IV. Conclusion

We have investigated the stability of neural networks with two additive time-varying delay components. By constructing a new Lyapunov functional and estimating the derivative of the Lyapunov functional less conservatively, a new delay-dependent stability criterion was derived. When one of the two time-delays is constant, a new stability criterion was also given for neural networks with interval time-varying delays. An example demonstrated the reduced conservatism of the stability criteria.

References

[1] L. Chua and L. Yang, "Cellular neural networks: Applications," *IEEE Trans. Circuits Syst.*, vol. 35, no. 10, pp. 1273–1290, Oct. 1998.

[2] A. N. Michel and D. Liu, *Qualitative Analysis and Synthesis of Recurrent Neural Networks*. New York: Marcel Dekker, 2002.

[3] J. Liang and J. Cao, "A based-on LMI stability criterion for delayed recurrent neural networks," *Chaos, Solitons, Fractals*, vol. 28, no. 1, pp. 154–160, Apr. 2006.

[4] N. Ozcan and S. Arik, "Global robust stability analysis of neural networks with multiple time delays," *IEEE Trans. Circuits Syst. I*, vol. 53, no. 1, pp. 166–176, Jan. 2006.

[5] P. Liu and Q.-L. Han, "On stability of recurrent neural networks-an approach from volterra integro-differential equations," *IEEE Trans. Neural Netw.*, vol. 17, no. 1, pp. 264–267, Jan. 2006.

[6] X. Liao, Q. Liu, and W. Zhang, "Delay-dependent asymptotic stability for neural networks with distributed delays," *Nonlinear Anal.: Real World Appl.*, vol. 7, no. 5, pp. 1178–1192, Dec. 2006.

[7] T. Li, L. Guo, C. Sun, and C. Lin, "Further results on delay-dependent stability criteria of neural networks with time-varying delays," *IEEE Trans. Neural Netw.*, vol. 19, no. 4, pp. 726–730, Apr. 2008.

[8] Y. He, G. P. Liu, D. Rees, and M. Wu, "Stability analysis for neural networks with time-varying interval delay," *IEEE Trans. Neural Netw.*, vol. 18, no. 6, pp. 1850–1854, Nov. 2007.

[9] J. Cao, D. S. Huang, and Y. Qu, "Global robust stability of delayed recurrent neural networks," *Chaos, Solitons, Fractals*, vol. 23, no. 1, pp. 221–229, 2005.

[10] Q. Zhang, X. Wei, and J. Xu, "Delay-dependent global stability results for delayed Hopfield neural networks," *Chaos, Solitons, Fractals*, vol. 34, no. 2, pp. 662–668, 2007.

[11] X.-M. Zhang and Q.-L. Han, "New Lyapunov–Krasovskii functionals for global asymptotic stability of delayed neural networks," *IEEE Trans. Neural Netw.*, vol. 20, no. 3, pp. 533–539, Mar. 2009.

[12] S. Xu, J. Lam, and D. W. C. Ho, "A new LMI condition for delay-dependent asymptotic stability of delayed Hopfield neural networks," *IEEE Trans. Circuits Syst. II*, vol. 52, no. 3, pp. 230–234, Mar. 2006.

[13] H. Shao, "Delay-dependent approaches to globally exponential stability for recurrent neural networks," *IEEE Trans. Circuits Syst. II*, vol. 55, no. 6, pp. 591–595, Jun. 2008.

[14] H. Shao, "Delay-dependent stability for recurrent neural networks with time-varying delays," *IEEE Trans. Neural Netw.*, vol. 19, no. 9, pp. 1647–1651, Sep. 2008.

[15] H. Shao, "Improved delay-dependent globally asymptotic stability criteria for neural networks with a constant delay," *IEEE Trans. Circuits Syst. II*, vol. 55, no. 10, pp. 1071–1075, Oct. 2008.

[16] Y. He, G. Liu, and D. Rees, "New delay-dependent stability criteria for neural networks with time-varying delay," *IEEE Trans. Neural Netw.*, vol. 18, no. 1, pp. 310–314, Jan. 2007.

[17] Y. Zhao, H. Gao, and S. Mou, "Asymptotic stability analysis of neural networks with successive time delay components," *Neurocomputing*, vol. 71, nos. 13–15, pp. 2848–2856, Aug. 2008.

[18] L. Hu, H. Gao, and W. Zheng, "Novel stability of cellular neural networks with interval time-varying delay," *Neural Netw.*, vol. 21, no. 10, pp. 1458–1463, Dec. 2008.

[19] Q.-L. Han, "New results for delay-dependent stability of linear systems with time-varying delay," *Int. J. Syst. Sci.*, vol. 33, no. 3, pp. 213–228, 2002.

[20] X. Meng, J. Lam, B. Du, and H. Gao, "A delay-partitioning approach to the stability analysis of discrete-time systems," *Automatica*, vol. 46, no. 3, pp. 610–614, Mar. 2010.

Modeling of Complex-Valued Wiener Systems Using B-Spline Neural Network

Xia Hong and Sheng Chen

***Abstract*— In this brief, a new complex-valued B-spline neural network is introduced in order to model the complex-valued Wiener system using observational input/output data. The complex-valued nonlinear static function in the Wiener system is represented using the tensor product from two univariate B-spline neural networks, using the real and imaginary parts of the system input. Following the use of a simple least squares parameter initialization scheme, the Gauss–Newton algorithm is applied for the parameter estimation, which incorporates the De Boor algorithm, including both the B-spline curve and the first-order derivatives recursion. Numerical examples, including a nonlinear high-power amplifier model in communication systems, are used to demonstrate the efficacy of the proposed approaches.**

***Index Terms*— B-spline, complex-valued neural networks, De Boor algorithm, system identification, Wiener system.**

I. Introduction

A popular approach to nonlinear systems identification is to use the so-called block-oriented nonlinear models which comprise the linear dynamic models and static or memoryless nonlinear functions [1]–[4]. For example, the Wiener model comprises a linear dynamic model followed by a nonlinear static functional transformation. The Wiener model is a reasonable model for any linear systems with a nonlinear measurement device, nonlinear high power amplifier (HPA) in broadband communication transmitters [5], or some industrial/biological systems [6]–[11]. The model characterization/representation of the unknown nonlinear static function in the Wiener model is fundamental to its identification, control, and/or other signal processing applications. Various approaches have been developed in order to capture the *a priori* unknown nonlinearity in the Wiener system including the nonparametric method [12], subspace model identification methods [11], fuzzy modeling [13], and the parametric method [1], [7], [9].

With its best conditioning property, the B-spline curve has been widely used in computer graphics and computer-aided geometric design [14]. The B-spline curves consist of many polynomial pieces, offering versatility. The early work on the construction of B-spline curve is mathematically involved and numerically unstable [15]. The De Boor algorithm uses recurrence relations and is numerically stable [15]. The B-spline basis functions for nonlinear systems modeling have been widely applied [16]–[19].

Many signal processing applications involve complex-valued functional representations for signals and systems.

Manuscript received May 13, 2010; revised February 14, 2011; accepted February 17, 2011. Date of publication April 5, 2011; date of current version May 4, 2011.

