Generalised Neurofuzzy Network Modelling Algorithms using Bézier Bernstein Polynomial Functions and Additive Decomposition

X. Hong and C. J. Harris

Image, Speech and Intelligent Systems Group
Department of Electronics and Computer Science
University of Southampton, Southampton SO17 1BJ, UK
Email: xh, cjh@ecs.soton.ac.uk; WWW: http://www.isis.ecs.soton.ac.uk;
Tel: +44 (0)1703 596653; Fax: +44 (0)1703 594498

Abstract — This paper introduces a new neurofuzzy model construction algorithm for nonlinear dynamic systems based upon basis functions that are Bézier Bernstein polynomial functions. This paper is generalised in that it copes with n-dimensional inputs by utilising an additive decomposition construction to overcome the curse of dimensionality associated with high n. This new construction algorithm also introduces univariate Bézier Bernstein polynomial functions for the completeness of the generalised procedure. Like the B-spline expansion based neurofuzzy systems, Bézier Bernstein polynomial function based neurofuzzy networks hold desirable properties such as nonnegativity of the basis functions, unity of support, and interpretability of basis function as fuzzy membership functions, moreover with the additional advantages of structural parsimony and Delaunay input space partition, essentially overcoming the curse of dimensionality associated with conventional fuzzy and RBF networks. This new modelling network is based on additive decomposition approach together with two separate basis function formation approaches for both univariate and bivariate Bézier Bernstein polynomial functions used in model construction. The overall network weights are then learnt using conventional least squares methods. Numerical examples are included to demonstrate the effectiveness of this new data based modelling approach.

Keywords — Bézier surface, Bernstein polynomial function, neurofuzzy network, backpropagation, barycentric coordinates.

1 Introduction

Neurofuzzy network systems have been widely studied in recent years [1, 2, 3]. The advantages of realising complex non-linear mappings using neural networks are well known [4]. Various neural networks like radial basis function (RBF) networks, can be expressed as a linear-in-the-parameters model structure where the system output is a linear combination of non-linear basis functions. Provided that there is a separate mechanism for determining the order and positioning of these basis functions, the basis weights can be trained using linear optimisation techniques. The architecture or topology of the class of linear-in-the-parameters modelling networks enables them to be readily assessed in terms of their modelling capability, structure, learning, construction and numeric stability, since the results of linear optimisation and algebra are directly applicable [5]. Retaining this advantage, neurofuzzy modelling systems combine fuzzy logic with quantitative artificial neural networks via a concept of fuzzification by using a fuzzy membership function usually based on B-spline and algebraic operators [2]. The combination of qualitative based reasoning via fuzzy logic and quantitative adaptive numeric/data processing via ANNs, is a potentially powerful concept, since it allows within a single framework intelligent qualitative and quantitative reasoning (IQ^2) to be achieved.

Fundamental to neurofuzzy network modelling is the idea of building a bridge between fuzzy logic and quantitative adaptive numeric/data processing via a concept of fuzzification by using a fuzzy membership function. Neurofuzzy network modelling using a B-spline as fuzzy membership basis function has been widely studied in recent years [1], with applications ranging from state estimation [6, 7] to multi-sensor data fusion for helicopter guidance [8]. Neural networks provide a highly sophisticated capability for non-linear function approximation and the B-spline neurofuzzy network has desirable properties such as a compact support, a partition of unity, locality, logicality and transparency. Traditionally B-spline neurofuzzy network modelling involves building multivariate B-spline basis function from a tensor product of univariate basis functions. As B-spline neurofuzzy network, like the majority of associative memory networks or other rule based systems, suffer from the curse of dimensionality in which the computational cost and accuracy increases exponentially with the networks input dimension, an adaptive construction algorithm based on additive decomposition techniques can overcome this problem [9]. Other efforts have been proposed to partly overcome the curse of dimensionality, such as piecewise linear models using a Delaunay partition of the input space [10, 11]. Delaunay partition of the input space has been shown to be more appropriate than the conventional fuzzy lattice partition [10] in obtaining a parsimonious and smooth piecewise linear model. As a neurofuzzy network can also be interpreted as a locally piecewise polynomial model which naturally has a stronger approximation capability than locally piecewise linear model, it is highly desirable that neurofuzzy networks based on Delaunay partition of the input space can be developed, so that all the advantages associated with the neurofuzzy network can be extended to the case of Delaunay input space partition, whilst in turn minimises the number of model free parameters. Recently a new neurofuzzy network construction algorithm using the Bézier Bernstein polynomial function has been introduced for the 2-D input space modelling problem [12]. A neurofuzzy network based on the expansion of Bézier Bernstein polynomial function will maintain most properties held by B-spline expansion, such as nonnegativity of the basis functions, the unity of support, approximation capability, with additional advantages of structural parsimony and Delaunay input space partition. Extending previous work [12], this paper derives a generalised neurofuzzy network modelling algorithms using B$\acute{e}$zier Bernstein polynomial functions based on an additive decomposition approach via the well known analysis of variance (ANOVA) expansion [9].

$$f(\mathbf{x}) = f_0 + \sum_{k=1}^{n} f_k(x_k) + \sum_{k_1=1}^{n} \sum_{k_2=k_1+1}^{n} f_{k_1k_2}(x_{k_1}, x_{k_2}) + e(\mathbf{x}) \tag{1}$$

where $f(\mathbf{x})$ is a general multivariate nonlinear mapping for input $\mathbf{x} = [x_1, x_2, \cdots, x_n]^T \in \Re^n$. $f_k(x_{k_1})$ and $f_{k_1k_2}(x_{k_1}, x_{k_2})$ are univariate and bivariate nonlinear submodels of $f(\bullet)$. For almost all practical problems the ANOVA expansion Eq.(1) can be truncated at the bivariate submodel level with minimal truncation error $e(\mathbf{x})$.

