_{i-1}]$$
$$\times[\tilde{U}_jP_r\ \tilde{U}_jP_r^2 \cdots \tilde{U}_jP_r^k]^{\mathrm{T}} \quad \text{for} \quad i = 1, 2, \ldots, q \tag{58}$$

$$f_r(t)u(t)_j^{\mathrm{T}} f_r(t) = \tilde{U}_j f_r(t) \tag{59}$$

Then Eq. (51) can be replaced by an orthogonal expression in the form

$$x(t) = \left(A_o + \sum_{i=1}^{q} A_i\right) f_r(t) \tag{60}$$

This makes the study of controllability and observability of bilinear models much easier via orthogonal functions.

Narma Representation

In this section, the *nonlinear autoregressive moving-average* (*NARMA*) model is discussed. Detailed nonlinear system modeling using this model can found in Refs. (10) and (11).

Now let us consider the following discrete-time system:

$$\begin{aligned} x(t+1) &= f[x(t), u(t)] \\ y(t) &= g[x(t), u(t)] \end{aligned} \tag{61}$$

where $y(t) \in Y$, the output set of dimension q; $x(t+1), x(t) \in X$, the state set of dimension n; $u(t) \in U$, the input set of dimension m; $t \in Z$, the set of integers; $f : Z \times X \times U \to X$ is the one-step-ahead state transition function; and $g : Z \times X \times U \to Y$ is the output function. Assuming that the system is at the zero equilibrium point at time $t = 1$, the zero-state response for an input sequence of length t is given by

$$y(t) = h^l[u(1), u(2), \ldots, u(t)], \tag{62}$$

where $h^t : U^t \to Y$. The function h^t is a different function for every $t = 1, 2, \ldots$, because the domain of definition U^t is different. Assume that the response functions h^t are continuously differentiable functions, and let the vector $u^t(t) = [u(t), u(t-1), \ldots, u(1)]^{\mathrm{T}}$; then

$$\begin{aligned} y(t) &= h^t[u(t), u^{t-1}] \\ y(t+1) &= h^{t+1}[u(t+1), u^t] \\ &\vdots \\ y(t+k-1) &= h^{t+k-1}[u(t+k-1), u^{t+k-2}] \end{aligned} \tag{63}$$

Let $y^t{}_k(t) = [y(t), y(t-1), \ldots, y(t+k-1)]^{\mathrm{T}}$ and $u^t{}_k(t) = [u(t+k-1), u(t+k-2), \ldots, u(t)]^{\mathrm{T}}$. Then Eq. (63) can be rewritten as

$$y_k(t) = H_k^t[u_k^t, u^{t-1}] \tag{64}$$

where $y_k(t) \in Y_k$, $u^t{}_k \in U^k$, $u^{t-1} \in U^{t-1}$. The function $H^t{}_k : U^k \times U^t \to Y^k$ can be defined for any $k = 1, 2, \ldots, t = 1, 2, \ldots$, when the response function h^t is known for $t = 1, 2, \ldots$. The function $H^t{}_k$ is continuously differentiable, since it is the Cartesian product of k response functions of continuously differentiable. For simplicity, we let $z = u^{t+1}{}_k \in U^k$, $x = u^t \in U^t$. The assumptions on the function $H^t{}_k$ are the following:

Assumption 1. $\max[\operatorname{rank} dH^t{}_k(z,x)/dx] = n$ for any $z = u_k{}^{t+1} \in U^k$, $x = u^t \in U^t$ and any t, $k = 1, 2, \ldots$.

Assumption 2. $\operatorname{rank} dH^t{}_k(0,0)/dx = n$ for some t and some k.

We can prove the following theorem.

Theorem 2. For any nonlinear system satisfying Assumptions 1 and 2, there exists a number l and a continuous function $\Phi : y_l \times u_l \to y$ such that the recursive input–output representation given as

$$\begin{aligned} y(k+1) = \Phi[&y(k), y(k-1), \ldots, y(k-l+1), \\ &\times u(k), u(k-1), \ldots, u(k-l+1)] \end{aligned} \tag{65}$$

has the same input–output behavior as the original system in a restricted operating region and around the zero equilibrium point.

Recall the autoregressive moving-average (*ARMA*) model in the linear case,

$$y(k+1) = a_k y(k) + a_{k-1} y(k-1) + \cdots + a_{k-l+1} y(k-l+1) + b_k u(k) + b_{k-1} u(k-1) + \cdots + b_{k-l+1} u(k-l+1)] \quad (66)$$

The model (65) is called the NARMA model (10) by extension. The NARMA model can be further extended to a nonlinear stochastic system. Let the estimate $\hat{y}(t)$ denote the prediction of $y(t)$ that is in the form

$$\hat{y} = E[y(t) \mid y^{t-1}, u^t] = f(y^{t-1}, u^t) \quad (67)$$

The prediction error vector $e(t)$ can be calculated as $e(t) = y(t) - \hat{y}(t)$; it is a vector random variable and shows how much the actual output at time t differs from the predicted one. Then the stochastic NARMA model can be expressed as (12)

$$y(t+k) = \Phi[y(t+k-1), \ldots, y(t);\ u(t+k), \ldots, u(t);\ \times e(t+k-1), \ldots, e(t)] + e(t+k) \quad (68)$$

Fuzzy-Logic Nonlinear Representation

The real physical world is very complex, so that problems may always arise from many uncertain factors. The uncertainty in all engineering problems motivates us to establish appropriate methods or models to express the uncertainty. Fuzzy logic is a mathematical approach developed by Zadeh (13) to tackle this issue. Because of their outstanding ability to relate fuzzy sets with humanistic variables, fuzzy-logic nonlinear models have recently been widely used in the areas of control, signal processing, and communication applications. They have demonstrated their versatility in representing complex nonlinear physical systems using linguistic rules of the IF–THEN form. For example, Fig. 4 shows the nonlinear relation between the probability of a machine fault and the vibration magnitude. The mapping shown in Fig. 4 can be best described by fuzzy logic.

A fuzzy-logic model consists of an identification algorithm that adjusts the parameters. The model is designed from certain IF–THEN rules using fuzzy-logic principles. In general, there are two strategies for combining numerical and linguistic information using adaptive fuzzy models. In the first, one uses linguistic information to construct an initial fuzzy-logic model. The parameters of that model are then adjusted in accordance with numerical information. The final fuzzy-logic model can then be obtained by combining numerical and linguistic information. In the second strategy, two separate fuzzy-logic models are initially constructed from numerical information and linguistic information. These models are then averaged to obtain the final fuzzy-logic model.

The most popular fuzzy logic models may be classified into three types: the pure fuzzy-logic model, Takagi and Sugeno's fuzzy model, and fuzzy-logic models with fuzzifier and defuzzifier (14,15).

Pure Fuzzy-Logic Model. Basically, this first type consists of a fuzzy rule base and a fuzzy inference engine. The rule base is a collection of fuzzy IF–THEN rules, and the inference engine is based on fuzzy-logic principles. The IF–THEN rules are to determine a mapping from fuzzy sets in the input universe to fuzzy sets

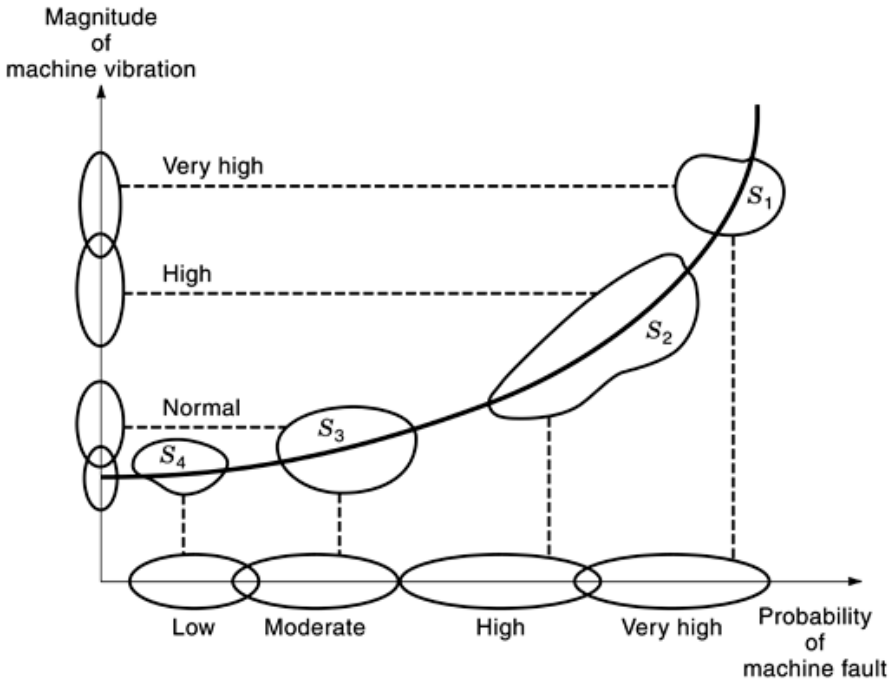

Fig. 4. A fuzzy nonlinear relation mapping from an input space to an output space.

in the output. They are of the following form:

$$R^{(l)} : \text{IF } u_1 \text{ is } F_1^l \text{ and } \cdots \text{ and } u_n \text{ is } F_n^l, \text{ THEN } y \text{ is } G^l \tag{69}$$

where ${F^l}_i$ and G^l are fuzzy sets, and $u = [u_1, u_2, \ldots, u_n]^{\mathrm{T}} \subset U$ and $y \subset V$ are input and output linguistic variables, respectively. The major advantage of fuzzy logic is its remarkable ability to incorporate common sense, as reflected in decisions, into numerical information. Each fuzzy IF–THEN rule of Eq. (69) defines a fuzzy set ${F^l}_1 \times {F^l}_2 \times \cdots \times {F^l}_n \to G^l$ in the product space $U \times V$.

The most commonly used fuzzy-logic principle is the so-called sup–star composition. Let A be the input to the pure fuzzy-logic model. Then the output determined by each fuzzy IF–THEN rule of Eq. (69) is a fuzzy set $A \circ (R^{(l)})$ in V whose membership function is

$$\mu_{A \circ R^{(l)}}(y) = \sup_{u \in U} [\mu_A(u) * \mu_{F_1^l \times \cdots \times F_n^l \to G^l}(u, y)] \tag{70}$$

where $*$ is an operator such as minimization or multiplication, and μ_A is the membership function of the fuzzy set A. The output of the pure fuzzy-logic system is a fuzzy set $A \circ (R^{(1)}, R^{(2)}, \ldots, R^{(M)})$ in V that combines the M fuzzy sets in accordance with Eq. (70):

$$\mu_{A \circ (R^{(1)}, R^{(2)}, \ldots, R^{(M)})}(y) = \mu_{A \circ R^{(1)}}(y) \dot{+} \mu_{A \circ R^{(2)}}(y) \dot{+} \cdots \mu_{A \circ R^{(M)}}(y) \tag{71}$$

where the operator $\dot{+}$ is maximization, that is, $u \dot{+} y = u + y - uy$. The model becomes a fuzzy dynamic model if there is feedback in the fuzzy-logic model.

