Approximation of Nonlinear Systems with Radial Basis Function Neural Networks

Robert J. Schilling, *Senior Member, IEEE*, James J. Carroll, Jr., *Member, IEEE*, and Ahmad F. Al-Ajlouni, *Member, IEEE*

***Abstract*—A technique for approximating a continuous function of n variables with a radial basis function (RBF) neural network is presented. The method uses an n-dimensional raised-cosine type of that RBF is smooth, yet has compact support. The RBF network coefficients are low-order polynomial functions of the input. A simple computational procedure is presented which significantly reduces the network training and evaluation time. Storage space is also reduced by allowing for a nonuniform grid of points about which the RBFs are centered. The network output is shown to be continuous and have a continuous first derivative. When the network is used to approximate a nonlinear dynamic system, the resulting system is bounded-input bounded-output stable. For the special case of a linear system, the RBF network representation is exact on the domain over which it is defined, and it is optimal in terms of the number of distinct storage parameters required. Several examples are presented which illustrate the effectiveness of this technique.**

***Index Terms*—Function approximation, neural network, nonlinear system, radial basis function (RBF).**

I. Introduction

MANY of the applications of neural networks, particularly in the areas of nonlinear system identification and control, reduce to the problem of approximating unknown functions of one or more variables from discrete measurements [1]–[7]. A number of authors have established that multilayer feedforward neural networks, with a variety of activation functions, serve as universal approximators [8]–[18]. For example, Hornik *et al.*in [8] use the Stone–Weierstrass theorem to prove that a two-layer feedforward network can approximate a continuous real function on a compact subset of R^n arbitrarily closely in the L_∞ norm. More generally, continuous nonlinear functionals can be approximated with feedforward neural networks, and these can be used to directly approximate the outputs of dynamical systems [5], [6]. This article focuses on the problem of approximating a continuous function of n variables over a compact region using a radial basis function (RBF) neural network. RBFs implement localized representations of functions, and are easier to initialize and train than multilayer perceptrons [19]–[26].

A raised-cosine type of RBF is introduced that is smooth, yet has compact support. The n-dimensional RBF is represented as a product of n one-dimensional (1-D) functions. The network structure consists of a sum of RBF terms with each term centered about a grid point and scaled by a coefficient function. The coefficient functions are either zeroth or first-degree polynomials of the input vector. In either case the weights of the network can be easily initialized. For the case of constant coefficients, the weights can be chosen such that the network output is exact at each of the grid points. For first-degree polynomial coefficients the weights can be chosen such that both the network output and its derivative are exact at the grid points.

The proposed network has a fixed structure, rather than one that can grow or shrink as the network is trained [7], [25]–[26]. However, if the function to be approximated is known to vary rapidly in some regions, but be relatively flat in others, then the total number of grid points can be reduced by allowing for a nonuniform distribution of the grid points with a higher density grid used in regions of rapid variation. The localized nature of the representation can also be exploited to substantially reduce the time required to evaluate and train the network.

A network based on a raised-cosine RBF has a number of useful qualitative properties in terms of interpolation. Included among these are the fact that the network output and its first derivative are continuous functions of the input. When a raised-cosine RBF network is used to approximate a nonlinear dynamic system, the resulting system is bounded-input bounded-output (BIBO) stable. For the special case of a linear system, the RBF network reduces to an exact representation on the domain over which it is defined. The resulting linear system is also optimal in the sense that requires the minimum number of storage parameters.

The remainder of this article is organized as follows. A zeroth-order network based on a raised-cosine RBF is introduced in Section II. The network is generalized to a first-order RBF network in Section III. Section IV discusses implementation issues including a simple technique for increasing network speed by exploiting the compact support property of the raised-cosine RBF. The application of the RBF network to the approximation of continuous-time nonlinear dynamic systems is discussed in Section V. Examples of applications involving the modeling and simulation of linear and nonlinear dynamic systems are presented in Section VI. Conclusions are presented in Section VII.

II. Zeroth-Order RBF Network

A. Center Point Grid

Let $f: S \to R$ denote the function to be approximated where $S \subset R^n$ is a closed bounded rectangular region. More specifically, if the $n \times 1$ vectors a and b denote the lower and upper limits of x, respectively, then

$$S = \{x \in R^n | a_i \le x_i \le b_i,\ 1 \le i \le n\}. \tag{1}$$

Manuscript received February 29, 2000; revised September 18, 2000.

R. J. Schilling and J. J. Carroll are with the Department of Electrical and Computer Engineering, Clarkson University, Potsdam, NY 13699-5720 USA.

A. F. Al-Ajlouni is with the Hijjawi Faculty of Engineering Technology, Department of Communication Engineering, Yarmouk University, Irbid, Jordan.

Publisher Item Identifier S 1045-9227(01)00754-8.

It is assumed that values of $f(x)$ are available at discrete points within S. The RBF network includes r terms with the kth term consisting of RBF $\phi_k(x)$ centered about point x^k and scaled by weight w_k. If w and $\phi(x)$ denote $r \times 1$ column vectors, then the network output can be expressed in compact form as

$$f_0(x) = w^T \phi(x). \tag{2}$$

Suppose there are $d_i \geq 2$ grid points along dimension i, for $1 \leq i \leq n$, ordered as follows:

$$x_{i1} < x_{i2} < \ldots < x_{id_i}, \qquad 1 \leq i \leq n. \tag{3}$$

Here $x_{i1} = a_i$ and $x_{id_i} = b_i$. Since there are d_i grid points per dimension, the total number of grid points is

$$r = d_1\, d_2\, \ldots\, d_n. \tag{4}$$

The total number of grid points can also be expressed as $r = \overline{d}^n$ where $\overline{d} = (d_1\, d_2\, \ldots\, d_n)^{1/n}$ is the geometric mean of the numbers of grid points per dimension.

If the function f is known to vary rapidly in some regions, and be relatively flat in others, then the number of grid points, r, can be reduced by allowing for a nonuniform distribution of points. Specifically, a higher density grid can be used in those regions of S where the variation of $f(x)$ is large, while a lower density grid can be used in regions where $f(x)$ is relatively flat. To provide for an efficient implementation, which reduces the time required to evaluate $f_0(x)$, it is useful to develop a function which maps the nonuniform grid in (3) into a grid with uniform unit spacing. Consider, in particular, the following piecewise-linear interpolating polynomial for the points in (3).

$$p_i(z) = \begin{cases} 1 + \dfrac{z - x_{i1}}{x_{i2} - x_{i1}}, & z < x_{i1} \\ j + \dfrac{z - x_{ij}}{x_{i,j+1} - x_{ij}}, & x_{ij} \leq z < x_{i,j+1} \\ & 1 \leq j \leq d_i, \quad 1 \leq i \leq n \\ d_i + \dfrac{z - x_{i,d_i-1}}{x_{i,d_i} - x_{i,d_i-1}}, & z \geq x_{i,d_i} \end{cases} \tag{5}$$

The function $p_i: [a_i, b_i] \rightarrow [1, d_i]$. Furthermore, by construction, p_i maps the jth grid value x_{ij} into the integer j

$$p_i(x_{ij}) = j, \qquad 1 \leq j \leq d_i,\ 1 \leq i \leq n. \tag{6}$$

It follows that the function $p: S \rightarrow R^n$ maps the nonuniform grid into a uniform grid with unit spacing between adjacent points. As a special case, if the original grid in (3) is uniform, but with a different constant spacing for each dimension, then $p_i(z)$ in (5) simplifies to

$$p_i(z) = 1 + \frac{z - x_{i1}}{x_{i2} - x_{i1}}, \qquad 1 \leq i \leq n. \tag{7}$$

Next, let $q \in Q^n$ be a *vector index* which species the qth grid point, $x(q)$, where Q represents the positive integers and $1 \leq q_i \leq d_i$ for $1 \leq i \leq n$. Thus the ith element of the vector $x(q)$ is

$$x_i(q) = x_{iq_i}, \qquad 1 \leq q_i \leq d_i,\ 1 \leq i \leq n \tag{8}$$

In order to avoid the need to use n separate summations in the formulation of the network output $f_0(x)$, it is helpful to reorder the r grid points into a 1-D array of points $C = \{x^1, x^2, \ldots, x^r\}$ by introducing a *scalar index* k which can computed from the $n \times 1$ vector index q as follows:

$$\begin{aligned} k = {} & q_1 + d_1(q_2 - 1) + d_1 d_2(q_3 - 1) + \ldots \\ & + d_1\, d_2\, \ldots\, d_{n-1}(q_n - 1). \end{aligned} \tag{9}$$

There will be a RBF $\phi_k(x)$ centered at each grid point with $\phi,(0) = 1$ and $|\phi,(x)|$ decreasing as the radius $|x - x^k|$ increases. Using the scalar index k in (9), the output of the RBN can then be expressed as in (2) which corresponds to the single summation

$$f_0(x) = \sum_{k=1}^{r} w_k \phi_k(x). \tag{10}$$

B. RBFs

RBFs are constructed such that $|\phi_k(x)|$ vanishes, or becomes extremely small, for sufficiently large values of the radius $|x - x^k|$. It is this feature that makes the network an efficient local representation because for a given x, only a limited number of terms in (10) will be active and contribute to the output.

To construct a general RBF $\phi_k(x)$ it is useful to first examine the special case of a 1-D RBF $\psi(z)$ centered at $z = 0$. Webb and Shannon [25] discuss several candidate functions, the most popular one being

