PAPER INDUSTRY, SYSTEM IDENTIFICATION AND MODELING

In many industries the processes used to manufacture a product are at such a high level of complexity that no working model of the process has been constructed. Such processes generally consist of many variables (dependent and independent) that interact through time to produce a system response. Proper and efficient control of such systems relies heavily on experienced operators who gain their system knowledge from other operators, trial and error, and educated guesses. In essence, prior experience allows operators to form their own system model in order to best control the process. An operator's system model, however, is highly simplified, since an operator cannot be expected to track the relationships between hundreds of variables. It is also difficult, if not impossible to quantify an operator's knowledge of a system. It would be useful if a tool could be constructed that, like an operator, observed the system in action, only instead of forming unstructured rules about the system like an operator would, it would attempt to quantify the functional relationship between the system response and the variables. Such a tool could be used to predict the system response to a change of one or more of the variables. Using such predictions, operators or control systems would have a better chance at efficiently controlling a process. The task of system identification is to provide such a tool.

A system identification tool would be useful in the analysis of many human-made and naturally occurring systems. An example of such a human-made system is a common pulp digester found in the pulp and paper industry. A simplified description of a pulp digester is a tank in which wood chips enter the top while the pulp (the raw material of paper) exits the bottom. The system response this research effort is concerned with is the height of the chip mixture in the digester, known as the digester level. It is desired to keep this level as constant as possible. However, the level commonly displays erratic behavior as shown in Fig. 1.

In many paper plants the digester level is controlled by human operators. When its level rises beyond a certain point, the flow out the bottom of the digester (blow flow) is increased. Likewise, the blow flow is decreased when the digester level drops too low. In addition to blow flow, many variables within the system determine the digester level's dynamic response. However, the exact relationship between the variables and digester level is unknown. A system identi-

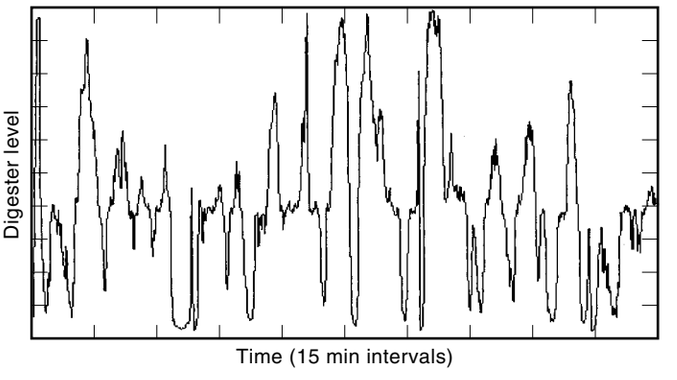

Figure 1. Digester level at a 15 min sampling rate.

fication tool that is able to predict the change of the digester level due to an adjustment in blow flow or some other variable would improve process control, resulting in a more stable digester level.

One method of performing system identification is to use the auto-regressive moving average (ARMA) model (1). This model is equipped with enough flexibility so that it is able to model an arbitrary linear dynamic system. The ARMA model performs poorly, however, when the system contains nonlinear characteristics. To get adequate performance, several ARMA models would have to be used as linear incremental models over small regions of the input space, which is impractical for problems using large data sets. Just as linear modeling techniques such as ARMA are troubled by nonlinear characteristics, nonlinear modeling techniques have trouble identifying exact linear relationships within a system. It is advantageous to have a modeling technique which has the ability to capture both the linear and nonlinear components of a system equally well. A method of dealing with this problem is discussed in Ref. 2 and is described later in this article.

In this research, artificial neural networks (ANNs) are applied to the task of dynamic nonlinear system identification. ANNs were inspired by naturally occurring neural systems whose learning and computational abilities in many tasks are far superior to the current capabilities of much faster digital computers. This is apparent, for example, when comparing the superior pattern recognition ability of a small child with that of the most powerful supercomputer. A small child is able to distinguish the spoken word from sound wave inputs and is even able to assign meaning to the words, whereas the most reliable way that words can be put into a computer is still via the keyboard. The field of neural networks represents an attempt to understand how neural systems process information. By doing so, it is possible to implement algorithms that attempt to model neural information processing. For many problems, these systems produce superior results when compared to those obtained by traditional computational methods. Generally, neural systems consist of many simple, independent, and interconnected processing units known as neurons (also referred to as nodes). The neurons process information in parallel and are connected to each other via synapses (also called weights in ANN terminology). The weights are adjustable, and they are responsible for the storage of information within the neural system. Learning in a neural system corresponds to the adjustment of weights. The ability to learn is one of the most useful properties of neural systems, because it often allows them to learn relationships among input variables which the human observer cannot discern.

In recent years the field of neural network has become a popular paradigm for both function approximation and classification. In this article, function approximation is the main concern. The goal of a function approximation problem is summed up as follows: Given a system with an unknown functional relationship [Eq. (1)], construct an approximation to $f(\boldsymbol{x})$ based on a set of N input and output observations [Eq. (2)].

$$y = f(\boldsymbol{x}), \qquad f(\boldsymbol{x}) : R^n \mapsto R \tag{1}$$
$$\{\boldsymbol{x}_i\}\, y_i, \qquad i = 1, 2, \ldots, N$$

$$y_i = f(\boldsymbol{x}_i) \tag{2}$$

The approximation is denoted as $\hat{f}(\boldsymbol{x})$ as shown in Eq. (3), where e_i represents the *observation error* (also called the residual error) of $\hat{f}(\boldsymbol{x})$ for observation i:

$$y_i = \hat{f}(\boldsymbol{x}_i) + e_i \tag{3}$$

However, the observation error alone is not an adequate measure of the approximation. The *generalization* of $\hat{f}(\boldsymbol{x})$ must also be considered. Generalization describes how well $\hat{f}(\boldsymbol{x})$ matches the values of $f(\boldsymbol{x})$ when $\boldsymbol{x} \notin \{\boldsymbol{x}_i\}$, $\boldsymbol{x} \in D$, where D is the domain over which $f(\boldsymbol{x})$ operates. If the model $\hat{f}(\boldsymbol{x})$ is only valid at the observation points (corresponding to poor generalization), then it is useless, since a simple look-up table could perform the same task. One pitfall that should be avoided when using Artificial Neural Networks (ANNs) is *overfitting,* which occurs when the network becomes too specifically tuned to the training observations, resulting in poor generalization.

While there are many proposed techniques to perform off-line training, only a few of these are discussed in this article. One distinction of this research is that it uses local, rather than global, networks. A global network is made up of neurons whose activation functions respond strongly to inputs over a large portion of the input space. One disadvantage of global networks is that a single node can be easily activated by very dissimilar inputs, which causes network training to be more difficult and time-consuming. A global activation function is also a disadvantage when on-line training is considered, since a very large amount of neurons and their corresponding weights must be altered each time the network needs adjustment in order to better fit the system. In contrast, the neurons in local networks respond only to limited regions of the input space, known as *receptive fields.* Local nodes have an advantage over globally active units in that they can adjust themselves to model the particular characteristics of each subregion of the input space. The importance of this property is evident in the discussion of multiresolution analysis (MRA) presented later in this article.

This article consists of four major sections. First, background information on function approximation using local basis functions is given. In this section, the mathematics necessary to understand later concepts are reviewed. Also, the use of wavelet functions as an MRA basis is discussed, followed by a discussion of radial basis functions (RBFs). Second, several off-line locally active network architectures are presented. Both wavelet-based and RBF-based networks are described and compared. In the third section, the three network architectures used for this study are described. Finally, experimental results are given for several problems, including the prediction of digester level.

BACKGROUND

This section reviews the mathematical tools used to construct and train networks of locally active units for function approximation. Following a description of dynamic system modeling using static function approximators is an explanation of the expansion of functions over a local basis and the basic theory behind multiresolution analysis using a wavelet basis. Finally, the use of radially symmetric basis functions and their ability to perform multiresolution analysis are considered.

Dynamic System Modeling Using a Static Model

The systems to be modeled in this research are multivariable, nonlinear dynamic systems. The neural networks considered are static in nature, meaning that the network response at discrete time step, k, is a function only of the current input, $\boldsymbol{x}(k)$. In contrast, the response of a dynamic system is a function not only of the current input but also of the past inputs and responses. It is possible to model such dynamic systems using a static model as expressed in Eqs. (4) and (5), where $\hat{y}(k+1)$ is the predicted system response and $\boldsymbol{u}(k)$ is an input vector whose elements consist of the dependent variables in the system:

$$\hat{y}(k+1) = \hat{f}(\boldsymbol{x}(k)) \tag{4}$$

$$\boldsymbol{x}(k) = [y(k),\, y(k-1),\, \ldots,\, y(k-n_y),\, \boldsymbol{u}(k),\, \boldsymbol{u}(k-1),\, \ldots,\, \boldsymbol{u}(k-n_u)] \tag{5}$$

By including n_y previous responses and n_u previous inputs into $\boldsymbol{x}(k)$, the static function approximator is able to model the dynamic system by constructing a mapping between the space spanned by $\boldsymbol{x}(k)$ and the future system response $y(k+1)$. When using this method it is important to select good values for n_y and n_u. If these values are too small, the mapping between $\boldsymbol{x}(k)$ and $y(k+1)$ may not represent a functional relationship and the approximation is poor. If the values are larger than necessary, the dimension of $\boldsymbol{x}(k)$ is larger than it needs to be, which makes the approximation harder to achieve. In general, the difficulty of forming a function approximation increases with the dimension of input space. This concept is known as the *curse of dimensionality*.

