Wiener System identification using B-spline functions with De Boor recursion

X. Hong *, R. J. Mitchell *, S. Chen **

* *School of Systems Engineering, University of Reading, UK*
** *School of Electronics and Computer Science, University of Southampton, UK*

***Abstract*—A simple and effective algorithm is introduced for the system identification of Wiener system based on the observational input/output data. The B-spline neural network is used to approximate the nonlinear static function in the Wiener system. We incorporate the Gauss-Newton algorithm with De Boor algorithm (both curve and the first order derivatives) for the parameter estimation of the Wiener model, together with the use of a parameter initialization scheme. The efficacy of the proposed approach is demonstrated using an illustrative example.**

I. Introduction

The Wiener system has been applied as a model for some industrial/biological systems [1], [2], [3], [4], [5], [6]. It comprises a linear dynamical model followed a nonlinear static functional transformation. Fundamental to the identification and control of the Wiener system is the characterization/representation of the unknown nonlinear static function. Various approaches have been researched including the nonparametric method [7], subspace model identification methods [8], [6], fuzzy modelling [9] and the parametric method [10], [3], [4], [2]. For the parametric method, the unknown nonlinear function is restricted by some parametric representation with a finite number of parameters, and the system identification includes the estimation of the unknown parameters using nonlinear optimization algorithms based on input/output observational data

Based on the approximation theory, the polynomial functions are appropriate in approximating the unknown nonlinear static functions. The spline curves consist of many polynomial pieces offering versatility. The use of piecewise linearity [11], [12] and various spline functions [13], [2] in the modeling of the Wiener system have been researched. With its best conditioning property, the B-spline curve has been widely used in computer graphics and computer aided geometric design (CAGD) [14]. 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 modelling have been widely applied [16], [17], [18]. In this paper we model the nonlinear static function in the Wiener system using a B-spline neural network. We point out that there are clear differences between the proposed approach to other splines functions based methods [13], [2]. 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. The Gauss-Newton algorithm is combined with De Boor algorithm (both curve and the first order derivative) for the parameter estimation of the Wiener model, following a parameter initialization scheme. The proposed model based on B-spline functions with De Boor recursion has several advantages over many existent Wiener system modeling paradigms. Firstly, unlike B-spline functions, the spline functions used in Wiener system modeling [13], [2] do not have the property of partition of unity (convexity), which is a desirable property in achieving numerical stability. Secondly the proposed algorithm based on De Boor recursion 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 paper. Finally, rather than just using the most commonly used cubic splines, the modeler has the freedom/flexibility in terms of coping with with different model setting such as number of knots and polynomial order.

II. The Wiener system and B-spline neural network

A. The Wiener system

The Wiener system consists of a cascade of two subsystems, a linear filter as the first subsystem, followed by a nonlinear memoryless function $\Psi(\bullet) : \mathcal{R} \to \mathcal{R}$ as the second subsystem. The system can be represented by

$$\begin{aligned} v(t) &= \frac{z^{-d}B(z)}{A(z)}u(t) \\ &= u(t-d) + b_1 u(t-d-1)... + b_{n_b}u(t-d-n_b) \\ &\quad -a_1 v(t-1) - a_2 v(t-2)... - a_{n_a}v(t-n_a) \quad (1) \\ y(t) &= \Psi\left(v(t)\right) + \xi(t) \quad (2) \end{aligned}$$