Takagi and Sugeno's Fuzzy Model. The pure fuzzy-logic model constitutes the essential part of all fuzzy-logic models. By itself, however, it has the disadvantage that its inputs and outputs are fuzzy sets, whereas in most physical systems the inputs and outputs of a system are real-valued variables. In order to overcome this shortcoming, Takagi and Sugeno designed another fuzzy-logic model of which inputs and outputs are real-valued variables.

In this model (16), the following fuzzy IF–THEN rules are proposed:

$$L^{(l)}: \text{IF } u_1 \text{ is } F_1^l \text{ and } \cdots \text{ and } u_n \text{ is } F_n^l, \text{ THEN } y^l = c_0^l + c_1^l u_1 + \cdots + c_n^l u_n \quad (72)$$

where $F^l{}_i$ are fuzzy sets, $c^l{}_i$ are real-valued parameters, y^l is the model output due to the rule L^l, and $l = 1, 2, \ldots, M$. That is, they considered rules of which the IF part is fuzzy, but the THEN part is crisp. Also, the output is a linear combination of input variables. For a real-valued input vector $u = [u_1, u_2, \ldots, u_n]^{\mathrm{T}}$, the output $y(u)$ is a weighted average of the y^l's:

$$y(u) = \frac{\sum_{l=1}^{M} w^l y^l}{\sum_{l=1}^{M} w^l} \quad (73)$$

where the weight w^l represents the overall truth value of the premise of the rule L^l for input and can be determined by

$$w^l = \prod_{i=1}^{n} \mu_{F_i^l}(u_i) \quad (74)$$

where $\mu_{F_i^l}$ is the member function of the fuzzy set $F_i{}^l$. The major advantage of this model lies in the fact that it provides a compact system equation (73). Hence, parameter estimation and order determination methods can be developed to estimate the parameters $c^l{}_i$ and the order M. A shortcoming of this model is, however, that the THEN part is not fuzzy, so that it cannot incorporate fuzzy rules from common sense and human decisions.

Fuzzy-Logic Models with Fuzzifier and Defuzzifier. Models of this type (17) add a fuzzifier to the input and a defuzzifier to the output of the pure fuzzy-logic model. The fuzzifier maps crisp points in U to fuzzy sets in U, while the defuzzifier maps fuzzy sets in V to crisp points in V. The fuzzy rule base and fuzzy inference engine are the same as those described in the pure fuzzy-logic model.

Evidently the fuzzy-logic model with fuzzifier and defuzzifier has certain advantages over other nonlinear modeling techniques. First, it is suitable for engineering systems in that its input and outputs are real-valued variables. Second, it is capable of incorporating fuzzy IF–THEN rules from common sense and human decisions. Third, there is much freedom in the selection of fuzzifier, fuzzy inference engine, and defuzzifier, in order to obtain the most appropriate fuzzy-logic model for a given problem. Finally, we are able to design different identification algorithms for this fuzzy-logic model, so as to integrate given numerical and human information.

The fuzzifier that performs the mapping from a crisp point into a fuzzy set A in U can be a *singleton* or a *nonsingleton* fuzzifier. Singleton fuzzifiers have mainly been used, but nonsingleton fuzzifiers can be useful when the inputs are noise-corrupted.

There are at least three possible choices for the defuzzifer that performs a mapping from fuzzy sets in V into a crisp point $y \in V$:

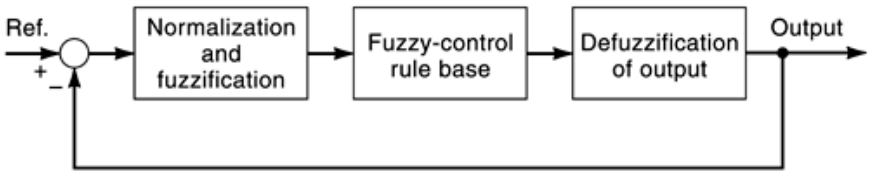

Fig. 5. A typical fuzzy control system.

(1) The *maximum defuzzifier*, defined as

$$y = \arg\sup_{y \in V} \mu_B(y) \tag{75}$$

where B is a fuzzy set with support y.

(2) The *center average defuzzifier*, defined as

$$y = \frac{\sum_{l=1}^{M} y^l \mu_{B^l}(y^l)}{\sum_{l=1}^{M} \mu_{B^l}(y^l)} \tag{76}$$

where y^l is the center of the fuzzy set G^l, that is, the point in V at which $\mu_{G^l}(y)$ achieves its maximum value.

(3) The *modified center average defuzzifier*, which is given as

$$y = \frac{\sum_{l=1}^{M} y^l \left[\mu_{B^l}(y^l)/\delta^l\right]}{\sum_{l=1}^{M} \mu_{B^l}(y^l)/\delta^l} \tag{77}$$

where δ^l is a parameter characterizing the shape of $\mu_{G^l}(y)$ such that the narrower the shape of $\mu_{G^l}(y)$, the smaller is δ^l. For example, if $\mu_{G^l}(y) = \exp[-(y - y^l)/\delta^l]^2$, then δ^l is such a parameter.

Fuzzy Control Summarized. For nonlinear system modeling, fuzzy logic can be viewed as a versatile tool to perform nonlinear mapping. But in such an input–output nonlinear system model, the input signals should be representative and sufficiently rich. In most cases, the fuzzy nonlinear model is based on inputs such as the rate of change of error, and the fuzzy model may be in continuous time or discrete time.

Figure 5 shows a typical fuzzy control system, in which input values are normalized and fuzzified, the fuzzy-model rule base is the control rule base used to produce a fuzzy region, and a defuzzification process is used to find the expected output value. A typical fuzzy learning control-system design is that proposed in 1993 by Harris et al. (18). Wang and Vachtsevanos (19) detailed an indirect fuzzy-control approach in 1992. In their early research, a separate fuzzy model was developed. A well-designed procedure was then used to calculate the control signal. More detailed discussion of control of nonlinear systems using fuzzy logic can be found in Refs. 18,19,20.

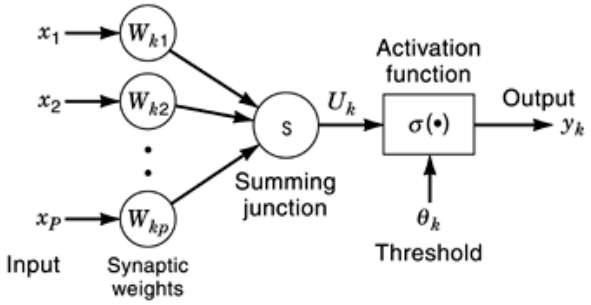

Fig. 6. A neuron with p inputs and a nonlinear activation function σ.

Nonlinear Representation Using Neural Networks

Conventional model-based theoretical methods have dominated nonlinear-representation research over the last few decades. These methods depend on a mathematical characterization of the monitored system. The main disadvantage of such approaches is that they are very sensitive to the selection of model type, modeling errors, parameter variations, and measurement noise. The success of model-based representation approaches is heavily dependent upon the quality of the models as well. For most practical nonlinear physical systems, it is often very difficult, if not impossible, to describe them by sufficiently simplified analytical models. In view of these, there is great interest in developing a robust and less model-dependent methodology for representing a complex nonlinear dynamic system. To avoid the difficulties experienced in the classical nonlinear system modeling, neural-network-based nonlinear system modeling methods appeared to be an appealing alternative. The rationale behind this approach lies in the fact that a multilayer neural network (Fig. 6) with an appropriate nonlinear activation function can approximate any nonlinear relationship.

In using neural networks for nonlinear system modeling, their nonlinear functional approximation capability can be enhanced by using higher-order architectures. In Refs. (32) and (33) strong theoretical arguments were introduced for the functional-approximation capability of the random-vector version of functional-link nets (*RVFL*). (See Fig. 7.) The distinctive aspects of this network are that the parameters of the hidden layer, such as the weights and the thresholds, are selected randomly and independently in advance. The parameters of the output layer are learned using simple quadratic optimization, whereas under conventional approaches all parameters need to be learned using complicated nonquadratic optimization.

Two types of RVFL have been proved to be universal approximators. One [Fig. 7(a)] can be formulated as $f_{\omega_n}(x) = \sum_{k=1}^{n} a_k g(w_k \cdot x + b_k$, where $\omega_n = (n, a_1, a_2, \ldots a_n, b_1, b_2, \ldots, b_n, w_1, w_2, \ldots, w_n)$ is the whole parameter vector of the net, $f(x) \in C(I^d)$, and $w_k \cdot x$ is the inner product of the vectors w_k and x. The random part of ω_n is denoted as $\lambda_n = (b_1, b_2, \ldots, b_n, w_1, w_2, \ldots, w_n)$. Each λ_n is selected uniformly in a specified probabilistic space. The second type of RVFL [Fig. 7(b)] is formulated as $f_{\omega_n}(x) = \sum_{k=1}^{n} a_k \prod_{i=1}^{d} g(w_{ki} x_i + b_{ki}$, where $x = (x_1, x_2, \cdots x_d)$, $w_k = (w_{k1}, w_{k2}, \cdots w_{kd})$, and $b_k = (b_{k1}, b_{k2}, \cdots b_{kd})$. The parameters w_k, b_k are selected uniformly in a specified probabilistic space, and a_k are learned from the information of $f(x)$. The parameter a_k represents a dth-order weight from the kth group in the hidden layer to the output unit. This is an interesting structure in that it is also a partially connected higher-order neural network. Apart from the network architecture, the self-learning ability is another appealing property, enabling the neural network to extract the system features through a learning process. This provides great flexibility and convenience for modeling nonlinear systems.

Many types of neural networks have been developed for tackling different problems, but two types have received the most attention in recent years (22,23): (1) multilayer feedforward neural networks and (2) recurrent

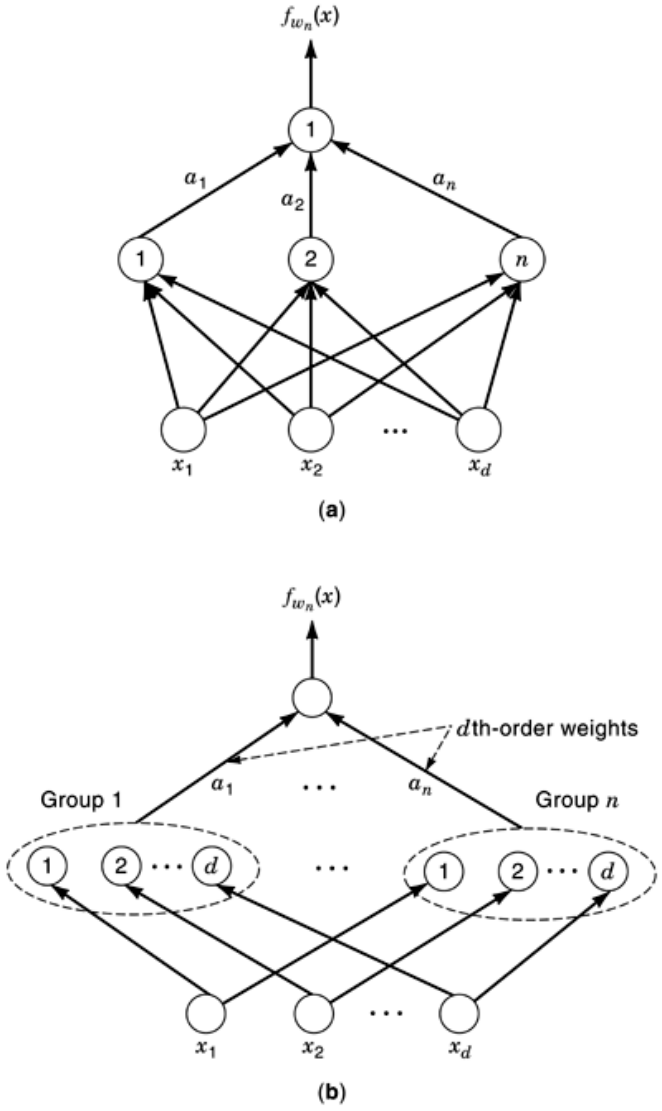

Fig. 7. (a) The structure of the an RVFL network with weights a_k, $1 \leq k \leq n$, that are first-order and adjustable. The weights and the thresholds of the hidden units are distributed uniformly in the selection spaces and determined in advance. (b) The structure of the RVFL of the second type. This network is an extremely sparsely connected higher-order network. The weights a_k, $1 \leq k \leq n$, are dth-order and adjustable. The weights and the thresholds of the hidden units are distributed uniformly in the selection spaces and determined in advance.

networks. Multilayer feedforward networks have proved extremely successful in pattern recognition problems, and recurrent networks for dynamical system modeling and time-series forecasting.