Function Expansion on a General Basis

The networks used in this article can be described as functional expansions over a set of local basis functions. Rather than describing a specific basis in this section, an arbitrary basis $\boldsymbol{\Phi} \subset \boldsymbol{H}$ is defined, where $\boldsymbol{H}$ is a Hilbert space with an inner product defined by Eq. (6):

$$\langle f(\boldsymbol{x}),\, g(\boldsymbol{x})\rangle = \int_{x\in R^n} f(\boldsymbol{x}) \cdot g(\mathbf{x})\, d\boldsymbol{x} \tag{6}$$

Length (norm) is identical to the standard definition given by Eq. (7):

$$\|f(\boldsymbol{x})\| = \sqrt{\langle f(\boldsymbol{x}),\, f(\boldsymbol{x})\rangle} \tag{7}$$

$\boldsymbol{\Phi}$ contains M basis functions, each one denoted as $\phi_i(\boldsymbol{x})$, $i = 1, 2, \ldots, M$. An expansion over this basis is expressed in Eq. (8), where w_i is the expansion coefficient for the ith basis function:

$$\hat{f}(\boldsymbol{x}) = \sum_{i=1}^{M} w_i \cdot \phi_i(\boldsymbol{x}) \tag{8}$$

In order to exactly represent an arbitrary function $f(\boldsymbol{x})$, the basis must satisfy $f(\boldsymbol{x}) \in \text{span}(\boldsymbol{\Phi})$. If this is the case, then $f(\boldsymbol{x})$ can be represented by Eq. (9), with the error residual, $e(\boldsymbol{x})$, equal to zero:

$$f(\boldsymbol{x}) = \hat{f}(\boldsymbol{x}) + e(\boldsymbol{x}) \tag{9}$$

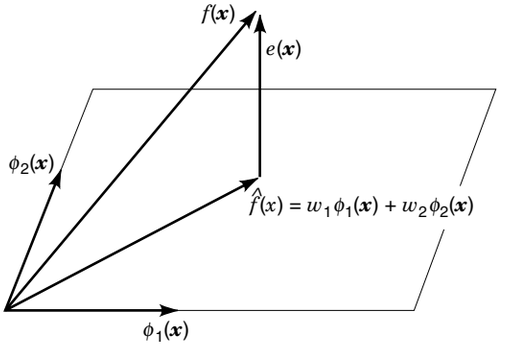

Figure 2. Geometric interpretation of Eq. (8).

When $f(\boldsymbol{x}) \notin \text{span}(\boldsymbol{\Phi})$, the best approximation to $f(\boldsymbol{x})$ is achieved by selecting the expansion coefficients so that $\hat{f}(\boldsymbol{x})$ represents the projection of $f(\boldsymbol{x})$ onto span($\boldsymbol{\Phi}$). In such a case, $\|e(\boldsymbol{x})\| \geq 0$, but is minimized, and $e(\boldsymbol{x})$ is orthogonal to span($\boldsymbol{\Phi}$). A simple geometric interpretation of Eq. (9) is shown in Fig. 2.

Equation (9) indicates that the task of function expansion is to select a basis set which spans a subspace of $\boldsymbol{H}$ and is as close as necessary to $f(\boldsymbol{x})$. The expansion coefficients are then obtained by projecting $f(\boldsymbol{x})$ onto this subspace. A special case is when the basis set is orthogonal. In order to represent an orthogonal basis set, the condition in Eq. (10) must be met, which states that the energy contained in each basis element is independent of the others.

$$\langle \phi_i(\boldsymbol{x}),\, \phi_j(\boldsymbol{x})\rangle = 0, \qquad i \neq j \tag{10}$$

Finding the optimal expansion coefficients for an orthogonal basis set is a simple matter of projecting $f(\boldsymbol{x})$ onto each basis element, as given by Eq. (11):

$$w_i = \frac{\langle f(\boldsymbol{x}),\, \phi_i(\boldsymbol{x})\rangle}{\|\phi_i(\boldsymbol{x})\|^2} \tag{11}$$

Another special case is when the basis set is biorthogonal. In this case there is another basis set $\tilde{\boldsymbol{\Phi}}$ which corresponds to $\boldsymbol{\Phi}$. The expansion coefficient is found by projecting $f(\boldsymbol{x})$ on each element in $\tilde{\boldsymbol{\Phi}}$, as shown by Eq. (12):

$$w_i = \frac{\langle f(\boldsymbol{x}),\, \tilde{\phi}_i(\boldsymbol{x})\rangle}{\|\tilde{\phi}_i(\boldsymbol{x})\|^2} \tag{12}$$

For nonorthogonal basis sets, the expansion coefficients are found using pseudoinversion techniques. For more information on Hilbert spaces and function expansions the reader is referred to Ref. 3.

Multiresolution Analysis (MRA) Using a Wavelet Basis

When approximating a function from observations, it is important to consider the local characteristics of the function. In some regions of the spatial domain, observations are dense and the function may contain high frequencies (*details*), while other spatial regions have less observations and contain mostly low frequencies (*coarse* function characteristics). A function with such local characteristics may be difficult to

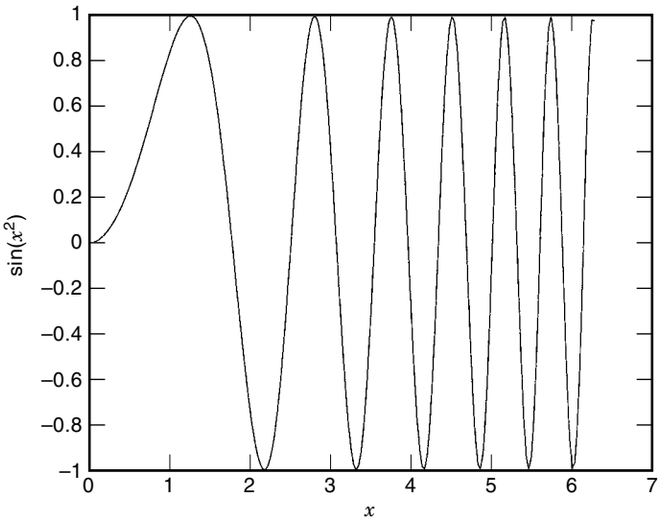

Figure 3. Plot of $y = \sin(x^2)$.

model and analyze using traditional Fourier techniques which use sinusoidal basis functions. Although sinusoidal basis functions have excellent frequency localization properties (each sinusoid represents one frequency), they are not localized in the spatial domain, because they are global functions. This lack of spatial localization does not allow the Fourier basis to model the frequency content of a particular region without also affecting its approximation over the rest of the spatial domain. It is desired that a basis be constructed which has both spatial and frequency localization. By doing so, the local characteristics of a function could be modeled without affecting the approximation outside of the local area. Such a basis would have more success at modeling signals with local irregularities than a spatially global basis. Figure 3 shows a plot of the function $y = \sin(x^2)$.

It is apparent that as the value of x increases, so does the frequency of y (the frequency of y is a linear function of x). If a spatially and frequency localized basis exists, it is possible to model the signal by spatially placing basis functions which represent lower frequencies further to the left. Each basis function would essentially act as a filter which captures the frequency content of a signal within a spatially localized area.

MRA is a modeling and analysis technique where a functional relationship $f(\boldsymbol{x})$ can be decomposed into approximations at multiple resolution levels as shown in Eq. (13), where $\hat{f}_k(\boldsymbol{x})$ represents the approximation at the kth resolution level:

$$f(\boldsymbol{x}) = \sum_{k=0}^{L} \hat{f}_k(\boldsymbol{x}) \tag{13}$$

The resolution level corresponding to $k = 0$ represents the finest details (highest frequencies) of the signal while the resolution level $k = L$ represents the coarsest characteristics of the signal. A discussion is now given concerning how an MRA representation can be constructed using a wavelet basis expansion.

A wavelet basis is made up of translations and dilations of a single mother wavelet $\psi(x)$ as shown in Eq. (14), where s and t are the dilation and translation parameters which give each wavelet their localization properties:

$$\psi_{x,t}(x) = s^{-1/2} \cdot \psi\left(\frac{x-t}{s}\right), \qquad t \in R,\ s \in R^{+} \tag{14}$$

Translation localizes a wavelet in space, while dilation localizes a wavelet in frequency. Wavelets must satisfy three main conditions. First, they must *contain oscillations* above and below the x axis. Second, they must *decay to zero quickly* in both the positive and negative directions. Third, the wavelet should have *zero mean.* These properties are discussed more precisely in Ref. 4. Figure 4(a–c) shows the effect of translation and dilation in the spatial and frequency domains for the Mexican hat wavelet. It is apparent from Fig. 4(b,c) that wavelets which contain wider and higher-frequency bands are more localized in space than wavelets which contain narrower and lower-frequency bands. In other words, spatially wide wavelets are used to represent low-frequency regions, whereas spatially narrow wavelets are used to represent lower frequencies, which agrees with intuition. This property of wavelets is governed by the uncertainty principle, which states that we cannot simultaneously improve the frequency and time resolutions (5).

For digital implementations a discrete form of the wavelet exist which takes the form of Eq. (15), where a and b are discrete scale and translation step sizes:

$$\psi_{kl}(x) = a^{-k/2} \cdot \psi(a^{-k} \cdot x - l \cdot b), \qquad k,\ l \in Z \tag{15}$$

In general for this research, $a = 2$ and $b = 1$, which indicates that the spatial resolution changes by a factor of two for each discrete scale step. A multiresolution basis is constructed by creating a discrete lattice of wavelet functions, where each element in the lattice has a unique k, l combination. Such a discrete lattice is called a frame and is introduced in Ref. 6 and expanded upon in Ref. 7.