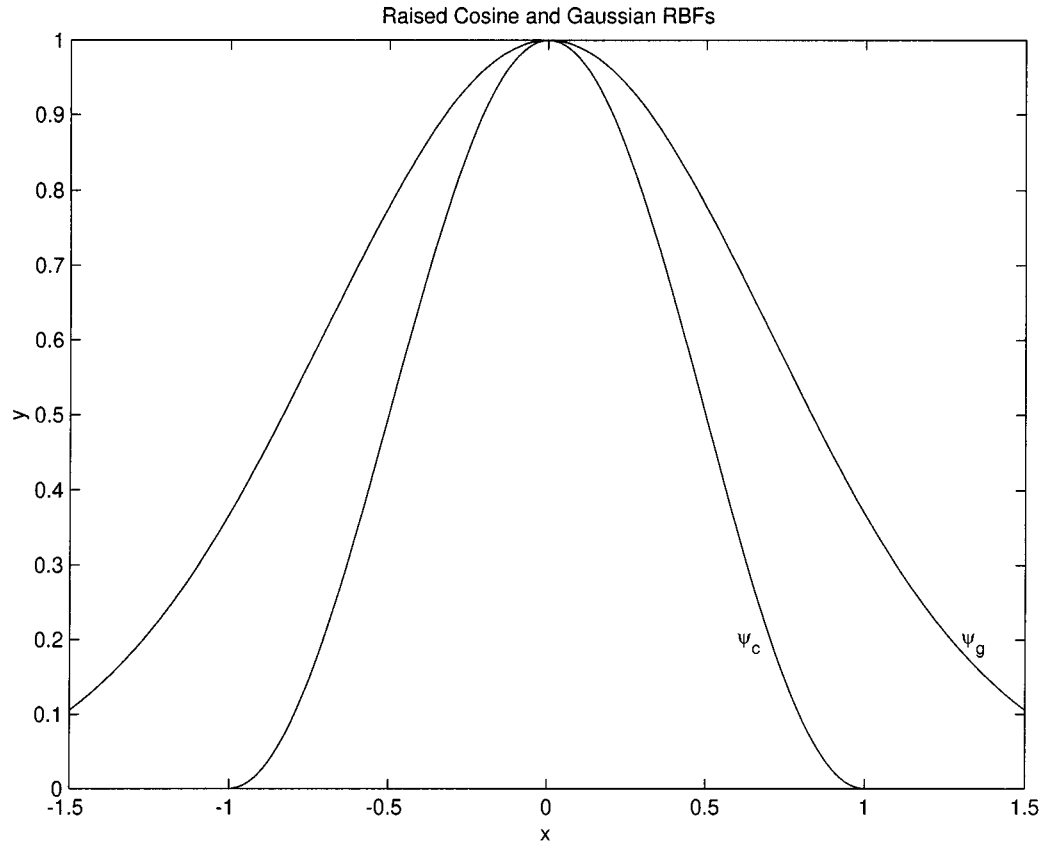
$$\psi_g(z) = \exp(-z^2). \tag{11}$$

The function $\psi_g(z)$ is the normal or *Gaussian* RBF. The function $\psi_g(z)$ can be shown to be optimal, in a least squares sense, for fitting data that has normally distributed noise on the input [23]. The Gaussian function is also infinitely differentiable and all of the derivatives are continuous. Suppose $\phi_k(x) = \psi_g[(x - x^k)/\sigma_k]$ is used as an RBF. If σ_k is sufficiently small, then $f_0(x)$ will be a good approximation of $f(x)$ at the grid points when the weights are set to $w_k = f(x^k)$. However, midway between the grid points $f_0(x) \approx 0$ and the approximation is poor. This can be remedied by using larger σ_k but then the representation becomes less localized with more terms contributing to the output. This is a consequence of the fact that $\psi_g(z) > 0$ for all finite z.

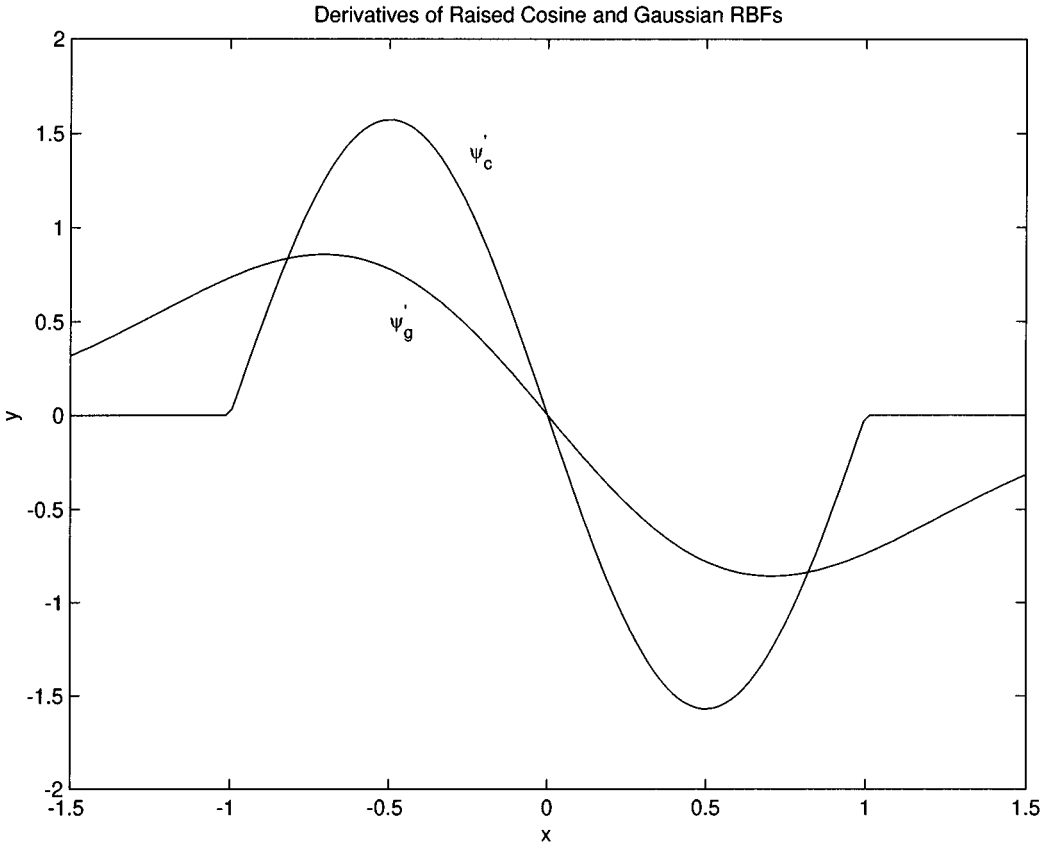
An alternative approach is to use the following *raised cosine* as a 1-D RBF centered about zero

$$\psi_c(z) \triangleq \begin{cases} \dfrac{1 + \cos(\pi z)}{2} & |z| \leq 1n, 0 \\ 0 & |z| > 1 \end{cases}. \tag{12}$$

Note that $\psi_c(0) = 1$ and $\psi_c(z) = 0$ for $|z| \geq 1$. Unlike the Gaussian RBF, $\psi_c(z)$ is zero outside the closed bounded region, $|z| \leq 1$ which makes $\psi_c(z)$ a function with compact support. The raised-cosine function in (12) can be regarded as an analog version of the Hanning window sequence used in digital filter design. A graphical comparison of $\psi_g(z)$ and $\psi_c(z)$ is shown in Fig. 1(a), and comparison of the derivatives, $\psi_g'(z)$ and $\psi_c'(z)$, is shown in Fig. 1(b).



(a)



(b)

Fig. 1. 1-D RBFs: (a) Gaussian (ψ_g) and raised-cosine (ψ_c) functions. (b) First derivatives of Gaussian (ψ_g') and raised-cosine (ψ_c') functions.

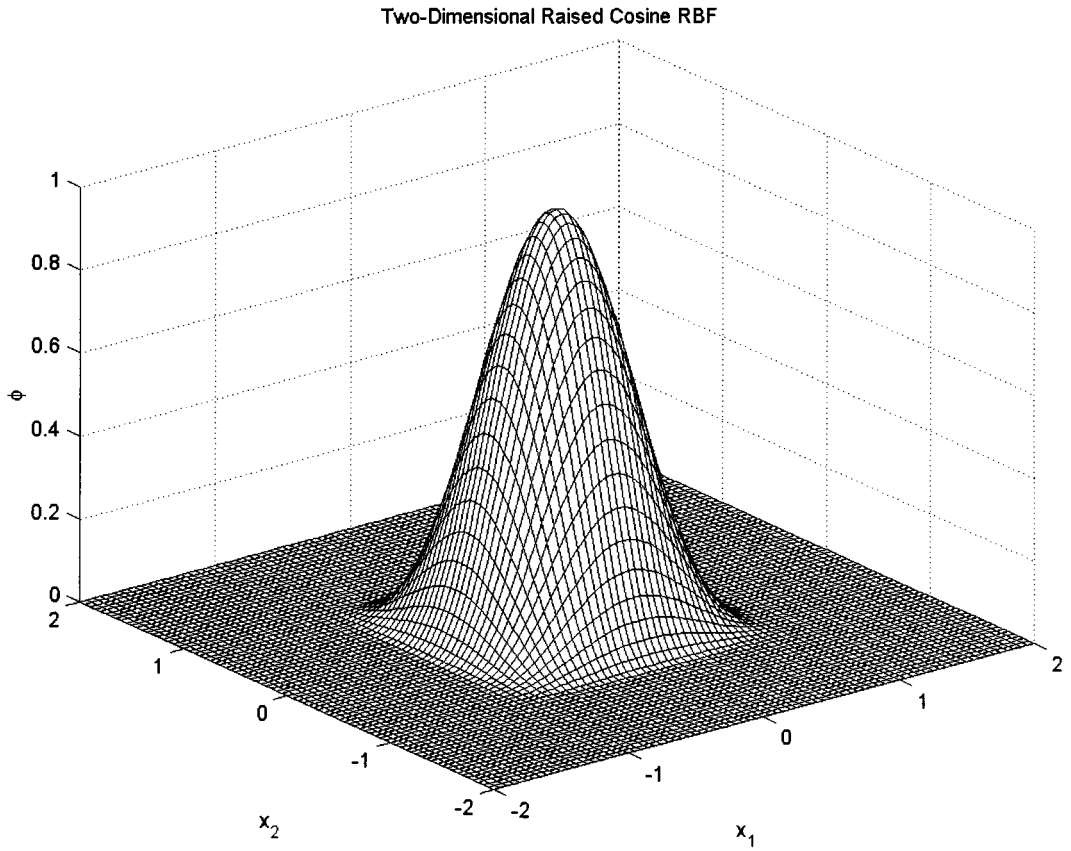


Fig. 2. A two–dimensional (2-D) raised-cosine RBF, $\phi(x)$, centered at the origin with a uniform unit grid.

The 1-D case is generalized to n dimensions by expressing $\phi_k(x)$ as a product of n 1-D RBFs.

$$\phi_k(x) \triangleq \prod_{i=1}^{n} \psi_c[p_i(x_i) - p_i(x_i^k)] \tag{13}$$

The basic RBF network in (2) has a number of useful qualitative properties. Using (6), (12), and (13), the RBF $\phi_k(x)$ satisfies the following *orthogonal* property with respect to the grid points:

$$\phi_k(x^j) = \begin{cases} 1, & k = j \\ 0, & k \neq j \end{cases} \qquad 1 \leq k, j \leq r. \tag{14}$$

The orthogonal property is a direct consequence of the fact that the raised cosine in (12) has compact support. Since the Gaussian RBF does not have compact support, it does not satisfy (14). Using (14), it follows from (2) that $f_0(x^k) = w_k$ for $1 \leq k \leq r$. Consequently, if $w_k = f(x^k)$ for $1 \leq k \leq r$, then the zeroth-order RBF network output in (2) is exact at the r grid points

$$f_0(x^k) = f(x^k), \qquad 1 \leq k \leq r. \tag{15}$$

Another important qualitative property is the relative smoothness of the network output. From (12) and Fig. 1(a), it is evident that $\psi_c(z)$ is a continuous function taking on values in the interval [0, 1]. Since the piecewise-linear interpolating polynomial $p_i(z)$ in (6) is also continuous, it follows from (13) that $\phi_k(x)$ is a product of compositions of continuous functions and is therefore itself a continuous function whose value lies in the interval [0, 1]. A plot of $\phi_k(x)$ for the case $n = 2$, $x^k = 0$, and a uniform unit grid is shown in Fig. 2. Since the raised-cosine RBF $\phi_k(x)$ is continuous, the network output $f_0(x)$ in (2) is continuous.