A multilayer network is a network of neurons organized in the form of layers. Figure 8 shows a typical form of a multilayer network, in which an input layer of source nodes projects onto the hidden layer composed of hidden neurons. The output of the hidden neurons then projects onto an output layer. In general, one hidden layer is adequate to handle most engineering problems. The nonlinear activation function of the hidden

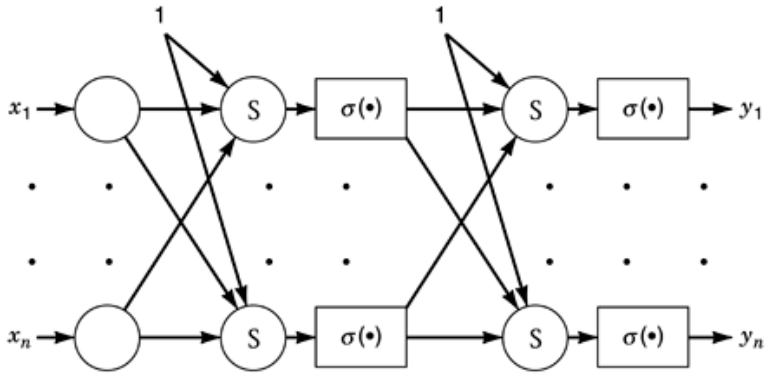

Fig. 8. A simple multilayer feedforward neural network.

neurons, which is generally sigmoidal, is chosen to intervene between the external input and the network output.

For hidden neurons to be useful in modeling nonlinear systems, they must be sufficiently numerous. Though there has been much research on determining the optimal number of hidden neurons, there is no straightforward rule for doing so. When the number of hidden neurons reaches a certain threshold, the overall performance will not be significantly affected by small further increases. In this respect, the design criterion is rather loose.

The source nodes of the network supply corresponding elements of the activation pattern (input vector), which constitute the input signals applied to the neurons in the hidden layer. The set of output signals of the neurons in the output layer of the network constitutes the overall response of the network to the activation pattern supplied by the source nodes at the input layer.

A neural network is said to be *fully connected* when every node in each layer of the network is connected to every other node in the adjacent forward layer. We say that the network is *partially connected* if some of the links are missing. Evidently, fully connected networks are relatively complex, but they are usually capable of much better functional approximation. A form of partially connected multilayer network of particular interest is the *locally connected network*. In practice, the specialized structure built into the design of a connected network reflects prior information about the characteristics of the activation pattern being classified.

As multilayer feedforward neural networks have become generally recognized as a suitable architecture for representing unknown nonlinearities in dynamic systems, numerous algorithms for training the networks on the basis of observable input–output information have been developed by many researchers. In this article, a training algorithm for neural networks called the cumulant-based weight-decoupled extended Kalman filter (*CWDEKF*) is described (24). Third-order cumulants are employed to define output errors for the network training. By this means, Gaussian disturbances or non-Gaussian noises with symmetric probability density function among the output signals can be rejected in the cumulant domain. Thus we can obtain clean neural network output information for nonlinear mapping implementation. The weight-decoupled extended Kalman filter training algorithm is applied because it provides faster convergence (learning rate) then other training algorithms. This feature is very important for neural-network-based analysis of nonlinear dynamic systems.

To ease the mathematical burden, we first present a summary of the notation used in the network learning algorithm.

Notation.

- The index i refers to different layers in the network, where $1 \le i \le M$, and M is the total number of layers (including the hidden and output layers) in the network.
- The index j refers to different neurons in the ith layer, where $1 \le j \le n_i$, and n_i is the neuron number of the ith layer.
- The index s refers to different neurons in the $(i-1)$th layer, where $1 \le s \le n_{i-1} + 1$.
- The index v refers to different neurons in the output layer, where $1 \le v \le n_M$.
- The iteration index k refers to the kth training pattern (example) presented to the network.
- The symbol $w^i_{js}(k)$ denotes the synaptic weight connecting the output of neurons in the $(i-1)$th layer to the input of neuron j in the ith layer at iteration k.
- The learning-rate parameter of the weight $w^i_{js}(k)$ at iteration k with respect to the vth output error is denoted by $\eta^i_{jsv}(k)$.
- The symbol $e_v(k)$ refers to the error signal between the target output and the actual output at the output of neuron v in the output layer at iteration k.
- The symbol $h^i_{jsv}(k)$ denotes the derivative of the output error $e_v(k)$ at iteration k with respect to the weight $w^i_{js}(k-1)$ at iteration $k-1$.
- The symbol $p^i_{js}(k)$ denotes the variance of the estimated weight $w^i_{js}(k)$ at iteration k.
- The symbol $a_v(k)$ refers to the central adjustment parameters for the output of neuron v in the output layer at iteration k.
- Suppose the network output is corrupted by a Gaussian noise $\{n_v(k)\}$ at iteration k. Then the symbol $r_v(k)$ denotes the variance of $\{n_v(k)\}$.
- The symbol $y(k)$ refers to the target output of the network, $N(w(k-1), u(k))$ refers to the actual output of the network, its weight matrix is $w(k-1)$ at iteration $k-1$, and the network input at iteration k is $u(k)$.

The training mechanism can be described by the following:

$$w^i_{js}(k) = w^i_{js}(k-1) + \sum_{v=1}^{n_M} \eta^i_{jsv}(k) h^i_{jsv}(k) e_v(k) \tag{78}$$

$$\eta^i_{jsv}(k) = p^i_{js}(k-1) a_v(k) \tag{79}$$

$$p^i_{js}(k) = \left(1 - \sum_{v=1}^{n_M} \eta^i_{jsv}(k) \left[h^i_{jsv}(k)\right]^2\right) p^i_{js}(k-1) \tag{80}$$

$$a_v(k) = \frac{1}{r_v(k) + \sum_{i=1}^{M} \sum_{j=1}^{n_i} \sum_{s=1}^{n_{i-1}+1} \left[h^i_{jsv}(k)\right]^2 p^i_{js}(k-1)} \tag{81}$$

$$e_v(k) = \left[\text{Cum}_3[y(k) - N(w(k-1), u(k))]\right]^2 \tag{82}$$

$$h^i_{jsv}(k) = \frac{\partial e_v(k)}{\partial w^i_{js}(k-1)} \tag{83}$$

where Cum_3 [·] denotes the third-order cumulant operation

$$\text{Cum}_3[f] = m_3[f] - 3m_1[f]m_2[f] + 2(m_1[f])^3 \tag{84}$$

with

$$\begin{aligned} m_3[f] &= E[f(k)f(k+m)f(k+n)] \approx \frac{1}{\text{pp}}\sum_{\text{pp}} f(k)f(k+m)f(k+n) \\ m_2[f] &= E[f(k)f(k+m)] \approx \frac{1}{\text{pp}}\sum_{\text{pp}} f(k)f(k+m) \\ m_1[f] &= E[f(k)] \approx \frac{1}{\text{pp}}\sum_{\text{pp}} f(k) \end{aligned} \tag{85}$$

where pp is the sample number.

A number of network initialization procedures have been developed for general feedforward networks (25,26). These algorithms are based on linear-algebraic methods to determine the optimal initial weights. With the optimal initial weights, the initial network error is much smaller, thus speeding up the overall training procedure (25,26). In this article, a conventional randomized-weight initialization procedure is presented:

(1) Weights are initialized as random numbers with normal distribution, typically in the range of ± 0.1.
(2) We initialize the matrices $P_i(k)$:$\{p^i_{js}(k)\}$ and $R(k)$:$\{r_v(k)\}$ as $P_i(0) = 100.0I$ and $R(0) = I$, where I refers to the unit matrix.

Thereafter, the CWDEKF algorithm can be applied to train the neural network on which the nonlinear representation is based.

A *recurrent* neural network differs from a multilayer feedforward neural network in that it has at least one feedback loop, which represents the dynamical characteristics of the network. For example, a recurrent network may consist of a single layer of neurons with each neuron feeding its output signals back to the inputs of all the other neurons. In this structure, there are no *self-feedback* loops in the network. Self-feedback refers to a situation where the output of a neuron is fed back to its own input. The feedback connections originate from the hidden neurons as well as the output neurons. The presence of feedback loops has a profound influence on the learning capability of the network and on its dynamical performance. Moreover, the feedback loops involve the use of special branches composed of unit-delay elements, which result in nonlinear dynamical behavior by

virtue of the nonlinear nature of the neurons. Nonlinear dynamics plays a key role in the storage function of a recurrent network.

The Hopfield network is a typical recurrent network that is well known for its capability of storing information in a dynamically stable configuration. It was Hopfield's paper (27) in 1982, elaborating the remarkable physical capacity for storing information in a dynamically stable network, that sparked off the research on neural networks on the eighties. One of the most fascinating findings of his paper is the realization of the associative memory properties. This has now become immensely useful for pattern recognition.

Physically, the Hopfield network operates in an unsupervised fashion. Thus, it may be used as a content-addressable memory or as a computer for solving optimization problems of a combinatorial kind. The classical traveling-salesman problem is a typical example. Koch applied Hopfield networks to the problem of vision. There has been other work on depth computation and on reconstructing and smoothing images.

In tackling a combinatorial optimization problem we are facing a discrete system that has an extremely large but finite number of possible solutions. The task is to find one of the optimal solutions through minimizing a cost function, which provides a measure of system performance. The Hopfield network requires time to converge to an equilibrium condition. The time required depends on the problem size and the possible stability problem. Hence it is never used online unless special-purpose hardware is available for its implementation.