To perform MRA using a wavelet basis, the function considered must work in tandem with the wavelet called a *scaling function.* For each wavelet, ψ_{kl}, there is a corresponding scaling function, ϕ_{kl}. Scaling functions have a useful property which is expressed below in Eqs. (16) and (17), where $F_k(k)$ is an approximation to $f(x)$ at resolution level k, and c_{kl} represents the expansion coefficients for each scaling function, which are obtained by projecting $f(x)$ onto the space spanned by the scaling functions:

$$\hat{F}_k(x) = \sum_{l \in Z} c_{kl} \cdot \phi_{kl}(x) \tag{16}$$

$$\hat{F}_{k-1}(x) = \hat{F}_k(x) + \hat{f}_k(x) \tag{17}$$

Equations (16) and (17) show that an expansion over the scaling functions at resolution $k - 1$ (finer resolution) is equivalent to an expansion over the scaling functions and wavelet functions at resolution k (coarser resolution). The wavelet expansion, $\hat{f}_k(x)$, represents the detail added when progressing from a coarser approximation at resolution k to a more detailed approximation at resolution $k - 1$.

The most useful wavelet basis sets are orthogonal. Given $\boldsymbol{W}_k \subset \boldsymbol{H}$, which is the space spanned by the orthogonal wavelet at resolution k, Eq. (18) shows that the wavelet spaces

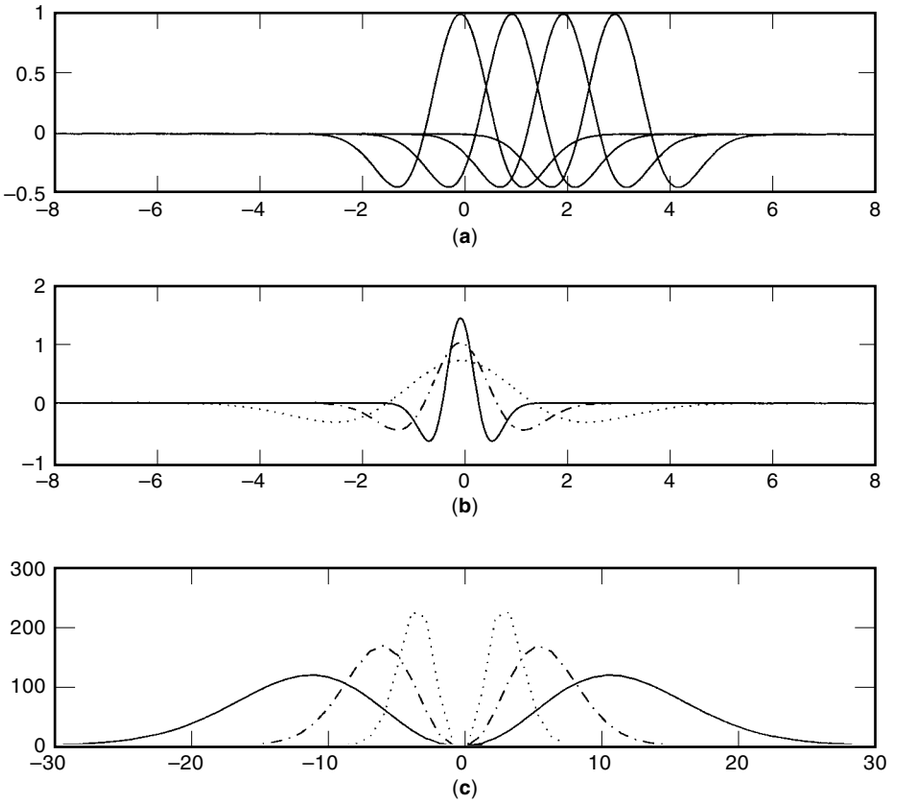

Figure 4. (a) Translations of the Mexican hat wavelet. (b) Dilations of the Mexican hat wavelet (spatial domain). (c) Dilation of the Mexican hat wavelet (frequency domain).

spanned at each resolution level for an orthogonal wavelet basis are completely independent:

$$\boldsymbol{W}_k \perp \boldsymbol{W}_j, \qquad k \neq j \tag{18}$$

Also, the space spanned by the scaling functions at resolution k is defined as $\boldsymbol{V}_k \subset \boldsymbol{H}$, which also represents an orthogonal subspace. The subspaces of the scaling functions at different resolutions are not orthogonal as is the case in Eq. (18). Instead, these spaces are subsets of one another.

$$\ldots \subset \boldsymbol{V}_2 \subset \boldsymbol{V}_1 \subset \boldsymbol{V}_0 \subset \boldsymbol{V}_{-1} \subset \boldsymbol{V}_{-2} \subset \ldots \tag{19}$$

Equation (17) demonstrates a very important relation between the wavelet and scaling function spaces.

$$\boldsymbol{V}_{k-1} = \boldsymbol{V}_k \oplus \boldsymbol{W}_k \tag{20}$$

Using the relation in Eq. (20), it is possible to form approximating subspaces at arbitrarily high resolutions by beginning with a low-resolution scaling space and including the wavelet subspaces at successively higher resolutions until the desired approximation accuracy is obtained. Finding the coefficients for an orthogonal wavelet space requires that the function to be approximated is projected onto each scaling and wavelet function to be considered, as was demonstrated by Eq. (11).

In recent years several orthogonal, one-dimensional wavelet sets have been constructed. The functions have found many uses in signal processing applications. The extension of these functions to two dimensions is discussed in Ref. 7, and the results are shown below:

$$\Phi_{mn}(x_1, x_2) = \phi_{mn_1}(x_1) \cdot \phi_{mn_2}(x_2) \tag{21a}$$

$$\Psi^1_{mn}(x_1, x_2) = \phi_{mn_1}(x_1) \cdot \psi_{mn_2}(x_2) \tag{21b}$$

$$\Psi^2_{mn}(x_1, x_2) = \psi_{mn_1}(x_1) \cdot \phi_{mn_2}(x_2) \tag{21c}$$

$$\Psi^3_{mn}(x_1, x_2) = \psi_{mn_1}(x_1) \cdot \psi_{mn_2}(x_2) \tag{21d}$$

These functions form an orthogonal basis in $L^2(R^2)$. It is necessary to have three wavelet functions in order to completely define the basis; each function filters information at different orientations in the input space. The above result can be generalized to R^n. The number of wavelet functions, however, is equal to $(2^n - 1)$, which means that $(2^n - 1)$ expansion coefficients must be computed for each frame element. This becomes computationally expensive for higher dimensions. One initial concern when considering wavelet neural network implementations is how to keep the advantageous properties of wavelets when the dimension of the input space is large. The wavelet neural networks discussed afterward confront multidimensional spaces in different ways. For more information on wavelets and their applications, the reader is referred to Refs. 3 and 8.

Using Radially Symmetric Basis Functions

Radial basis functions (RBFs) are commonly used for function approximation in the field of neural networks. RBFs are local functions which are defined by a center, $\boldsymbol{c} \in R^n$, and an $n \times n$ diagonal matrix, Σ^{-1}, having $1/\sigma_j^2$ for the diagonal compo-

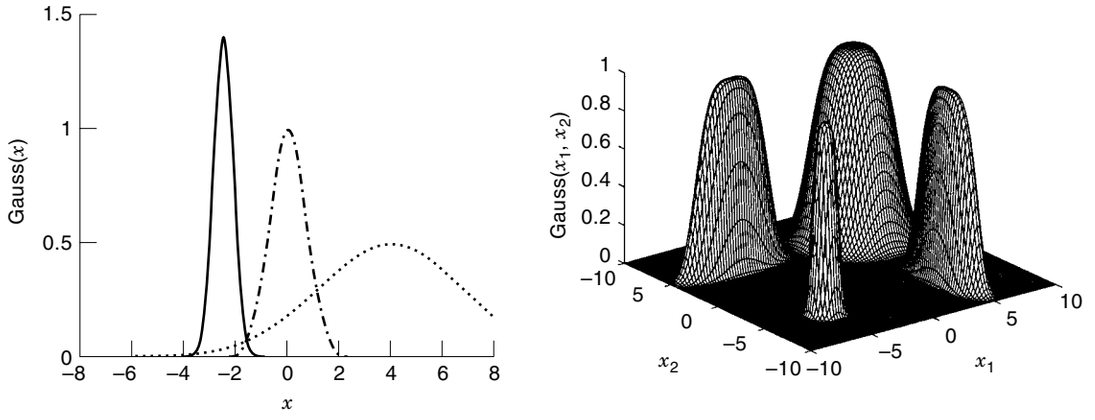

Figure 5. (a) One-dimensional Gaussian functions. (b) Two-dimensional Gaussian functions.

nents. The center defines the location of the basis function in the input space, while Σ^{-1} defines the shape of the functions receptive field. One of the most commonly used RBFs is the multidimensional Gaussian function given by Eq. (22), where $d_i(\boldsymbol{x})$ represents a weighted distance measure which defines an n-dimensional hyperellipsoid [Eq. (23)]:

$$\phi_i(\boldsymbol{x}) = e^{-d_i(\boldsymbol{x}))^2/2} \tag{22}$$

$$d_i(\boldsymbol{x}) = \sqrt{(\boldsymbol{x} - \boldsymbol{c}_i) \cdot \Sigma_i^{-1} \cdot (\boldsymbol{x} - \boldsymbol{c}_i)^{\mathrm{T}}} = \sqrt{\sum_{j=1}^{n} \frac{(x_j - c_{ij})^2}{\sigma_{ij}^2}} \tag{23}$$

Figure 5(a) shows examples of one-dimensional Gaussian functions. The width of the receptive field can be changed by modifying Σ^{-1}. Functions with narrow receptive fields contain higher frequencies than those with wide receptive fields. Changing Σ^{-1} is analogous to changing the dilation value of a wavelet. Figure 5(b) shows examples of two-dimensional Gaussian functions with different centers and Σ^{-1}.