The derivative of the 1-D raised-cosine RBF in (12) is shown in Fig. 1(b) and can be expressed

$$\psi_c'(z) = \begin{cases} \dfrac{\pi \sin(\pi z)}{2}, & |z| \leq 1 \\ 0, & |z| > 1 \end{cases}. \tag{16}$$

Thus $\psi_c'(z)$ is continuous. First consider the case of a uniform grid in (7). In this case the derivative of the piecewise-linear interpolating polynomial, $p_i'(z) = 1/(x_{i2} - x_{i1})$, is constant. As a result, the following derivative is then continuous:

$$\frac{d}{dx_i}\,\psi_c[p_i(x_i) - p_i(x_i^k)] = \frac{\pi \sin(\pi[p_i(x_i) - p_i(x_i^k)])}{2(x_{i2} - x_{i1})}. \tag{17}$$

Since $\psi_c[p_i(x_i) - p_i(x_i^k)]$ is a continuous function of x_i, it then follows from (13) that the gradient $\nabla\phi_k(x)$ is continuous which means that $\nabla f_0(x)$ is continuous.

For the more general case of a nonuniform grid, the derivative of the piecewise-linear interpolating polynomial $p_i(z)$ is not defined at the grid point coordinates where the slope abruptly

changes. However if, for example, the right-hand side derivative, $p_i'(x_i^+)$, is used, then from (5)

$$p_i'(z) = \begin{cases} \dfrac{1}{x_{i2} - x_{i1}} & z < x_{i1} \\ \dfrac{1}{x_{i,j+1} - x_{ij}} & x_{ij} \le z < x_{i,j+1} \\ & 1 \le j \le d_i \quad 1 \le i \le n \\ \dfrac{1}{x_{i,d_i} - x_{i,d_i-1}} & z \ge x_{i,d_i} \end{cases} \tag{18}$$

Thus the only thing that changes in (17) is that the denominator is replaced by $2(x_{i,j+1} - x_{ij})$ where j depends on x_i and changes when x_i crosses a grid boundary. However, recall from (6) that the numerator of the right-hand side of (17) is zero at the grid points. Thus the expression in (17) is again continuous assuming the right-hand side derivative is used. It then follows from (13) that the gradient $\nabla\phi_k(x)$ is continuous which means that $\nabla f_0(x)$ is continuous for a nonuniform grid when the right-hand side derivative is used. A similar argument could have been applied using the left-hand side derivative.

Next, recall from (6) that when $x = x^j$, the argument $p_i(x^j) - p_i(x_i^k)$ in (17) is an integer which means that the derivative in (17) evaluates to zero. Using (13) one can then conclude that at the grid points

$$\nabla\phi_k(x^j) = 0, \qquad 1 \le k, j \le r. \tag{19}$$

The simplest possible function to approximate is the 1-D constant function, $f(x) = \alpha$, which can be thought of as a polynomial of degree zero. Consider the case $n = 1$ with $d_1 = 2$ grid points at $x^1 = a$ and $x^2 = b$. Then from (6) we have $p_1(x^1) = 10$ and $p_1(x^2) = 2$. Finally, using (12) and (13) it can be shown that the following *constant interpolation* property holds in this case

$$\phi_1(x) + \phi_2(x) = 1, \qquad a \le x \le b. \tag{20}$$

Consequently, when $w = [\alpha, \alpha]^T$ the network output $f_0(x) = \alpha$ is exact. That is, the raised-cosine RBF network can represent a constant function exactly using two terms. Although this is not a profound result, it is a useful one (see, e.g., [24]). Moreover, it is worth noting that when the Gaussian RBF in (11) is used, an exact representation of a constant is not possible unless a limiting special case (the inclusion of a bias term, $w_0\phi_0(x) \equiv w_0$) is used. However, the presence of a bias term destroys the local nature of the representation because, in general, the value of w_0 will affect the values of all of the other weights. The constant interpolation property in (20) can be extended to the general case, $n \ge 1$, and it also holds when $d_i > 2$ (see the Appendix). We summarize the properties of the zeroth-order RBF network as follows.

Proposition 1 (Zeroth-Order RBF Network): When the raised-cosine RBF is used, the zeroth-order RBF network in (2) has the following properties.

1) The network output, $f_0(x)$, and its gradient, $\nabla f_0(x)$ are continuous.
2) If $w_k = f(x^k)$ for $1 \le k \le r$, then the network output is exact on the center point grid, and the gradient is zero on the grid
$$f_0(x^k) = f(x^k), \qquad 1 \le k \le r$$
$$\nabla f_0(x^k) = 0, \qquad 1 \le k \le r.$$
3) If $f(x) = \alpha$, then the RBF network is an exact representation of $f(x)$ on S when $d_i = 2$ for $1 \le i \le n$ and $w_i = \alpha$ for $1 \le i \le 2^n$.
4) The number of weights needed to train the network is
$$r = d_1\, d_2 \ldots d_n.$$
5) The number of nonzero terms in $f_0(x)$ is, at most, 2^n.

The last two properties are concerned with the size and speed of the network. Property 5) is a result of the fact that there are, at most, two nonzero RBF terms per dimension because $\psi_c(z)$ is a function with compact support.

III. First-Order RBF Network

The exact representation property, in Property 2), can be extended by considering a simple generalization of the zeroth-order RBN to a first-order RBN. In particular, the constant weighting coefficients in (2) can be replaced by first-degree polynomial functions of the input

$$f_1(x) = (w + Vx)^T\phi(x). \tag{21}$$

Here the parameters to be identified during training now include a $r \times 1$ constant *weight vector* w and a $r \times n$ linear *weight matrix* V. In this case, the number of weights has increased by a factor of $n + 1$.

To verify that the first-order RBN satisfies the basic interpolation property on the center point grid, note from (14) and (21) that $f_1(x^k) = w_k + v^k x^k$ for $1 \le k \le r$ where v^k denotes the kth row of V. Therefore the first-order RBF network in (21) satisfies the following basic interpolation property when $w_k = f(x^k) - v^k x^k$ for $1 \le k \le r$.

$$f_1(x^k) = f(x^k), \qquad 1 \le k \le r. \tag{22}$$

Since $\phi(x)$ is continuous, it follows that $f_1(x)$ is continuous. Next observe that the derivative of $f_1(x)$ with respect to x_i is

$$\frac{\partial f_1(x)}{\partial x_i} = \sum_{k=1}^{r} V_{ki}\phi_k(x) + (w_k + v^k x)\frac{\partial\phi_k(x)}{\partial x_i}, \quad 1 \le i \le n. \tag{23}$$

Since $\phi_k(x)$ and $\nabla\phi_k(x)$ are both continuous, it follows that $\partial f_1(x)/\partial x_i$ is continuous for $1 \le i \le n$. Therefore $\nabla f_1(x)$ is continuous.

Next recall from (19) that $\nabla\phi_k(x) = 0$ at the grid points. If the expression in (23) is evaluated at grid point $x = x^j$, then using (14), the partial derivative reduces to

$$\frac{\partial f_1(x^j)}{\partial x_i} = V_{ji}, \qquad 1 \le j \le r,\ 1 \le i \le n. \tag{24}$$

Thus the jth row of the linear weight matrix V is the gradient of $f_1(x)$ evaluated at $x = x^j$. Consequently, the gradient of the

network output at the grid points can be specified though the choice of V.

Following the constant function, the next simplest function to approximate is the n-dimensional affine function, $f(x) = \alpha + \beta^T x$, which is just a polynomial of degree one. Using the constant interpolation property in (20), one can show that a linear polynomial can be represented exactly by the first-order RBF network using $d_k = 2$ grid points per dimension. We summarize the properties of the first-order RBF network as follows.

Proposition 2 (First-Order RBF Network): When the raised-cosine RBF is used, the first-order RBF network in (21) has the following properties:

1) The network output, $f_1(x)$, and its gradient, $\nabla f_1(x)$, are continuous.
2) If $w_k = f(x^k) - v^k x^k$ where $v^k = \nabla f(x^k)$ for $1 \le k \le r$, then the network output and its gradient are exact on the center point grid

$$f_1(x^k) = f(x^k), \qquad 1 \le k \le r$$
$$\nabla f_1(x^k) = \nabla f(x^k), \qquad 1 \le k \le r.$$

3) If $f(x) = \alpha + \beta^T x$, then the RBF network is an exact representation of $f(x)$ on S when $d_i = 2$ for $1 \le i \le n$, $w_i = \alpha$ for $1 \le i \le 2^n$, $V_{ij} = \beta_j$ for $1 \le i \le 2^n$ and $1 \le j \le n$.
4) The number of weights needed to train the network is

$$s = (n+1)d_1\, d_2 \ldots d_n.$$

5) The number of nonzero terms in $f_1(x)$ is, at most, 2^n.

Comparison of Proposition 2 with Proposition 1 reveals that the size of the first-order RBF network is $n+1$ times larger than the size of the zeroth-order RBF network. This increase in the number of weights allows one to specify both the value of the output and the value of its gradient on the center point grid. This, in turn, means that an exact representation of a more general class of functions is possible as in Property 3). The increase in the number of weights from r to $(n+1)r$ will increase the time needed to train the network. However, if the network evaluation speed is measured in the number of RBF evaluations, then the first-order network is just as fast as the zeroth-order network because there are still, at most, 2^n nonzero terms for each $x \in S$. Evaluation of each term is somewhat more expensive because the number of floating point multiplications increases from one to $1 + n$.

Information regarding the gradient of the function to be approximated at the grid points may or may not be available. If it is not, then a good initial guess for the linear weight matrix can be obtained by approximating the gradient of f numerically. For example, the rows of V can be initialized by approximating the gradient numerically using central differences at the interior grid points and forward or backward differences, as appropriate, along the grid boundary [29].

The first-order network in (21) can be generalized still further by replacing the linear polynomial by a quadratic polynomial in x. In this case an exact representation of a quadratic $f(x) = \alpha + \beta^T x + x^T \Gamma x$ can be obtained using two grid points per dimension. However, the second derivative of the network output is not continuous at the grid points. Furthermore, for a second-order RBF network, the number of weights, $(1 + n + n^2)r$ can become excessive, particularly for larger values of n.

IV. Implementation

The zeroth-order RBF network in (2) is a special case of the first-order RBF network in (21) with the $r \times n$ weight matrix V set to $V = 0$. Consequently, we focus our attention on the first-order RBF network. The number of terms in (21) grows rapidly with the number of grid points per dimension, d_i, and the number of dimensions, n. Fortunately, most of the terms are zero and therefore do not have to be evaluated. This can speed up the calculation of $f_1(x)$ significantly, particularly for larger values of n. For a given x, the number of nonzero RBFs in each dimension is either one or two. Consequently, the number of nonzero terms in (21) is, at most, 2^n. The subscripts of the nonzero terms can be efficiently identified by first finding the grid point hypercube in R^n that contains the point x. The terms associated with the 2^n vertices of this hypercube are then examined and their associated RBFs are evaluated. The procedure is summarized in the following algorithm which takes w, V and x as inputs and produces $y = f_1(x)$ as an output.

```
Alg. 1 (Network Evaluation)
1.  Set y = 0.
2.  For i = 1 to n do
  {
    a.  u_i = int[p_i(x_i)]
    b.  If u_i < 1, set u_i = 1.
    c.  If u_i > d_i - 1, set u_i = d_i - 1.
  }
3.  For j = 0 to 2^n - 1 do
  {
    a.  Convert j from decimal to n-bit
    binary c = c_{n-1} c_{n-2} ... c_0.
    b.  For i = 1 to n compute q_i = u_i + c_{i-1}.
    c.  Convert q to k using (9)
    d.  Compute y = y + (w_k + v^k x)φ_k(x)
  }
```

The implementation in Alg. 1 evaluates 2^n terms and returns $y = f_1(x)$. The increase in speed in using Alg. 1 in comparison with a direct computation of (21) can be substantial. Recall that $\overline{d}$ is the geometric mean of the number of grid points in each dimension. Whereas direct evaluation of (21) requires $r = \overline{d}^n$ function evaluations, Alg. 1 requires 2^n function evaluations. Thus for $\overline{d} = 10$ and $n = 3$, Alg. 1 is more than two orders of magnitude faster than direct evaluation. For larger values of n and a finer grid, the savings are even more dramatic.