In the proposed construction algorithm based on Eq.(1), the formation of the B$\acute{e}$zier Bernstein polynomial function based network basis, is formulated via two different approaches for each of the general univariate terms $f_k(x_k)$ and bivariate terms $f_{k_1k_2}(x_{k_1}, x_{k_2})$. The network construction is based on the concept of barycentric coordinates and the de Casteljau algorithm which have been widely used in B$\acute{e}$zier Bernstein curve and surface design in the field of computer aided geometric design [13]. Since the neural network are constructed from only input/output data, the principle in the proposed B$\acute{e}$zier Bernstein neurofuzzy network construction algorithms for each of the univariate terms $f_k(x_k)$ and bivariate terms $f_{k_1k_2}(x_{k_1}, x_{k_2})$ is a new concept, which can be interpreted as inverse procedures of the de Casteljau algorithms used in B$\acute{e}$zier Bernstein curve and surface design. The previous approach [12] is only applicable to modelling of the bivariate terms $f_{k_1k_2}(x_{k_1}, x_{k_2})$, where the B$\acute{e}$zier Bernstein polynomial function based network basis, are formed by using an input vector's barycentric coordinates, which are in turn obtained by using an inverse de Casteljau procedure using backpropagation. The B$\acute{e}$zier Bernstein neurofuzzy network construction algorithm for the bivariate terms $f_{k_1k_2}(x_{k_1}, x_{k_2})$ can be interpreted as an inverse procedure of the de Casteljau algorithm used in B$\acute{e}$zier Bernstein $surface$ design. Whereas here for the modeling of the univarite term $f_k(x_k)$, the B$\acute{e}$zier Bernstein polynomial function based network basis, are generated using a new inverse procedure of the de Casteljau algorithm which is used in B$\acute{e}$zier Bernstein *curve* design. The network weights are then trained by conventional least squares methods used in other linear in the weights artificial neural networks [5, 14]. The structure of B$\acute{e}$zier Bernstein polynomial function based network via the additive decomposition is shown in Fig. 1 (see the context for detailed illustration).

This paper is organised as follows. In Section 2, a brief description of neurofuzzy modelling methodology is given to provide the motivation of the proposed new modelling algorithm. Section 3 introduces the new neurofuzzy modelling approach using B$\acute{e}$zier Bernstein polynomial function for univariate terms $f_k(x_k)$. Section 4 develops the neurofuzzy modelling using B$\acute{e}$zier Bernstein polynomial function for bivariate terms $f_{k_1k_2}(x_{k_1}, x_{k_2})$. Section 5 provides a summary of the complete construction algorithm. Section 6 provides some typical multivariate data based modelling