X. Hong is with the School of Systems Engineering, University of Reading, Reading RG6 6LY, U.K. (e-mail: x.hong@reading.ac.uk).

S. Chen is with the School of Electronics and Computer Science, University of Southampton, Southampton SO17 1BJ, U.K. (e-mail: sqc@ecs.soton.ac.uk).

Color versions of one or more of the figures in this brief are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNN.2011.2119328

Complex-valued artificial neural networks have been studied theoretically and applied widely in nonlinear signal and data processing [20]–[27]. Note that most artificial neural networks cannot be automatically extended from a real domain to a complex domain because the resultant model would in general violate the Cauchy–Riemann conditions, which means the training algorithms become unusable. A number of analytic functions were introduced for the fully complex-valued multilayer perceptrons [21]. Alternatively, the problem can be avoided by two real-valued artificial neural networks, one processing the real part and the other processing the real imaginary part. In this brief, we propose a general complex-valued B-spline neural network for the modeling of the complex-valued Wiener system. Specifically, the complex-valued nonlinear static function in the Wiener system is modeled on the basis of the tensor product constructed from the two univariate B-spline neural networks which use the real and imaginary parts of the system input, respectively. We point out that the proposed approach is different from the existing complex-valued neural network based on spline functions [20], [28], [29] in either model representation or the identification algorithms (see the discussions at the end of Section II). It is shown that, by minimizing the mean square error (MSE) between the model output and the system output, the Gauss–Newton algorithm is readily applicable for the parameter estimation in the proposed model. We introduce the use of a simple least squares parameter initialization scheme, followed by the Gauss–Newton algorithm incorporating the De Boor recursion for both curve and the first-order derivatives. The motivation of the proposed methods is twofold. Firstly, this extends the B-spline model to accommodate a general complex-valued Wiener systems. Secondly, the proposed model based on B-spline functions has a significant advantage over many other modeling paradigms in that this enables stable and efficient evaluations of functional and derivative values, as required in nonlinear optimization algorithm, e.g., the Gauss–Newton algorithm used in this brief. The efficacy of the proposed approaches is demonstrated by using numerical examples. In particular, a nonlinear HPA system modeling example, which is crucial in any linearization techniques of broadband communication systems employing power-efficient nonlinear HPA transmitter [30], [31], is included.

II. Complex-Valued Wiener System and the Proposed Complex-Valued B-Spline Neural Network

A. Complex-Valued Wiener System

The complex-valued Wiener system consists of a cascade of two subsystems: a linear filter of order n representing the memory effect on the input signal as the first subsystem, followed by a nonlinear memoryless function $\Psi(\bullet) : \mathcal{C} \to \mathcal{C}$ as the second subsystem. The system can be represented by

$$\begin{aligned} w(t) &= H(z)y(t) \\ &= y(t) + h_1 y(t-1) \cdots + h_n y(t-n) \qquad (1) \\ d(t) &= \Psi(w(t)) + \xi(t) \qquad (2) \end{aligned}$$

with z transfer function $H(z)$ defined by

$$H(z) = \sum_{i=0}^{n} h_i z^{-i}, \quad h_0 = 1 \tag{3}$$

where $d(t) = d_R(t) + j \cdot d_I(t) \in \mathcal{C}$ is the system output and $y(t) = y_R(t) + j \cdot y_I(t) \in \mathcal{C}$ is the system input. $j = \sqrt{-1}$. $\xi(t) = \xi_R(t) + j\xi_I(t) \in \mathcal{C}$ is assumed to be a white complex-valued noise sequence independent of $y(t)$. Both $\xi_R(t)$ and $\xi_I(t)$ are zero mean and have a variance of σ^2. $w(t) = w_R(t) + j \cdot w_I(t) \in \mathcal{C}$ is the output of linear filter subsystem and the input to the nonlinear subsystem. $h_i = h_{i,R} + j \cdot h_{i,I}$, $(i = 1, \ldots, n)$ are complex-valued coefficients of the linear filter. n is assumed to be known. Denote $\mathbf{h} = [h_1, ..., h_n]^T \in \mathcal{C}^n$.

From (1)

$$\begin{cases} w_R(t) = y_R(t) + \sum_{i=1}^{n} (h_{i,R} y_R(t-i) - h_{i,I} y_I(t-i)) \\ w_I(t) = y_I(t) + \sum_{i=1}^{n} (h_{i,I} y_R(t-i) + h_{i,R} y_I(t-i)) \end{cases} \tag{4}$$

and for $i = 1, \ldots, n$, we have

$$\begin{cases} \frac{\partial w_R(t)}{\partial h_{i,R}} = y_R(t-i) \\ \frac{\partial w_R(t)}{\partial h_{i,I}} = -y_I(t-i) \\ \frac{\partial w_I(t)}{\partial h_{i,R}} = y_I(t-i) \\ \frac{\partial w_I(t)}{\partial h_{i,I}} = y_R(t-i). \end{cases} \tag{5}$$

Without significantly losing generality, the following assumptions are initially made about the problem.

Assumption 1: $\Psi(\bullet)$ is a one-to-one mapping, i.e., it is an invertible and continuous function.

Assumption 2: $d_R(t)$, $d_I(t)$, $w_R(t)$, and $w_I(t)$ are upper and lower bounded by some finite real values.

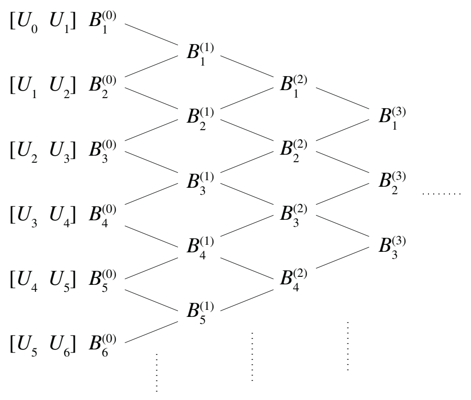
Our aim is to identify the system for the above Wiener model, i.e., given an observational input/output data set $D_N = \{y(t), d(t)\}_{t=1}^{K}$, to identify the underlying nonlinear function $\Psi(\bullet)$ and to estimate the parameters h_i's of the linear filter. Note that the signal $w(t)$ between the two subsystems are unavailable. In this brief, we introduce a complex-valued B-spline network in order to model $\Psi(\bullet)$, as introduced in the following.

B. New Complex-Valued B-Spline Neural Network

Consider the modeling of a complex mapping $a = f(b) = f(u + jv) : \mathcal{C} \to \mathcal{C}$, where u, v are real numbers.

Assumption 3: u is bounded by $U_{\min} < u < U_{\max}$, and v is bounded by $V_{\min} < v < V_{\max}$, where $U_{\min}$, $U_{\max}$, $V_{\min}$, and $V_{\max}$ are assumed known finite real values.