It is important to notice that for different Σ^{-1}, the major axis of the ellipsoidal receptive field along each dimension changes length. This permits control of how fast the Gaussian function decays to zero along each dimension. This is an important property since it allows the scaling of a Gaussian function to cover a particular input space. For example, consider a two-dimensional space for which one variable ranges from -1000 to 1000, and the second variable ranges from -0.1 to 0.1. If a standard euclidean distance measure is used to define the receptive field by setting $\Sigma^{-1} = \boldsymbol{I}$, the Gaussian response is the same in each input direction. Such a Gaussian function models the input space poorly since it barely detects a variation in the second dimension compared to the first dimension. It is necessary to adjust Σ^{-1} so that the Gaussian decay is related to the data range covered by each dimension. If this is done properly, it is equivalent to normalizing each input dimension and using a circular receptive field ($\Sigma^{-1} = \boldsymbol{I}$).

A basis made up of RBFs is not orthogonal by nature, as opposed to a wavelet basis. Like wavelets, however, an RBF basis can be set up to perform MRA by having several different resolution levels of RBF units. The chief advantage of using a wavelet basis to perform MRA is its orthogonality property. Because the levels of an RBF MRA lattice are not orthogonal, the lattice contains redundant information (the energy contained by each element is not independent), and an approximation on such a grid may not be as efficient as that obtained using an orthogonal basis. Also, the coefficients may be more time-consuming to compute for an RBF basis. The following section considers the ability for a wavelet basis to maintain these advantages for high-dimensional, irregularly sampled spaces. It is shown that the advantages offered by a wavelet basis do not come into effect for many practical problems.

NETWORK IMPLEMENTATIONS

This section presents the construction of basis function networks to be used as function approximators. The general method of network construction is discussed, followed by a description of various networks which use wavelet basis functions and various techniques used to construct networks utilizing an RBF basis set.

General Network Construction from an Arbitrary Basis

Figure 6 represents the network model used for this research. Each circle corresponds to a functional node, ϕ_i, of the network which accepts an input, $\boldsymbol{x} \in R^n$, and generates an output. All of the outputs, $\phi_i(\boldsymbol{x})$, are multiplied by a corresponding weight, w_i, and then summed to obtain the network output, $\hat{f}(\boldsymbol{x})$. This architecture matches the model defined by Eq. (8) exactly. Generally, the network makes an approximation of a function based on a set of training observations as shown by Eq. (2). The problem can then be viewed in matrix form, Eq. (24), where $\boldsymbol{Y}$ is an $M \times 1$ column vector whose elements are the set $\{y_i\}$, $\boldsymbol{\Phi}$ is an $M \times N$ matrix whose columns are made up of the activation values of a single node, $\phi_i(\boldsymbol{x})$, for each $\boldsymbol{x} \in \{\boldsymbol{x}_i\}$, and $\boldsymbol{w}$ is an $N \times 1$ column vector whose elements represent the weight or expansion coefficient for each node:

$$\boldsymbol{Y} = \boldsymbol{\Phi} \cdot \boldsymbol{w} \tag{24}$$

Training the network corresponds to solving the system of linear equations, Eq. (24), while still attempting to ensure

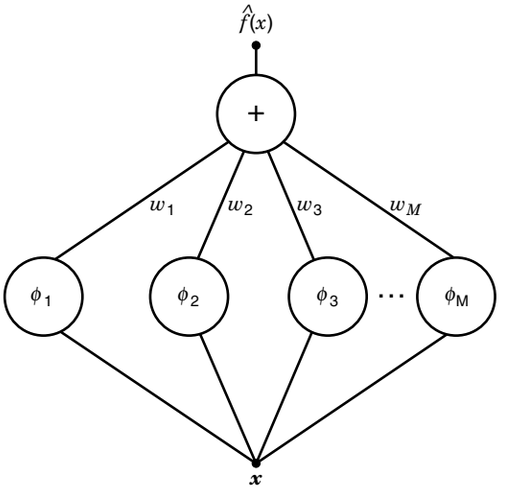

Figure 6. General basis function network.

proper generalization. Every network discussed in this article can be described by this basic model. Two main tasks are involved when training a basis function network which is represented by Eqs. (8) and (24). First, the network structure ($\mathbf{\Phi}$) must be determined, which corresponds to selecting appropriate basis functions which give the network enough flexibility to perform the desired approximation. Second, the weights (expansion coefficient) must be determined which provide the best approximation to the target function. The differences between the various networks that are presented are due to different ways of determining the network structure and weights.

Wavelet Networks

In this section, work done in the field of function approximation using wavelet networks is considered. It is important to pay attention to how each network contends with high dimensional spaces, since this is a potentially troublesome problem when using wavelets. Also, it is important to consider how the orthogonality of a wavelet basis aids the network, if at all for high-dimensional spaces.

Zhang and Benveniste present a wavelet based network in Ref. 9. In this design, the number of nodes to be used in the network is arbitrarily selected. Each node used the Mexican Hat wavelet (second derivative of the Gaussian function) for an activation function. This wavelet constitutes a nonorthogonal basis; therefore, the results obtained do not yield any information about the advantages of using an orthogonal wavelet basis. To deal with multiple dimensions, the activation function of each node is defined by Eq. (25), where $\boldsymbol{D}_i$ is the dilation matrix (same role as Σ^{-1} for RBFs), $\boldsymbol{t}_i$ is the translation vector (same role as $\boldsymbol{c}_i$ for RBFs), and $\boldsymbol{R}_i$ is a rotation matrix which rotates the receptive field of the wavelet in order to account for the various possible orientations:

$$\begin{aligned} \phi_i(\boldsymbol{x}) &= \Psi(\boldsymbol{D}_i\boldsymbol{R}_i(\boldsymbol{x}-\boldsymbol{t}_i)) \\ \Psi(x) &= \psi(x_1)\psi(x_2)\ \ldots\ \psi(x_n), \qquad x \in R^n \end{aligned} \tag{25}$$

A *gradient descent* method is used to adjust the network parameters for network training. This method is similar to the way a backpropagation network (10) is trained. An error measure for a particular parameter set, $\boldsymbol{P}$, is defined to be the sum-squared error between the network approximation and the desired network responses [given by Eq. (2)] and is expressed by Eq. (26), where $\hat{f}_{\boldsymbol{P}}(\boldsymbol{x})$ is the network output for parameter set $\boldsymbol{P}$:

$$E(\boldsymbol{P}) = \sum_{j=1}^{N} (y_j - \hat{f}_{\boldsymbol{P}}(\boldsymbol{x}_j))^2 \tag{26}$$

It is desired to find the parameter set which makes this error small (note that minimizing the error may cause poor generalization). The partial derivative of $E(\boldsymbol{P})$ is found for each parameter ($\boldsymbol{D}_i$, $\boldsymbol{R}_i$, $\boldsymbol{t}_i$, w_i), and the parameters are iteratively updated using Eq. (27), where $\boldsymbol{p}_k$ is the kth parameter in $\boldsymbol{P}$:

$$\boldsymbol{p}_k(t+1) = \boldsymbol{p}_k(t) - \frac{\partial E(\boldsymbol{P})}{\partial \boldsymbol{p}_k} \tag{27}$$

The error surface for the parameter space is very rough and plagued by local minima. This makes the convergence of such a technique very unstable and slow. The authors describe a method for selecting initial values for the parameters, which slightly improves the performance. Results are given for only one- and two-dimensional functions since the network would be very difficult to train for problems of higher complexity. As pointed out in Ref. 11, this approach is essentially a slight variation of an RBF network (to be described later) trained using gradient descent, a training method which has been shown to be inferior to other training methods.

Zhang et al. present another implementation of a wavelet network in Ref. 11. In this network, scaling functions from a single resolution are used for the node activation levels. To contend with multiple dimensions, the node activations are products of the scaling function calculated for each dimension. The following brute force training method is used. First, select a scale level of $k = k_0$ and use the scaling functions at that resolution to generate the $\mathbf{\Phi}$ matrix of Eq. (24). Then, use the generalized matrix inverse to solve for the weights of the least-squares solution as given by Eq. (28):

$$\boldsymbol{w} = ((\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi})^{-1}\mathbf{\Phi}^{\mathrm{T}})\boldsymbol{Y} \tag{28}$$

If the solution using the weights given by Eq. (28) does not produce an acceptable result (large error), then the resolution is decreased by one (resulting in a finer scale) and the process is continued until the appropriate scale level is determined. One problem that becomes apparent is that for high-dimensional spaces, far too many scaling functions are needed to adequately cover the space. In an attempt to solve this problem, the researchers use a principal component analysis (PCA) technique which reduces the dimension of the input space. This technique may perform well for some input spaces; but for high-dimensional spaces in which most components are significant, PCA loses information, leading to poor results. This network implementation does not take advantage of MRA (all the functions are from the same scale). The results given compare RBF performance to that of the wavelet network. It is shown that the wavelet network and RBF network achieved results of the same accuracy using the same number of nodes. The chief advantage given for using the wavelet network over the RBF network was that the RBF network requires an amount of trial and error during training while the wavelet network does not. This wavelet network,

however, uses an automated trial-and-error process to find the correct scale level.