The operational procedure for the Hopfield network may be summarized as follows:

(1) Storage (Learning) Let $\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_p$ denote a known set of N-dimensional memories. Construct the network by using the outer-product rule to compute the synaptic weights of the network as

$$w_{ji} = \begin{cases} \frac{1}{N} \sum_{\mu=1}^{p} \varepsilon_{\mu,j} \varepsilon_{\mu,i}, & j \neq i \\ 0, & j = i \end{cases} \tag{86}$$

where w_{ji} is the synaptic weight from neuron i to neuron j. The elements of the vector ε_μ equal ± 1. Once they are determined, the synaptic weights are kept fixed.

(2) Initialization Let X denote an unknown N-dimensional input vector presented to the network. The algorithm is initialized by setting

$$s_j(0) = x_j, \qquad j = 1, 2, \ldots, N \tag{87}$$

where $s_j(0)$ is the state of neuron j at time $n = 0$, and x_j is the jth element of the vector X.

(3) Iteration until Convergence Update the elements of the state vector $s_j(n)$ asynchronously (i.e., randomly and one at a time) according to the rule

$$s_j(n+1) = \operatorname{sgn}\left(\sum_{i=1}^{N} w_{ji} s_i(n) \right) \tag{88}$$

Repeat until the state vector s remains unchanged.

(4) Outputting Let s_f denote the fixed point (stable state) computed at the end of step 3. The resulting output vector y of the network is

$$y = s_f \tag{89}$$

It is clear that a neural network is a massively parallel-distributed network that has a natural capacity for storing experimental knowledge and making it available for use.

The primary characteristics of knowledge representation are twofold: (1) what information is actually made explicit, and (2) how the information is physically encoded. In real-world applications of intelligent machines, neural networks represent a special and versatile class of modeling techniques that are significantly different from conventional mathematical models. Neural networks also offer a convenient and reliable approach for the modeling of highly nonlinear multiinput multioutput systems.

Model-Free Representation

The traditional qualitative modeling techniques discussed above focus on means for abstracting the value spaces of variables that are used to represent system input–output relationships as well as system states and for simplifying the constraints that hold among the values of those variables. It is obvious that no general nonlinear representations exist for modeling an arbitrary nonlinear system. The approximation accuracy of a nonlinear model depends on what physical system is modeled and what type of model is used. In some cases, where all possible nonlinear models are found to be unsuitable, model-free representations for nonlinear systems will be a better alternative. Rather than a model-based methodology, a flexible model-free technique based on signal processing provides a set of enhanced, sensitive, and definitive characterizations of nonlinear systems. Higher-order statistics have been found to be useful in such analyses (28).

Given an arbitrary real, stationary random signal $\{y(t)\}$ at the discrete times $t = 0, \pm 1, \pm 2, \ldots$, its second-order moment (*SOM*) $M^y{}_2\,(m)$ and third-order moment (*TOM*) $M^y{}_3\,(m, n)$ provide a measure of how the sequence is correlated with itself at different time points:

$$M_2^y(m) = E[y(t)y(t+m)] \tag{90}$$

$$M_3^y(m) = E[y(t)y(t+m)y(t+n)] \tag{91}$$

where m is the time lag; $m, n = 0, \pm 1, \pm 2, \ldots$; and $E[\cdot]$ denotes statistical expectation.

Third-Order Cumulant in Time Domain

For the random signal $\{y(t)\}$, its third-order cumulant (*TOC*) $C^y{}_3\,(m, n)$ is given by

$$\begin{aligned} C_3^y(m,n) &= E([y(t) - E[y(t)]][y(t+m) - E[y(t+m)]][y(t+n) - E[y(t+n)]]] \\ &= M_3^y(m,n) - M_2^y(0)[M_2^y(m) + M_2^y(n) + M_2^y(n-m)] + 2[M_2^y(0)]^3 \end{aligned} \tag{92}$$

Note that the SOM $M^y{}_2\,(m)$ in Eq. (90) is a symmetric function about $m = 0$, that is, $M^y{}_2\,(-m) = M^y{}_2\,(m)$. Hence, $M^y{}_2(m)$ is a zero-phase function, which means that all phase information about $y(t)$ is lost in $M^y{}_2(m)$. With regard to the TOC $C^y{}_3\,(m, n)$, important symmetry conditions follow from the properties of SOM and

TOM and Eq. (92):

$$\begin{aligned} C_3^y(m,n) &= C_3^y(n,m) = C_3^y(-n,m-n) = C_3^y(m-n,-n) \\ &= C_3^y(n-m,-m) = C_3^y(m,n-m) \end{aligned} \tag{93}$$

Thus $C^y{}_3\ (m,\ n)$ is a phase-bearing function, and knowing the TOC in any of the six sectors delimited by Eq. (93) would enable us to find it everywhere.

If $\{y(t)\}$ is corrupted by an independent Gaussian measurement noise $\{v(t)\}$, then the SOM, TOM, and TOC of the noisy random sequence $\{x(t) = y(t)+v(t)\}$ are found to be

$$M_2^x(m) \neq M_2^y(m) \tag{94}$$

$$M_3^x(m,n) \neq M_3^y(m,n) \tag{95}$$

$$C_3^x(m,n) = C_3^y(m,n) \tag{96}$$

which means that the TOC is blind to any kind of a Gaussian process, whereas the SOM and TOM are not.

The TOC, on the other hand, can be used to deal with quadratic phase-coupling phenomena. Suppose

$$y(t) = y_1(t) + y_2(t) \tag{97}$$

where $y_1(t)$ is the phase-decoupled component and $y_2\ (t)$ the phase-coupled component. Then only the second component appears in the TOC of $y(t)$, that is,

$$C_3^y(m,n) = f(y_2(t)) \tag{98}$$

where f stands for a nonlinear relationship between the TOC and the phase-coupled component. The fact that only phase-coupled components contribute to the TOC of a random process is what makes the TOC a useful tool for detecting quadratic phase coupling and discriminating phase-coupling components from those that are not.

Bispectrum in the Frequency Domain. According to its definition, the bispectrum $C^y{}_3\ (f_1, f_2)$ of the random signal $\{y(t)\}$ is defined as its two-dimensional Fourier transform:

$$C_3^y(f_1,\ f_2) = \sum_{m=-\infty}^{+\infty} \sum_{n=-\infty}^{+\infty} C_3^y(m,n)\ \exp[-j(f_1 m + f_2 n)] \tag{99}$$

where $|f_1|, < \pi$, $|f_2| < \pi$, $|f_1 + f_2| < \pi$. In general, $C^y{}_3\ (f_1, f_2)$ is complex, that is, it has magnitude and phase:

$$C_3^y(f_1,\ f_2) = \left|C_3^y(f_1,\ f_2)\right|\ \exp[j\psi_3^y(f_1,\ f_2)] \tag{100}$$

It is clear from Eqs. (99) and (100) that the bispectrum is periodic with period 2π, and preserves both the magnitude and phase information associated with the random sequence $\{y(t)\}$. Thus, it provides one easy way to extract useful information directly, in contrast with complicated nonlinear models.

In view of the "filtering" nature of the TOC in Eq. (96), one can obtain that

$$C_3^x(f_1, f_2) = C_3^y(f_1, f_2) \tag{101}$$

Bearing Eq. (98) in mind, the bispectrum of $y(t)$ can be easily obtained as

$$C_3^y(f_1, f_2) = g(y_2(t)) \tag{102}$$

where g stands for a nonlinear relationship between the bispectrum and the phase-coupled component.

The above discussion has shown that the TOC and bispectrum statistical measures provide an enhanced system characterization capability over conventional second-order measures. As a result, model-free TOC- and bispectrum-based analysis techniques are particularly suitable for extracting internal chararacteristics of a nonlinear system.

Features of Nonlinear Representations

Identifiability. Identifiability is a concept that is central in system identification problems. Loosely speaking, the problem is whether the system modeling procedure will yield a unique value of the parameter and whether the resulting model is faithful to the true system. The issue involves whether the measured data set is informative enough to distinguish between different models as well as properties of the model structure itself: If the data are informative enough to distinguish between unequal models, then the question is whether different values of parameters can give equal models. Depending on the availability of system information, the corresponding models can be distinguished as: (1) *gray-box* models, whose structures are known on physical grounds but that have certain of parameters to be estimated from observed data; and (2) *black-box* models, whose structures as well as parameters are totally unavailable. In the following, we define different kinds of identifiability corresponding to different models.

Definition 1. *A gray-box model $\Sigma(y(t), u(t), \theta)$, where the observed input–output pair $\{y(t), u(t)\}$ is given, is internally identifiable at parameter vector θ^* if $\Sigma(y(t), u(t), \theta) = \Sigma(y(t), u(t), \theta^*)$, $\theta \in \Omega_\Sigma$, leads to $\theta = \theta^*$, where θ comprises the true parameters of the model, and θ^* their estimates. A gray-box model $\Sigma(y(t), u(t), \theta)$ is globally internally identifiable if it is internally identifiable at almost all $\theta^* \in \Omega_\Sigma$. Moreover, the gray model $\Sigma(y(t), u(t), \theta)$ is blindly internally identifiable if its input $u(t)$ is unknown.*

Definition 2. *A black-box model $\Gamma(y(t), u(t))$, where the observed input–output pair $\{y(t), u(t)\}$ is given, is externally identifiable if there exists a determined model structure Δ and parameters θ such that $\Gamma(y(t), u(t)) = \Delta(y(t), u(t), \theta)$, $\theta \in \Omega_\Sigma$. A black-box model $\Gamma(y(t), u(t))$ is globally externally identifiable if it is externally identifiable at almost all $u \in \Omega_u$, $y \in \Omega_y$.*

A nonlinear differential algebraic representation (7) and a state-space representation (46) are globally internally identifiable if and only if one can obtain differential equations in the form

$$\phi_i(u, y) + \theta_i \psi_i(u, y) = 0 \tag{103}$$

where i=1, 2, ,. . ., n, by differentiating, adding, scaling and multiplying the equations (7) and (46). Then the identification algorithm gives the parameter estimate (29)

$$\hat{\theta}_i = -\frac{\int_a^b \phi_i(u, y)\psi_i(u, y)\, dt}{\int_a^b \psi_i^2(u, y)\, dt} \tag{104}$$

As a special case of Eq. (46), a bilinear model (49) is internally identifiable if the above condition (103) is satisfied.