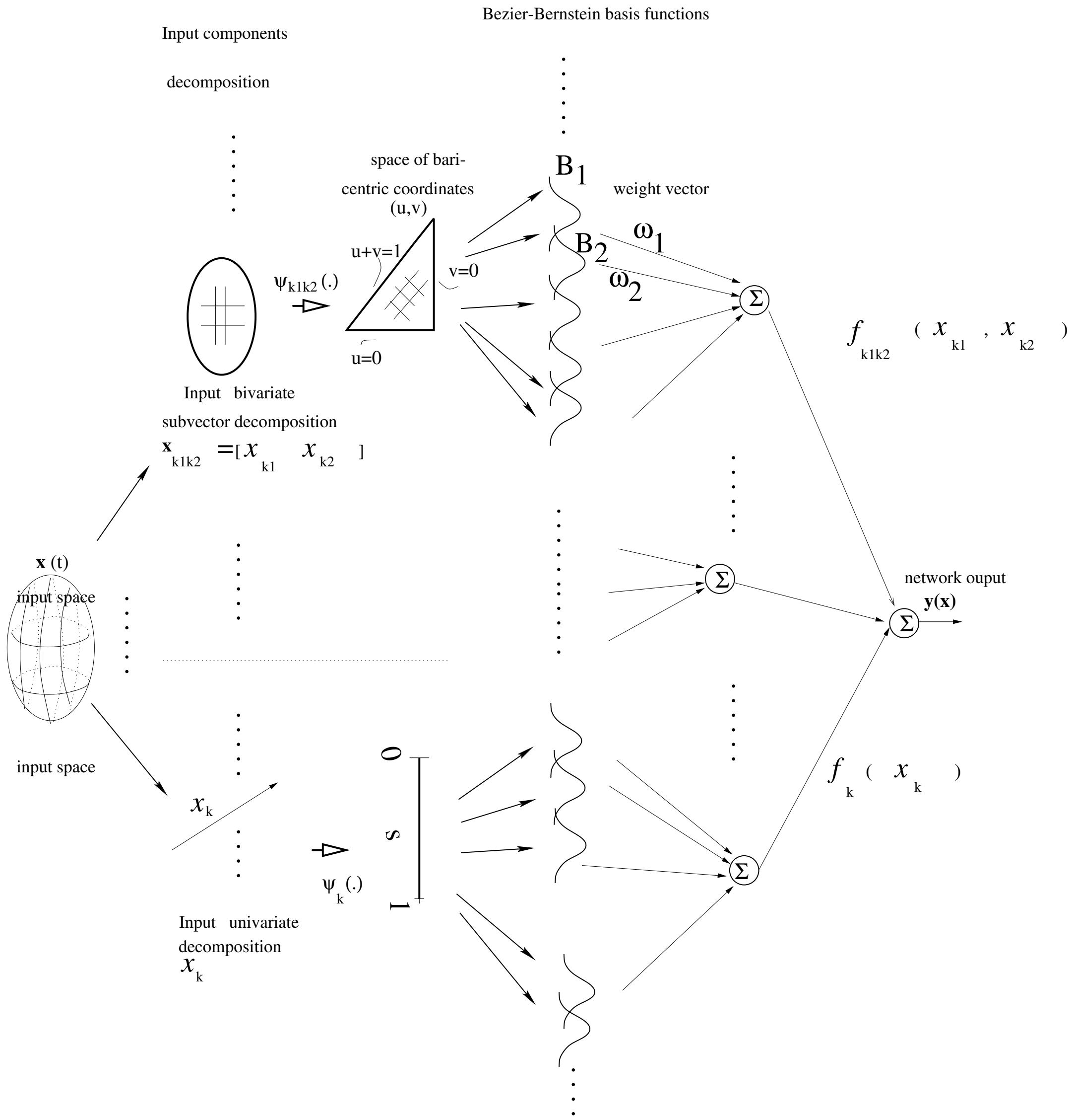


Figure 1: Bézier Bernstein polynomial function based neurofuzzy network via the additive approach

problems to demonstrate the effectiveness of the new algorithm, followed by some conclusions in Section 7.

2 Problem formulation

Fuzzy logic generalises the concept of a characteristic function in Boolean logic, whereby a fuzzy membership function $\mu_A(x) \in [0,1]$, in which A is the fuzzy label describing the variable x, is used to represent vague statements such as *x is negative small.* A standard fuzzy logic system utilises the logical functions of AND, OR, IF(.) and THEN(.). If the *algebraic* operators of *product* and *sum*, rather than the more usual truncation operators *min* and *max*, then the resultant system (i) produces smoother interpolation (ii) provides an equivalence between ANN's and fuzzy logic if $\mu_A(x)$ are radial basis functions, or B-splines and (iii) enables fuzzy system to be readily analysed [1, 2]. The following result [1] is significant in this context,

Theorem 1 *When algebraic operators are used to implement the fuzzy logical functions, the real valued inputs are represented via fuzzy membership functions, and a centre of defuzzification method is used and the rule confidences are normalised, then the neurofuzzy output is given by:*

$$\hat{y}(t) = f(\mathbf{x}) = \sum_{i=1}^{p}\left(\frac{\mu_{A^i}(\mathbf{x})}{\sum_{j=1}^{p}\mu_{A^j}(\mathbf{x})}\right)w_i = \sum_{i=1}^{p}\mu_{A^i}(\mathbf{x})w_i, \mathbf{x} \in R^n \tag{2}$$

where $\mu_{A^i}(\mathbf{x})$ is the ith $(i = 1, 2, \ldots, p)$ fuzzy membership function of a multivariate input $\mathbf{x}$, and w_i is the ith weight.

The network weights $w_i = \sum_{j=1}^{p} c_{ij} y_j^c$, where y_j^c's are centres of the jth output fuzzy membership function $\mu_{B^j}(y)$. c_{ij} is the confidence in the rule: if $\mathbf{x} \in A^i$ then $y \in B^j$, (with confidence c_{ij} where $\sum_{j=1}^{p} c_{ij} = 1$, $c_{ij} \in [0,1]$), where A^i and B^j are fuzzy labels defined on the input and output variables, respectively. If $\sum_{j=1}^{p}\mu_{A^j}(\mathbf{x}) = 1$ without normalisation, the basis function can be especially simple, this desirable property is called a unity of support.

Theorem 1 shows that a neurofuzzy system as a nonlinear system can be described as a linear-in-the-parameters structure enabling fast, simple training methods to be implemented. A neurofuzzy network must be able to learn inherent relationships contained in input/output data sets $\{\mathbf{x}(t), y(t)\}_{t=1}^{N}$ for some desired output $y(t)$. Suppose a basis function $B(\bullet)$ with a unity of support property is used to represent fuzzy membership function $\mu_{A^i}(\mathbf{x})$, Eq.(1) can be expressed as

$$\hat{y}(t) = f(\mathbf{x}) = \mathbf{B}^T(\mathbf{x}(t))\Omega \tag{3}$$

where $\hat{}$ denotes estimates. $\Omega = [\omega_1, \cdots, \omega_p]^T \in \Re^p$ is a weight vector, $\mathbf{B}^T(k)$ is the p-dimensional input basis(fuzzy) functions: $\mathbf{x} \in \Re^n \to \mathbf{B}(\mathbf{x}) \in \Re^p$. Typical linear learning methods such as least squares [14] can be used, via the minimisation of cost $\sum_{t=1}^{N}(y(t) - \hat{y}(t))^2$, to obtain the weights Ω, following the determination of the membership function basis $\mathbf{B}(\mathbf{x})$ by some preprocessing or network construction procedure.

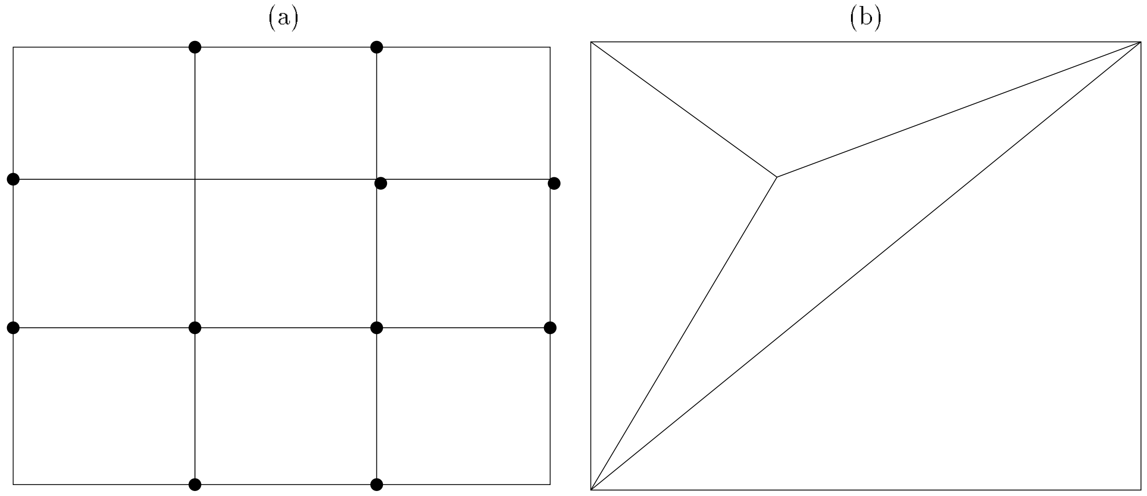


Figure 2: An illustrative example of hidden nodes required in two modelling approaches; (a). lattice partition, where • denotes nodes redundancy compared with Delaunay partitioning approach of figure (b);