Bakshi and Stephanopoulos present another wavelet-based network, Wave-Net, in Ref. 12. This network utilizes the concepts of both orthogonal wavelets and MRA. Both scaling and wavelet functions are used as the node activation functions. For multidimensional spaces, generalizations of Eq. (21) are used. This produces $(2^n - 1)$ wavelet functions for each grid point of an MRA lattice. Wave-Net is trained by first defining an MRA lattice, which is done by setting the finest resolution level ($k = 0$) in each dimension to have a spacing equal to the smallest sampling rate for the particular dimension. The rest of the lattice is constructed by decreasing the resolution of the coarser levels by a factor or two, until only two units are present in each dimension. The scaling functions at an adequately coarse resolution are then used to construct a network and find a solution using Eq. (28). If this solution is not acceptable, then wavelets at successively finer resolutions are added to the network (increasing the detail of the approximation) until the solution is acceptable. Wavelets and scaling functions at each resolution level which contribute least to the approximation (small weights) are then removed from the network. This network takes full advantage of the orthogonal MRA. It can be seen from this network, however, that using orthogonal wavelets does not help solve for the weights when the space is *nonuniformly sampled.* In this case, the weights must be found using Eq. (28) rather than using the projection of the target function onto each basis unit. Also, this network is still plagued by the curse of dimensionality, since the number of nodes can be very large for high-dimensional spaces. Results are not given for problems with a dimension greater than two.

Finally, Chen and Bruns present the WaveARX network in Ref. 2. This network uses the nonorthogonal Mexican Hat wavelet as an activation function. To confront multiple dimensions, a radial wavelet function is defined to be the activation of each node, as shown in Eq. (29), where $\boldsymbol{t}_i$ and s_i are the translation and dilation parameters, respectively:

$$\phi_i(\boldsymbol{x}) = \psi\left(\left\|\frac{\boldsymbol{x} - \boldsymbol{t}_i}{s_i}\right\|\right) \tag{29}$$

By defining the activation function in this way, the orthogonality property of wavelets for higher dimensions is lost, although in this case it did not matter since the wavelet is already nonorthogonal. This method reduces the number of wavelet units needed in the network for higher dimensions. To train the network, an MRA lattice is defined over the input space, and those wavelets for which training data fall within their receptive fields are selected as network candidate neurons. Another set of $n + 1$ neurons, where n is the input dimension, is added to the candidate set. These are called the autoregressive external (ARX) input nodes, whose responses are linear with respect to the input variables. An extra neuron is added as a bias unit which produces a constant output. Thus the wavelet nodes represent the nonlinear portion of the approximation, while the ARX nodes represent the linear portion. An *orthogonal search* procedure, which is described below, is then used to select candidate neurons to be added to the network in the order of their importance to the approximation. As each neuron is added, the weights of the network are found using Eq. (28) to solve for the approximation error. When the approximation error is small, no additional neurons are added and training ends. The method of orthogonal search has been used before (as will be seen in the RBF section), but the chief advantage of this method over previous orthogonal-search-based methods is that the candidate set of neurons represents many resolutions, offering more flexibility to the network structure. The authors claim that the use of wavelets in the WaveARX network was a chief advantage. However, it seems that the chief advantage lies in the MRA orthogonal search combination, and not in the fact that a wavelet basis function was used. In this article it is proposed that using RBF nodes instead of wavelet nodes in this network would produce nearly the same results.

For completeness, a discussion of the orthogonal search algorithm is now given. Using the form in Eq. (24), it is helpful to view $\boldsymbol{Y}$ as a linear combination of the columns of $\boldsymbol{\Phi}$, each of which represents the trace of a single basis function over the training observations. Some of these columns are more important to the approximation than others. The orthogonal search algorithm is a way to measure the importance of each column (and thus each basis function) in $\boldsymbol{\Phi}$. The basis functions are then included in the order of their importance until an acceptable network performance is reached. The importance of each column is determined by calculating how much unique energy a column has in the direction of $\boldsymbol{Y}$. The uniqueness criteria implies that each time another column is added to the network, it is the one which has the maximum amount of useful energy which is not already present in the approximation. To determine this, the Gram–Schmidt orthogonalization procedure is used. First, an energy reduction ratio (ERR) is defined, which indicates how much energy a column, ϕ, contains in the direction of $\boldsymbol{Y}$:

$$\mathrm{ERR}(\phi) = \left(\frac{\phi^{\mathrm{T}}\boldsymbol{Y}}{\|\phi\|}\right)^2 \tag{30}$$

The algorithm proceeds as follows:

1. *Initialization.* Set the columns of $\boldsymbol{\Phi}$ to be equal to the traces of each basis function, $\phi_i(\boldsymbol{x})$. Create an empty set, $\boldsymbol{I} = \{\ \}$ to store the indexes of the selected nodes.
2. *Node Selection.* Determine the index, k, which satisfies

   $$\max(\mathrm{EER}(\phi_k)), \qquad 1 \le k \le N,\ k \notin \boldsymbol{I}$$

   Add k as an element of set $\boldsymbol{I}$.
3. *Orthogonalization Step.* Remove the component of ϕ_k from all candidates in $\boldsymbol{\Phi}$ as shown below:

   $$\phi_i = \phi_i - \left(\frac{\phi_k^{\mathrm{T}}\phi_i}{\|\phi_k\|^2}\right)\phi_k, \qquad 1 \le i \le N,\ i \notin \boldsymbol{I}$$

4. *Calculate a Termination Condition.* This condition is the approximation error found using the basis functions contained in $\boldsymbol{I}$.
5. *Loop.* If the termination criterion was not satisfied, go to step 2.

RBF Networks

The RBF network with a single layer of nodes, each having the same width, has been shown to possess the universal ap-

proximation property in Ref. 13, and it is discussed further in Ref. 14. This property, however, reveals nothing about methods to obtain such an approximation. Much work has been done trying to solve this problem, part of which is reviewed in this section.

The goal is to use the system defined by Eq. (24) to form the approximation $\hat{f}(\boldsymbol{x})$. For all cases in this section the weights, $\boldsymbol{W}$, are found using Eq. (28). The main task is to select a set of basis functions which result in a stable solution of Eq. (24). The stability of a solution in this case corresponds to the degree of smoothness of the approximation. The degree of smoothness represents how the approximation behaves for the input space outside of the training set and dictates the generalization ability of the network. For example, if the widths of an RBF network are selected to be too small, then the approximation may pass through the training points as desired; however, the rest of the approximation may be very unstable, constituting poor generalization. Also, if the widths are selected to be too large, then the basis function matrix, $\boldsymbol{\Phi}$, may be ill-conditioned (since its columns will be strongly dependent), resulting in unstable weights.

To obtain a stable solution, the RBF nodes must cover the input space of the approximation, which is accomplished by selecting appropriate centers, and the node widths are selected to yield good generalization, by selecting good values for the elements of Σ^{-1}. It is also desired to keep the number of RBF nodes as small as possible. The simplest method of selecting the centers of a network is to create an *exact network,* which means that one center is selected for each training data point. There are several ways of selecting the widths for such a network, one being to assign each node the same Σ^{-1} and use trial and error to find an appropriate value. Another approach is to use a P-nearest-neighbor heuristic which considers the P nearest neighbors of a particular node to calculate Σ^{-1}. Some of these heuristics are discussed in Ref. 15. The obvious dilemma with an exact network is that the number of nodes grows very large for large training data sets.

It is generally possible to get a better or comparable approximation of an exact network using far fewer nodes. One way to reduce the number of nodes is to use a clustering algorithm to distribute the nodes relatively evenly over the input space. A common clustering algorithm is known as k-means clustering, which is used in several RBF implementations, including Ref. 16. Figure 7 shows the results of using 50 nodes (stars) to cluster a two-dimensional random data set consisting of 1000 points (small dots).

It is observed that the centers are distributed uniformly over the input space. This clustering algorithm, as well as the others used in this research, is performed using a slight modification to the k-means algorithm. The k-means algorithm is explained below, followed by the modification which is referred to as the Robin Hood variation.

The k-means algorithm performs the operation of assigning K nodes to the means of K clusters of data. This is accomplished by the following iterative algorithm.

1. Initialization
   - Let $\{\boldsymbol{x}_i\}$, $\boldsymbol{x}_i \in \boldsymbol{R}^n$, $i = 1, \ldots, M$, be the set of input vectors.
   - Generate a set of K randomly distributed cluster centers $\{\boldsymbol{c}_j\}$, $\boldsymbol{c}_j \in \boldsymbol{R}^n$, $j = 1, \ldots, K$.

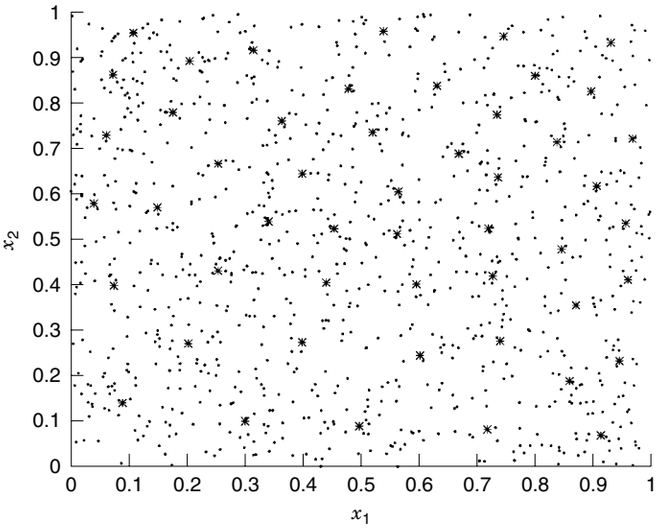

Figure 7. Clustering example. *, clusters; ., data points.