De Boor's algorithm is a fast and numerically stable algorithm for evaluating B-spline spline curves. A set of univariate B-spline basis functions based on u, which is the real part of b, is parameterized by the order of a piecewise polynomial of order $(k-1)$, and also by a knot vector which is a set of values defined on the real line which break it up into a number of intervals. Supposing that there are M_R basis functions, the knot vector is specified by (M_R+k) knot values, $\{U_1, U_2, \ldots, U_{M_R+k}\}$. At each end, there are k knots satisfying



$[U_0\ U_1]\ B_1^{(0)}$
$[U_1\ U_2]\ B_2^{(0)}$
$[U_2\ U_3]\ B_3^{(0)}$
$[U_3\ U_4]\ B_4^{(0)}$
$[U_4\ U_5]\ B_5^{(0)}$
$[U_5\ U_6]\ B_6^{(0)}$
$B_1^{(1)}$ $B_2^{(1)}$ $B_3^{(1)}$ $B_4^{(1)}$ $B_5^{(1)}$
$B_1^{(2)}$ $B_2^{(2)}$ $B_3^{(2)}$ $B_4^{(2)}$
$B_1^{(3)}$ $B_2^{(3)}$ $B_3^{(3)}$

Fig. 1. Visualizing De Boor algorithm.

the condition of being external to the input region, and as a result the number of internal knots is $(M_R - k)$. Specifically

$$U_1 < U_2 < U_k = U_{\min} < U_{k+1} < U_{k+2} < \cdots < U_{M_R} < U_{\max} = U_{M_R+1} < \cdots < U_{M_R+k}. \quad (6)$$

Given the predetermined knot vector, a set of M_R B-spline basis functions can be formed by using De Boor recursion [15], given by

$$B_l^{(\Re,0)}(u) = \begin{cases} 1, & \text{if } U_l \le u < U_{l+1} \\ 0, & \text{otherwise} \end{cases} \quad (7)$$
$$l = 1, \ldots, (M_R + k)$$

$$\left.\begin{array}{l} B_l^{(\Re,i)}(u) = \frac{u-U_l}{U_{i+l}-U_l} B_l^{(\Re,i-1)}(u) \\ \quad + \frac{U_{i+l+1}-u}{U_{i+l+1}-U_{l+1}} B_{l+1}^{(\Re,i-1)}(u), \\ \quad l = 1, \ldots, (M_R + k - i) \end{array}\right\} i = 1, \ldots, k. \quad (8)$$

The derivative of the B-spline basis function $B_l^{(\Re,k)}(u)$ can be readily computed as

$$\frac{d}{du}[B_l^{(\Re,k)}(u)] = \frac{k}{U_{k+l} - U_l} B_l^{(\Re,k-1)}(u) - \frac{k}{U_{k+l+1} - U_{l+1}} B_{l+1}^{(\Re,k-1)}(u) \quad (9)$$
$$l = 1, \ldots, M_R.$$

De Boor recursion can be graphically illustrated with reference to Fig. 1, in which the superscript $^{(\Re)}$ used in (7) and (8) has been removed from the plot for clarity. Note that the early work on the construction of B-spline curve is mathematically involved. Hence, another advantage of using De Boor's recursion is the flexibility in terms of the evaluations of functional and derivative values, since it can cope with different settings such as number of knots and polynomial order.

Similarly, a set of univariate B-spline basis functions can be established based on v, which is the imaginary part of b. Supposing that the order of the piecewise polynomial is predetermined also as $(k-1)$, a knot vector is then defined on the imaginary line in a similar manner. Supposing that there are M_I basis functions, the knot vector is then specified by $(M_I + k)$ knot values, $\{V_1, V_2, \ldots, V_{M_I+k}\}$. At each end, there are k knots satisfying the condition of being external to the input region, and as a result the number of internal knots is $(M_I - k)$. Specifically

$$V_1 < V_2 < V_k = V_{\min} < V_{k+1} < V_{k+2} < \cdots < V_{M_I} < V_{\max} = V_{M_I+1} < \cdots < V_{M_I+k}. \quad (10)$$

Similarly, a set of M_I B-spline basis functions are given by

$$B_m^{(\Im,0)}(v) = \begin{cases} 1, & \text{if } V_m \le v < V_{m+1} \\ 0, & \text{otherwise} \end{cases} \quad (11)$$
$$m = 1, \ldots, (M_I + k)$$

$$\left.\begin{array}{l} B_m^{(\Im,i)}(v) = \frac{v-V_m}{V_{i+m}-V_m} B_m^{(\Im,i-1)}(v) \\ \quad + \frac{V_{i+m+1}-v}{V_{i+m+1}-V_{m+1}} B_{m+1}^{(\Im,i-1)}(v), \\ \quad m = 1, \ldots, (M_I + k - i) \end{array}\right\} i = 1, \ldots, k. \quad (12)$$

The derivative of B-spline basis function $B_m^{(\Im,k)}(v)$ can be readily computed as

$$\frac{d}{dv}[B_m^{(\Im,k)}(v)] = \frac{k}{V_{k+m} - V_m} B_m^{(\Im,k-1)}(v) - \frac{k}{V_{k+m+1} - V_{l+1}} B_{m+1}^{(\Im,k-1)}(v) \quad (13)$$
$$m = 1, \ldots, M_I.$$

Using the tensor product between the two sets of univariate B-spline basis functions of $B_l^{(\Re,k)}(u)$'s and $B_m^{(\Im,k)}(v)$'s [18], a set of new B-spline basis functions $B_{l,m}^{(k)}(b)$ can be formed and used in the complex-valued B-spline neural network, given by

$$a = f(b) = f(u, v) = \sum_{l=1}^{M_R} \sum_{m=1}^{M_I} B_{l,m}^{(k)}(b)\omega_{l,m}$$
$$= \sum_{l=1}^{M_R} \sum_{m=1}^{M_I} B_l^{(\Re,k)}(u) B_m^{(\Im,k)}(v)\omega_{l,m} \quad (14)$$

where $\omega_{l,m} \in \mathcal{C}$ $(l = 1, \ldots, M_R,\ m = 1, \ldots, M_I)$ are complex-valued weights. Equation (14) can be decomposed as two real-valued neural networks, i.e., $a = a_R + j \cdot a_I$, where

$$a_R = \sum_{l=1}^{M_R} \sum_{m=1}^{M_I} B_l^{(\Re,k)}(u) B_m^{(\Im,k)}(v)\mathrm{Re}(\omega_{l,m}) \quad (15)$$

$$a_I = \sum_{l=1}^{M_R} \sum_{m=1}^{M_I} B_l^{(\Re,k)}(u) B_m^{(\Im,k)}(v)\mathrm{Im}(\omega_{l,m}). \quad (16)$$

Because of the piecewise nature of B-spline functions, for any point functional evaluation there are only k basis functions with nonzero values for each of the real and the imaginary part, leading to k^2 nonzero terms in both (15) and (16). This is advantageous as k can be set quite low. The computational cost of De Boor recursion is in $O(k^2)$. Thus the computational cost of calculating both (15) and (16) scales up to about three times of the De Boor recursion, including both real and imaginary parts evaluation and the tensor product calculations. Notably, there is a minimal additional cost for derivative evaluation, as (9) and (13) can be regarded as a byproduct of the De Boor recursion. Note that there are also only k nonzero first-order derivative terms in each of (9) and (13).