The choice of membership function to represent system inputs is important for the modelling capability of a neurofuzzy model. Desired properties include compactness, and a partition of unity. Neurofuzzy modelling using B-spline basis function as membership function has been well studied [1], where traditionally B-spline neurofuzzy network modelling involves building multivariate B-spline basis function from the tensor product of univariate basis functions. As a B-spline neurofuzzy network, like the majority of associative memory networks or other rule based systems, suffer from the curse of dimensionality, an adaptive construction algorithm based on an additive decomposition techniques via Eq.(1) can overcome the problem [9, 15]. Other efforts have been proposed to partly overcome the curse of dimensionality, such as piecewise linear models using a Delaunay partition of the input space and local linear modelling tree (LOLIMOT) construction algorithm [10, 16, 17]. Delaunay partition of the input space has been shown to be more appropriate than the conventional lattice partition [10] in obtaining a parsimonious and smooth piecewise linear model, since it avoids the use of redundant nodes caused by the orthogonal splits inherent in a lattice based input space structure, see Fig.2(a). Also piecewise local linear models based on Delaunay partition can overcome the discontinuity in conventional lattice partitioned based locally piecewise linear models, this is simply illustrated in Fig.3, where $z = f(x, y)$ is supposed to be modelled by two versions of local piecewise linear model. Because a neurofuzzy network can also be interpreted as a locally piecewise polynomial model which inherently has stronger approximation capability than local piecewise linear models, it is highly desirable that neurofuzzy networks based on Delaunay partition of the input space can be developed, so that that all the advantages associated with the neurofuzzy network can be extended to the case of Delaunay input space partitioning. In this

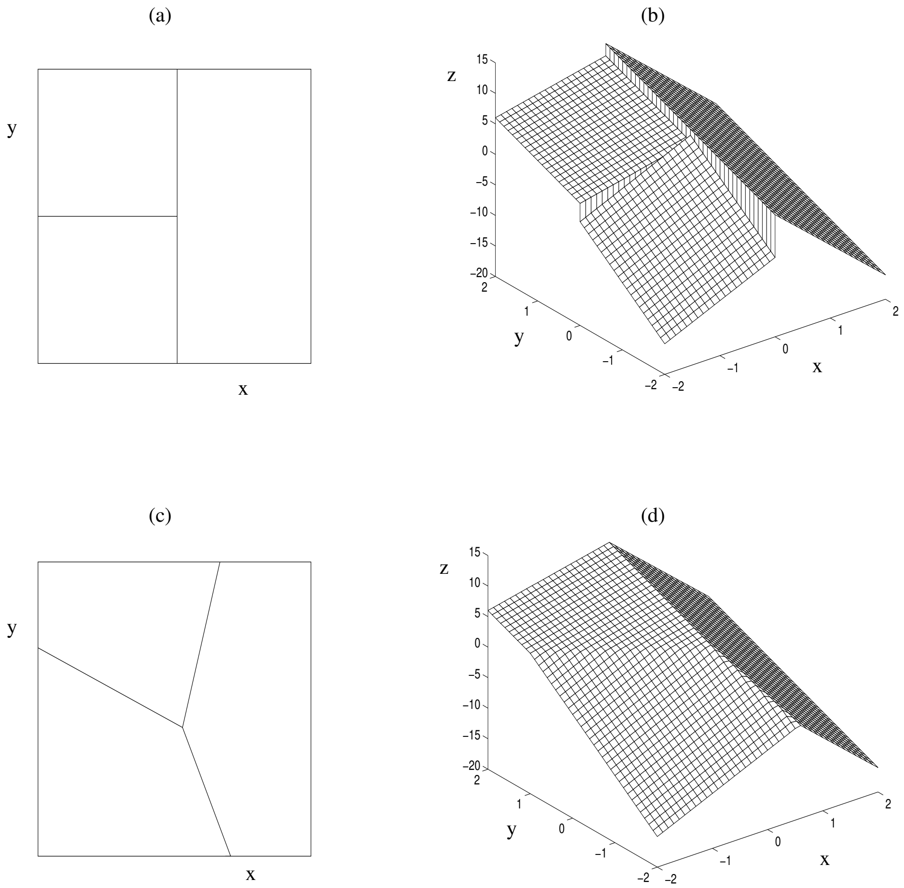


Figure 3: Input space partitions and the corresponding piecewise linear surfaces in modelling $z = f(x, y)$; (a). Orthogonal/lattice decomposition; (b). resultant locally piecewise linear models of (a); (c). Delaunay partitioning; (d). resultant locally piecewise linear models of (c).