with z transfer functions $A(z)$ and $B(z)$ which are defined by

$$A(z) = \sum_{j=0}^{n_a} a_j z^{-j}, \qquad a_0 = 1 \quad (3)$$

$$B(z) = \sum_{j=0}^{n_b} b_j z^{-j}, \qquad b_0 = 1 \quad (4)$$

$u(t) \in \mathcal{R}$ is the system input and $y(t) \in \mathcal{R}$ is the system input. $\xi(t)$ is assumed to be a white noise sequence independent of $u(t)$, with zero mean and variance σ^2. $v(t) \in \mathcal{R}$ is the output of the linear filter subsystem and the input to the nonlinear subsystem. a_j, b_j are the coefficients of the linear filter. $d \geq 1$ is assumed known positive integer representing the delay of the system. n_a and n_b are assumed known positive integers. Denote $\mathbf{a} = [a_1, ..., a_{n_a}]^T \in \mathcal{R}^{n_a}$ and $\mathbf{b} = [b_1, ..., b_{n_b}]^T \in \mathcal{R}^{n_b}$. The objective of system identification for the above Wiener model is that, given an observational input/output data set $D_N = \{y(t), u(t)\}_{t=1}^N$, to identify $\Psi(\bullet)$ and to estimate the parameters a_j, b_j in the linear subsystems. Note that this model is a special case of the general Wiener systems [1].

Without significantly loss of generality the following assumptions are initially made about the problem.

Assumption 1: $A(z)$ has all zeros inside the unit circle.

Assumption 2: $v(t)$ is bounded by $V_{min} \leq v(t) \leq V_{max}$, where V_{min} and V_{max} are finite real values.

Note that although the signals between the two subsystems are unavailable, Assumption 2 is possible due to the constraint $b_0 = 1$. V_{min} and V_{max} need not to be known precisely and they can be set based on an auxiliary signal $\{\hat{v}(t)\}$ as defined in (18) in the modeling process. Note that if the nonlinear subsystem is modeled using other local basis functions, e.g. piecewise linear models or radial basis functions (RBF), there is a need to impose constraints on the range of $v(t)$, and determine the required parameters for the associated models (knots or centers).

In this work the B-spline basis functions are adopted in order to model $\Psi(\bullet)$. Specifically, the De Boor algorithm [15] is used in the construction of the B-spline basis functions, as described below.

B. Modelling of $\Psi(\bullet)$ using B-spline function approximation with De Boor's algorithm

De Boor's algorithm is a fast and numerically stable algorithm for evaluating B-spline spline curves. Univariate B-spline basis functions are 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 that break it up into a number of intervals. Supposing that there are M basis functions, the knot vector is specified by $(M+k)$ knot values, $\{V_1, V_2, \cdots, V_{M+k}\}$. 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 $(M-k)$. Specifically

$$\begin{aligned} V_1 < V_2 < V_k < V_{min} < V_{k+1} < V_{k+2} < \cdots \\ < V_M < V_{max} < V_{M+1} < \cdots < V_{M+k}. \end{aligned} \tag{5}$$

Given these predetermined knots, a set of M B-spline basis functions can be formed by using De Boor recursion [15],

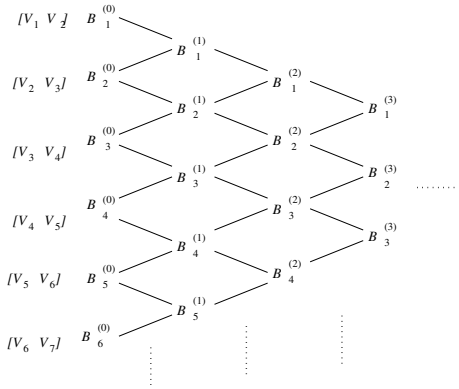


Fig. 1. Visualizing De Boor algorithm.

given by

$$\mathcal{B}_j^{(0)}(v) = \begin{cases} 1 & \text{if } V_j \leq v \leq V_{j+1} \\ 0 & \text{otherwise} \end{cases} \tag{6}$$
$$j = 1, \cdots, (M+k)$$

$$\left.\begin{aligned} \mathcal{B}_j^{(i)}(v) &= \tfrac{v - V_j}{V_{i+j} - V_j} \mathcal{B}_j^{(i-1)}(v) \\ &+ \tfrac{V_{i+j+1} - v}{V_{i+j+1} - V_{j+1}} \mathcal{B}_{j+1}^{(i-1)}(v), \\ j &= 1, \cdots, (M+k-i) \\ i &= 1, \cdots, k \end{aligned}\right\} \tag{7}$$