Complex-valued neural networks based on different spline functions have been researched [20], [28], [29], and our approach is clearly different in terms of either model representation or the identification algorithms. In [20], the model construction is based on a different type of spline function of Campmull–Rom cubic spline basis. The network introduced in [28] based on natural cubic splines is very limited because it only has complex weights, but do not take complex signal at all. Note that the basis functions in [20] and [28] do not have the property of partition of unity (convexity), which is a desirable property in achieving numerical stability. The work in [29] is more related to ours as it uses B-spline function. It is initially based on the concept of a splitting function $a = f_R(u) + j.f_I(v)$, where $f_R(u)$ are $f_I(v)$ is univariate real functional mapping. A generalized splitting function is also introduced, for which, due to the different construction approaches, it is not easy to establish an equivalence to our model or otherwise. Despite this, we point out that in [20], [28], and [29] the basis functions are constructed on the basis of cubic models without making use of De Boor recursion, which is highly relevant in our proposed system identification algorithm in order to achieve computational efficiency, numerical stability, and modeling flexibility. Finally, [20], [28], [29] has been used in the problem of system identification of the complex-valued Wiener system considered in this brief.

III. System Identification Algorithm Based on the Complex-Valued B-Spline Neural Network

Consider the system identification of the complex-valued Wiener system given by (1) and (2) based on a block of training data $\{y(t), d(t)\}_{t=1}^K$. For notational simplicity, assuming that *a set of knots are appropriately set based on both the real and imaginary parts of* $w(t)$ [see (6) and (10)] and that the associated polynomial degree is still denoted as $(k-1)$ and the number of basis functions for both the real and imaginary parts are still denoted by M_R and M_I, the complex-valued B-spline neural network used in representing $\Psi(\bullet)$ is given by

$$\hat{d}(t) = \Psi(w(t)) = \sum_{l=1}^{M_R}\sum_{m=1}^{M_I} B_l^{(\Re,k)}(w_R(t))B_m^{(\Im,k)}(w_I(t))\omega_{l,m}. \tag{17}$$

Denoting $\hat{d}(t) = \hat{d}_R(t) + j\cdot\hat{d}_I(t)$, (17) is decomposed into two real valued networks as

$$\hat{d}_R(t) = \sum_{l=1}^{M_R}\sum_{m=1}^{M_I} B_l^{(\Re,k)}(w_R(t))B_m^{(\Im,k)}(w_I(t))\mathrm{Re}(\omega_{l,m}) \tag{18}$$

$$\hat{d}_I(t) = \sum_{l=1}^{M_R}\sum_{m=1}^{M_I} B_l^{(\Re,k)}(w_R(t))B_m^{(\Im,k)}(w_I(t))\mathrm{Im}(\omega_{l,m}). \tag{19}$$

Denote the total number of terms in (18) or (19) as $M = M_R\cdot M_I$. Let the error between the Wiener system output $d(t)$ and the B-spline network output $\hat{d}(t)$ be denoted by $e(t) = d(t) - \hat{d}(t) = e_R(t) + j\cdot e_I(t) \in \mathcal{C}$. Our task is to estimate $\mathbf{h}$ and $\omega_{l,m}$'s. This could be achieved by minimizing

$$J = \sum_{t=1}^K [e_R(t)]^2 + \sum_{t=1}^K [e_I(t)]^2 \tag{20}$$

using the Gauss–Newton algorithm. As the objective function of (20) is highly nonlinear, it is important that $\mathbf{h}$ and $\omega_{l,m}$'s are properly initialized so that it is close to optimal solution. Hence a simple least squares parameter initialization scheme is introduced in Sections III-A and III-B.

A. Initialization of the Linear Filter Parameters h_i's

The initialization of the linear filter parameters is illustrated with reference to Fig. 2(a). Denote the inverse function of $\Psi(\bullet)$ or $\Psi^{-1}(\bullet)$, as $\varphi(\bullet)$. Consider also using the proposed complex-valued B-spline neural network (14) for the modeling of $\varphi(\bullet)$. For notational simplicity, it is assumed that the polynomial degree used is still denoted as $(k-1)$ and the numbers of basis functions used in the modeling of the real and imaginary parts are still denoted as M_R and M_I, respectively. With the two knot vectors for the real and imaginary parts being set based on $d_R(t)$ and $d_I(t)$, respectively, we have

$$\varphi(d(t)) = \sum_{l=1}^{M_R}\sum_{m=1}^{M_I} B_{l,m}^{(k)}(d(t))\,\alpha_{l,m} \tag{21}$$

where $\alpha_{l,m} \in \mathcal{C}$, $(l = 1,\ldots,M_R,\ m = 1,\ldots,M_I)$ are complex-valued weights. Let the error between $w(t)$ and $\varphi(d(t))$ be defined as $\varepsilon(t)$. Note that Fig. 2(a) describes the error feedback for parameter optimization, in which $w(t)$ is used as the target for $\varphi(d(t))$.

Applying (1) and (2) yields

$$\begin{aligned} y(t) &= -\sum_{i=1}^n h_i y(t-i) - \sum_{l=1}^{M_R}\sum_{m=1}^{M_I} B_{l,m}^{(k)}(d(t))\,\alpha_{l,m} + \varepsilon(t) \\ &= [\mathbf{p}(\mathbf{x}(t))]^T\boldsymbol{\vartheta} + \varepsilon(t) \end{aligned} \tag{22}$$

where $\mathbf{x}(t) = [-y(t-1),\ldots,-y(t-n),d(t)]^T$, $\boldsymbol{\vartheta} = [\vartheta_1,\ldots,\vartheta_{M+n}]^T = [-h_1,\ldots,-h_n,\alpha_{1,1},\ldots,\alpha_{l,m},\ldots,\alpha_{M_R,M_I}]^T \in \mathcal{C}^{M+n}$. $\mathbf{p}(\mathbf{x}(t)) = [p_1(\mathbf{x}(t)),\ldots,p_{M+n}(\mathbf{x}(t))]^T = [-y(t-1),\ldots,-y(t-n),B_{1,1}^{(k)}(d(t)),\ldots,B_{l,m}^{(k)}(d(t)),\ldots,B_{M_R,M_I}^{(k)}(d(t))]^T \in \mathcal{C}^{M+n}$.

Over the training dataset, (22) can be written in matrix form as

$$\mathbf{y} = \mathbf{P}\boldsymbol{\vartheta} + \boldsymbol{\varepsilon} \tag{23}$$

where $\mathbf{y} = [y(1),\ldots,y(K)]^T$, $\boldsymbol{\varepsilon} = [\varepsilon(1),\ldots,\varepsilon(K)]^T$, and $\mathbf{P}$ is the regression matrix $\mathbf{P} = [\mathbf{p}(\mathbf{x}(1)),\ldots,\mathbf{p}(\mathbf{x}(K))]^T$. The parameter vector $\boldsymbol{\vartheta}$ can be found as the least squares solution of

$$\boldsymbol{\vartheta}_{LS} = \left(\mathbf{P}^H\mathbf{P}\right)^{-1}\mathbf{P}^H\mathbf{y}. \tag{24}$$

A subvector of the resultant $\boldsymbol{\vartheta}_{LS}$, consisting of its first n elements, forms our initial estimate $\hat{\mathbf{h}}^{(0)}$.