Since a RBF network has only one hidden layer and a linear output layer, the update formulas for the weights are relatively simple. Let S and T denote the set of inputs used to train and test the network, respectively

$$S = \{s^k \in R^n \,|\, 1 \le k \le n_s\} \tag{25}$$
$$T = \{t^k \in R^n | 1 \le k \le n_t\}. \tag{26}$$

Next, let $E(x) = f(x) - f_1(x)$ denote the *error* between the actual output and the network output evaluated at x. If we minimize $E^2(x)/2$ by adjusting the weights after each training input using a gradient search with a step size of $\mu > 0$, this leads to the following least-squares algorithm for updating w and the V.

```
Alg. 2 (Network Training)
1.  Pick E_max ≥ 0 and p_max ≥ 1. Initialize V
    and w using Property 2). Set p = 0.
2.  Do
  {
    a.  Set E = 0 and p = p + 1
    b.  For h = 1 to n_s do
      {
        i.  Compute E = f(s^h) − f_1(s^h) using
            Alg. 1.
        ii.  Compute u_i = int[p_i(s_i^h)] for
             1 ≤ i ≤ n using (5).
        iii.  For j = 0 to 2^n − 1 do
          {
            a)  Convert j from decimal to
  n-bit binary c = c_{n−1} c_{n−2} ... c_0.
            b)  Compute q_i = u_i + c_{i−1} for 1 ≤ i ≤ n.
            c)  Convert q to k using (9)
            d)  Compute
              {

                w_k = w_k + μE φ_k(s^h)
                v^k = v^k + μE φ_k(s^h)(s^h − x^k)^T
              }
          }
    c.  For k = 1 to n_t use Alg. 1 to
      compute

                E = max{E, |f(t^k) − f_1(t^k)|}
  }
3.  While (E > E_max) and (p < p_max).
```

Note that the training and test outputs, $f(s^k)$ and $f(t^k)$, can all be precomputed ahead of time. In addition, from step 2biii, only 2^n terms are used to adjust the weights during training. This substantially reduces the training time. Use of the initial guesses for V and w based on Property 2) further reduces the training time. Indeed, for a sufficiently fine grid, the initial values for V and w in Property 2) should be quite close to the optimal values.

V. Nonlinear Systems

The raised-cosine RBF network can be used to approximate any continuous function of n variables [27], [28]. In this article we focus on approximating the right-hand side of a continuous-time m-dimensional nonlinear dynamic system of the following general form:

$$\frac{dz(t)}{dt} = g[z(t), u(t)] \tag{27}$$

$$y(t) = h[z(t), u(t)]. \tag{28}$$

Here it is assumed that the function $g: R^m \times R \to R^m$ is Lipschitz continuous in z and continuous in u and the function $h: R^m \times R \to R$ is continuous in both arguments. Under these conditions, if the input $u(t)$ is piecewise-continuous, then for each initial condition $z(0) = z^0$, a unique solution $z(t)$ exists for $0 \le t < \infty$ [30].

A. Stability

Suppose that the right-hand side functions g_k and h in (27) and (28) are each approximated with a first-order raised-cosine RBF network of form $f_1(x)$ where $x = [z^T, u]^T$ and $n = m + 1$. Thus there are n networks, each having n inputs. Let $S_1 \subset R^n$ be an extension of the network domain S in (1) that includes a boundary around S whose width, in dimension i, is equal to the grid width. That is

$$c_i = (b_i - a_i)/(d_i - 1), \qquad 1 \le i \le n \tag{29}$$

$$S_1 = \{x \in R^n | a_i - c_i \le x_i \le b_i + c_i, 1 \le i \le n\}. \tag{30}$$

Thus S_1 is a closed bounded region of R^n. Furthermore, because the raised-cosine RBF has finite support in the form of a 2×2 grid, the network output $f_1(x)$ has finite support. In particular

$$f_1(x) = 0, \qquad x \notin S_1. \tag{31}$$

A simple four-point illustration of the region S_1 is shown in Fig. 3 where $n = 2$, $d = [2, 2]^T$, $a = [-1, -1]^T$, $b = [1, 1]^T$, $w = [1, 1, 1, 1]^T$, and $V = 0$. In this instance it is evident that $f_1(x) = 0$ outside the region $S_1 = [-3, 3] \times [-3, 3]$. Note that Fig. 3 also illustrates the constant interpolation property in (20) for the case $n = 2$.

A common way to characterize the stability of a nonlinear nonautonomous system is to say that the system is BIBO stable if and only if every bounded input $u(t)$ generates a bounded output $y(t)$. Having a system that is BIBO stable is useful for simulation purposes because this avoids possible numerical overflow conditions during simulation runs.

Suppose $u(t) = x_n(t)$ lies in the interval $a_n \le u(t) \le b_n$. Thus $u(t)$ is bounded with bound $u_{\max} = \max\{|a_n|, |b_n|\}$. To show that $y(t)$ is bounded, suppose $g_k(z, u)$ for $1 \le k \le m$ and $h(z, u)$ are each approximated with a RBF network of the form $f_1(x)$ where $x = [z^T, u]^T$. Then from (31), $dz/dt = 0$ for $x \notin S_1$. We examine two cases. If $x(0) \in S_1$, it then follows that $x(t) \in S_1$ for $t \ge 0$. Alternatively, if $x(0) \notin S_1$, then $z(t) = z(0)$ for $t \ge 0$. Thus in either case, $x(t)$ is bounded. Since the output equation function, $g(z, u)$, in (28) is also approximated by a continuous RBF network, it then follows that the RBF output $y(t)$ must be bounded.

Proposition 3 (Stability): Suppose the nonlinear system in (27) and (28) is approximated by using raised-cosine RBF networks for the right-hand side functions. Then the resulting nonlinear system is BIBO stable.

B. Linear Systems

A very important special case of a nonlinear dynamic system is a linear m-dimensional system of the following form:

$$\frac{dz(t)}{dt} = Fz(t) + gu(t) \tag{32}$$

$$y(t) = h^T z(t) + cu(t). \tag{33}$$

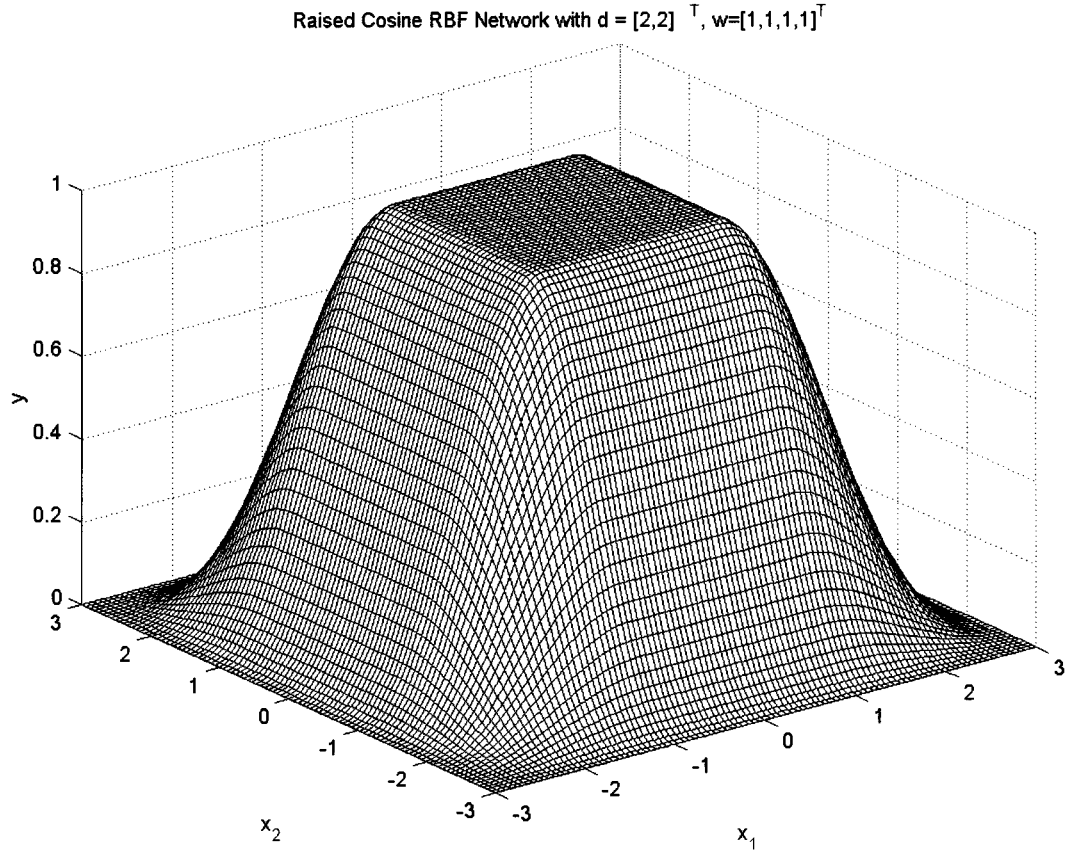


Fig. 3. Output of a four-point 2-D raised-cosine RBF network with $a = [-1\ 1]^T$, $b = [1,\ 1]^T$, $w = [1,\ 1,\ 1,\ 1]^T$, $V = 0$, $S = [-1,\ 1] \times [-1,\ 1]$ and $S_1 = [-3,\ 3] \times [-3,\ 3]$.

Here F is $m \times m$, g is $m \times 1$, h^T is $1 \times m$ and c is 1×1. Thus a total of $p = m^2 + 2m + 1$ parameters are needed to completely specify the linear system. It is of interest to examine how effective the RBF network is in approximating this simplified system. Using Property 3), it can be shown that the RBF network representation is exact on the domain S.

Proposition 4 (Linear Systems): Suppose the linear system in (32) and (33) is approximated by using first-order raised-cosine RBF networks for the right-hand side functions.

1) Let $d_i = 2$ for $1 \le i \le n$ and $w = 0$. For the ith network, let the jth row of V be as follows for $1 \le j \le 2^n$.

$$v^j = \begin{cases} [F_{i1},\ F_{i2},\ \ldots,\ F_{im},\ g_i], & 1 \le i \le m \\ [h_1,\ h_2,\ \ldots,\ h_m,\ c], & i = m+1. \end{cases}$$

Finally, let $y_1(t)$ be the RBF network output for the output equation in (33). Suppose $a_n \le u(t) \le b_n$ and $x(0) = [z(0)^T,\ u(0)]^T \in S$. If $x(t) \in S$ for $t \ge 0$, then $y_1(t) = y(t)$ for $t \ge 0$.