2. Cluster Formation
   - Assign each input $\boldsymbol{x}_i$ to a set $\{\boldsymbol{g}\}_j$ of vectors whose closest center is $\boldsymbol{c}_j$:

$$\boldsymbol{x}_i \in \{\boldsymbol{g}\}_j \rightarrow \|\boldsymbol{x}_i - \boldsymbol{c}_j\| = \min\{\|\boldsymbol{x}_i - \boldsymbol{c}_l\|\}, \qquad l = 1, \ldots, K$$

3. Center Modification
   - Move each center $\boldsymbol{c}_j$ in the direction of the mean of all the input vectors in $\{\boldsymbol{g}\}_j$. The learning rate parameter η determines the fraction of the distance the center is moved.

$$\boldsymbol{c}_j = \boldsymbol{c}_j + \eta \cdot (\text{mean}(\{\boldsymbol{g}\}_j) - \boldsymbol{c}_j)$$

4. Termination Criteria
   - If the average movement of the centers is below a certain threshold, terminate the algorithm. Else, repeat steps 2 and 3.

$$\text{IF} \frac{1}{K} \sum_{l=1}^{K} \|\text{mean}(\{g\}_j) - \boldsymbol{c}_j\| \geq \alpha$$

THEN GOTO Step 2
ELSE END

Although this algorithm performs well, it has one weakness. There is no guarantee that all K centers represent valid clusters. If an initial center is not closest to any input vector, it is not updated and becomes useless. In order to solve this problem, a Robin Hood variation of the K-means algorithm is used, which robs the rich centers to feed the poor centers. The richness of the jth center equals the number of input vectors assigned to the set $\{\boldsymbol{g}\}_j$. Essentially, input vectors are stolen from the richest centers and given to the centers with $\{\boldsymbol{g}\}_l = \{\,\}$, $l = 1, \ldots, K$. This procedure relocates isolated centers to the most heavily populated regions of the input space. Thus, the clusters provide a better approximation of the input space because more centers are present in the densest regions. Step 3 of the above algorithm can be modified in order to implement the Robin Hood variation.

3. Center Modification
   A. Richness Sorting
      • Let the set $\{\boldsymbol{r}_i\}$, $i = 1, \ldots, K$, contain the elements of $\{\boldsymbol{c}_j\}$ sorted in descending order according to their richness.
   B. Center Adjustment
      • For each center $\boldsymbol{c}_j$, if $\{\boldsymbol{g}\}_j = \{\ \}$, move $\boldsymbol{c}_j$ very close to the richest center that hasn't already been robbed; else move $\boldsymbol{c}_j$ in the direction of the mean of all the input vectors in $\{\boldsymbol{g}\}_j$.

        SET $i = 1$
        IF $\{\boldsymbol{g}\}_j = \{\ \}$,
        THEN $\boldsymbol{c}_j = \boldsymbol{r}_i + \boldsymbol{\delta}$; $i = i + 1$
        ELSE $\boldsymbol{c}_j = \boldsymbol{c}_j + \eta \cdot (\text{mean}(\{\boldsymbol{g}\}_j) - \boldsymbol{c}_j)$

      • Each element of the vector $\boldsymbol{\delta}$ is randomly selected to be $\pm\epsilon$, where ϵ is a very small number.

After the clustering process is complete, the width of the RBF nodes must be selected so as to cover the entire input space. This can be done using various P-nearest-neighbor techniques.

Another method of selecting the node locations and width is presented in Ref. 17. Here, a genetic algorithm is used to evolve populations of RBF nodes toward a good approximation structure. The initial results indicate that this method outperforms the k-means clustering algorithm. However, further study is necessary to draw definite conclusions. One possible weakness of the genetic approach is that it is very sensitive to changes in its many parameters.

Finally, an RBF approach is implemented in Ref. 18, which combines clustering with the orthogonal search. The standard k-means clustering technique and P-nearest-neighbor heuristics are used to select a set of centers and corresponding widths. The orthogonal search algorithm is then used to select the most useful centers until the training error is acceptable. This method achieves more efficient results, with less nodes, than an approximation which uses all of the centers found by the clustering algorithm. This efficiency becomes increasingly important as the dimension of the input space is increased. There are a few differences between this approach and the approach used by the WaveARX network. First, the WaveARX network creates the candidate set of nodes using an MRA grid, where the orthogonal RBF implementation does not include nodes at multiple resolutions for a single spatial region. This means that the WaveARX network has a wider variety of nodes to select from, which gives it an advantage. Second, the RBF implementation does not include the linear terms in the model. There is no reason, however, that an RBF network cannot use linear terms just as effectively as the WaveARX network. Third, the WaveARX network uses the Mexican hat function as opposed to a Gaussian function. It is not believed that the activation function is responsible for the advantages exhibited by the WaveARX network. If an RBF network is given a multiresolution set of nodes and is trained using an orthogonal search procedure, the performance should be very similar to the WaveARX network.

NETWORKS USED FOR THIS STUDY

This section discusses the three networks which will be used in the experimental results section. It is desired that these networks have some form of MRA built into their structure, and are trained using the orthogonal search algorithm to limit the number of nodes. It is assumed that such a network generates a solution which is at least as good as a network trained without the technique. The disadvantage to using an orthogonal search is that for large data sets with large numbers of nodes, the training time can be quite lengthy. To help overcome this problem, the fast orthogonal search (FOS) algorithm is used instead of the more straightforward orthogonal search. The FOS algorithm is presented in Ref. 19 and simply represents a faster way to perform an orthogonal search.

WaveARX Like Methods

The WaveARX network, as described above, is used in the experimental results section of this article. In Ref. 2 the WaveARX network is shown to yield results superior to that of a standard RBF network and also superior to that of previous attempts at wavelet-based networks (no comparison with Wave-Net was given). Because of the high dimension of the system identification problem presented in this research, Wave-Net is not implemented. Recall that Wave-Net requires $(2^n - 1)$ nodes for each grid point. This is impractical for high-dimensional applications. The main goal, then, is to determine what gives WaveARX its advantage. To aid in answering this question, an RBF version of the WaveARX network is implemented, which differs from WaveARX only in the fact that it uses Gaussian activation functions rather than Mexican hat functions. This is similarly called the GaussianARX network. For each case the MRA grid is set up as follows. Begin with the coarsest resolution level so that there are two nodes along each input dimension. The spacing between nodes in each dimension is set according to the span of the training data in each dimension. For successively higher resolution layers, the number of nodes along each dimension is doubled. For the GaussianARX network, the widths are selected so that each Gaussian unit has a value of b at the center of its nearest neighbor in the grid. Generally, a value of b between 0.2 and 0.5 works well. Although many nodes are present in the full MRA grid, only the nodes whose receptive fields cover the input data are used. The FOS algorithm is then used to select the most important basis functions until an acceptable result is reached.

When using the WaveARX or GaussianARX networks, one potential problem concerns the memory and time required for network construction, specifically for the FOS algorithm. This is because with a moderate number of resolution levels (generally three or more), the initial group of nodes given to the FOS algorithm is very large for problems with several dimensions and a large number of data points. Often there are so many initial nodes that the number of resolution levels must be decreased in order to perform the FOS computation in a reasonable time. It is desirable to give the FOS algorithm a multiresolution set of candidate nodes for which the number of nodes is not dependent on the number of data points. This would make the FOS algorithm running time less dependent on the number of data points. The following network is an attempt to address this problem.

Multiresolution Clustering Network

This network uses a multiresolution clustering (MRC) technique in order to generate a candidate set of RBF nodes for the FOS algorithm to select from. MRC is similar to a structure called a multilayer self-organizing feature map which is

used for image segmentation (20). The idea behind both algorithms is to cluster data at successive resolution levels. Essentially the MRC technique begins by using the k-means algorithm to cluster the training data using N_0 nodes. Next, the values of the N_0 node locations are clustered using N_1 centers, where $N_0 > N_1$. The successive clustering is continued until the number of nodes used for clustering is less than 2. For each level of clustering, the widths are selected using a P-nearest-neighbor technique. Such a clustering method produces a set of nodes at multiple resolutions, since as the number of nodes is decreased for each level, the widths of the nodes on that level increase in order to cover the space, which yields a coarser resolution than the previous level.

The goal is to end up with the sets $\{\boldsymbol{c}_i\} \subset R^n$ and $\{a_i\} \subset R$ which represent the center locations and width parameters, respectively, for each candidate node. The width parameter set stores a scalar value for each node which defines a width with respect to the other nodes. It would appear that more than one scalar is necessary to represent the width for each dimension. However, the P-nearest-neighbor heuristic used here simply assigns a Σ_i^{-1} matrix to each node which is a scalar multiple of a common Σ^{-1}, whose diagonal elements are the variances of the training data in each input dimension. The reason this is done is so that the Σ^{-1} matrix has a normalization effect on the input space. Scaling this matrix permits change in the receptive field size of a particular node, which corresponds to changing its resolution. The activation for each node is found using Eqs. (22), (23), and (31), where $\boldsymbol{x}_i$ represents the training data inputs:

$$\begin{aligned} \Sigma_i^{-1} &= a_i \Sigma^{-1} \\ \Sigma^{-1} &= \mathrm{var}(\{\boldsymbol{x}_i\}) \end{aligned} \tag{31}$$

***P*-Nearest-Neighbor Heuristic.** To calculate the scalar width values, a_i, for a set of centers, $\{\boldsymbol{c}_i\}$, solve for a_i using Eqs. (22), (23), and (31) such that the RBF node defined by $\boldsymbol{c}_i$ has an activation value of b at the coordinate defined by the center of its Pth nearest neighbor. Generally $P = 2$ and $b = 0.5$ yield good results.