B. Initialization of the B-Spline Network Weights $\omega_{l,m}$'s

The initialization of the B-spline network weights $\omega_{l,m}$'s is illustrated in Fig. 2(b). Consider generating an auxiliary signal $\{\hat{w}(t)\}_{t=1}^K$ over training dataset $\{y(t), d(t)\}_{t=1}^K$, based on the initialized parameter estimates $\hat{\mathbf{h}}^{(0)}$, given by

$$\hat{w}(t) = y(t) + \hat{h}_1^{(0)}y(t-1) + \hat{h}_2^{(0)}y(t-2) + \cdots + \hat{h}_n^{(0)}y(t-n). \tag{25}$$

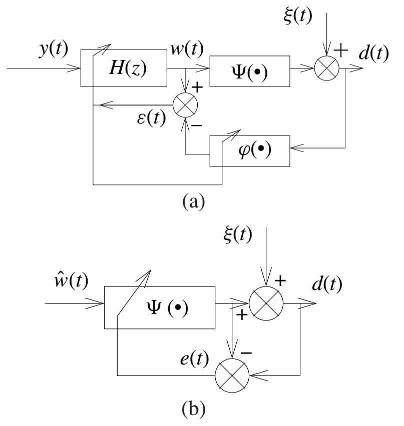


Fig. 2. Successive initialization for (a) linear filter parameters h_i's and (b) B-spline network weights $\omega_{l,m}$'s.

A block of training dataset $\{\hat{w}(t), d(t)\}_{t=1}^{K}$ will be used in the parameter initialization for modeling of the nonlinear memoryless function. Using (17), but with $w(t)$ replaced by its estimates $\hat{w}(t)$, we have

$$\begin{aligned} d(t) &= \sum_{l=1}^{d_R} \sum_{m=1}^{M_I} B_{l,m}^{(k)}\left(\hat{w}(t)\right)\omega_{l,m} + e(t) \\ &= [\mathbf{q}(\hat{w}(t))]^T \boldsymbol{\omega} + e(t) \end{aligned} \tag{26}$$

where $\boldsymbol{\omega} = [\omega_{1,1}, \ldots, \omega_{l,m}, \ldots, \omega_{M_R,M_I}]^T \in \mathcal{C}^M$, $\mathbf{q}(\hat{w}(t)) = [q_1(\hat{w}(t)), \ldots, q_M(\hat{w}(t))]^T = [B_{1,1}^{(k)}\left(\hat{w}(t)\right), \ldots, B_{l,m}^{(k)}\left(\hat{w}(t)\right),$ $\ldots, B_{M_R,M_I}^{(k)}\left(\hat{w}(t)\right)]^T \in \mathcal{C}^M$. Over the training dataset, (26) can be written in matrix form

$$\mathbf{d} = \mathbf{Q}\boldsymbol{\omega} + \boldsymbol{e} \tag{27}$$

where $\mathbf{d} = [d(1), \ldots, d(K)]^T$, $\mathbf{e} = [e(1), \ldots, e(K)]^T$ and $\mathbf{Q} = [\mathbf{q}(\hat{w}(1)), \ldots, \mathbf{q}(\hat{w}(K))]^T$. The least squares solution of $\boldsymbol{\omega}$ given by

$$\boldsymbol{\omega}_{LS} = \left(\mathbf{Q}^H\mathbf{Q}\right)^{-1}\mathbf{Q}^H\mathbf{d} \tag{28}$$

is used as the initial estimate of $\hat{\boldsymbol{\omega}}^{(0)}$. Note that the initial parameter estimates obtained are only near, but not, optimal. For example, in the noisy environment, they are not consistent. This is because the B-spline basis functions (regressors) in (21) are subject to the output noise, such that the output noise will in general propagate to the parameter estimates, yielding a bias. However, our final parameter estimates via minimizing (20) are optimal in the sense that these are the maximum likelihood estimates in the case that $\xi(t)$ is Gaussian. The final parameter estimate is obtained using the Gauss–Newton algorithm combined with the De Boor algorithm, as presented below.

C. Gauss–Newton Algorithm Combined with the De Boor Algorithm

Define $\boldsymbol{\theta} = [\theta_1, \theta_2, \ldots, \theta_{2(M+n)}]^T = [\mathrm{Re}(\omega_{1,1}), \ldots, \mathrm{Re}(\omega_{l,m}), \ldots, \mathrm{Re}(\omega_{M_R,M_I}), \mathrm{Im}(\omega_{1,1}), \ldots, \mathrm{Im}(\omega_{l,m}), \ldots, \mathrm{Im}(\omega_{M_R,M_I}), h_{1,R}, \ldots, h_{n,R}, h_{1,I}, \ldots, h_{n,I}]^T \in \Re^{2(M+n)}$. Note that the first $2M$ elements in $\boldsymbol{\theta}^{(0)}$ are formed using the real and imaginary parts of $\hat{\boldsymbol{\omega}}^{(0)}$, and the last $2n$ elements in $\boldsymbol{\theta}^{(0)}$ are formed using the real and imaginary parts of $\hat{\mathbf{h}}^{(0)}$. Denote $\boldsymbol{\epsilon} = [\epsilon_1, \epsilon_2, \ldots, \epsilon_{2K}]^T = [e_R(1), \ldots, e_R(K), e_I(1), \ldots, e_I(K)]^T \in \Re^{2K}$.

Denote an iteration step variable by a superscript $^{(\tau)}$. With the initial value $\boldsymbol{\theta}^{(0)}$, the iteration formula is given by

$$\boldsymbol{\theta}^{(\tau)} = \boldsymbol{\theta}^{(\tau-1)} + \alpha\left\{\left[\mathbf{J}^{(\tau)}\right]^T\mathbf{J}^{(\tau)}\right\}^{-1}\left[\mathbf{J}^{(\tau)}\right]^T\boldsymbol{\epsilon}\left(\boldsymbol{\theta}^{(\tau-1)}\right) \tag{29}$$

where $\alpha > 0$ is a small positive step size. $\mathbf{J}$ denotes the Jacobian of $\boldsymbol{\epsilon}(\boldsymbol{\theta})$ and is given by $\mathbf{J} = (\{\partial\epsilon_p/\partial\theta_q\}), p = 1, \ldots, 2K, q = 1, \ldots, 2(M+n)$. For $p = 1, \ldots, K$, and $t = p$

$$\frac{\partial\epsilon_p}{\partial\theta_q} = \begin{cases} \frac{\partial e_R(t)}{\partial \mathrm{Re}(\omega_{l,m})} = -B_l^{(\Re,k)}(w_R(t))B_m^{(\Im,k)}(w_I(t)), \\ \qquad \text{for} \ \ q = 1, \ldots, M \\ \frac{\partial e_R(t)}{\partial \mathrm{Im}(\omega_{l,m})} = 0, \quad \text{for} \ \ q = M+1, \ldots, 2M \\ \frac{\partial e_R(t)}{\partial h_{i,R}} = -\sum_{l=1}^{M_R}\sum_{m=1}^{M_I} \\ \left(\frac{d}{dw_R(t)}\left[B_l^{(\Re,k)}(w_R(t))\right]B_m^{(\Im,k)}(w_I(t))y_R(t-i)\right. \\ \left.+B_l^{(\Re,k)}(w_R(t))\frac{d}{dw_I(t)}[B_m^{(\Im,k)}(w_I(t))]y_I(t-i)\right)\mathrm{Re}(\omega_{l,m}) \\ \text{for} \ \ q = 2M+1, \ldots, 2M+n, \ (i = q-2M) \\ \frac{\partial e_R(t)}{\partial h_{i,I}} = -\sum_{l=1}^{M_R}\sum_{m=1}^{M_I} \\ \left(\frac{d}{dw_R(t)}[B_l^{(\Re,k)}(w_R(t))]B_m^{(\Im,k)}(w_I(t))(-y_I(t-i))\right. \\ \left.+B_l^{(\Re,k)}(w_R(t))\frac{d}{dw_I(t)}[B_m^{(\Im,k)}(w_I(t))]y_R(t-i)\right)\mathrm{Re}(\omega_{l,m}) \\ \text{for} \ q = 2M+n+1, \ldots, 2(M+n)(i = q-2M-n) \end{cases} \tag{30}$$