2) The number of distinct scalar parameters that must be stored for the RBF network representation of the m-dimensional linear system in (32) and (33) is

$$p = m^2 + 2m + 1.$$

Thus for a linear system, the RBF network approximation is exact when the input $u(t)$ and solution trajectory $z(t)$ remain within the domain S over which the network is defined. The domain S can be made arbitrarily large without increasing the number of grid points $r = 2^n$. Since a total of p parameters are required to specify the linear system coefficients $\{F,\ g,\ h,\ c\}$, the number of distinct parameter values that must be stored for the RBF network is the minimum possible. That is, the network is optimal in that sense.

VI. Examples

The following examples illustrate the effectiveness of the raised-cosine RBF network when it is used to approximate the right-hand side of first-order systems of differential equations.

Example 1—Unstable Linear System: To verify the behavior summarized in Propositions 3 and 4, consider the following three-dimensional (3-D) linear system:

$$\frac{dz}{dt} = \begin{bmatrix} 0 & 1 & 0 \\ -20 & -4 & 0 \\ 0 & 0 & 0 \end{bmatrix} z + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u$$
$$y = [1 \quad -3 \quad 0.2]\, z + [2]u.$$

This system has eigenvalues at $\lambda_{1,2} = -2 \pm j4$ and $\lambda_3 = 0$. Thus there is a stable second-order mode and an unstable first-order mode. In this case $x = [z_1,\ z_2,\ z_3,\ u]^T$ and $n = 4$. Suppose the domain S is defined by lower limits $a = [-10, -10, -10, -10]^T$ and upper limits

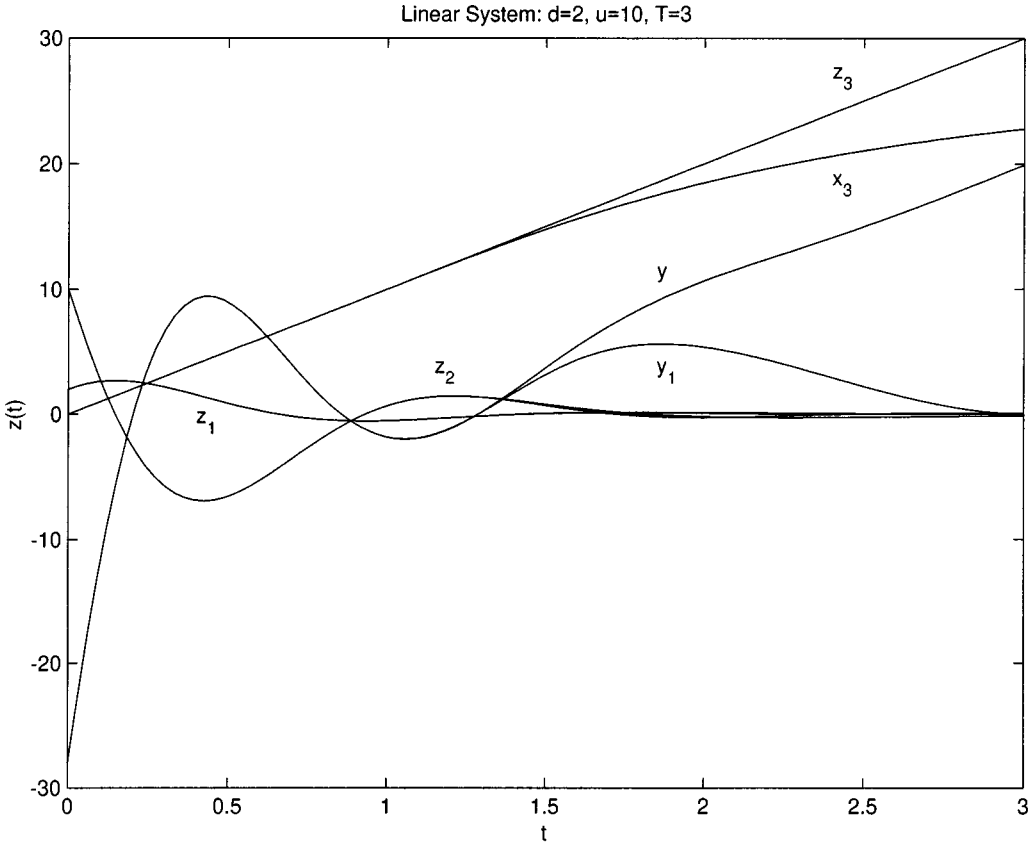


Fig. 4. Solution of 3-D unstable linear system with $d = [2,\ 2,\ 2,\ 2]^T$, $a = [-10,\ -10,\ -10,\ -10]^T$ and $b = [10,\ 10,\ 10,\ 10]^T$.

$b = [10, 10, 10, 10]^T$. Thus from (30), the domain S and compact support region S_1 are as follows:

$$S = \{x \in R^4 | -10 \le x_k \le 10,\ 1 \le k \le 4\}$$
$$S_1 = \{x \in R^4 | -30 \le x_k \le 30,\ 1 \le k \le 4\}.$$

In this case there are two grid points per dimension, $d = [2, 2, 2, 2]$, and the total number of terms is $r = 16$. There are four RBF networks, each with four inputs. Assigning w and V as in Proposition 4, the number of distinct values that need to be stored is $p = 16$ which is the minimum possible.

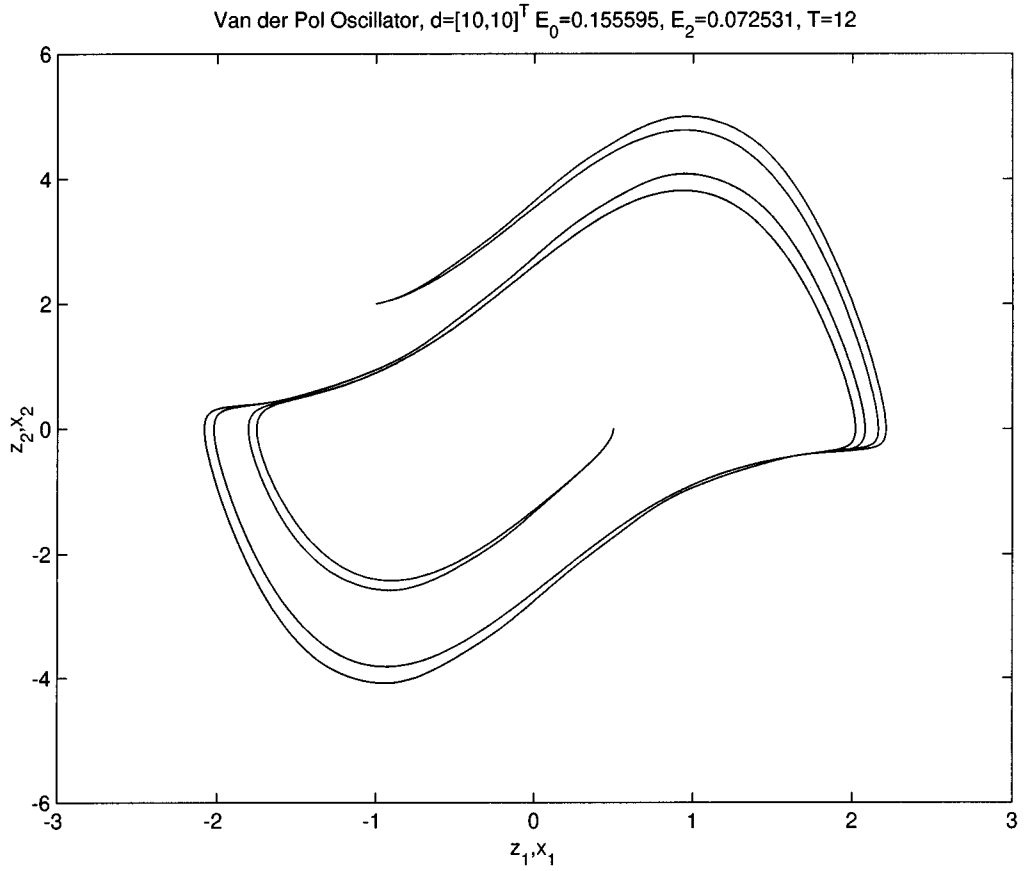
Suppose the initial state is $z(0) = [2, 10, 0]^T$ and the input is $u(t) = 10$. A plot of the three state variables $z(t)$ and the output $y(t)$ is shown in Fig. 4 for $0 \le t \le 3$. Also plotted is the solution $s(t) = [x_1(t), x_2(t), x_3(t), y_1(t)]^T$ obtained by representing the right-hand side functions with four first-order RBF networks. Note that over the time interval $0 \le t \le 1$, $x(t) \in S$ and the network solution $s(t)$ exactly matches the actual solution $r(t) = [z_1(t), z_2(t), z_3(t), y(t)]^T$ as predicted by Property 4. When $t > 1$, state $z_3(t)$ crosses the boundary of S, and the RBF network representation is no longer exact. This becomes evident by about $t = 1.4$ where the solutions $x_3(t)$ and $z_3(t)$ and the outputs $y_1(t)$ and $y(t)$ begin to separate. Since this linear system is not BIBO stable, the solution $||z(t)||$ continues to grow without bound and escapes the region S_1 at $t = 3$. However, the RBF network solution satisfies $x(t) \in S_1$ for $t \ge 0$. This is evident from Fig. 4 where the state variable $x_3(t)$ begins to saturate as it approaches the boundary of S_1, the point beyond which $dx/dt = 0$. Note that even though the RBF network is BIBO stable due to its compact support S_1, it can still be used to accurately approximate the solution of an unstable system for as long as the solution remains within the domain S over which the network is defined. Within this domain the approximation is exact.