Now that a method of selecting widths has been devised, the MRC technique is given below:

1. *Initialization.*
   - Select P_0, the number of neurons for the initial layer.
   - Create sets $\boldsymbol{C}$ and $\boldsymbol{A}$ to store the centers, $\boldsymbol{c}$, and width parameters, a, of the neuron candidates.
   - Set $\boldsymbol{C} = \{\ \}$ and $\boldsymbol{A} = \{\ \}$.
   - Define a set of vectors to be clustered, $\boldsymbol{D} = \{\boldsymbol{d}_i\}$. Initialize D to contain the input training vectors.

     $$\{\boldsymbol{d}_i\} = \{\boldsymbol{x}_i\}, \qquad i = 1, \ldots, M$$

   - Set the iteration counter, $k = 0$.
2. *Clustering.*
   - Create a new set of centers, $\boldsymbol{D}' = \{\boldsymbol{d}_i'\}$, $i = 1, \ldots, P_k$, by clustering $\boldsymbol{D}$ with the K-means algorithm and "Robin Hood" criteria.
   - Calculate a set of width parameters, $\boldsymbol{A}' = \{a_i'\}$, $i = 1, \ldots, P_k$, for the centers in $\boldsymbol{D}'$.
   - Add the new centers in $\boldsymbol{D}'$ to $\boldsymbol{C}$ and the new width parameters in $\boldsymbol{A}'$ to $\boldsymbol{A}$.
   - Set $\boldsymbol{D} = \boldsymbol{D}'$, so that the most recent set of centers will be clustered into a smaller set with the next iteration.
   - Calculate the number of neurons for the next layer.

     $$P_{k+1} = \mathrm{int}(\lambda \cdot P_k),\ 0 < \lambda < 1$$

     In general, $\lambda = 0.5$ is used, so that the number of neurons decreases by one-half with each iteration.
   - Increase the counter

     $$k = k + 1$$

3. *Termination Criteria.*
   - If the number of neurons in the current layer ≥ 2, the next layer has at least one neuron and the above process is repeated.

     IF $P_k \geq 2$,
     THEN GOTO Step 2
     ELSE END

After performing the MRC algorithm, use the sets $\boldsymbol{C}$, $\boldsymbol{A}$ as candidate nodes for the FOS algorithm.

EXPERIMENTAL RESULTS

This section presents the results of two experiments performed to address the following concerns. First, it is desired to observe if there is a significant difference between the results obtained by the WaveARX network and those obtained by its Gaussian counterpart. If there is not a significant difference, then it is probable that the advantages achieved by the WaveARX network are due to the MRA orthogonal search combination rather than the activation function. Second, the results of the multiresolution clustering network are compared to those of the WaveARX-like networks in order to see if it possesses any significant advantages or disadvantages. Finally, the ability of such networks to model a real-world nonlinear dynamic systems is discussed.

The first experiment is a one-dimensional function approximation problem which is designed to demonstrate the MRA capabilities of the networks. The second experiment is to model the level of the pulp digester at a paper plant, which is a seven-dimensional problem. Each experiment follows the same procedure. Each of the three networks are trained using a training data set and likewise tested with an independent testing data set. The performances of the networks are compared based on the training error, testing error, and the number of nodes used for the approximation. The error measure to be used is the mean-squared error (MSE) and is given by Eq. (32), where M is the number of observations (samples), y_i is the ith output observation, and $\hat{f}(\boldsymbol{x}_i)$ is the network output for the ith input observation:

$$\mathrm{MSE} = \frac{1}{M} \sum_{i=1}^{M} (y_i - \hat{f}(\boldsymbol{x}_i))^2 \tag{32}$$

One-Dimensional Function

The function to be modeled is the example function shown in Fig. 3, with a linear term added as given by the expression below:

$$y = \sin(x^2) + x, \qquad 0 \leq x \leq 2\pi$$

The training data consisted of 40 evenly spaced function samples, while the testing data contained 200 evenly spaced sam-

Table 1. Results of One-Dimensional Function Approximation

	Training MSE	Testing MSE	Number of Nodes	w_{lin}
MRC	0.0021	0.0041	23	1.122
WaveARX	0.0028	0.0030	26	0.987
GaussianARX	0.0021	0.0047	23	1.098
Standard RBF	0.0025	0.0081	35	0.975

ples. For this example, the three networks described above were trained, along with a standard RBF network which was trained using *K*-means clustering and the *P*-nearest-neighbor heuristic described earlier. For the standard RBF network, the number of nodes was increased until the desired error was reached. Note that a linear term was included in the model so that it could be better compared to the other networks. The standard RBF network is included in this experiment to demonstrate the advantage gained by using MRA, which the standard RBF doesn't support. The networks were all trained until the training MSE reached 0.003 (an arbitrary value which yields good results). Then, the testing MSE was calculated and the number of nodes was recorded. As expected, the linear node corresponding to the input x was selected by the FOS algorithm for each network, and the weight associated with the linear node, w_{lin}, was recorded. It is important to note that the results given for the MRC network represent averages taken over five trials. This is to account for the fact that performance will vary for the MRC networks since the clustering is not exactly the same for each run. This is one disadvantage of the MRC network. The results are given in Table 1.

It is clear from the table that the networks utilizing MRA (the first three in the table) all achieve significantly more efficient network structure (fewer nodes) than the solution reached by the standard RBF. This is the expected result since the frequency concentration of this function changes across the spatial domain, which means that an MRA structure is better suited to make the approximation. No conclusion can be drawn, however, regarding the approximation made by the three MRA-based networks. The Gaussian-based network used three fewer nodes than the WaveARX network. However, the WaveARX network achieved a slightly better testing error. It can be seen that for all networks, the weight associated with the linear node represented the linear portion of the signal very well since the values are all very close to 1.0, which is the slope of the linear portion. To further demonstrate the MRA structure of the networks, Fig. 8 shows the function to be approximated along with the first 10 nodes selected by the FOS algorithm for the GaussianARX network.

Note that the Gaussian nodes are more narrow on the right portion of the graph. This is expected, since the frequency content of the signal increases on the right.

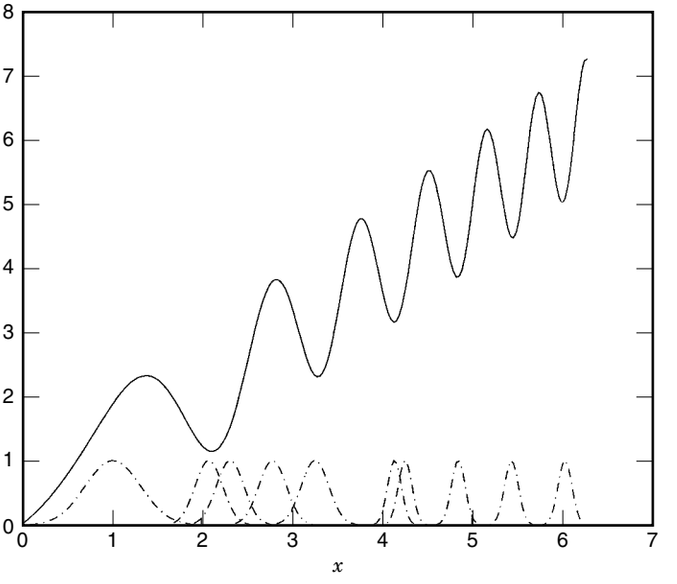

Figure 8. MRA demonstration.

Pulp Digester Level

The goal of the second experiment is to model the multidimensional dynamic system defined by the pulp digester level in a paper mill. As described earlier, the digester level is a process variable indicating the approximate height of the pulp in the digester. This level fluctuates freely, which is dependent on many factors. The four independent variables that are used to predict the digester level in this study are the blow flow, which is a measure of the flow out of the bottom of the digester, and three strain gauges, which measure the strain at different points on the digester. The present and past values of the digester level are also used in the model. These variables may not be the only ones the digester level is dependent on, but they are the ones available which produced the best experimental results. The network was also given the current digester level, as well as the digester levels 15 min and 30 min prior to the current time (the sampling rate for all the data is 15 min).

The training set consists of about 7 days of digester level operation which yields 700 training data points. The testing data consisted of 300 data points, or about 3 days of digester level operation. Each network was trained until the training MSE became less than 55 (the MSE of 55 was found to yield good generalization for all networks), and then the testing MSE and number of neurons used was recorded. Table 2 shows the results for this experiment.