but for $p = K+1, \ldots, 2K$, and $t = (p-K)$

$$\frac{\partial\epsilon_p}{\partial\theta_q} = \begin{cases} \frac{\partial e_I(t)}{\partial \mathrm{Re}(\omega_{l,m})} = 0 \ \ \text{for} \ \ q = 1, \ldots, M \\ \frac{\partial e_I(t)}{\partial \mathrm{Im}(\omega_{l,m})} = -B_l^{(\Re,k)}(w_R(t))B_m^{(\Im,k)}(w_I(t)) \\ \ \ \text{for} \ q = M+1, \ldots, 2M \\ \frac{\partial e_I(t)}{\partial h_{i,R}} = -\sum_{l=1}^{M_R}\sum_{m=1}^{M_I} \\ \left(\frac{d}{dw_R(t)}[B_l^{(\Re,k)}(w_R(t))]B_m^{(\Im,k)}(w_I(t))y_R(t-i)\right. \\ \left.+B_l^{(\Re,k)}(w_R(t))\frac{d}{dw_I(t)}[B_m^{(\Im,k)}(w_I(t))]y_I(t-i)\right)\mathrm{Im}(\omega_{l,m}) \\ \ \ \text{for} \ \ q = 2M+1, \ldots, 2M+n \ (i = q-2M) \\ \frac{\partial e_I(t)}{\partial h_{i,I}} = -\sum_{l=1}^{M_R}\sum_{m=1}^{M_I} \\ \left(\frac{d}{dw_R(t)}[B_l^{(\Re,k)}(w_R(t))]B_m^{(\Im,k)}(w_I(t))(-y_I(t-i))\right. \\ \left.+B_l^{(\Re,k)}(w_R(t))\frac{d}{dw_I(t)}[B_m^{(\Im,k)}(w_I(t))]y_R(t-i)\right)\mathrm{Im}(\omega_{l,m}) \\ \text{for} \ q = 2M+n+1, \ldots, 2(M+n) \ \ (i = q-2M-n) \end{cases} \tag{31}$$

in which (5) has been applied. Note we propose that De Boor algorithm [(7)–(9) and (11)–(13)] is applied in evaluating all entries for (30) and (31). Effectively, this enables stable and efficient evaluations of B-spline functional and derivative values, which would be very difficult for many other nonlinear models including some spline-functions-based nonlinear models. The iterative equation (29) can be

TABLE I
RESULTS OF LINEAR SUBSYSTEM PARAMETER ESTIMATION (EXAMPLE 1)

	True parameters	Initial estimates	Final estimates
h_1	0.7692	$0.7692 + j0.0001$	$0.7692 + j7 \times 10^{-6}$
h_2	0.1538	$0.1537 - j0.0002$	$0.1538 + j1 \times 10^{-5}$
h_3	0.0769	$0.0772 - j0.0001$	$0.0769 + j7 \times 10^{-6}$

terminated when $\boldsymbol{\theta}^{(\tau)}$ converges, or by predetermining a sufficiently large number of iterations.

Summary of the System Identification Algorithm: The system identification algorithm can be summarized as follows.

1) Predetermine two sets of knot vectors that break the domain of $d_R(t)$ and $d_I(t)$ up, with k knots satisfying the condition of being external to the regions for $d_R(t)$ and $d_I(t)$ at each end.
2) Form $\mathbf{P}$ and $\mathbf{y}$, and apply (24). Obtain $\mathbf{h}^{(0)}$ as the subvector of the resultant $\boldsymbol{\vartheta}_{LS}$, consisting of the first n elements.
3) Construct an auxiliary signal $\{\hat{w}(t)\}_{t=1}^{K}$ based on (25).
4) Predetermine two sets of knot vectors that break the domain $w_R(t)$, $w_I(t)$ up respectively, with k knots satisfying the condition of being external to regions of $w_R(t)$ and $w_I(t)$ respective at each end.
5) Form $\mathbf{Q}$ and $\mathbf{d}$, obtain $\boldsymbol{\omega}^{(0)}$ using (28).
6) Form $\boldsymbol{\theta}^{(0)}$ from the real and imaginary parts of $\mathbf{h}^{(0)}$ and $\boldsymbol{\omega}^{(0)}$.
7) Apply the Gauss–Newton algorithm (29)–(31).

The optimization of model output with respect to the number/location of knots is an intractable mixed-integer problem, for which an iterative trial-and-error approach can be used to yield a good model (but not optimal). For a prior unknown system, the initial knots' location should be set as evenly spread out in the input region. With the number of knots and their location determined, conventional nonlinear optimization algorithms are applicable, e.g., the proposed algorithm. In practice, the number of knots is predetermined to produce a model as small as possible (to avoid overfitting) that can still provide good modeling capability. A simple iteration of the proposed approach can be used. The number of knots is increased, and the model performance is monitored until the improvement becomes insignificant. For many problems, the model performance is not sensitive to the location of knots to a large extent if these are evenly spread out. However, if there is severe local nonlinearity, the location of knots can be empirically set by the user by inserting more knots at higher density in regions with high curvatures. These regions can be identified by trial and error (through identifying the data points with high modeling errors) during an iterative modeling process.

IV. EXPERIMENTAL RESULTS

Example 1: Consider a complex-valued Wiener system consisting of a linear filter $H(z) = 1 + 0.7692z^{-1} + 0.1538z^{-2} + 0.0769z^{-3}$, followed by a combined complex mapping of rotation and translation, given by

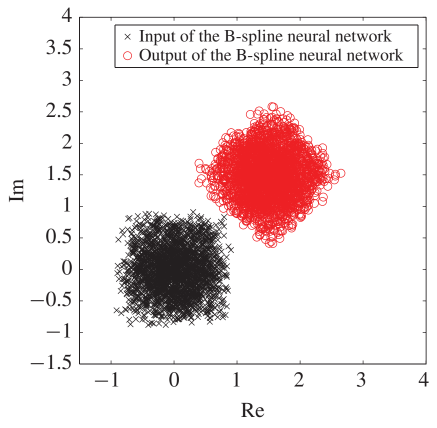


Fig. 3. Modeling results of the combined complex mapping of rotation and translation (Example 1).