Example 2—Nonlinear System with Limit Cycle: As an example of a nonlinear system, consider the following 2-D autonomous system called the van der Pol oscillator

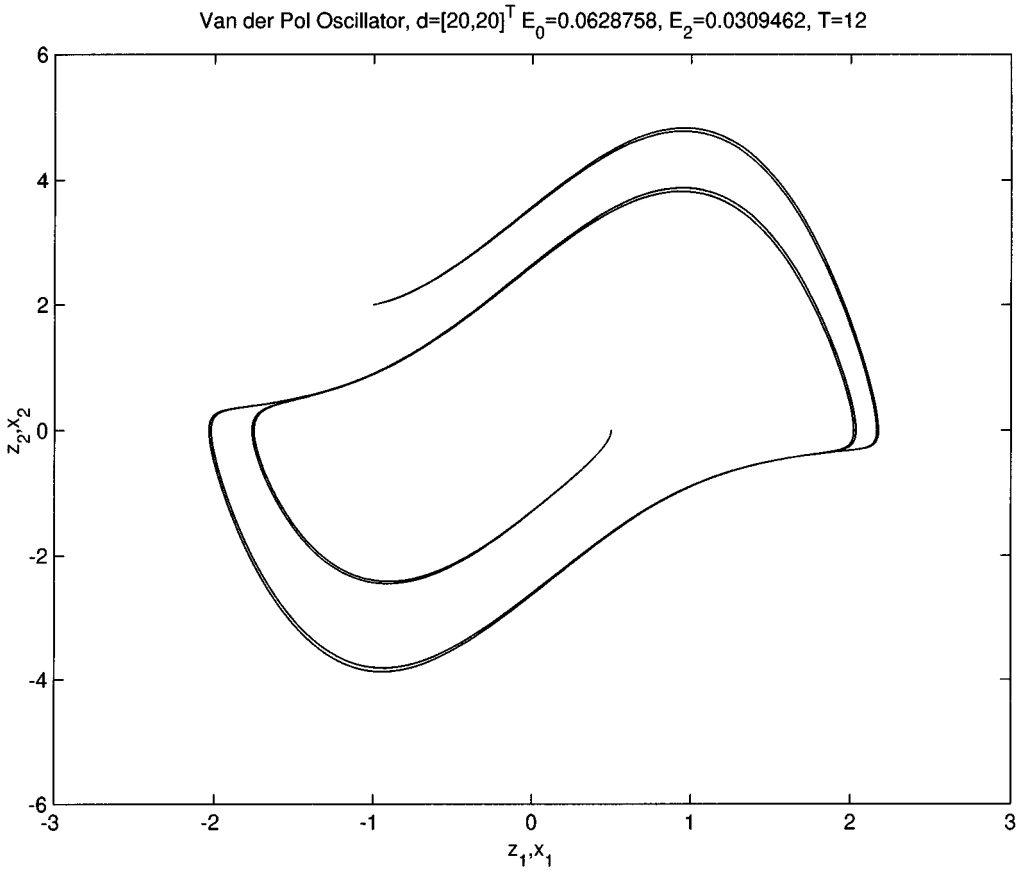
$$\frac{dz_1}{dt} = z_2$$
$$\frac{dz_2}{dt} = -z_1 + \mu(1 - z_1^2)z_2.$$

When $\mu = 0$ this is a simple linear harmonic oscillator, whereas when $\mu > 0$ the steady-state solution is a limit cycle with the shape of the limit cycle becoming less circular as μ increases. To approximate this system we use two RBF networks, each with input $x = z$. Note that the first network is actually linear, so from Property 4(a), we have $w_k = 0$ and $v^k = [0, 1]$ for $1 \le k \le r$. Suppose the lower limits of S are $a = [-3, -5]^T$ and the upper limits are $b = [3, 5]^T$. Let the number of grid points per dimension be $d = [\delta, \delta]^T$. Two initial conditions are considered, $z^0 = [0.5, 0]^T$ which is inside the limit cycle, and $z^1 = [-1, 2]^T$ which is outside.

The solution is computed at $p = 1000$ points uniformly distributed over the time interval $[0, 12]$. Fig. 5(a) shows the actual solution and the RBF network solution for the two initial conditions and $\delta = 10$ grid points per dimension. Thus using $r = 100$

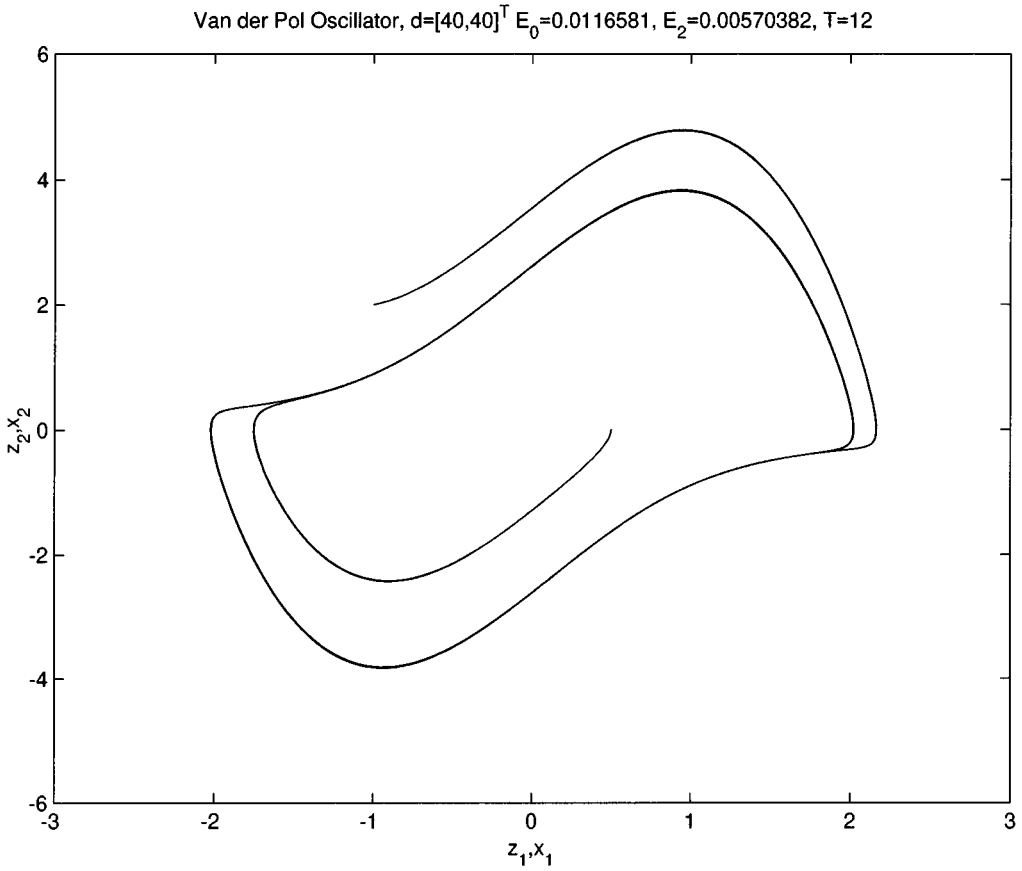


(a)



(b)

Fig. 5. Solutions of van der Pol oscillator with $a = [-3, -5]^T$ and $b = 3, 5]^T$: (a) the case $d = [10, 10]^T$ with $E_0 = 0.156$, $E_2 = 0.073$. (b) the case $d = [20, 20]^T$ with $E_0 = 0.063$, $E_2 = 0.031$.



(c)

Fig. 5. *(Continued.)* Solutions of van der Pol oscillator with $a = [-3, -5]^T$ and $b = 3, 5]^T$: (c) the case $d = [40, 40]^T$ with $E_0 = 0.012$, $E_2 = 0.006$.

terms, the RBF solution is generally similar to the actual solution, but clearly it exhibits some error. Let t_k denote the kth solution time for $1 \leq k \leq p$. If $||x||_\infty$ denotes the infinity norm and $||x||_2$ denotes the Euclidean norm, then the normalized errors over the solution trajectory can be computed as follows:

$$E_0 = \frac{\max_{k=1}^{p}\{||x(t_k) - z(t_k)||_\infty\}}{\max_{k=1}^{p}\{||z(t_k)||_\infty\}}$$

$$E_2 = \frac{\sqrt{\frac{1}{p}\sum_{k=1}^{p}||x(t_k) - z(t_k)||_2^2}}{\sqrt{\frac{1}{p}\sum_{k=1}^{p}||z(t_k)||_2^2}}.$$

For the case $d = [10, 10]^T$ shown in Fig. 5(a), the infinity norm error is $E_0 = 0.156$ and the root mean square (RMS) error is $E_2 = 0.073$. When the density of the grid increases, the error is reduced. For example, when $\delta = 20$, this leads to $r = 400$ terms and the errors are $E_0 = 0.063$ and $E_2 = 0.031$. The corresponding plot of the two superimposed solutions is shown in Fig. 5(b). Finally, for the case $\delta = 40$, shown in Fig. 5(c), a total of $r = 1600$ terms are used and the errors are reduced to $E_0 = 0.012$ and $E_2 = 0.006$. In this case the two solutions are indistinguishable in terms of viewing the plot. Although the size of the network grows by more than an order a magnitude in going from Fig. 5(a)–(c), recall that the execution speed is the same for all three cases because there are, at most, only four nonzero terms for each point z.

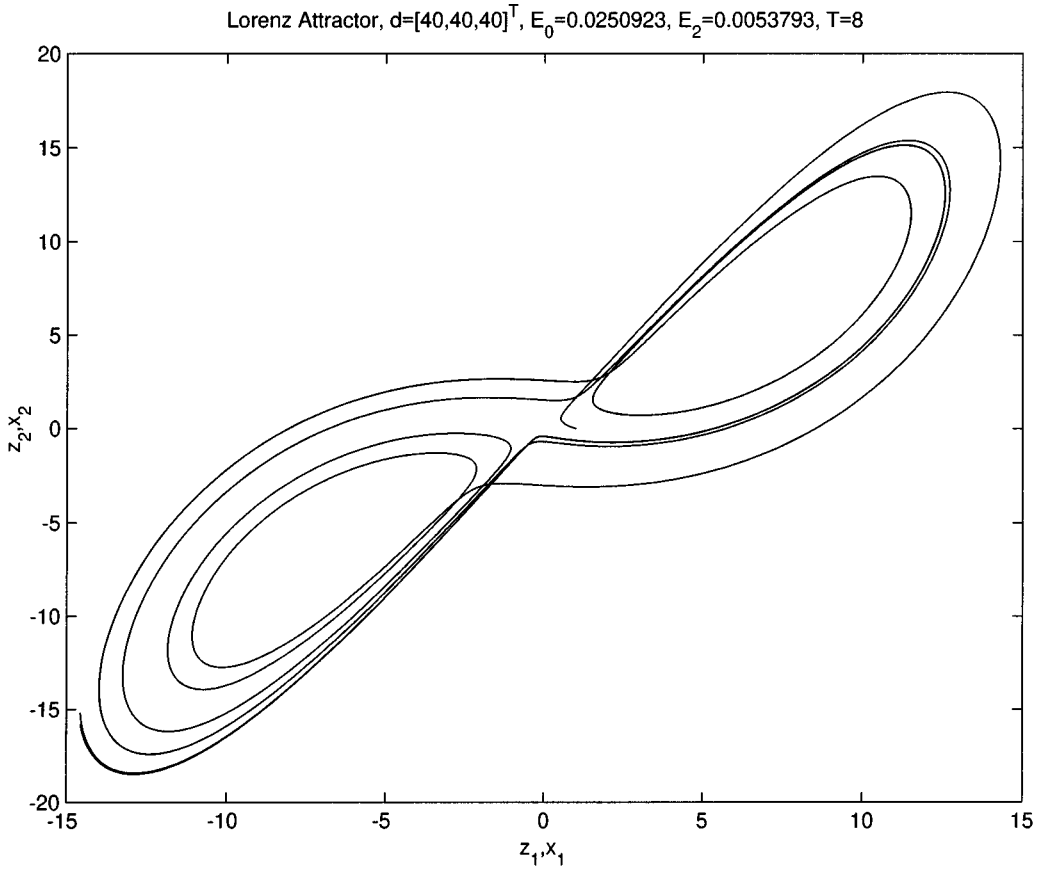
Example 3—Nonlinear System with Chaotic Motion: As a final example, one that is challenging to approximate, consider the following 3-D autonomous nonlinear system

$$\begin{aligned}\frac{dz_1}{dt} &= \sigma(z_2 - z_1)\\ \frac{dz_2}{dt} &= (1 + \lambda - z_3)z_1 - z_2\\ \frac{dz_3}{dt} &= z_1 z_2 - \gamma z_3.\end{aligned}$$

This system, known as the Lorenz attractor, can be used to model turbulent convection in fluids. In particular, the solution exhibits chaotic motion when the parameters satisfy the following inequalities [31]:

$$\begin{aligned}\sigma &> \gamma + 1\\ \lambda &> \frac{(\sigma + 1)(\sigma + \gamma + 1)}{\sigma - \gamma - 1.}\end{aligned}$$

To verify that an RBF approximation can exhibit chaotic motion, consider the case $(\sigma, \lambda, \gamma) = (10, 24, 2)$. Suppose the lower limits of the domain S are $a = [-20, -20, 0]^T$ and the upper limits are $b = [20, 20, 40]^T$. Finally, suppose that for each of the three RBF networks there are $d = [40, 40, 40]^T$ grid points. Although this is a substantial network with a total of $r = 64\,000$ terms, there are, at most, eight nonzero terms for



(a)

Fig. 6. Solutions of Lorenz attractor with $a = [-20, -20, 0]^T$, $b = [20, 20, 40]^T$, $d = [40, 40, 40]^T$, $E_0 = 0.025$, and $E_2 = 0.005$. (a) Projection onto z_1 versus z_2 plane.

any given z. The solution is computed at $p = 2000$ points uniformly distributed over the time interval $[0, 8]$ starting from an initial condition of $z(0) = [1, 0, 20]^T$.