work, the Bézier Bernstein polynomial function is introduced as an alternative choice of fuzzy function in the expansion of a general nonlinear function. The neurofuzzy network based on the expansion of the Bézier Bernstein polynomial function will maintain most properties held by B-spline expansion, such as nonnegativity of the basis functions, unity of support with an additional advantage of structural parsimony and a Delaunay input space partition. In practice, many high input dimensional modelling problem can be decomposed into a sum of univariate and bivariate submodels (if submodels with intersection degree higher than 2 are truncated), via the analysis of variance (or Gabor-Kolmogrov) expansion as [9] as given in Eq.(1). Based on the additive decomposition approach, this work presents a new generalised neurofuzzy system model construction algorithm for multi-dimentional systems that can be decomposed into a series of univariate and bivariate functions. In practice this is applicable to almost all real data process. In the proposed construction algorithm based on Eq.(1), the formation of the Bézier Bernstein polynomial function based network basis, is formulated via two different approaches for each of the univariate terms $f_k(x_k)$ and bivariate terms $f_{k_1k_2}(x_{k_1}, x_{k_2})$. In previous work [12], a new neurofuzzy network construction algorithm using the Bézier Bernstein polynomial function was introduced for the two dimensional input space problem, which is applicable to the modelling of the bivariate terms $f_{k_1k_2}(x_{k_1}, x_{k_2})$, as developed in Section 4. To complete the theory of the new Bézier Bernstein polynomial function neurofuzzy network construction algorithm for multi-dimensional systems that can be decomposed into a series of univariate and bivariate functions based on the additive decomposition approach, a new construction algorithm algorithm for the modelling of the univariate term $f_k(x_k)$ is required, which is introduced in the following section.

3 New Neurofuzzy modelling using Bézier Bernstein polynomial function for univariate term $f_k(x_k)$

The neurofuzzy modelling using Bézier Bernstein polynomial function for univariate term $f_k(x_k)$ is based on the de Casteljau algorithm which have been widely used in Bézier Bernstein curve design in the field of computer aided geometric design [13]. Since the neural networks are constructed from only input/output data, it will be initially shown that an inverse procedure of the de Casteljau algorithm that is used in Bézier Bernstein curve design is needed for approximation problem, followed by the introduction of a new inverse procedure of the de Casteljau algorithm in constructing the univariate term $f_k(x_k)$.

In the univariate case, univariate Bernstein polynomials are used as basis functions. Univariate Bernstein polynomial basis $B_j^{(d)}(s)$ are the expansion of $[s + (1 - s)]^d$, defined by

$$B_j^{(d)}(s) = \frac{d!}{j!(d-j)!} s^j (1-s)^{(d-j)} \tag{4}$$

where j and d are nonnegative integer number, $j \leq d$ over the region $s \in [0, 1]$. The total number of dth order univariate Bernstein polynomials $B_j^{(d)}(s)$ is $(d + 1)$. A set of different order

Bernstein polynomial basis functions are shown in Fig.4. In a Bézier Berstein polynomial function based neurofuzzy network, these polynomials are used as fuzzy membership functions as they satisfy $B_j^{(d)}(s) \in [0, 1]$ (representing fuzzy membership function) and $\sum_{j=0}^{d} B_j^{(d)}(s) = 1$ (a unity of support). It can be shown [13] that Bernstein polynomials also satisfy the following recursion:

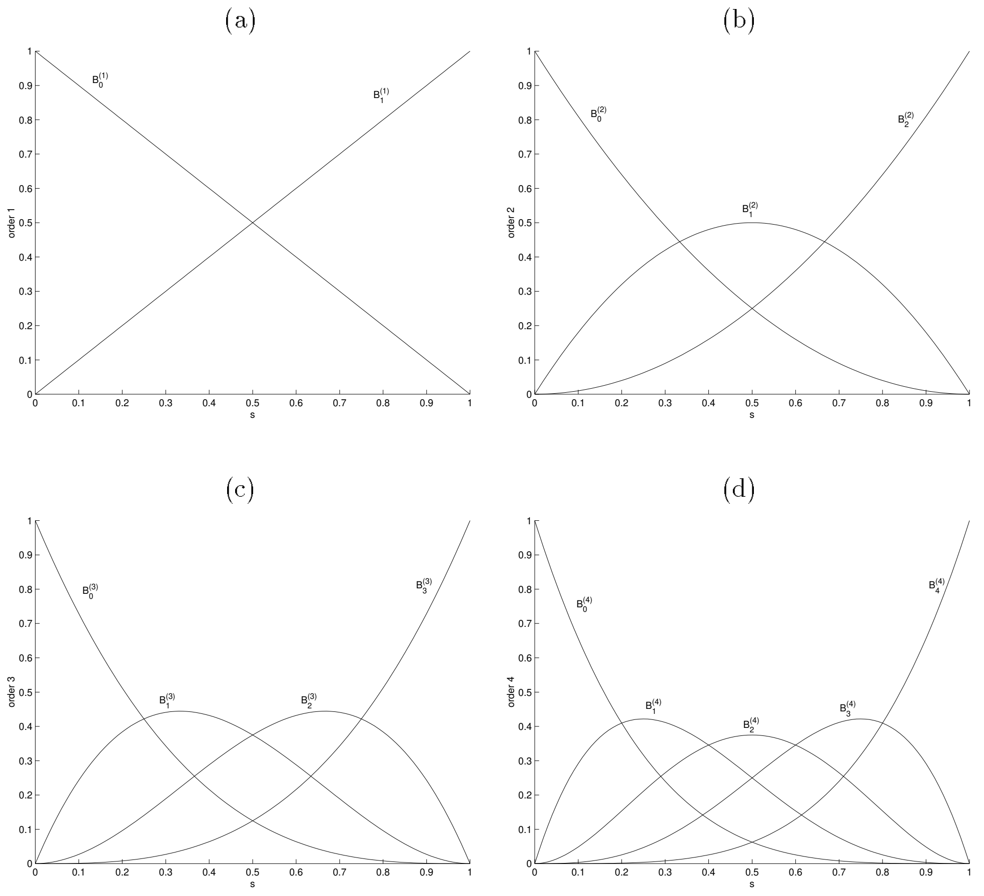


Figure 4: A set of univariate Bernstein polynomials of different orders defined over s =[0, 1]; (a) order 1, linear; (b) order 2, quadratic; (c) order 3, cubic; and (d) order 4, quartic