$$\Psi(w(t)) = w(t)\exp\left(-j\frac{\pi}{4}\right) + 1.5 + 1.5j. \qquad (32)$$

Two thousand training data samples and 500 validations data samples $d(t)$ were generated by using (1), (2), and (32). $y(t)$ was a uniformly distributed complex random variable with $y_R(t) \in [-0.5, 0.5]$ and $y_I(t) \in [-0.5, 0.5]$. The variance of the additive noise to the system output is set as zero ($\sigma^2 = 0$). The polynomial degree of the B-spline basis functions for both the real and imaginary parts was set as two (i.e., $k = 3$, piecewise quadratic). The knot sequence

$$[-0.5, 0.3, 0.5, 0.7, 1.0, 1.5, 2.0, 2.3, 2.5, 2.7, 3.5]$$

is initially set for both $d_R(t)$ and $d_I(t)$ in order to generate basis functions used in (21). The knot sequence

$$[-2, -1.2, -1, -0.8, -0.5, 0, 0.5, 0.8, 1, 1.2, 2]$$

is then set for both $w_R(t)$ and $w_I(t)$ in order to generate basis functions used in (17). The parameter estimation results are shown in Table I for the linear subsystem. Fig. 3 plots data samples from the B-spline neural network model data samples, which clearly matches the complex mapping (32). The MSE between $d(t)$ and $\hat{d}(t)$ over the validation dataset is very small at 2.4635×10^{-4}.

Example 2: We further illustrate the modeling of the complex-valued Wiener system using a HPA model with high importance in communication systems. The HPA model consists of a linear filter followed by a nonlinearity of the traveling wave tube (TWT) [5]. For the baseband HPA model, the input to the TWT nonlinearity can be expressed as

$$w(t) = |w(t)|\exp(j\angle w(t)) = r_w(t)\exp(j\phi_w(t)) \qquad (33)$$

where $r_w(t)$ and $\phi_w(t)$ denote the amplitude and phase of $w(t)$, respectively. The output of TWT, $\Psi(w(t))$, is distorted in both amplitude and phase, with the distortion dependent mainly on the input signal amplitude, i.e., $r_w(t)$. So $\Psi(w(t))$ can be expressed by [5]

$$\Psi(w(t)) = |\Psi(w(t))|\exp(j\Psi(w(t))) = r_\Psi(t)\exp(j\phi_\Psi(t)) \qquad (34)$$

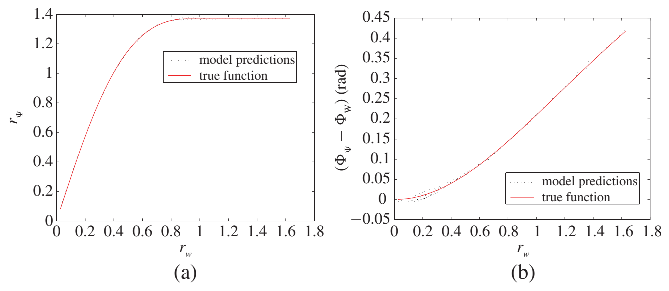


Fig. 4. TWT nonlinearity modeling results (Example 2). (a) Amplitude distortion with respect to the amplitude of the input. (b) Phase shift with respect to the amplitude of the input.

where $r_\Psi(t)$ and $\phi_\Psi(t)$ denote the amplitude and phase of $\Psi(w(t))$, respectively, and these are given by

$$r_\Psi(t) = \begin{cases} \frac{\alpha_1 r_w(t)}{1+\alpha_2 r_w^2(t)}, & 0 \le r_w(t) \le r_{Sat} \\ \Psi_{max}, & r_w(t) > r_{Sat} \end{cases} \tag{35}$$

$$\phi_\Psi(t) = \phi_w(t) + \frac{\beta_1 r_w^2(t)}{1+\beta_2 r_w^2(t)} \tag{36}$$

where $\alpha_1, \alpha_2, \beta_1, \beta_2$ are unknown parameters. $r_{Sat} = \sqrt{1/\alpha_2}$ and $\Psi_{\max} = \alpha_1/2\sqrt{\alpha_1}$.

Two thousand training data samples and 500 validations data samples $d(t)$ were generated by using (1) and (2) [via (35) and (36)], where $H(z) = 1+0.7692z^{-1}+0.1538z^{-2}+0.0769z^{-3}$, in which the TWT nonlinearity is used to generate the training dataset and specified by $\alpha_1 = 3$, $\alpha_2 = 1.2$, $\beta_1 = \pi/12$, $\beta_2 = 0.25$. $y(t)$ was uniformly distributed complex random variable with $y_R(t) \in [-0.75, 0.75]$ and $y_I(t) \in [-0.75, 0.75]$. The variances of the additive noise to the system output are set as $\sigma^2 = 0$ (noise free) and $\sigma^2 = 0.05^2$ (high noise), respectively.

The polynomial degree of the B-spline basis functions for both the real and imaginary parts was set as two (i.e., $k = 3$, piecewise quadratic). The system identification algorithm outlined in Section III were carried out, with the following predetermined knot sequences. The knot sequence

$$[-2.25, -1.8, -1.5, -1.2, -0.75, 0, 0.75, 1.2, 1.5, 1.8, 2.25]$$

is initially set for both $d_R(t)$ and $d_I(t)$ in order to generate basis functions used in (21). The knot sequence

$$[-7.5, -3, -1.2, -0.75, -0.3, 0, 0.3, 0.75, 1.2, 3, 7.5]$$

is then set for both $w_R(t)$ and $w_I(t)$ in order to generate basis functions used in (17).

The modeling results are shown in Table II for the linear subsystem. It is seen that the proposed system identification method is consistently effective in capturing the true model parameters. In order to demonstrate the nonlinear approximation capability, the model predictions of the B-spline model $\Psi(w(t))$ in the polar coordinate system were reconstructed over the validation dataset, and this is compared with the true model used to generate the data set in Fig. 4(a) and (b), using the noise-free case dataset. It is seen that the proposed approaches have excellent approximation results for modeling the complex-valued nonlinear static function.

TABLE II
RESULTS OF LINEAR SUBSYSTEM PARAMETER ESTIMATION (EXAMPLE 2). (a) NOISE FREE. (b) HIGH NOISE

(a)

	True parameters	Initial estimates	Final estimates
h_1	0.7692	$0.7254 - j0.0012$	$0.7656 + j0.0001$
h_2	0.1538	$0.1405 + j0.0002$	$0.1526 + j7 \times 10^{-6}$
h_3	0.0769	$0.0691 - j0.0046$	$0.0764 + j0.0003$

(b)

	True parameters	Initial estimates	Final estimates
h_1	0.7692	$0.6913 - j0.001$	$0.7655 - j0.0003$
h_2	0.1538	$0.1352 + j0.0004$	$0.1527 - j0.0006$
h_3	0.0769	$0.0629 - j0.0072$	$0.0748 + j0.0011$

V. CONCLUSION

A new complex-valued B-spline neural network model has been proposed for the modeling of general complex-valued Wiener system. The complex-valued nonlinear static function in the Wiener system is modeled on the basis of the tensor product from two univariate B-spline neural networks that are constructed using the real and imaginary parts of the system input. For parameter estimation, the Gauss–Newton algorithm has been applied by incorporating the De Boor algorithm, using both curve and the first-order derivatives recursion. A simple least squares parameter initialization scheme was also included for completeness. The efficacy of the proposed approaches has been demonstrated using modeling examples, which included a nonlinear HPA with high importance in communication systems.