Notably the first order derivatives of the B-spline function has a similar recursion

$$\begin{aligned} \tfrac{d}{dv}\mathcal{B}_j^{(k)}(v) = \tfrac{k}{V_{k+j} - V_j} \mathcal{B}_j^{(k-1)}(v) \\ \tfrac{k}{V_{i+j+1} - V_{j+1}} \mathcal{B}_{j+1}^{(k-1)}(v), \end{aligned} \qquad j = 1, \cdots, M \tag{8}$$

De Boor recursion can be graphically illustrated with reference to Figure 1. 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 setting such as number of knots, and polynomial order.

We model $\Psi(\bullet)$ in (2) as

$$\hat{\Psi}(v) = \sum_{j=1}^{M} \mathcal{B}_j^{(k)}(v) \omega_j \tag{9}$$

where $\widehat{\ }$ denotes estimate, ω_j's are weights to be determined. We denote $\boldsymbol{\omega} = [\omega_1, \cdots, \omega_M]^T \in \Re^M$. Note that model (9) satisfies the property of partition of unity (convexity), i.e. $(\mathcal{B}_j^{(k)}(v) \geq 0, \sum_{j=1}^{M} \mathcal{B}_j^{(k)}(v) = 1)$, which is a desirable property in achieving numerical stability [19].

Note that due to the piecewise nature of B-spline functions, there are only k basis functions with nonzero values, and nonzero first order derivatives, for any point v. Hence the computational cost for the evaluation of $\Psi(v)$ based on the De Boor algorithm is to do with k, rather than the number of knots, and this is in the order of $O(k^2)$. The evaluation of the first order derivatives can be regarded as a byproduct, with the additional computational cost in the order of $O(k)$.

III. The proposed system identification algorithm

Let the prediction error [20] between the Wiener system output $y(t)$ and $\hat{y}(t)$, the model predicted output for $y(t)$, be denoted by $e(t) = y(t) - \hat{y}(t)$, and $\mathbf{e} = [e(1), e(2), \cdots, e(N)]^T$. With the B-spline approximation, the model predicted output $\hat{y}(t)$ can be written as

$$\hat{y}(t) = \Psi(v(t, \mathbf{a}, \mathbf{b}), \boldsymbol{\omega}) + e(t) = \sum_{j=1}^{M} \mathcal{B}_j^{(k)}(v(t, \mathbf{a}, \mathbf{b}))\omega_j + e(t) \tag{10}$$

The specific system identification task is to jointly estimate $\mathbf{a}$, $\mathbf{b}$ and $\boldsymbol{\omega}$. This could be achieved by minimizing

$$V = \sum_{t=1}^{N} [e(t)]^2 \tag{11}$$

via the Gauss Newton algorithm. The solution obtained via (11) is optimal in the sense that it is the maximum likelihood estimates (MLE) in the case that $\xi(t)$ is Gaussian. As the objective function of (11) is highly nonlinear, the solution of Gauss Newton algorithm is dependent on the initial condition. It is important that $\mathbf{a}$, $\mathbf{b}$ and $\boldsymbol{\omega}$ are properly initialized so that they converge to optimal solution. An initialization scheme is proposed below.

A. Initialization of parameter vectors $\mathbf{a}$ and $\mathbf{b}$

Denote the inverse function of $\Psi(\bullet)$, or $\Psi^{-1}(\bullet)$, as $\varphi(\bullet)$. Consider using also a B-spline neural network for the modeling of $\varphi(\bullet)$, as shown in Figure 2. For convenience, we still denote the polynomial degree as k in the modeling of $\varphi(\bullet)$. The number of basis functions is denoted as d_y. A set of $(d_y + k)$ knots is predetermined (see (5)) based on the domain of the system output $y(t)$, so that there are k external knots outside each side of the boundary of the system output $y(t)$. The model used for modeling $\varphi(\bullet)$ is