$$B_j^{(d)}(s) = (1 - s)B_j^{(d-1)}(s) + sB_{j-1}^{(d-1)}(s) \tag{5}$$

Suppose that the $f_k(x_k)$ term (Eq.(1)) is modelled as

$$f_k(x_k) = \sum_{j=0}^{d} B_j^{(d)}(s(x_k))\omega_j \tag{6}$$

where $x_k \in \Re^1$ is the kth component of the input vector $\mathbf{x}(t)$, $B_j^{(d)}(s(x_k))$'s are dth order Bernstein polynomials and ω_{kj}'s are weights determined from input-output data. In order to derive these basis functions, it is necessary to find a one-to-one mapping: $\Psi_k(\bullet) : x_k \in [\min(x_k), \max(x_k)] \rightarrow s(x_k) \in [0, 1]$. It is well known [13] that the univariate de Casteljau algorithm which has been used in Bézier curve construction, can realise a mapping $s \in [0, 1] \rightarrow x_k(s) \in [\min(x_k), \max(x_k)]$

with respect to a set of predetermined knots. The de Casteljau algorithm, as illustrated in Fig.5, is described next.

Suppose that there are a set of plane knots $\mathbf{a}_j = [b_j, a_j]^T \in \Re^2$, $j = 1, \cdots, d$, and $s \in [0,1]$, the de Casteljau algorithm is a recursion defined on these knots by

$$\mathbf{a}_j^{(r)}(s) = (1-s)\mathbf{a}_j^{(r-1)}(s) + s\mathbf{a}_{j+1}^{(r-1)}(s) \tag{7}$$

where $\mathbf{a}_j^{(0)}(s) = \mathbf{a}_j$, $r = 1, 2, \cdots, d$ and $j = 0, \cdots, d-r$. Denote $\mathbf{a}_j^{(r)}(s) = [b_j^{(r)}(s), a_j^{(r)}(s)]^T$, Eq.(7) can be written in a decomposed form and only the first component is used here for derivation as

$$b_j^{(r)}(s) = (1-s)b_j^{(r-1)}(s) + sb_{j+1}^{(r-1)}(s) \tag{8}$$

for $b_j^{(0)}(s) = b_j(s)$, $r = 1, 2, \cdots, d$ and $j = 0, \cdots, d-r$. $b_0^{(d)}(s)$ is the first component of the point $\mathbf{a}_j^{(0)}(s)$ on the Bézier curve with parameter s. It can be shown [13] that by using both recursions Eq.(5) and (8), the $b_0^{(d)}$ (denoted as $\hat{x}_k(s)$) can be written as

$$\hat{x}_k(s) = b_0^{(d)}(s) = \sum_{j=0}^{d} b_j(s) B_j^{(d)}(s) \tag{9}$$

It is clear that the de Casteljau algorithm as a decomposed form in Eq.(8), is a recursive algorithm which realises the mapping from $s(x_k) \in [0,1]$ to $\hat{x}_k(s) \in [\min(x_k) \max(x_k)] \in [b_0, b_d]$, with known supporting knot components b_j. To model $f_k(x_k)$ using a neurofuzzy network construction algorithm based on Bézier-Bernstein functions, the objective is to learn the system characteristics only from input/output data such that an inverse procedure of de Casteljau algorithm is utilised. In the following, the proposed inverse de Casteljau algorithm for univariate term $f_k(x_k)$ attempts to construct the mapping $\Psi_k : x_k \in [b_0, b_d] \to s \in [0,1]$ using backpropagation. Note that the present stage only involves the formation of basis functions, but not the precise identification of $f_k(x_k)$, and also only the first component b_j of knots $\mathbf{a}_j = [b_j, a_j]^T$ need to be predetermined, but not the second component a_j. Given an input vector component $x_k(t)$, the method aims to obtain a desired s by the minimisation the loss function $||x_k(t) - \hat{x}_k(s)||$. By taking account of the decomposed form of de Casteljau algorithm given in Eq.(8), the well-known backpropagation algorithm can be applied iteratively to obtain the new inverse procedure of de Casteljau algorithm for the univariate case, as below.

Denote the iteration step in the minimisation procedure as superscript (m).

1. Predetermine a set of knot components b_j, $j = 0, \cdots, d$, satisfying $[\min(x_k) \max(x_k)] \in [b_0, b_d]$. Initial set $m = 1$, s as random number $s^{(m)}$, satisfying the constraints $0 < s^{(m)} < 1$.

2. Calculate the corresponding first components of Bézier curve points using the de Casteljau recursive formula Eq.(8) until $r = d$.

$$\hat{x}_k(s^{(m)}) = (1-s^{(m)})b_j^{(d-1)}(s^{(m)}) + s^{(m)}b_{j+1}^{(d-1)}(s^{(m)}) \tag{10}$$

3. A new point is generated by using the backpropagation rule as

$$\tilde{x}_k(s^{(m)}) = \hat{x}_k(s^{(m)}) + \alpha[x_k(t) - \hat{x}_k(s^{(m)})] \tag{11}$$

where α is a very small positive number representing the learning rate $0 < \alpha \ll 1$. Note that α should be sufficiently small such that $\tilde{x}(s^{(m)})$ is bounded in the range: $\tilde{x}_k(s^{(m)}) \in [b_0^{(d-1)}(s^{(m)}), b_1^{(d-1)}(s^{(m)})]$.