Plots of the projections of the two solutions in R^3 onto the three orthogonal planes are shown in Fig. 6(a)–(c), respectively. In this case the errors defined in Example 2 are $E_0 = 0.025$ and $E_2 = 0.005$. It is evident from Fig. 6(a)–(c) that the RBF network provides an effective approximation even for this fairly complex nonlinear system whose solution exhibits chaos. When the length of the solution interval is decreased, the same or improved accuracy can be obtained with fewer grid points.

VII. Conclusion

A simple but effective technique for approximating a continuous function of n variables with an RBF network has been presented. The method uses an n-dimensional raised-cosine type of RBF that is smooth, yet has compact support. The coefficients of the RBF network are low-order polynomial functions of the input. A simple computational procedure is presented which significantly reduces the network training and evaluation time. Storage space is also reduced by allowing for a nonuniform grid of points about which the RBFs are centered. The network output is shown to be continuous and have a continuous first derivative. When the network is used to approximate a nonlinear dynamic system, the resulting system is BIBO stable. For the special case of a linear m-dimensional system, the RBF network representation is exact on the domain over which it is defined, and it is optimal in terms of the number of distinct storage parameters required. Several examples are presented which illustrate the effectiveness of this technique. These include an unstable linear system, a nonlinear system whose steady-state solution is a limit cycle, and a nonlinear system whose solution exhibits chaotic motion.

Appendix

A. Proposition 5 (Constant Interpolation)

Let S, r, and $\phi_k(x)$ be as defined in (1), (4), and (13), respectively. Then

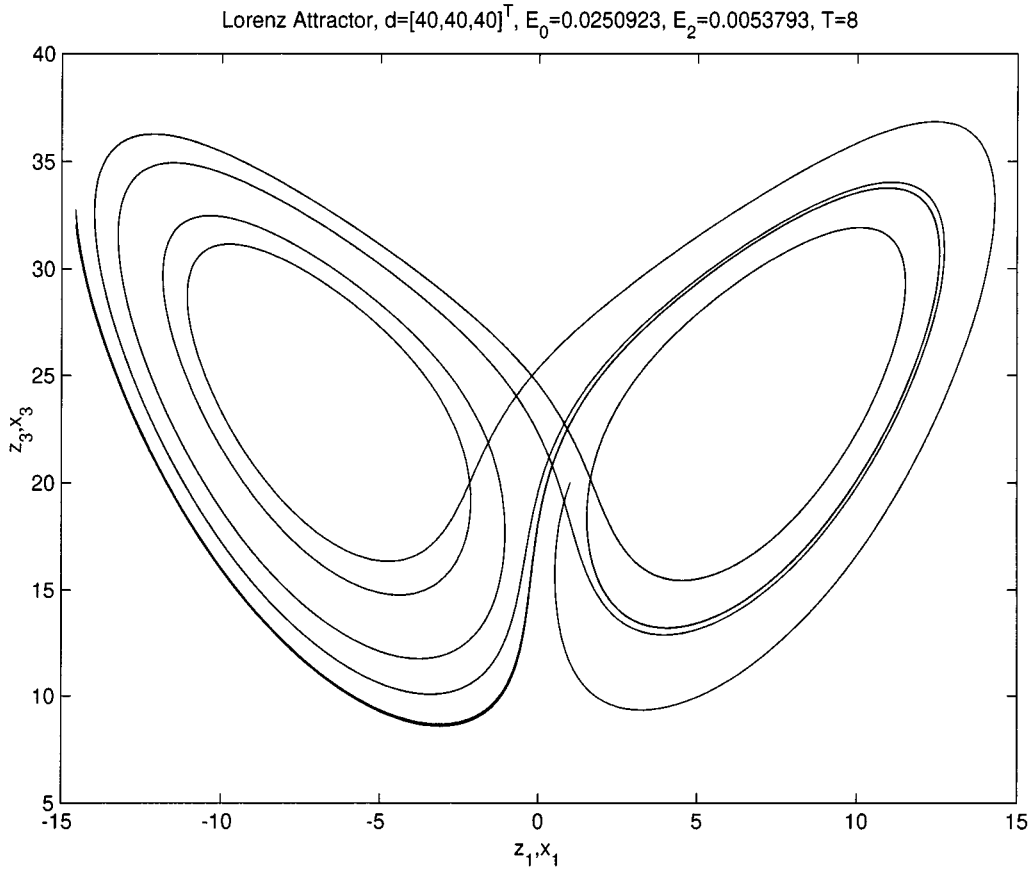
$$\sum_{k=1}^{r} \phi_k(x) = 1, \qquad x \in S.$$

To show that Proposition 5 holds, we first consider the special case when $r = 2^n$ which corresponds to $d_i = 2$, $x_{i1} = a_i$ and $x_{id_i} = b_i$ for $1 \le i \le n$. In this case the piecewise linear grid interpolation function in (5) simplifies to

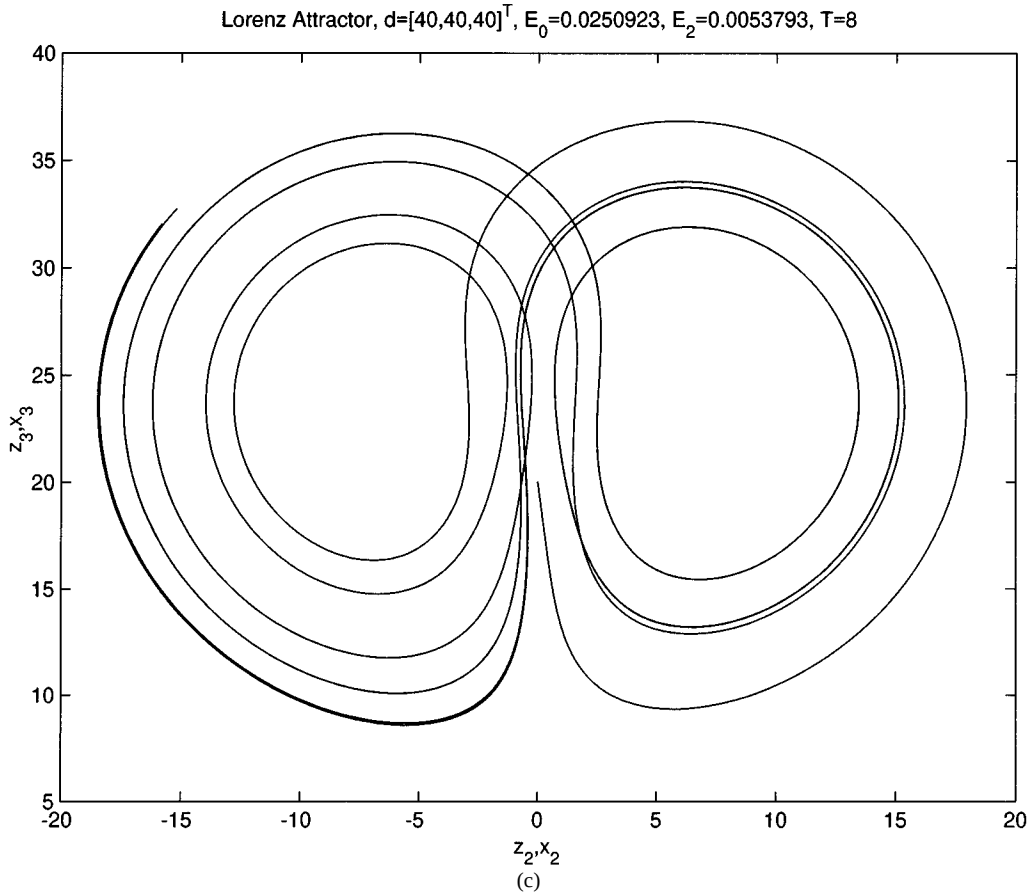
$$p_i(x_i) = 1 + \frac{x_i - a_i}{b_i - a_i}, \qquad 1 \le i \le n. \tag{34}$$

Next let $\psi_c(z)$ be the 1-D raised-cosine RBF in (12) and define

$$u_i(x_i) = \psi_c[p_i(x_i) - p_i(a_i)] + \psi_c[p_i(x_i) - p_i(b_i)] \qquad 1 \le i \le n. \tag{35}$$



(b)



(c)

Fig. 6. (*Continued.*) Solutions of Lorenz attractor with $a = [-20, -20, 0]^T$, $b = [20, 20, 40]^T$, $d = [40, 40, 40]^T$, $E_0 = 0.025$, and $E_2 = 0.005$. (b) Projection onto z_2 vs. z_3 plane. (c) Projection onto z_1 versus z_3 plane.

When $r = 2^n$, the terms in Proposition 5 correspond to RBFs centered at the 2^n vertices of the n-dimensional hypercube

$$S = [a_1, b_1] \times [a_2, b_2] \times \cdots \times [a_n, b_n]. \tag{36}$$

In particular, the sum of $r = 2^n$ terms can be expressed as a product of n sums as follows:

$$\sum_{k=1}^{2^n} \phi_k(x) = \prod_{i=1}^{n} u_i(x_i). \tag{37}$$

Consider the ith factor in the product. Using (34), the definition of $\psi_c(z)$ in (12), and a trigonometric identity yields

$$\begin{aligned} u_i(x_i) &= \psi_c[p_i(x_i) - p_i(a_i)] + \psi_c[p_i(x_i) - p_i(b_i)] \\ &= \psi_c[p_i(x_i) - 1] + \psi_c[p_i(x_i) - 2] \\ &= \frac{1 + \cos[\pi(x_i - a_i)/(b_i - a_i)]}{2} \\ &\quad + \frac{1 + \cos[\pi(x_i - a_i)/(b_i - a_i) - \pi]}{2} \\ &= 1 + \frac{\cos[\pi(x_i - a_i)/(b_i - a_i)]}{2} \\ &\quad - \frac{\cos[\pi(x_i - a_i)/(b_i - a_i)]}{2} \\ &= 1, \qquad 1 \le i \le n. \end{aligned} \tag{38}$$

From (37) and (38) it then follows that the constant interpolation property holds when $r = 2^n$.