For the Volterra representation, it should be pointed out, there is no general identifiability principle. However, if we consider the finite-support Volterra model (34) excited by a zero-mean Gaussian signal, then the model is globally internally identifiable if and only if the power spectral process of the input signal is nonzero at at least t distinct frequencies (30). Moreover, if we consider the finite-support second-order Volterra model (37) excited by an unknown arbitrary zero-mean independent and identically distributed (*i.i.d.*) random signal, then the model is of blind internal identifiability if and only if the following equation has a unique solution (24):

$$C_3^y(m, n) = M_3^y(m, n) - M_1^y\left[M_2^y(m) + M_2^y(n) + M_2^y(n - m)\right] + 2\left(M_1^y\right)^3 \tag{105}$$

and

$$\begin{aligned}
M_1^y &= \gamma_{2u} p(0) \\
M_2^y(m) + M_2^y(n) + M_2^y(n - m) &= \gamma_{4u} g_1(m, n) + \gamma_{2u}^2[g_2(m, n) + g_3(m, n)] \\
M_3^y(m, n) &= \gamma_{6u} f_1(m, n) + \gamma_{4u}\gamma_{2u}[f_2(m, n) + f_3(m, n)] \\
&\quad + \gamma_{2u}^3[f_4(m, n) + f_5(m, n) + f_6(m, n)] \\
&\quad + \gamma_{3u}^2[f_7(m, n) + f_8(m, n)]
\end{aligned}$$

where

$$\begin{aligned}
g_1(m, n) &= \varphi_1(m) + \varphi_1(n) + \varphi_1(n - m) \\
g_2(m, n) &= \varphi_2(m) + \varphi_2(n) + \varphi_2(n - m) \\
g_3(m, n) &= \varphi_3(m) + \varphi_3(n) + \varphi_3(n - m) \\
f_1(m, n) &= \phi_1(0, m, n) \\
f_2(m, n) &= \phi_{21}(0, 0, m) + \phi_{22}(0, 0, n) + \phi_{23}(0, 0, n - m) \\
f_3(m, n) &= \phi_3(0, -m, n - m) + \phi_3(0, m, n) + \phi_3(0, n, m) \\
f_4(m, n) &= \phi_4(m, n, 0) \\
f_5(m, n) &= \phi_5(-m, 0, n - m) + \phi_5(m, 0, n) + \phi_5(n, 0, m) \\
f_6(m, n) &= \phi_6(m, m - n, n) \\
f_7(m, n) &= \phi_7(0, n, n - m) + \phi_7(0, n - m, n) + \phi_7(0, m, m - n) \\
&\quad + \phi_7(0, m - n, m) + \phi_7(0, -m, -n) + \phi_7(0, -n, -m) \\
f_8(m, n) &= \phi_8(0, m, n)
\end{aligned}$$

and

$$
\begin{aligned}
p(0) &= \sum_{i=0}^{q} h(i,i) \\
\varphi_1(k) &= \sum_{i=0}^{q} h(i,i)h(i+k,i+k) \\
\varphi_2(k) &= \sum_{i=0}^{q} h(i,i) \sum_{\substack{j=0 \\ j\neq i+k}}^{q} h(j,j) \\
\varphi_3(k) &= 4\sum_{i=0}^{q-1} \sum_{j=i+1}^{q} h(i,j)h(i+k,j+k) \\
\phi_1(k_1,k_2,k_3) &= \sum_{i=0}^{q} h(i+k_1,i+k_1)h(i+k_2,i+k_2)h(i+k_3,i+k_3) \\
\phi_{21}(k_1,k_2,k_3) &= \sum_{i=0}^{q} \sum_{\substack{j=0 \\ j\neq i+n}}^{q} h(i+k_1,i+k_1)h(j+k_2,j+k_2)h(i+k_3,i+k_3) \\
\phi_{22}(k_1,k_2,k_3) &= \sum_{i=0}^{q} \sum_{\substack{j=0 \\ j\neq i+m}}^{q} h(i+k_1,i+k_1)h(j+k_2,j+k_2)h(i+k_3,i+k_3) \\
\phi_{23}(k_1,k_2,k_3) &= \sum_{i=0}^{q} \sum_{\substack{j=0 \\ j\neq i+m}}^{q} h(i+k_1,i+k_1)h(j+k_2,j+k_2)h(j+k_3,\mathrm{j}+k_3) \\
\phi_3(k_1,k_2,k_3) &= 4\sum_{i=0}^{q-1} \sum_{j=i+1}^{q} h(i+k_1,j+k_1) \\
&\quad \times[h(i+k_2,i+k_2)+h(j+k_2,j+k_2)]h(i+k_3,j+k_3) \\
\phi_4(k_1,k_2,k_3) &= \sum_{\substack{i,j,k=0 \\ i\neq j-k_1 \\ \neq k-k_2}}^{q} h(i+k_3,i+k_3)h(j,j)h(k,k) \\
\phi_5(k_1,k_2,k_3) &= 4\sum_{i=0}^{q-1} \sum_{j=i+1}^{q} \sum_{\substack{k=0 \\ k\neq i+k_1 \\ \text{or} k\neq j+k_1}}^{q} h(i,j)h(k+k_2,k+k_2)h(i+k_3,j+k_3) \\
\phi_6(k_1,k_2,k_3) &= 8\sum_{i=0}^{q-1} \sum_{j=i+1}^{q} \sum_{k=0}^{q-1} h(i,\mathrm{j})h(j+k_1,k+k_2)h(i+k_3,j+k_3) \\
\phi_7(k_1,k_2,k_3) &= 2\sum_{i=0}^{q} \sum_{j=0}^{q} h(i,i)h(j+k_1,j+k_1)h(i+k_2,j+k_3) \\
\phi_8(k_1,k_2,k_3) &= 8\sum_{i=0}^{q-1} \sum_{j=i+1}^{q} h(i+k_1,j+k_1)h(i+k_2,j+k_2)h(i+k_3,j+k_3)
\end{aligned}
$$

where $C^y{}_3(m,n)$ is the TOC of the output signal $y(t)$ defined in Eq. (92), and γ_{iu}, i=2, 3, 4, 6, is the ith-order autocorrelation of the input $u(t)$. Clearly, one must estimate the $(q+1)^2$ TOCs $C^y{}_3(m,n)$ for $0 \le m,n \le q$, and then invert the equations to solve for the unknown quantities $\hat{h}(i,j)$. This inversion involves solving $(q+1)^2$ simultaneous cubic equations with $(q+1)^2$ unknowns. The parameter-coupled terms in Eq. (105) are of high complexity, and conventional model-based estimation theories cannot directly applied. However, the following theorem provides a possible way to estimate the model parameters based on neural networks:

Let $(m,n) \in M \subset \Re^p$ and $H_{ij} \in H \subset \Re^q$, where $H_{ij} = [h(0,0)\ h(0,1) \cdots h(0,q) \cdots \cdots h(q,0)\ h(q,1) \cdots h(q,q)]$ be open. Let g be continuous on $M \times H$. Consider the equation (105).

Theorem 3. Let $D_{H_{ij}} g(\overline{H}_{ij}, \overline{(m,n)})$ be nonsingular for $\overline{H}_{ij}, \in H$ and $\overline{(m,n)} \in M$. Given the class of neural networks, N, that map $\Re^p \times \Re^q$ to $\Re^q$, there exist $\varepsilon_1, \varepsilon_2 > 0$ and a neural network $NN \in N$ such that, given $\varepsilon > 0$,

$$\sup_{\left(C_3^y(m,n),(m,n)\right)\in\phi} \left|H_{ij} - \mathrm{NN}\left(C_3^y(m,n),(m,n)\right)\right| < \varepsilon \tag{106}$$

where $\phi = \{(C^y{}_3(m,n), (m,n)) : |C^y{}_3(m,n) - g(\overline{H}_{ij}, \overline{(m,n)})| < \varepsilon_1, |(m,n) - \overline{(m,n)}| < \varepsilon_2\}$.

As far as the identifiability of NARMA models is concerned, Eqs. (103) and (104) are also suitable, because a NARMA model is a special type of nonlinear differential algebraic representation and can be characterized by a state-space representation. On the other hand, relying on the approximation capabilities of multilayer neural networks, the functions Φ in Eq. (65) can be approximated by a neural network with appropriate input and output dimensions. This can be stated as the following theorem (31).

Theorem 4. For generic Φ, the input–output behavior of the NARMA model (65) is externally identifiable with arbitrary precision by a multilayer feedforward neural-network-based input–output model of the form

$$\begin{aligned} y(k+1) = \mathrm{NN}[&y(k), y(k-1), \ldots, y(k-l+1), \\ &u(k), u(k-1), \ldots, u(k-l+1)] \end{aligned} \tag{107}$$

where $\mathrm{NN}(\cdot)$ is a multilayer feedforward neural network with $2l$ inputs and one output characterizing the monitored nonlinear system. Here l is the order of the model.

An important problem faced herein is the choice of the most suitable network topology to perform the identification and modeling of unknown nonlinear dynamic systems efficiently. It mainly lies in the choice of the optimal number of neurons for each of the networks employed, which plays a crucial role in network performance. Considering the external identifiability of fuzzy-logic models, the following theorem shows that the fuzzy-logic models with fuzzifier and defuzzifier are capable of uniformly approximating any nonlinear function over U to any degree of accuracy if U is compact (32).

Theorem 5. For any given real continuous function g on a compact set $U \subset \Re^n$ and arbitrary $\varepsilon > 0$, there exists a fuzzy-logic model f such that

$$\sup_{x\in U} |f(u) - g(u)| < \varepsilon \tag{108}$$

This theorem provides a justification for applying the fuzzy-logic representations to almost any nonlinear modeling problem. It also provides an explanation for the practical success of the fuzzy logic models in engineering applications.

Some nonlinear systems are very difficult, if not impossible, to describe with analytical equations. Hence, there is a great need to develop a robust and less model-dependent methodology for the representation of a complex nonlinear dynamic system. To avoid the difficulties of the classical nonlinear-system modeling approach,

neural-network-based modeling methods for nonlinear systems have been proposed recently. The main motivation is that any nonlinear relationship can be approximated by a neural network given suitable weighting factors and architecture. Another important property of such networks is their self-learning ability: a neural network can extract the system features from previous training data through learning, whilst requiring little or no *a priori* knowledge about the system. This provides great flexibility for modeling general nonlinear physical systems and guarantees robust external identifiability of neural-network-based system representations. A multilayer network trained with the CWDEKF algorithm may be viewed as a practical vehicle for performing general nonlinear input–output mapping, due to the following universal approximation theorem (23,33).

Theorem 6. Let $\varphi(\cdot)$ be a nonconstant, bounded, and monotone-increasing continuous function. Let I_p denote the p-dimensional unit hypercube $[0,1]^p$. The space of continuous functions on I_p is denoted by $C(I_p)$. Then, given any function $f \in C(I_p)$ and $\varepsilon > 0$, there exist an integer M and sets of real constants α_j, θ_i, and w_{ij}, where $i=1, 2, \ldots, M$ and $j=1, 2, \ldots, p$, such that we may define

$$F(x_1, x_2, \ldots, x_p) = \sum_{i=1}^{M} \alpha_i \varphi \left(\sum_{j=1}^{p} w_{ij} x_j - \theta_i \right) \tag{109}$$

as an approximate realization of the function $f(\cdot)$; that is, $|F(x_1, x_2, \ldots, x_p) - f(x_1, x_2, \ldots, x_p)| < \varepsilon$ for all $(x_1, x_2, \ldots, x_p) \in I_p$.

This theorem is directly applicable to multilayer networks. We first note that the logistic function used as the nonlinearity in a neuron node for the construction of a multilayer network is indeed a nonconstant, bounded, and monotone-increasing function. It therefore satisfies the conditions imposed on the function $\varphi(\cdot)$. Next, we note that the above equation represents the output of a multilayer network described as follows: (1) the network has p point nodes and a single hidden layer consisting of M neurons; the inputs are denoted by $(x_1, x_2, \ldots, x_p)$; (2) hidden neuron i has synaptic weights $(w_{i1}, W_{i2}, \ldots, w_{ip})$ and threshold θ_i; and (3) the network output is a linear combination of the outputs of the hidden neurons, with $(\alpha_1, \alpha_2, \ldots \alpha_M)$ as coefficients.