4. The desired solution of s at iteration step $(m+1)$ is determined as the solution of s such that $\tilde{x}_k(s^{(m)})$ is the first order (linear) Bézier point with respect to two end knots as $b_0^{(d-1)}(s^{(m)})$ and $b_1^{(d-1)}(s^{(m)})$, and determined by

$$\tilde{x}_k(s^{(m)}) = (1-s)b_0^{(d-1)}(s^{(m)}) + sb_1^{(d-1)}(s^{(m)}) \tag{12}$$

The solution is thus given by

$$s^{(m+1)} = \frac{\tilde{x}_k(s^{(m)}) - b_0^{(d-1)}(s^{(m)})}{b_1^{(d-1)}(s^{(m)}) - b_0^{(d-1)}(s^{(m)})} \tag{13}$$

5. The procedure continues while the cost function $\|x_k(t) - \hat{x}_k(s^{(m)})\|$ is monitored to see if $\|x_k(t) - \hat{x}_k(s^{(m)})\| \le \varepsilon$, where ε is a positive number close to zero. If $\|x_k(t) - \hat{x}_k(s^{(m)})\| \le \varepsilon$, then set $s = s^{(m)}$ and the iteration loop stops. Otherwise set $m = m+1$, goto step 2.

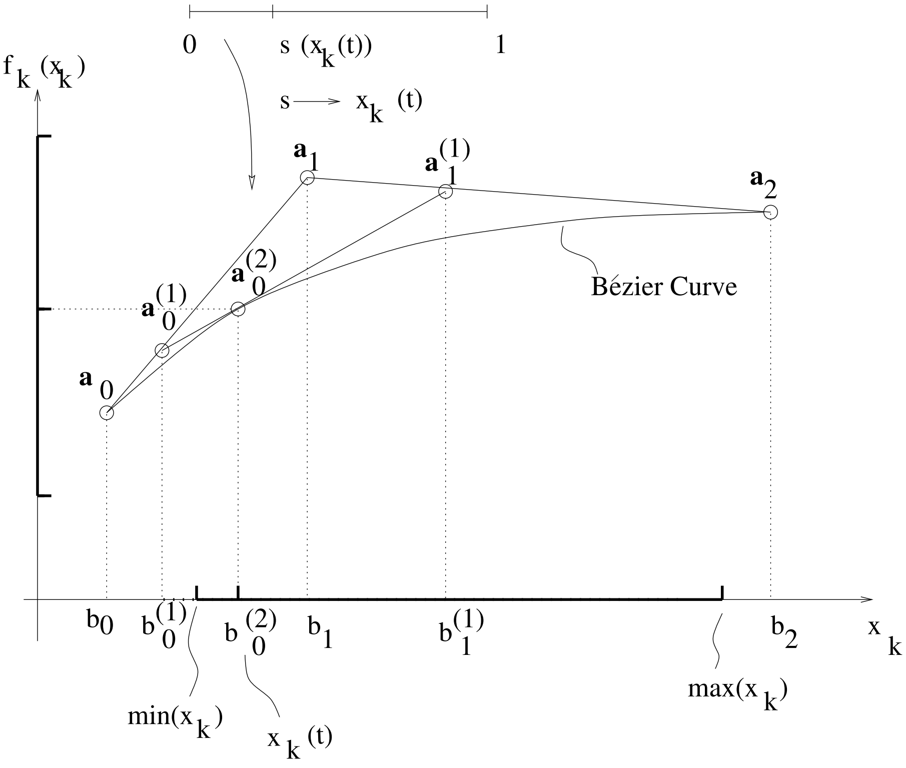


Figure 5: Bézier curve generation using de Casteljau algorithm

4 Neurofuzzy modelling using Bézier Bernstein polynomial function for bivariate term $f_{k_1k_2}(x_{k_1}, x_{k_2})$

The proposed neurofuzzy network construction algorithm is an extension of previous work [12], where a neurofuzzy network construction algorithm for modelling unknown two dimensional input system was developed, which is used here for the modelling of the bivariate term $f_{k_1k_2}(x_{k_1}, x_{k_2})$ used in the general ANOVA expansion of Eq.(1). The algorithm is based on a concept of barycentric coordinates and the bivariate de Casteljau algorithm which has been widely used in computer aided geometric design [13]. As a neurofuzzy network has to be constructed only from input/output data, the principle behind the construction algorithms for each of the univariate term $f_k(\bullet)$ and the bivariate term $f_{k_1k_2}(\bullet)$ based on the Bézier-Bernstein function is a new concept, which can be regarded as inverse procedures of the de Casteljau algorithms that are used in Bézier curve and surface geometric design, respectively. In the proposed construction algorithm for bivariate term $f_{k_1k_2}(x_{k_1}, x_{k_2})$, the Bézier Bernstein polynomial function based network basis, are initially formed by using the barycentric coordinates of the input vector, which are in turn obtained by using a new inverse de Casteljau procedure using backpropagation (see Section 4.3).

4.1 Basic idea

Bivariate Bernstein polynomial basis functions $B^{(d)}_{i,j,k}(u, v, w)$, (with $u+v+w = 1$) are the expansion of $(u + v + w)^d$, defined by

$$B^{(d)}_{i,j,k}(u, v, w) = \frac{d!}{i!j!k!} u^i v^j w^k, \qquad i + j + k = d \tag{14}$$

where i, j, k are nonnegative integer number. The total number of $B^{(d)}_{i,j,k}(u, v, w)$ is $\frac{1}{2}(d+1)(d+2)$. It can be shown [13] that Bernstein polynomials satisfy the following recursion :

$$B^{(d)}_{i,j,k}(u, v, w) = uB^{(d-1)}_{i-1,j,k}(u, v, w) + vB^{(d-1)}_{i,j-1,k}(u, v, w) + wB^{(d-1)}_{i,j,k-1}(u, v, w) \tag{15}$$