REFERENCES

[1] E.-W. Bai, "An optimal two-stage identification algorithm for Hammerstein–Wiener nonlinear systems," *Automatica*, vol. 34, no. 3, pp. 333–338, Mar. 1998.

[2] K. Hsu, T. Vincent, and K. Poolla, "A kernel based approach to structured nonlienar system identification part I: Algorithms, part II: Convergenge and consistency," in *Proc. 14th IFAC Symp. Syst. Identif.*, 2006, pp. 1–6.

[3] J. Schoukens, J. G. Nemeth, P. Crama, Y. Rolain, and R. Pintelon, "Fast approximate identification of nonlinear systems," *Automatica*, vol. 39, no. 7, pp. 1267–1274, Jul. 2003.

[4] Y. Zhu, "Estimation of an N-L-N Hammerstein–Wiener model," *Automatica*, vol. 38, no. 9, pp. 1607–1614, Sep. 2002.

[5] C. J. Clark, G. Chrisikos, M. S. Muha, A. A. Moulthrop, and C. P. Silva, "Time-domain envelope measurement technique with application to wideband power amplifier modeling," *IEEE Trans. Microw. Theory Tech.*, vol. 46, no. 12, pp. 2531–2540, Dec. 1998.

[6] A. Hagenblad, L. Ljung, and A. Wills, "Maximum likelihood identification of Wiener models," *Automatica*, vol. 44, no. 11, pp. 2697–2705, Nov. 2008.

[7] Y. Zhu, "Distillation column identification for control using Wiener model," in *Proc. Amer. Control Conf.*, vol. 5. San Diego, CA, Jun. 1999, pp. 3462–3466.

[8] A. D. Kalafatis, N. Arifinand, L. Wang, and W. R. Cluett, "A new approach to the identification of pH processes based on the Wiener model," *Chem. Eng. Sci.*, vol. 50, no. 23, pp. 3693–3701, Dec. 1995.

[9] A. D. Kalafatis, L. Wang, and W. R. Cluett, "Identification of Wiener-type nonlinear systems in a noisy environment," *Int. J. Control*, vol. 66, no. 6, pp. 923–941, 1997.

[10] I. W. Hunter and M. J. Korenberg, "The identification of nonlinear biological systems: Wiener and Hammerstein cascade models," *Biol. Cybern.*, vol. 55, nos. 2–3, pp. 135–144, 1986.

[11] J. C. Gomez, A. Jutan, and E. Baeyens, "Wiener model identification and predictive control of a pH neutralisation process," *IEE Proc. Control Theory Appl.*, vol. 151, no. 3, pp. 329–338, May 2004.

[12] W. Greblicki, "Nonparametric identification of Wiener systems," *IEEE Trans. Inf. Theory*, vol. 38, no. 5, pp. 1487–1493, Sep. 1992.

[13] I. Skrjanc, S. Blazic, and O. E. Agamennoni, "Interval fuzzy modeling applied to Wiener models with uncertainties," *IEEE Trans. Syst., Man, Cybern., Part B: Cybern.*, vol. 35, no. 5, pp. 1092–1095, Oct. 2005.

[14] G. Farin, *Curves and Surfaces for Computer-Aided Geometric Design: A Practical Guide*. New York: Academic, 1994.

[15] C. de Boor, *A Practical Guide to Splines*. New York: Springer-Verlag, 1978.

[16] T. Kavli, "ASMO: An algorithm for adaptive spline modelling of observation data," *Int. J. Control*, vol. 58, no. 4, pp. 947–967, Oct. 1993.

[17] M. Brown and C. J. Harris, *Neurofuzzy Adaptive Modelling and Control*. Englewood Cliffs, NJ: Prentice-Hall, 1994.

[18] C. J. Harris, X. Hong, and Q. Gan, *Adaptive Modelling, Estimation and Fusion from Data: A Neurofuzzy Approach*. New York: Springer-Verlag, 2002.

[19] Y. Yang, L. Guo, and H. Wang, "Adaptive statistic tracking control based on two-step neural networks with time delays," *IEEE Trans. Neural Netw.*, vol. 20, no. 3, pp. 420–429, Mar. 2009.

[20] A. Uncini, L. Vecci, P. Campolucci, and F. Piazza, "Complex-valued neural networks with adaptive spline activation function for digital-radio-links nonlinear equalization," *IEEE Trans. Signal Process.*, vol. 47, no. 2, pp. 505–514, Feb. 1999.

[21] T. Kim and T. Adali, "Approximation by fully complex multilayer perceptrons," *Neural Comput.*, vol. 15, no. 7, pp. 1641–1666, 2003.

[22] C.-C. Yang and N. Y. Bose, "Landmine detection and classification with complex-valued hybrid neural network using scattering parameters dataset," *IEEE Trans. Neural Netw.*, vol. 16, no. 3, pp. 743–753, May 2005.

[23] M.-B. Li, G.-B. Guang, P. Saratchandran, and N. Sundararajan, "Fully complex extreme learning machine," *Neurocomputing*, vol. 68, pp. 306–314, Oct. 2005.

[24] A. Hirose, *Complex Valued Neural Networks*. Berlin, Germany: Springer-Verlag, 2006.

[25] S. Chen, S. McLaughlin, and B. Mulgrew, "Complex-valued radial basic function network, part I: Network architecture and learning algorithms," *Signal Process.*, vol. 35, no. 1, pp. 19–31, Jan. 1994.

[26] S. Chen, X. Hong, C. J. Harris, and L. Hanzo, "Fully complex-valued radial basis function networks: Orthogonal least squares regression and classification," *Neurocomputing*, vol. 71, nos. 16–18, pp. 3421–3433, 2008.

[27] M. Kobayashi, "Exceptional reducibility of complex-valued neural networks," *IEEE Trans. Neural Netw.*, vol. 21, no. 7, pp. 1060–1072, Jul. 2010.

[28] B. Igelnik, "Kolmogorov's spline complex network and adaptive dynamic modeling of data," in *Complex-Valued Neural Networks: Unitizing High-Dimensional Parameters*, T. Nitta, Ed. Hershey, PA: IGI Global, 2009, pp. 56–78.

[29] M. Scarpiniti, D. Vigliano, R. Parisi, and A. Unicinis, "Flexible blind signal separation in the complex domain," in *Complex-Valued Neural Networks: Unitizing High-Dimensional Parameters*, T. Nitta, Ed. Hershey, PA: IGI Global, 2009, pp. 284–323.

[30] L. Ding, G. T. Zhou, D. R. Morgan, Z. Ma, J. S. Kenney, J. Kim, and C. R. Giardina, "A robust digital baseband predistorter constructed using memory polynomials," *IEEE Trans. Commun.*, vol. 52, no. 1, pp. 159–165, Jan. 2004.

[31] D. Zhou and V. E. DeBrunner, "Novel adaptive nonlinear predistorters based on the direct learning algorithm," *IEEE Trans. Signal Process.*, vol. 55, no. 1, pp. 120–133, Jan. 2007.