$$\hat{\varphi}\left(y(t)\right) = \sum_{j=1}^{d_y} \mathcal{B}_j^{(k)}\left(y(t)\right)\alpha_j \tag{12}$$

where $\alpha_j \in \mathcal{R}$, $(j = 1, \cdots, d_y)$ are the associated weights. Denote the error between $v(t)$ and $\varphi\left(y(t)\right)$ as $\epsilon(t)$ and let $n = d_y \times (n_a + 1) + n_b$. Applying (1), yields

$$\begin{aligned} u(t-d) &= -\sum_{i=1}^{n_b} b_i u(t-d-i) \\ &\quad + \sum_{i=0}^{n_a} a_i [(\sum_{j=1}^{d_y} \mathcal{B}_j^{(k)}\left(y(t-i)\right)\alpha_j] + \epsilon(t) \\ &= [\mathbf{p}(\mathbf{x}(t))]^T \boldsymbol{\vartheta} + \epsilon(t) \end{aligned} \tag{13}$$

where $\mathbf{x}(t) = [-u(t-d-1), \cdots, -u(t-d-n_b), y(t)]^T$, $\boldsymbol{\vartheta} = [\vartheta_1, \cdots, \vartheta_n]^T$=$[-b_1, \cdots, -b_{n_b}, \alpha_1, \cdots, \alpha_{d_y}, \alpha_1 a_1, \cdots, \alpha_j a_i, \cdots, \alpha_{d_y} a_{n_a}]^T \in \Re^n$. $\mathbf{p}(\mathbf{x}(t)) = [p_1(\mathbf{x}(t)), \cdots, p_n(\mathbf{x}(t))]^T$=$[-u(t-d-1), \cdots, -u(t-d-n_b), \mathcal{B}_1^{(k)}\left(y(t)\right), \cdots, \mathcal{B}_{d_y}^{(k)}\left(y(t)\right), \mathcal{B}_1^{(k)}\left(y(t-1)\right) \cdots, \mathcal{B}_j^{(k)}\left(y(t-i)\right), \cdots, \mathcal{B}_{d_y}^{(k)}\left(y(t-n_a)\right)]^T \in \mathcal{R}^n$. Define $\boldsymbol{\alpha} = [\alpha_1, ..., \alpha_{d_y}]^T$.

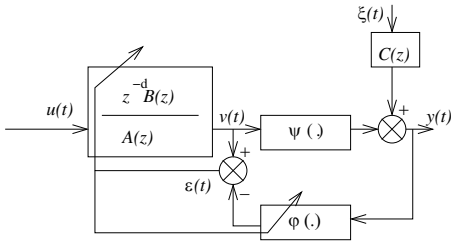


Fig. 2. The initialization for the linear filter parameter vector $\mathbf{a}$ and $\mathbf{b}$'s

Over the training data set, (13) can be written in matrix form as

$$\mathbf{u} = \mathbf{P}\boldsymbol{\vartheta} + \boldsymbol{\epsilon} \tag{14}$$

where $\mathbf{u} = [u(1-d), \cdots, u(N-d)]^T$. $\boldsymbol{\epsilon} = [\epsilon(1), ..., \epsilon(N)]^T$, and $\mathbf{P}$ is the regression matrix $\mathbf{P} = [\mathbf{p}(\mathbf{x}(1)), \cdots, \mathbf{p}(\mathbf{x}(N))]^T$. The parameter vector $\boldsymbol{\vartheta}$ can be found as the least squares solution of

$$\boldsymbol{\vartheta}_{LS} = \left(\mathbf{P}^T\mathbf{P}\right)^{-1}\mathbf{P}^T\mathbf{u} \tag{15}$$

Denote the iteration step variable m of the Gauss-Newton algorithm by a superscript (m). This procedure produces our initial estimate of the vector $\mathbf{b}^{(0)}$, which is simply taken as the subvector of the resultant $\boldsymbol{\vartheta}_{LS}$, consisting of its first n_b elements.