The universal approximation theorem 6 is an existence theorem in the sense that it provides the mathematical justification for the approximation of an arbitrary continuous function as opposed to exact representation.

Controllability. Controllability is one of the most important properties of systems modeled by state-space representations. A state-space model in Eq. (46) is said to be *controllable* if an available input $u(t)$ is sufficient to bring the model from any initial state $x(0)$ to any desired final state $x(t)$. The importance of controllability in the formulation of the control problems for linear systems is evident. A linear time-invariant system $\dot{x}(t) = Ax(t) + Bu(t)$ is completely controllable if and only if the matrix $E = [B\ AB \cdots A^{n-1}\ B]$ has rank n, where $x(t) \in \Re^n$, $u(t) \in \Re^m$.

In spite of many attempts to characterize controllability for nonlinear systems, similar generic results to those available for linear systems do not exist for nonlinear systems. Hence, the choice of controller models for nonlinear systems is a formidable problem, and successful control has to depend on several strong assumptions regarding the input–output behavior of the systems. For example, we consider the specified (affine) nonlinear model of the form

$$\dot{x}(t) = f(x, t) + g(x, t)u(t) \tag{110}$$

where $x(t) = [x_1, x_2, \ldots, x_n]^{\mathrm{T}}$ is the local coordinate on a smooth n-dimensional manifold M, the control $u(t)$ is a scalar piecewise smooth function, and $f(x, t)$, $g(x, t)$ are the local coordinate representations of smooth vector

fields globally defined on M. The model (110) is locally controllable at an equilibrium point x_e of $f(\cdot)$ if [B, AB, $\ldots$, $A^{n-1}B$] has full rank, where $B = g(x_e)$ and $A = \partial f(x_e)/\partial x$ (36).

On the other hand, the variable structure of bilinear systems allows them to be more controllable than linear systems, just as it frequently provides for a more accurate model. Let us consider the bilinear system given by Eq. (49). It is assumed that the class of admissible inputs $u(t)$ is the class of all piecewise continuous vector time functions with $[0, \infty)$ and range U, where U is a compact connected set containing the origin in $\Re^m$. The *reachable zone* from an initial state x_0, $R(x_0) \subset \Re^n$, is the set of all states to which the system can be transferred in finite time, starting at x_0. Similarly, the *incident zone* to a terminal state x_f, $I(x_f) \subset \Re^n$, is the set of all initial states from which x_f is reachable in finite time.

For each fixed $x \in U$, the bilinear system is a constant-parameter linear system with system matrix $F + \Sigma^m{}_{i=1} N_i u_i$. The term $\Sigma^m{}_{i=1} N_i u_i$ in the system matrix permits manipulation of the eigenvalues of the fixed-control system. With an appropriate controller it is often possible to shift these eigenvalues from the left half of the complex plane to the right half. The controllability analysis presented here can be summarized by the following sufficient conditions (5):

Theorem 7. The bilinear system (49) is completely controllable if: (1) there exist input values u^+ and u^- such that the real parts of the eigenvalues of the system matrix are positive and negative, respectively, and such that equilibrium states $x_e(u^+)$ and $x_e(u^-)$ are contained in a connected component of the equilibrium set, and (2) for each x in the equilibrium set with an equilibrium input $u_e(x) \in U$ such that $f(x, u_e(x)) = 0$, there exists a $v \in R^m$ such that g lies in no invariant subspace of dimensional at most $n - 1$ of the matrix E, where

$$\begin{aligned} E &= F + \sum_{i=1}^{m} N_i u_i(x) \\ g &= Gv - \sum_{j=1}^{m} v_j \left[N_j \left(F + \sum_{i=1}^{m} N_i u_i \right)^{-1} \right] Gu \end{aligned} \tag{111}$$

As one might expect, the conditions given by the above theorem are not as simple as the popular conditions for complete controllability of linear systems. For phase-variable systems, $x_1 = x, x_2 = \bar{a}, \ldots, x_n = x^{(n-1)}$, condition (2) is always satisfied if G is a nonzero matrix. Condition (2) is satisfied if all the eigenvalues of the system matrix $F + \Sigma^m{}_{i=1} N_i u_i$ can be shifted across the imaginary axis of the complex plane without passing through zero, as u ranges continuously over a subset of U.

Observability. A nonlinear state-space model is said to be *observable* if, given any two states x_1, x_2, there exists an input sequence of finite length, $u_l = [u(0), u(1), \ldots, u(l)]^{\mathrm{T}}$, such that $y_l(x_1, u_l) \neq y_l(x_2, u_l)$, where y_l is the output sequence. The ability to effectively estimate the state of a model or to identify it based on input–output observations is determined by the observability properties of the model.

Observability has been extensively studied in the context of linear systems and is now part of the standard control literature. An nth-order single-input and single-output time-invariant linear system is described by the set of equations

$$\begin{aligned} x(t+1) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) \end{aligned} \tag{112}$$

where $x(t) \in \Re^n$, $u(t) \in \Re$, $y(t) \in \Re$, A is an $n \times n$ matrix, and B, C are n-dimensional vectors. A basic result in linear control theory states that the above system will be observable if and only if the $n \times n$ matrix $M = [C\ CA \cdots CA^{n-1}]^{\mathrm{T}}$ is of rank n. M is called the observability matrix.

If the system (112) is observable and $[CB\ CAB \cdots CA^{d-2}\ B]^{\mathrm{T}} = 0$, but $CA^{d-1}B \neq 0$, then we have $y(t + d) = CA^d\ x(t) + CA^{d-1}Bu(t)$. This implies that the input at any instant t can affect the output only d instants later, where d denotes the delay in the propagation of the signals through the system and is called the *relative degree* of the system.

Observability of a linear system is a system-theoretic property and remains unchanged even when inputs are present, provided they are known. For an observable linear system of order n, any input sequence of length n will distinguish any state from any other state. If two states are not distinguishable by this randomly chosen input, they cannot be distinguished by any other input sequence. In that case, the input–output behavior of the system can be realized by an observable system of lower dimension, where each state in the new system represents an equivalence class that corresponds to a set of states that could not be distinguished in the old one.

Whereas a single definition is found to be adequate for linear time-invariant systems, the concept of observability is considerably more involved for nonlinear systems. A desirable situation would be if any input sequence of length l sufficed to determine the state uniquely, for some integer l. This form of observability will be referred to as *strong* observability. It readily follows that any observable linear system is strongly observable with $l = n$, n being the order of the system. A less restrictive form of observability is the notion of *generic* observability. A system of the form (46) is said to be generically observable if there exists an integer l such that almost any input sequence of length greater or equal to l will uniquely determine the state.

Now let us look at the observability of nonlinear systems using Eq. (46). This problem arises when the laws governing the system are known but the states of the system are not accessible or only partly accessible. By definition, complete knowledge of the states will enable accurate prediction of the system's behavior. Thus the observation problem in this case actually reduces to the estimation of the state based on input and output observations over a time interval $[t_0, t_0 + l]$, or in other words establishing an *observer*. Sufficient conditions for strong local observability of a system (46) around the origin can be derived from the observability properties of its linearization at the origin:

$$\begin{aligned} \delta x(t+1) &= f_x|_{0,0}\ \delta x(t) + f_u|_{0,0}\ \delta u(t) = A\ \delta x(t) + B\ \delta u(t) \\ \delta y(t) &= g_x|_0\ \delta x(t) = C\ \delta x(t) \end{aligned} \tag{113}$$

where $A = f_x|_{0,0}$, $B = f_u|_{0,0}$, $C = g_x|_0$ are the system's Jacobian matrices. This is summarized by the following theorem (33).

Theorem 8. Let Σ be nonlinear system (45), and Σ_l its linearization around the equilibrium as given in Eq. (113). If Σ_l is observable, then Σ is locally strongly observable.

A bilinear model (49) or (50) is called *observable* if there are no indistinguishable states in the model. Theorem 9 gives a necessary and sufficient condition for observability of bilinear models (8).

Theorem 9. The bilinear model described in Eq. (49) or (50) is observable if and only if rank $Q_n = n$, where $Q_n = [q_1, q_2, \ldots, q_n]^{\mathrm{T}}$, $q_1 = C$, $q_i = [q_{i-1}F\ q_{i-1}N_i]^{\mathrm{T}}$, $i = 2, 3, \ldots, n$.

Example

This example shows how a neural network together with a nonlinear model approach is used for short-term electric load forecasting (34). This neural-network model utilizes the full dynamic range of the neural network and is a nonlinear model for nonstationary time series. The model is used to provide hourly load forecasts one day ahead. Off-line simulation has been done on the Hong Kong Island electric load profile provided by the Statistics and Planning Division of Hong Kong Electric Company Limited.

The electric load forecast models can be summarized in the following two equations: for a static model (feedforward neural network),

$$x_{t+l} = h(x_{t-1}, x_{t-2}, \ldots, x_{x-p}, \hat{w}_{t+l}, w_{t-1}, w_{t-2}, \ldots, w_{t-q}) + e_{t+l}$$

and for a dynamical model (recurrent neural network),

$$x_{t+l} = h(x_{t-1}, x_{t-2}, \ldots, x_{x-p}, \hat{w}_{t+l}, w_{t-1}, \ldots, w_{t-q}, e_{t-1}, \ldots, e_{t-r}) + e_{t+l}$$

where x_t is the load consumption, $\hat{w}_{t+l}$ is the weather forecast, w_t is the weather information, and e_t is a noise residual at time t. The nonlinear function h is nonlinearly approximated by *ANN*. From the above models, the load forecast $\hat{x}_{t+l}$ is estimated mainly from the currently available load profile.