Next, consider the more general case where $d_i \ge 2$ for $1 \le i \le n$. In this case, the pair of terms in (35) is replaced by a sum of $d_i - 1$ pairs of terms

$$v_i(x_i) = \sum_{j=1}^{d_i - 1} \{\psi_c[p_i(x_i) - p_i(x_{ij})] + \psi_c[p_i(x_i) - p_i(x_{i,j+1})]\}, \quad 1 \le i \le n. \tag{39}$$

Using this generalization of $u_i(x_i)$, the sum of $r = d_1\, d_2\, \ldots\, d_n$ terms can again be expressed as a product as in (37), but this time with $u_i(x_i)$ replaced by $v_i(x_i)$.

$$\sum_{k=1}^{r} \phi_k(x) = \prod_{i=1}^{n} v_i(x_i). \tag{40}$$

Next, let $h_{ij}(x_i)$ denote the jth term of the sum in (39). That is,

$$h_{ij}(x_i) = \psi_c[p_i(x_i) - p_i(x_{ij})] + \psi_c[p_i(x_i) - p_i(x_{i,j+1})] \quad 1 \le j \le d_i - 1. \tag{41}$$

Using the compact support property of $\psi_c(z)$, we have

$$h_{ik}(x_i) = 0, \qquad x_{ij} \le x_i \le x_{i,j+1},\ k \ne j. \tag{42}$$

That is, over the subinterval $[x_{ij}, x_{i,j+1}]$, the only term in (39) that contributes to $v_i(x_i)$ is the term $h_{ij}(x_i)$. Consequently, it is sufficient to examine the value of $h_{ij}(x_i)$ over the subinterval $[x_{ij}, x_{i,j+1}]$. Using (6) and the definition of $\psi_c(z)$ from (12), yields

$$\begin{aligned} h_{ij}(x_i) &= \psi_c[p_i(x_i) - p_i(x_{ij})] + \psi_c[p_i(x_i) - p_i(x_{i,j+1})] \\ &= \psi_c[p_i(x_i) - j] + \psi_c[p_i(x_i) - (j+1)] \\ &= \frac{1 + \cos[\pi(x_i - a_i)/(b_i - a_i) - j\pi]}{2} \\ &\quad + \frac{1 + \cos[\pi(x_i - a_i)/(b_i - a_i) - (j+1)\pi]}{2} \\ &= 1 + \frac{(-1)^j \cos[\pi(x_i - a_i)/(b_i - a_i)]}{2} \\ &\quad + \frac{(-1)^{j+1} \cos[\pi(x_i - a_i)/(b_i - a_i)]}{2} \\ &= 1, \qquad x_{ij} \le x_i \le x_{i,j+1}. \end{aligned} \tag{43}$$

Using (39)–(43), we then have

$$v_i(x_i) = 1, \qquad a_i \le x_i \le b_i,\ 1 \le i \le n. \tag{44}$$

Proposition 5 then follows from (40) and (44). •

It is worth pointing out that the raised-cosine RBF in (12) can be recast, using a trigonometric identity, as one member of the following family of RBFs.

$$\psi_i(z) \triangleq \begin{cases} \cos^i\left(\frac{\pi z}{2}\right), & |z| \le 1) \\ 0, & |z| > 1 \end{cases}. \tag{45}$$

The raised-cosine RBF in (12) corresponds to the case $i = 2$. This case is optimal in the sense that it is the only value of i for which the constant interpolation property holds. An illustration of the constant interpolation property, corresponding to $n = 2$, is shown in Fig. 3.

REFERENCES

[1] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamic systems using neural networks," *IEEE Trans. Neural Networks*, vol. 1, pp. 4–27, 1990.

[2] S. Chen and S. A. Billings, "Neural networks for nonlinear dynamic system modeling and identification," *Int. J. Contr.*, vol. 56, pp. 319–346, 1992.

[3] K. I. Funahashi and Y. Nakamura, "Approximation of dynamical systems by continuous time recurrent neural networks," *Neural Networks*, vol. 6, no. 6, pp. 801–806, 1993.

[4] N. Sadegh, "A perceptron network for functional identification and control of nonlinear systems," *IEEE Trans. Neural Networks*, vol. 4, pp. 982–988, 1993.

[5] T. Chen and H. Chen, "Approximation of continuous functional by neural networks with application to dynamic systems," *IEEE Trans. Neural Networks*, vol. 4, pp. 910–918, Nov. 1993.

[6] ——, "Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems," *IEEE Trans. Neural Networks*, vol. 6, pp. 911–917, July 1995.

[7] G. P. Liu, V. Kadirkamanathan, and S. A. Billings, "Variable neural networks for adaptive control of nonlinear systems," *IEEE Trans. Syst., Man, Cybern. C*, vol. 29, pp. 34–43, 1999.

[8] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, pp. 359–366, 1989.

[9] K.-I. Funahashi, "On the approximate realization of continuous mappings by neural networks," *Neural Networks*, vol. 2, pp. 183–192, 1989.

[10] T. Poggio and F. Girosi, "Networks for approximation and learning," *Proc. IEEE*, vol. 78, no. 9, pp. 1481–1497, 1990.

[11] E. K. Blum and L. K. Li, "Approximation theory and feedforward networks," *Neural Networks*, vol. 4, pp. 511–515, 1991.

[12] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Networks*, vol. 4, pp. 251–257, 1991.
[13] C. K. Chui and X. Li, "Approximation by ridge functions and neural networks with one hidden layer," *J. Approximation Theory*, vol. 70, pp. 131–141, 1992.
[14] S. Geva and J. Sitte, "A constructive method for multivariate function approximation by multilayer perceptrons," *IEEE Trans. Neural Networks*, vol. 3, pp. 621–624, July 1992.
[15] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken, "Multilayer feedforward networks with a nonpolynomial activation function can approximate any function," *Neural Networks*, vol. 6, pp. 861–867, 1993.
[16] S.-S. Yang and C.-S. Tseng, "An orthogonal neural network for function approximation," *IEEE Trans. Syst., Man, Cybern. B*, vol. 26, pp. 779–783, 1996.
[17] D. R. Hush and B. Horne, "Efficient algorithms of function approximation with piecewise linear sigmoidal networks," *IEEE Trans. Neural Networks*, vol. 9, pp. 1129–1141, Nov. 1998.
[18] T.-T. Lee and J.-T. Jeng, "The Chebyshev-polynomials-based unified model neural networks for function approximations," *IEEE Trans. Syst., Man, Cybern. B*, vol. 28, no. 6, pp. 925–935, 1998.
[19] J. Park and I. W. Sandberg, "Universal approximation using radial-basis-function networks," *Neural Comput.*, vol. 3, pp. 246–257, 1991.
[20] S. Chen, C. F. N. Cowan, and P. M. Grant, "Orthogonal least squares learning algorithm for radial basis function networks," *IEEE Trans. Neural Networks*, vol. 2, pp. 302–309, 1991.
[21] J. A. Leonard, M. A. Kramer, and L. H. Ungar, "Using radial basis functions to approximate a function and its error bounds," *IEEE Trans. Neural Networks*, vol. 3, p. 622, July 1992.
[22] L. Yingwei, N. Sundararajan, and P. Saratchandran, "Performance evaluation of a sequential minimal radial basis function (RBF) neural network learning algorithm," *IEEE Trans. Neural Networks*, vol. 9, pp. 308–318, Mar. 1998.
[23] A. Webb and S. Shannon, "Shape-adaptive radial basis functions," *IEEE Trans. Neural Networks*, vol. 9, Nov. 1998.
[24] C.-C. Lee, P.-C. Chung, J.-R. Tsai, and C.-I. Chang, "Robust radial basis function neural networks," *IEEE Trans. Syst., Man, Cybern. B*, vol. 29, pp. 674–685, 1999.
[25] N. B. Karayiannis and G. W. Mi, "Growing radial basis neural networks: Merging supervised and unsupervised learning with network growth techniques," *IEEE Trans. Neural Networks*, vol. 8, pp. 1492–1506, 1997.
[26] G. P. Liu, V. Kadirkamanathan, and S. A. Billings, "Stable sequential identification of continuous nonlinear dynamical systems by growing RBF networks," *Int. J. Contr.*, vol. 65, no. 1, pp. 53–69, 1996.
[27] V. Cherkassky, D. Gehring, and F. Mulier, "Comparison of adaptive methods for function estimation from samples," *IEEE Trans. Neural Networks*, vol. 7, pp. 969–984, 1996.
[28] L. Prechelt, "Proben1—A set of neural network benchmark problems and benchmarking rules," Fakultat fur Informatik, Univ. Karlsruhe, Germany, Tech. Rep. 21/94, Sept. 1994.
[29] R. J. Schilling and S. L. Harris, *Applied Numerical Methods for Engineers Using MATLAB and C*. Pacific Grove, CA: Brooks/Cole, 2000.
[30] M. Vidyasagar, *Nonlinear Systems Analysis*. Englewood Cliffs, NJ: Prentice-Hall, 1978.
[31] P. A. Cook, *Nonlinear Dynamic Systems*. Englewood Cliffs, NJ: Prentice-Hall, 1986.

Robert J. Schilling (SM'89) received the B.E.E. degree in electrical engineering from the University of Minnesota, Minneapolis, in 1969 and the M.S. and Ph.D. degrees in electrical engineering from the University of California, Berkeley, in 1970 and 1973, respectively.

He was a Lecturer in the Department of Electrical Engineering and Computer Science at the University of California, Santa Barbara, from 1974 to 1978. In 1978, he joined Department of Electrical and Computer Engineering at Clarkson University in Potsdam, NY, where he is now a Professor. He has authored four textbooks in the areas of engineering analysis, robotics, and numerical methods. His current research interests include adaptive signal processing, active noise control, nonlinear system identification, and robot motion planning and control.

James J. Carroll, Jr. (M'90) received the B.S. degree in electrical engineering from Syracuse University, Syracuse, NY, in 1989 and the M.S. degree in electrical engineering from Georgia Institute of Technology, Atlanta, in 1990. He received the Ph.D. degree in electrical engineering with a specialization in control systems and robotics from Clemson University, Clemson, SC, in 1993.

He is currently an Associate Professor in Clarkson University's Department of Electrical and Computer Engineering, Potsdam, NY. His current research interests include the intelligent control of mechatronic systems, rapid prototyping and application development, real-time data acquisition, signal processing and control systems, and signal processing for nondestructive evaluation.

Ahmad F. Al-Ajlouni (M'97) received the Ph.D. degree in electrical and computer engineering from Clarkson University, Potsdam, NY, in 1997.

He joined the faculty of Yarmouk University, Jordan, where he is an Assistant Professor in the Department of Communication Engineering. He is currently engaged in research and teaching in the areas of computer networks, digital signal processing, and active noise cancellation.