In order to produce our initial estimate of the vector $\mathbf{a}^{(0)}$, the singular value decomposition (SVD) method based on the above $\boldsymbol{\vartheta}_{LS}$ [10] is used. Rearrange the last $(n_a + 1) \times d_y$ elements of $\boldsymbol{\vartheta}_{LS}$ to form the matrix given by

$$\begin{aligned} \boldsymbol{\Theta} &= \boldsymbol{\alpha}[1\ \mathbf{a}^T]^T \\ &= \begin{pmatrix} \widehat{\alpha_1} & \widehat{\alpha_1 a_1} & \ddots & \widehat{\alpha_1 a_{n_a}} \\ \widehat{\alpha_2} & \widehat{\alpha_2 a_1} & \ddots & \widehat{\alpha_2 a_{n_a}} \\ \vdots & \ddots & \vdots & \\ \widehat{\alpha_{d_y}} & \widehat{\alpha_{d_y} a_1} & \ddots & \widehat{\alpha_{d_y} a_{n_a}} \end{pmatrix} \\ &\in \Re^{d_y \times (n_a+1)} \end{aligned} \tag{16}$$

The idea in [10] is to determine $\boldsymbol{\alpha}$ and $\mathbf{a}$ from an overparameterized vector $\boldsymbol{\Theta}$ by minimizing

$$\|\boldsymbol{\Theta} - \boldsymbol{\alpha}[1\ \mathbf{a}^T]^T\|_F^2.$$

Construct the singular value decomposition (SVD) of $\boldsymbol{\Theta} = \sum_{i=1}^{\min(d_y, n_a+1)} \sigma_i \boldsymbol{\mu}_i \boldsymbol{\nu}_i{}^T$, where $\boldsymbol{\mu}_i$ $(i = 1, ...d_y)$ and $\boldsymbol{\nu}_i$ $(i = 1, ...(n_a + 1))$ are orthonormal vectors. Let $\boldsymbol{\nu}_1 = [\nu_{1,1}, \nu_{2,1}, ..., \nu_{n_a+1,1}]^T$. Using the fact that $\boldsymbol{\Theta}$ has rank one, we obtain

$$\mathbf{a}^{(0)} = [\nu_{2,1}, ..., \nu_{n_a+1,1}]^T / \nu_{1,1}. \tag{17}$$

We can also obtain $\boldsymbol{\alpha} = \sigma_1 \nu_{1,1} \boldsymbol{\mu}_1$, but it is no longer used in the remainder of our algorithm. Note that we used the constraint $a_0 = 1$ to get the unique solution that is different from [10] due to the different constraints.

B. The initialization of parameter vector ω

Consider initially generating an auxiliary signal $\{\hat{v}(t)\}_{t=1}^{N}$ over training data set $\{u(t), y(t)\}_{t=1}^{N}$, based on the initialized parameter estimates $\hat{\mathbf{b}}^{(0)}$ and $\hat{\mathbf{a}}^{(0)}$, as

$$\begin{aligned}\hat{v}(t) &= u(t-d)+\hat{b}_1^{(0)}u(t-d-1)+... \\ &\quad +\hat{b}_{n_b}^{(0)}u(t-d-n_b)-a_1^{(0)}\hat{v}(t-1) \\ &\quad -a_2^{(0)}\hat{v}(t-2)...-a_{n_a}^{(0)}\hat{v}(t-n_a) \end{aligned} \tag{18}$$