In general, a linear time series $\{x_t\}$ can be written in the form of an ARMA model as

$$\phi(B)x_t = \theta(B)e_t \tag{114}$$

where B is the backward shift operator, and the autoregressive operator $\phi(B)$ of order p is given by

$$\phi(B) = 1 - \phi_1 B - \phi_2 B^2 - \cdots - \phi_p B^p$$

The moving-average operator $\theta(B)$ of order q is defined by

$$\theta(B) = 1 - \theta_1 B - \theta_2 B^2 - \cdots - \theta_q B^q$$

where the white noise e_t with finite variance σ^2 is zero-mean, i.i.d., and independent of past x_t. Equation (114) can be rewritten in the form

$$x_t = \sum_{i=1}^{p} \phi_i x_{t-i} - \sum_{j=1}^{q} \theta_j e_{t-j} + e_t \tag{115}$$

If the roots of the equation

$$\phi(B) = 1 - \sum_{i=1}^{p} \phi_i B^i = 0$$

lie outside the unit circle, the time series $\{x_t\}$ is stationary; otherwise $\{x_t\}$ is nonstationary. For linear nonstationary time series $\{x_t\}$, the ARMA model for stationary time series can still be used, but the above ARMA

model (115) has to be reformulated as

$$\nabla^d x_t = \sum_{i=1}^{p} \phi_i \nabla^d x_{t-i} - \sum_{j=1}^{q} \theta_j e_{t-j} + e_t \tag{116a}$$

or

$$\nabla^d x_t = \sum_{i=1}^{r} \varphi_i x_{t-i} - \sum_{j=1}^{q} \theta_j e_{t-j} + e_t \tag{116b}$$

where $\triangledown^d x_t = (1 - B^d)x_t = x_t - x_{t-d}$ and $\phi_i = \varphi_i + \varphi_{i+d}$. The model 116a is the so-called *ARIMA* model. A natural generalization of the linear ARIMA model to the nonlinear case would be the nonlinear autoregressive integrated moving average (*NARIMA*) model, which is given by

$$\nabla^d x_t = h(\nabla^d x_{t-1}, \nabla^d x_{t-2}, \ldots, \nabla^d x_{t-p}, e_{t-1}, \ldots, e_{t-q}) + e_t \tag{117a}$$

or

$$\nabla^d x_t = h(x_{t-1}, x_{t-2}, \ldots, x_{t-r}, e_{t-1}, \ldots, e_{t-q}) + e_t \tag{117b}$$

where h is an unknown smooth function and, as in Eq. (114), it is assumed that the white noise e_t with variance σ^2 is zero-mean, i.i.d, and independent of past x_t. The minimum-mean-squared-error optimal predictor of x_t given $x_{t-1}, \ldots, x_{t-r}$ is the conditional expectation

$$\begin{aligned} \hat{x}_t &= E\{x_t \,|x_{t-1}, \ldots, x_{t-r}\} \\ &= E\{x_{t-d} + \nabla^d x_t \,|x_{t-1}, \ldots, x_{t-r}\} \\ &= x_{t-d} + h(x_{t-1}, \ldots, x_{t-r}, e_{t-1}, \ldots, e_{t-q}) \end{aligned}$$

This predictor has mean squared error σ^2. In this work, a nonlinear autoregressive integrated (*NARI*) model, the special case of the NARIMA, is considered and is defined by

$$\nabla^d x_t = h(\nabla^d x_{t-1}, \nabla^d x_{t-2}, \ldots, \nabla^d x_{t-p}) + e_t \tag{118a}$$

or

$$\nabla^d x_t = h(x_{t-1}, x_{t-2}, \ldots, x_{t-r}) + e_t \tag{118b}$$

We focus on a multilayer feedforward neural network (*FNN*) and how it may be used to forecast the hourly load consumption of the coming day. In many time-series predictions, the time-series model is based on nonlinear autoregressive (*NAR*) models of the form

$$x_t = h(x_{t-1}, x_{t-2}, \ldots, x_{t-p}) + e_t \tag{119}$$

The neural STLF model can be considered as a modified NAR model given by

$$x_{t+l} = h(x_{t-1}, x_{t-2}, \ldots, x_{x-24}, \boldsymbol{w}_t) + e_{t+l} \tag{120}$$

and the unknown smooth function h is nonlinearly approximated by a FNN. Hence, the neural optimal predictor is given by

$$\hat{x}_{t+l} = \sigma\left(\sum_{i=0}^{H} W_i^1 \sigma\left(\sum_{j=0}^{24} W_{ij}^0 x_{t-j} + \sum_{k=1}^{m} W_{ik}^0 w_k^t\right)\right) \tag{121}$$

where $0 \le l \le 24$ and the function σ is a smooth bounded monotonic function, $\tanh(0.5x)$. The vector $\boldsymbol{w}_t$ of m components contains the available weather information at time t. The parameters $W^0{}_{ij}$, $W^0{}_{ik}$, and $W^1{}_i$ are the neural-network weights.

To obtain an accurate load forecast, we should identify the most appropriate model in accordance with the nature of load consumption. In this example, the modified NARI model in Eq. (118b) is proposed for STLF. Several important weather factors are also included, because weather variation is one of the crucial disturbances to electric load demand. Consequently, the modified NARI model for STLF is given by

$$\hat{x}_{t+l} = x_{t+l-d} + \hat{h}(x_{t-1}, \ldots, x_{t-24}, \boldsymbol{w}_t) + e_{t+l}$$

and the neural optimal predictor is then formulated by

$$\hat{x}_{t+l} = x_{t+l-d} + \sigma\left(\sum_{i=0}^{H} W_i^1 \sigma\left(\sum_{j=0}^{24} W_{ij}^0 x_{t-j} + \sum_{k=1}^{m} W_{ik}^0 w_k^t\right)\right)$$

where $0 \le l \le 24$. The weather information vector $\boldsymbol{w}_t$ contains four components, namely, temperature, relative humidity, level of rainfall, and level of sunshine.

The architecture of our proposed neural network model is illustrated in Fig. 9(b). The structure of the modified NAR model for STLF is depicted in Fig. 9(a). This type of NARI neural-network model is called a weather compensation neural network because the weather-dependent component will be only determined from the weather information and load consumption of the previous day.

The characteristics of electric load consumption gradually change because of many uncontrollable factors. Our weather compensation neural network (NARI model), without keeping track of the change of load characteristic, will degrade in forecasting performance over the years. An adaptive tracking scheme is therefore proposed so that the weather compensation neural network will be retrained every day. This scheme can efficiently update the neural network to adapt to the changing conditions of the environment. Different sizes of window of the trained set have been studied. The size of the window determines the memory of past knowledge. Too small a window may produce catastrophic loss of information that severely degrades the forecasting performance and robustness. Three different sizes—10 days, 20 days, and 136 days—are examined. Updating is omitted when the standard deviation of the forecast error is less than 990 MW. The results are depicted in Fig. 10, and they show that the neural network provides the most robust forecasting when the window size of 136 days is used. Figure 11 displays a comparison of the actual load and the load forecast for seven consecutive working days using our proposed NARI neural network and adaptive tracking scheme. Figures 12 and 13

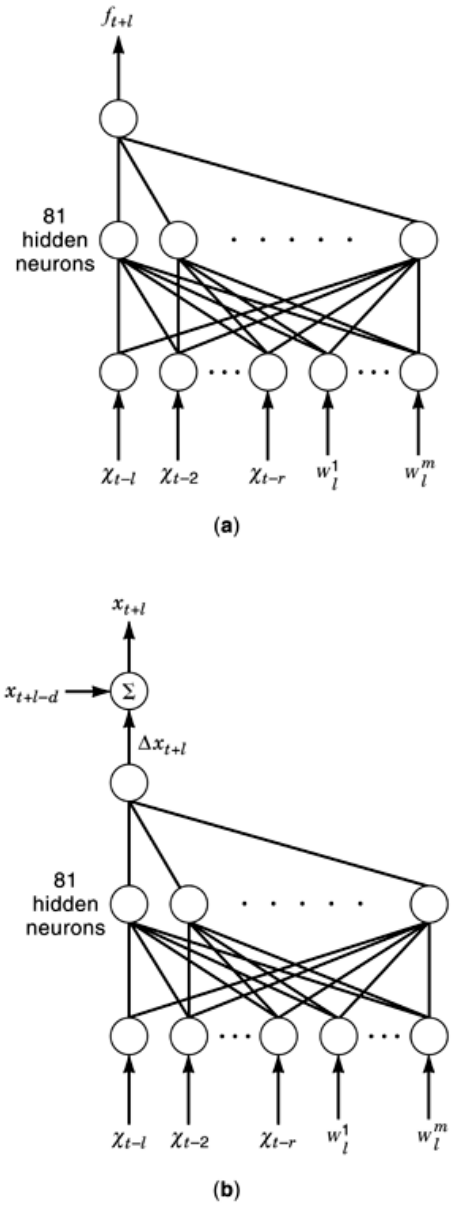

Fig. 9. (a) The architecture of modified NAR model for neural short-term load forecasting. (b) The architecture of the weather compensation neural network (modified NARI model).

display a comparison of the actual load and the load forecast for the worst cases using the NAR and the NARI neural network, respectively.

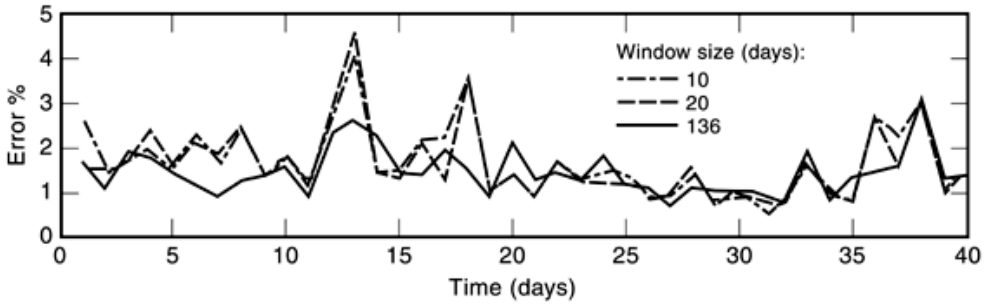

Fig. 10. Comparison of percentage error for different sizes of moving window using the neural-network NARI model.

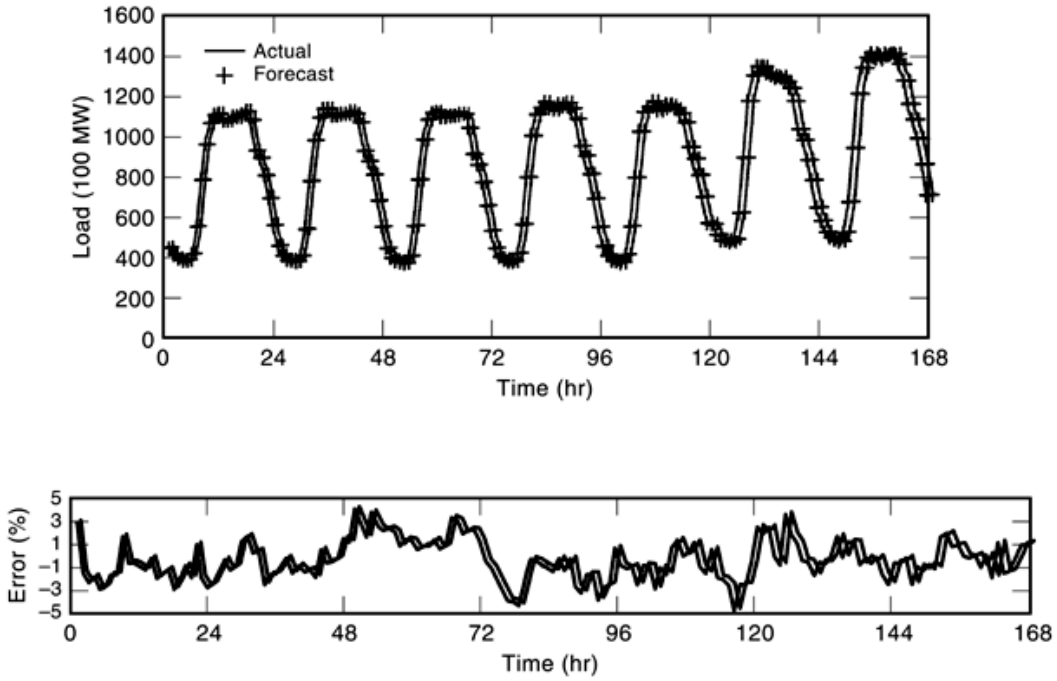

Fig. 11. Comparison of actual load and load forecast using neural network NARI model (136-day window) for seven consecutive working days.