The results show that the GaussianARX network gives the best overall performance with respect to testing MSE and number of neurons used. It is interesting to point out that both networks which use Gaussian activation function use 10 less nodes than the WaveARX network. Apparently for research effort, the Gaussian function is better suited to make the approximation than the Mexican hat function. This, however, gives no indications as to which activation function is best for the general case. If anything, the result indicates that

Table 2. The Training and Test Results of the Digester Level Experiment

	Training MSE	Testing MSE	Number of Nodes
MRC	54.91	77.86	60
WaveARX	54.94	75.49	71
GaussianARX	54.52	69.32	60

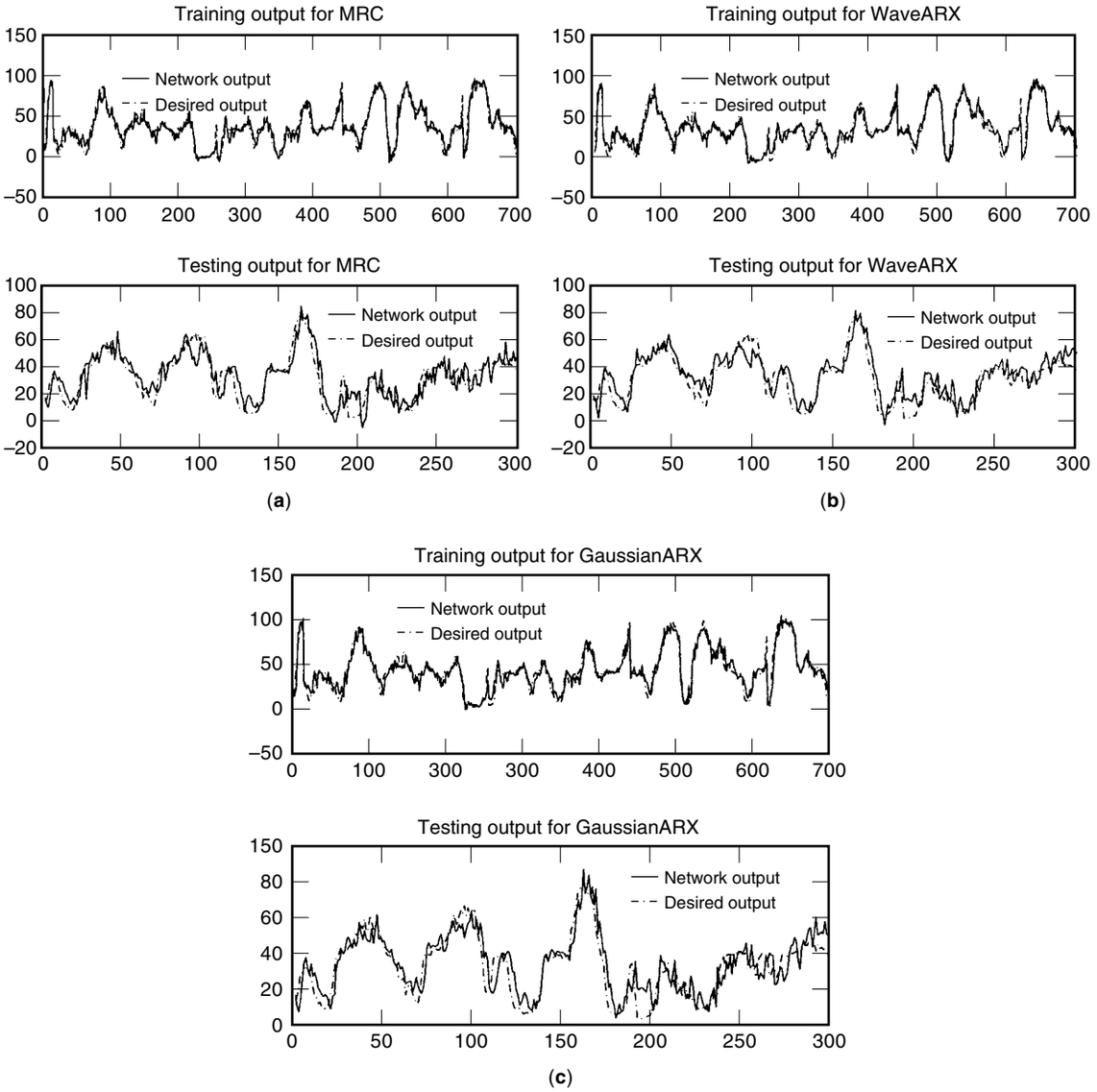

Figure 9. Training and testing outputs for (a) MRC network, (b) WaveARX network, and (c) GaussianARX network.

certain activation functions are better for certain problems. Figure 9 shows the training and testing results for the three networks.

The network output captures the general characteristics of the system. However, the network output curve possesses higher frequency components than the actual digester level curve. This is most likely due to the network becoming overly sensitive to one or more input variables which possess comparatively higher frequency components. It is also important to notice that the testing is worse on the right-hand portion of the plot. This is because the digester system characteristics change with time, and therefore the network yields the best results immediately after it has been trained and become worse as time passes. This problem is a good example of the importance of an on-line training procedure.

CONCLUSION

The basic theory of dynamic system identification using networks of locally active multiresolution units was given, including a discussion of wavelet based networks and radial basis function (RBF) networks. Arguments were given for the fact that the advantages a wavelet basis has over a nonorthogonal basis are not truly advantages when the function to be approximated is of high input dimension or is irregularly sampled. Three network architectures were described, implemented, and tested. The first was an RBF implementation which utilized a multiresolution clustering (MRC) algorithm and a fast orthogonal search (FOS) to train the network. Second, a wavelet network implementation called WaveARX was described, which extends the wavelet function to multidimen-

sions by using a norm for the wavelet input argument. The network is trained by defining a multiresolution analysis (MRA) grid of nodes which are used by an orthogonal search procedure to train the network. Third, an RBF network (GaussianARX) utilizing the same training techniques used by WaveARX was created. The only difference between GaussianARX and WaveARX was that the GaussianARX network used a Gaussian activation function as opposed to the Mexican hat wavelet. The networks were tested on two examples. First, a one-dimensional problem was given in which the frequency content of the signal changed with the spatial location. It was shown that the three networks described in this article, which all used MRA, produced better results than a standard RBF network which did not utilize MRA. Second, the networks were used to model the pulp level of a paper plant pulp digester, a seven-dimensional problem. The GaussianARX network produced the solution of smallest testing error, and at the same time it used the fewest number of nodes. Again, it was apparent that all the networks were able to learn the general characteristics of the system equally well, and no conclusion could be drawn as to the dominance of a particular network. The MRC network has an advantage over the other two networks in that the number of nodes given to the FOS algorithm was not dependent on the number of training data (which reduces training time for large data sets). A disadvantage of using the MRC architecture is that the results will not be the same from trial to trial due to the random initial conditions of the clustering algorithm. The results also indicate that the advantages exhibited by the WaveARX network over prior network architectures was due mainly to the fact that an orthogonal search procedure was used in combination with MRA, and not due to the fact that a wavelet activation function was used. It is reasonable to conclude that certain activation functions are better at solving some problems than others. The combination of MRA and the orthogonal search procedure was shown to be a promising method of system identification.

BIBLIOGRAPHY

1. L. Ljung and T. Soderstrom, *Theory and Practice of Recursive Identification,* Cambridge, MA: MIT Press, 1983.
2. J. H. Chen and D. D. Bruns, WaveARX neural network development for system identification using a systematic design synthesis, *Ind. Eng. Chem. Res.,* **34** (12): 4420–4435, 1995.
3. M. Vetterli and J. Kovacevic, *Wavelets and Subband Coding,* Englewood Cliffs, NJ: Prentice-Hall, 1995.
4. R. Young, *Wavelet Theory and Its Applications,* Dordrecht: Kluwer, 1992.
5. O. Rioul and M. Vetterli, Wavelets and signal processing, *IEEE Signal Process. Mag.,* **8** (4): 14–38, 1991.
6. R. Duffin and A. Schaffer, A class of nonharmonic Fourier series, *Trans. Am. Math. Soc.,* **72**: 341–353, 1952.
7. I. Daubechies, *Ten Lectures on Wavelets,* Philadelphia, PA: Society for Industrial and Applied Mathematics, 1992.
8. G. Strang and T. Nguyen, *Wavelets and Filter Banks,* Wellesley, MA: Wellesley-Cambridge Press, 1996.
9. Q. Zhang and A. Benveniste, Wavelet networks, *IEEE Trans. Neural Netw.,* **3** (6): 889–898, 1992.
10. S. Haykin, *Neural Networks: A Comprehensive Foundation,* New York: Macmillan, 1994.
11. J. Zhang et al., Wavelet neural networks for function learning, *IEEE Trans. Signal Process.,* **43** (6): 1485–1497, 1995.
12. B. Bakshi and G. Stephanopoulos, Wave-Net: A multiresolution, hierarchical neural network with localized learning, *AIChE J.,* **39** (1): 57–81, 1993.
13. J. Park and I. Sandberg, Universal approximation using radial-basis-function networks, *Neural Comp.,* **3** (2): 246–257, 1991.
14. T. Chen and H. Chen, Approximation capability to functions of several variables, nonlinear functionals, and operators by radial basis function neural networks, *IEEE Trans. Neural Netw.,* **6** (4): 904–910, 1995.
15. J. Moody and C. Darken, Fast learning in networks of locally-tuned processing units, *Neural Comput.,* **1** (2): 281–294, 1989.
16. C. Bishop, Improving the generalization properties of radial basis function neural networks, *Neural Comput.,* **3** (4): 579–588, 1991.
17. B. Whitehead and T. Choate, Cooperative-competitive genetic evolution of radial basis function centers and widths for time series prediction, *IEEE Trans. Neural Netw.,* **7** (4): 869–880, 1996.
18. S. Chen, C. Cowan, and P. Grant, Orthogonal least squares learning algorithm for radial basis function networks, *IEEE Trans. Neural Netw.,* **2** (2): 302–309, 1991.
19. W. Ahmed, Fast orthogonal search for training radial basis function neural networks, Master's thesis, Univ. of Maine, 1995.
20. J. Koh, M. Suk, and S. Bhandarkar, A multilayer self-organizing feature map for range image segmentation, *Neural Netw.,* **8** (1): 67, 86, 1995.

MOHAMAD T. MUSAVI
University of Maine

ALAN FERN
Purdue University

DAN R. COUGHLIN
Sappi Fine Paper