Then a block of training data set $\{\hat{v}(t), y(t)\}_{t=1}^{N}$ is used for the initialization of parameter vector $\boldsymbol{\omega}$. Using model form (2), in which $v(t)$ is replaced by its estimates $\hat{v}(t)$, and $\xi(t)$ replaced by $e(t)$, we have

$$\begin{aligned} y(t) &= \sum_{j=1}^{M}\mathcal{B}_j^{(k)}(\hat{v}(t))\,\omega_j + e(t) \\ &= [\mathbf{q}(\hat{v}(t))]^T\boldsymbol{\omega}+e(t) \end{aligned} \tag{19}$$

where $\mathbf{q}(\hat{v}(t))=[q_1(\hat{v}(t)),\cdots,q_M(\hat{v}(t))]^T$=$[\mathcal{B}_1^{(k)}(\hat{v}(t)),\cdots,\mathcal{B}_M^{(k)}(\hat{v}(t))]^T \in \Re^M$. Over the training data set, (19) can be written in matrix form

$$\mathbf{y}=\mathbf{Q}\boldsymbol{\omega}+\mathbf{e} \tag{20}$$

where $\mathbf{Q} = [\mathbf{q}(\hat{v}(1)),\cdots,\mathbf{q}(\hat{v}(N))]^T$ and $\mathbf{y} = [y(1),\cdots,y(N)]^T$. The initial estimate of $\boldsymbol{\omega}$ is obtained as the least square solution given by

$$\boldsymbol{\omega}^{(0)}=(\mathbf{Q}^T\mathbf{Q})^{-1}\mathbf{Q}^T\mathbf{y} \tag{21}$$

C. Gauss Newton algorithm combined with De Boor recursion

Note that the initial parameter estimates obtained so far are only near to, but not optimal in minimizing (11). This is because the regressors in (12) are subject to the output noise, which will in general propagate to the parameter estimates, yielding biased parameters [1]. In order to find the optimal value to minimize (11), Gauss Newton algorithm can be applied. Denote $\boldsymbol{\theta}=[\boldsymbol{\omega}^T\ \mathbf{a}^T\ \mathbf{b}^T]^T$. With an initial $\boldsymbol{\theta}^{(0)}$, the Gauss Newton algorithm is given by

$$\boldsymbol{\theta}^{(m)} = \boldsymbol{\theta}^{(m-1)}-\alpha\{[\mathbf{J}^{(m)}]^T\mathbf{J}^{(m)}\}^{-1}[\mathbf{J}^{(m)}]^T\mathbf{e}(\boldsymbol{\theta}^{(m-1)}) \tag{22}$$

where $\mathbf{J}=[\mathbf{J}_{\boldsymbol{\omega}}\ \mathbf{J}_{\mathbf{a}}\ \mathbf{J}_{\mathbf{b}}]$ is the Jacobian of $\mathbf{e}(\boldsymbol{\theta})$, and

$$\mathbf{J}_{\boldsymbol{\omega}}=-\begin{bmatrix}\mathcal{B}_1^{(k)}(v(1)) & \cdots & \mathcal{B}_M^{(k)}(v(1)) \\ \mathcal{B}_1^{(k)}(v(2)) & \cdots & \mathcal{B}_M^{(k)}(v(2)) \\ \vdots & \ddots & \vdots \\ \mathcal{B}_1^{(k)}(v(N)) & \cdots & \mathcal{B}_M^{(k)}(v(N))\end{bmatrix} \tag{23}$$

$$\mathbf{J}_{\mathbf{a}}=\begin{bmatrix}v(0)f(1) & \cdots & v(1-n_a)f(1) \\ v(1)f(2) & \cdots & v(2-n_a)f(2) \\ \vdots & \ddots & \vdots \\ v(N-1)f(N) & \cdots & v(N-n_a)f(N)\end{bmatrix} \tag{24}$$

$$\mathbf{J}_{\mathbf{b}}=-\begin{bmatrix}u(-d)f(1) & \cdots & u(1-d-n_b)f(1) \\ u(1-d)f(2) & \cdots & u(2-d-n_b)f(2) \\ \vdots & \ddots & \vdots \\ u(N-d)f(N) & \cdots & u(N-d-n_b)f(N)\end{bmatrix} \tag{25}$$