Concluding Remarks

Nonlinear system representations are classified by a simple dichotomy: a system is represented either linearly or nonlinearly. The number and variety of nonlinear representations is almost limitless, and one does not find the complete and elegant theory that exists for linear representations. It would be expecting too much if one hoped to find a universal analytical technique applicable to any nonlinear system representation with an arbitrary input signal.

In this article, we have discussed several nonlinear representations and their important features such as identifiability, controllability, and observability. These representations are widely used in signal processing,

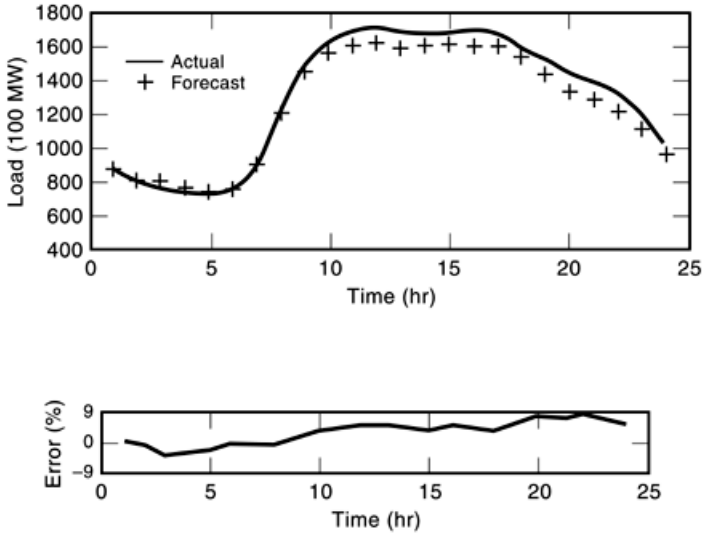

Fig. 12. Comparison of the actual load and the load forecast using the neural-network NAR model (worst case).

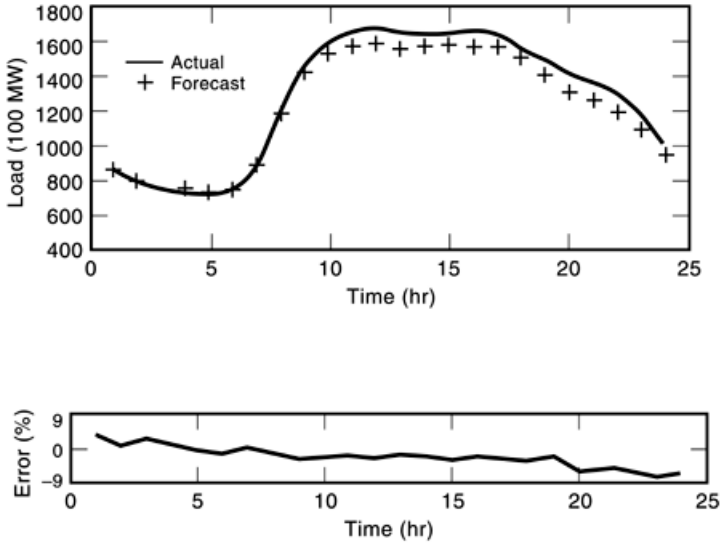

Fig. 13. Comparison of the actual load and the load forecast using the neural network NARI model with 136-day window (worst case).

circuit systems, control systems, and communication systems. A comparison of these representations is given in Table 1. It can be seen that, with different types of input signal, different nonlinear representations should be selected. When internal identifiability is important, Volterra, state-space, and bilinear representations are the better choices, due to the linear parameter relationship of the Volterra system and the compact description of the state-space and bilinear systems. For external identifiability, fuzzy logic and neural-network-based representations often appear to be useful for characterizing nonlinear systems in a black-box approach.

Table 1: Comparison of Different Nonlinear Representations

Representation	Input Signals	Identifiability	Controllability	Observability
Differential algebraic	Deterministic	√	×	×
Volterra	Random	√	×	×
State-space	Deterministic or random	√	√	√
Bilinear	Deterministic or random	√	√	√
NARMA	Random	√	×	×
Fuzzy logic	Deterministic	√	×	×
Neural network	Deterministic or random	√	×	×

On the other hand, a state-space representation can be used to guarantee high controllability and observability, since the corresponding theories in the linear domain can be extended to the nonlinear world. For system inversion, no one representation is the best, due to the complexity of the problem; but suboptimal alternatives may be obtained with Volterra, state-space, or bilinear representations.

Acknowledgment

The authors would like to thank the Hong Kong Electric Company Limited for providing the electric load data and weather information for this part of work.

BIBLIOGRAPHY

1. D. E. Thompson *Design Analysis: Mathematical Modeling of Nonlinear Systems*, New York: Cambridge Univ. Press, 1999.
2. M. Vidyasagar *Nonlinear Systems Analysis*, 2nd ed., Englewood Cliffs, NJ: Prentice-Hall, 1993.
3. A. J. Fossard D. Normand-Cyrot *Nonlinear Systems, Modeling and Estimation*, London: Chapman & Hall, 1995.

4. M. Schetzen *The Volterra and Wiener Theories of Nonlinear Systems*, New York: Wiley, 1980.
5. J. L. Casti *Nonlinear System Theory*, Orlando, FL: Academic Press, 1985.
6. P. G. Drazin *Nonlinear Systems*, New York: Cambridge Univ. Press, 1992.
7. J. S. Bendat A. G. Piersol *Engineering Applications of Correlation and Spectral Analysis*, New York: Wiley, 1993.
8. R. R. Mohler *Bilinear Control Processes*, New York: Academic Press, 1973.
9. P. N. Paraskevopoulos A. S. Tsirikos K. G. Arvanitis A new orthogonal series approach to state space analysis of bilinear systems, *IEEE Trans. Automat. Control*, **39**: 793–797, 1994.
10. S. Chen S. A. Billings Representation of nonlinear systems: the NARMA model, *Int. J. Control*, **49**(3): 1013–1032, 1989.
11. I. J. Leontaritis S. A. Billings Input–output parametric models for non-linear systems, part I: deterministic non-linear systems, *Int. J. Control*, **41**(2): 303–328, 1985.
12. E. D. Sontag Realization theory of discrete-time nonlinear systems: Part I—the bounded case, *IEEE Trans. Circuits Syst.* **26**: 342–356, 1979.
13. L. Zadeh Fuzzy sets, *Inf. Control*, **8**: 338–353.
14. C. C. Lee Fuzzy logic in control systems: fuzzy logic controller, part I, *IEEE Trans. Syst. Man Cybern.*, **SMC-20**: 404–418, 1990.
15. C. C. Lee Fuzzy logic in control systems: fuzzy logic controller, part II, *IEEE Trans. Syst. Man Cybern.*, **SMC-20**: 419–435, 1990.
16. T. Takagi M. Sugeno Fuzzy identification of systems and its applications to modeling and control, *IEEE Trans. Syst. Man Cybern.*, **15**: 116–132, 1985.
17. E. H. Mamdani Applications of fuzzy algorithms for simple dynamic plant, *Proc. IEE*, **121**(12): 1585–1588, 1974.
18. C. J. Harris C. G. Moore M. Brown *Intelligent Control: Some Aspects of Fuzzy Logic and Neural Networks*, London and Singapore: World Scientific Press, 1993.
19. B. H. Wang G. Vachtsevanos Learning fuzzy logic control: an indirect control approach, *Proc. IEEE Int. Conf. on Fuzzy Systems*, San Diego, CA, 1992, pp. 297–304.
20. T. J. Ross *Fuzzy Logic with Engineering Applications*, New York: McGraw-Hill, 1995, Chapter 13.
21. J. Y. Li T. W. S. Chow Functional approximation of higher-order neural networks, *J. Intell. Syst.*, **6**(3–4): 239–260, 1996.
22. J. J. Hopfield Learning algorithm and probability distributions in feed-forward and feed-back networks, *Proc. Nat. Acad. Sci. U.S.A.*, **84**, 8429–8433, 1987.
23. K. Hornik M. Stinchcombe H. White Multilayer feedforward networks are universal Approximators, *Neural Netw.*, **2**: 359–366, 1989.
24. H.-Z. Tan T. W. S. Chow Blind identification of quadratic nonlinear models using neural networks with higher-order cumulants, *IEEE Trans. Ind. Electron.*, **47**: 687–696 2000.
25. J. Y. F. Yam T. W. S. Chow A weight initialization method for improving training speed in feedforward neural network, *Neurocomputing*, **30**(1–4): 219–232, 1999.
26. J. Y. F. Yam T. W. S. Chow C. T. Leung A new method in determining initial weights of feedforward neural networks for training enhancement, *Neurocomputing*, **16**(1): 23–32, 1997.
27. J. J. Hopfield Neural networks and physical systems with emergent collective properties, *Proc. Nat. Acad. Sci. U.S.A.*, **79**: 2554–2558, 1982.
28. C. L. Nikias A. P. Petropuop *Higher-Order Spectra Analysis: A Non-linear Signal Processing Framework*, Englewood Cliffs, NJ: Prentice-Hall, 1993.
29. L. Ljung *System Identification: Theory for the User*, 2nd ed., Upper Saddle River, NJ: Prentice-Hall, 1999.
30. G. A. Glentis P. Koukoulas N. Kalouptsidis Efficient algorithm for Volterra system identification, *IEEE Trans. Signal Process.*, **47**: 3042–3057, 1999.
31. A. U. Levin K. S. Narendra Recursive identification using feedforward neural networks, *Int. J. Control*, **61**(3): 533–547, 1995.
32. L. X. Wang J. M. Mendel Fuzzy basis function, universal approximation, and orthogonal least squares learning, *IEEE Trans. Neural Netw.*, **3**: 807–814, 1992.
33. H.-Z. Tan Y. Fang T. W. S. Chow Wiener models identification based on the higher-order cumulants and orthogonal wavelet neural networks, *Int. J. Knowledge Based Intell. Eng. Syst.*, **3**(2): 102–107, 1999.
34. T. W. S. Chow C. T Leung A non-linear autoregressive intergrated neural network model for short-term load forecasting, *IEE Proc. Gener. Transm. Distrib.*, **143**(5): 500–506, 1996.

READING LIST

G. Anger *Inverse Problems in Differential Equations*, New York: Plenum, 1990.

B. Igelnik Y.-H. Pao Stochastic choice of basis functions in adaptive function approximation and the functional-link net, *IEEE Trans. Neural Netw.*, **6**: 1320–1329, 1995.

W. J. Rough An extended linearization approach to nonlinear system inversion, *IEEE Trans. Automat. Control*, **AC-31**(8): 725–733, 1986.

S. N. Singh A modified algorithm for invertibility in nonlinear systems, *IEEE Trans. Automat. Control*, **26**: 595–598, 1981.

TOMMY W. S. CHOW
City University of Hong Kong
HONG-ZHOU TAN
University of Manitoba
YONG FANG
Shanghai University