where $f(t)=\sum_{j=1}^{M}\frac{d}{dv}[\mathcal{B}_j^{(k)}(v(t)))]\omega_j$, $t=1,...,N$, $\alpha>0$ is a small positive step size. Note that in calculating (23)-(25), De Boor algorithm (6)-(8) is applied in evaluating all entries. In particular we point out the term $\frac{d}{dv}[\mathcal{B}_j^{(k)}(v(t)))]$ using (8) gives exact values at minimum extra computational cost (this is an advantage specific to our B-spline functions with De Boor recursion, but not [13], [2]). Effectively this enables stable and efficient evaluations of B-spline functional and derivative values to be possible, which could be problematic for many other nonlinear representation including some spline functions based nonlinear models. The above iteration can be terminated when $\boldsymbol{\theta}^{(m)}$ converges, or by predetermining a sufficiently large number of iterations.

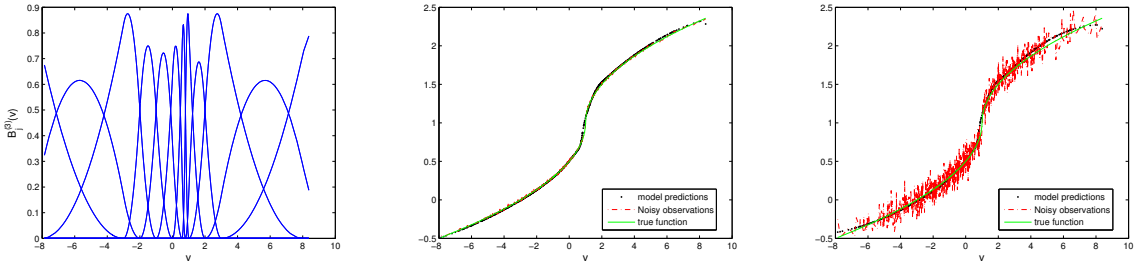
We point out that the optimization of model output with respect to the number/location of knots is an intractable mixed integer problem, for which an iterative trial and error approach can be used to yield a good model (not optimal). For a prior unknown system, the initial knots location should be set 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 are 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 are 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 (though identifying the data points with high modeling errors) during the iterative modeling process.

IV. Numerical examples

A Wiener system is simulated, in which the linear subsystem is set as $A(z)=1-1.2z^{-1}+0.5z^{-2}$, $B(z)=1+0.3z^{-1}+0.8z^{-2}+0.07z^{-3}$ and $d=2$. The nonlinear subsystem is given by

$$\Psi(v) = 0.5\text{sign}(v-1)\sqrt{|v-1|}+1 \tag{26}$$

1000 training data samples $y(t)$ were generated by using (1) and (2), where $u(t)$ was uniformly distributed random variable $u(t)\in[-1.5,1.5]$. The variances of the additive noise to the system output are set as 0.01^2 (low noise) and 0.1^2 (high noise) respectively. The polynomial degree of B-spline basis functions was set as two ($k=3$, piecewise quadratic). The proposed system identification algorithm is carried out with



(a) (b) (c)

Fig. 3. The modelling results of the nonlinear function $\Psi(u)$; (a) the B-spline functions and (b) low noise and (c) high noise.

the following empirically predetermined knot sequences. The knot sequence

$$[-1.5, -1, -0.75, -0.25, 0, 0.5, 0.75, 1, 1.25,$$
$$1.5, 2, 2.25, 2.75, 3, 3.5]$$

is initially set for $y(t)$ in order to generate basis functions used in (12). The knot sequence

$$[-11, -10, -8, -3, -2, -1, 0, 0.6, 0.8, 1, 2, 3, 8, 10, 11]$$

is used for $v(t)$ in order to generate basis functions used in (9). The modelling results are shown in Table I. It is shown that the proposed system identification method is particularly effective at high output noise level. Figure 3 show the excellent approximation results for the nonlinear static functions using the B-spline models.

TABLE I
RESULTS OF LINEAR SUBSYSTEM PARAMETER ESTIMATION; (A) LOW NOISE; AND (B) HIGH NOISE

(a)

	True parameters	Initial estimates	Final estimates
a_1	-1.2	-1.0240	-1.1864
a_2	0.5	0.3710	0.4919
b_1	0.3	0.3099	0.2782
b_2	0.8	0.8048	0.8081
b_3	0.07	0.0769	0.0873

(b)

	True parameters	Initial estimates	Final estimates
a_1	-1.2	-0.5802	-1.1759
a_2	0.5	0.1364	0.4853
b_1	0.3	0.4788	0.2808
b_2	0.8	0.7315	0.7931
b_3	0.07	0.3527	0.1038

V. CONCLUSIONS

This paper investigates a new system identification algorithm for the Wiener system based on B-spline neural network. The parameter estimation is based on the Gauss-Newton algorithm incorporating the De Boor algorithm for both curve and the first order derivatives, following the use of a parameter initialization scheme. The efficacy of the proposed approach has been demonstrated using an illustrative example.

REFERENCES

[1] A. Hagenblad, L. Ljung, and A. Wills, "Maximum likelihood identification of Wiener models," *Automatica*, vol. 44, pp. 2697–2705, 2008.

[2] Y. Zhu, "Distillution column identification for control using Wiener model," in *Proc. the American Control Conference*, San Diego, CA, USA, 1999, pp. 3462–3466.

[3] A. D. Kalafatis, N. Arifinand L. Wang, and W. R. Cluett, "A new approach to the identification of pH processes on the Wiener model," *Chemical Engineering Science*, vol. 50, no. 23, pp. 3693–3701, 1995.

[4] A. D. Kalafatis, L. Wang, and W. R. Cluett, "Identification of Wiener-type nonlinear systems in a noisy environment," *International Journal of Control*, vol. 66, pp. 923–941, 1997.

[5] I. W. Hunter and M. J. Korenberg, "The identification of nonlinear biological systems: Wiener and Hammerstein cascade models," *Biological Cybernetics*, vol. 55, pp. 135–144, 1986.

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

[7] W. Greblicki, "Nonparametric identification of Wiener systems," *IEEE Transactions on Information Theory*, vol. 38, no. 5, pp. 1487–1493, 1992.

[8] D. Westwick, "Identifying MIMO Wiener systems using subspace model identification model identification methods," *Signal Processing*, vol. 52, pp. 235–258, 1996.

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

[10] E. W. Bai, "An optimal two-stage identification algorithm for Hammerstein-wiener nonlinear systems," *Automatica*, vol. 34, pp. 333–338, 1998.

[11] T. Wigren, "Recursive prediction error identification using the nonlinear Wiener model," *Automatica*, vol. 29, no. 4, pp. 1011–1025, 1993.

[12] T. Wigren, "Convergence analysis of recursive identification algorithms based on the nonlinear Wiener model," *IEEE Transactions on Automatic Control*, vol. 39, no. 11, pp. 2191–2206, 1994.

[13] M. C. Hughes and D. T. Westwick, "Identification of IIR Wiener system with spline nonliearitis that have variable knots," *IEEE Transactions on Automatic Control*, vol. 50, no. 10, pp. 1617–1622, 2005.

[14] G. Farin, *Curves and Surfaces for Comnputer-aided Geometric Design: a Practical Guide*, Academic Press, Boston, 1994.

[15] De Boor, *A Practical Guide to Splines*, New York: Spring Verlag, 1978.

[16] T. Kavli, "ASMOD - an algorithm for adaptive spline modelling of observation data," *International Journal of Control*, vol. 58, no. 4, pp. 947–967, 1993.

[17] M. Brown and C. J. Harris, *Neurofuzzy Adaptive Modelling and Control*, Prentice Hall, Hemel Hempstead, 1994.

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

[19] R. T. Farouki and T. N. T. Goodman, "On the optimal stability of the Bernstein basis," *Mathematics of Computation*, vol. 65, no. 216, pp. 1553–1566, 1996.

[20] T. Soderström and P. Stoica, *System Identification*, Prentice Hall, 1989.