where u, v, w satisfies the constraints: $0 \le u,\ v,\ w \le 1$, and $u + v + w = 1$. Bernstein polynomials $B^{(d)}_{i,j,k}(u, v, w)$ are well suitable as a fuzzy membership function candidates as $B^{(d)}_{i,j,k}(u, v, w) \in [0, 1]$ (representing fuzzy membership function) and $\sum_{i+j+k=d} B^{(d)}_{i,j,k}(u, v, w) = 1$ (a unity of support). Consider the construction problem for the bivariate term $f_{k_1k_2}(x_{k_1}, x_{k_2})$. Denote a system input subvector $\mathbf{x}_{k_1k_2} = [x_{k_1}, x_{k_2}]^T \in \Re^2$, where x_{k_1} and x_{k_2}, $(k_1 \ne k_2)$ denote the k_1th and k_2th components of the input vector $\mathbf{x}(t)$. Suppose $f_{k_1k_2}(x_{k_1}, x_{k_2})$ is modelled using the neurofuzzy system with Bernstein polynomials $B^{(d)}_{i,j,k}(u, v, w)$ as basis functions as

$$f_{k_1k_2}(x_{k_1}, x_{k_2}) = \sum_{i+j+k=d} \omega_{i,j,k} B^{(d)}_{i,j,k} \left[u(\mathbf{x}_{k_1k_2}(t)), v(\mathbf{x}_{k_1k_2}(t)), w(\mathbf{x}_{k_1k_2}(t))\right] \tag{16}$$

where $u(\mathbf{x}_{k_1k_2}(t)) + v(\mathbf{x}_{k_1k_2}(t)) + w(\mathbf{x}_{k_1k_2}(t)) = 1$, and $0 \le u(\mathbf{x}_{k_1k_2}(t)),\ v(\mathbf{x}_{k_1k_2}(t)),\ w(\mathbf{x}_{k_1k_2}(t)) \le 1$. The total number of terms in Eq.(16) is $\frac{1}{2}(d + 1)(d + 2)$.

It follows that in order to construct the bivariate term $f_{k_1k_2}(\bullet)$ based on Eq.(16) it is necessary to find a one-to-one mapping: $\Psi_{k_1k_2} : \mathbf{x}_{k_1k_2}(t) \in \Re^2 \to \{u(\mathbf{x}_{k_1k_2}(t)), v(\mathbf{x}_{k_1k_2}(t))\} \in triangle\{u = 0, v = 0, u+v = 1\}$, or it is necessary to relate system input subvector $\mathbf{x}_{k_1k_2}(t)$ with the parameters $\{u(t), v(t), w(t)\}$. Based on this basic idea, a construction algorithm has been introduced by [1] which aims to construct the function mapping $\Psi_{k_1k_2} : \mathbf{x} \in \tilde{S} \to \{u, v\} \in triangle\{u = 0, v =$

$0, u+v=1\}$, and the procedure is described for completeness in Section 4.3. Fundamental to the construction is the concept of barycentric coordinates and the de Casteljau algorithm which has been widely used in computer aided geometric design [13]. For the purpose of exposition of this modelling algorithm, some background about barycentric coordinates and de Casteljau algorithm is initially described in Section 4.2.

4.2 Barycentric coordinates and de Casteljau algorithm for triangular blossoms

The construction of Bézier-Bernstein curve and surface has been widely used in computer aided geometric design [13]. Fundamental to the curve and surface design are the concept of barycentric coordinates and the de Casteljau algorithm.

Definition 1 *barycentric coordinates.*

The location of a point $\mathbf{x}_\circ \in \Re^2$ within a triangle S_1 with the vertices $\mathbf{b}_i$, $i=1,2,3$, where the subscript $_\circ$ is used to distinguish with input vector $\mathbf{x} \in \Re^n$, can be expressed uniquely in terms of barycentric coordinates $\{u,v,w\}$ with respect to the triangle S_1 as

$$\mathbf{x}_\circ = u\mathbf{b}_1 + v\mathbf{b}_2 + w\mathbf{b}_3, \quad \text{with} \quad u+v+w=1, \quad 0 \le u,v,w \le 1. \tag{17}$$

Definition 2 *Bézier triangle patch.*

A Bézier triangle patch of polynomial degree d can be constructed based on a polygon S terraced by d^2 triangles formed by $\frac{1}{2}(d+1)(d+2)$ vertices labelled as $\mathbf{b}_{i,j,k}$, with $i+j+k=d$. In the cubic case, as shown in Fig.6, the control net consists of vertices

$$\begin{array}{c}
\mathbf{b}_{3,0,0} \\
\mathbf{b}_{2,1,0}\mathbf{b}_{2,0,1} \\
\mathbf{b}_{1,2,0}\mathbf{b}_{1,1,1}\mathbf{b}_{1,0,2} \\
\mathbf{b}_{0,3,0}\mathbf{b}_{0,2,1}\mathbf{b}_{0,1,2}\mathbf{b}_{0,0,3}
\end{array}$$

The de Casteljau algorithm involves the construction of a point $\mathbf{b}^{(d)}_{0,0,0}(u,v,w)$ on a Bézier triangle patch (cubic polynomial surface if d=3) by using a recursive linear interpolation. Given a triangle array of points $\mathbf{b}_{i,j,k}$, with $i+j+k=d$ and a given barycentric coordinates $\{u,v,w\}$, $\quad$ *with* $\quad u+v+w=1, 0 \le u,v,w \le 1$.

$$\mathbf{b}^{(r)}_{i,j,k}(u,v,w) = u\mathbf{b}^{(r-1)}_{i+1,j,k}(u,v,w) + v\mathbf{b}^{(r-1)}_{i,j+1,k}(u,v,w) + w\mathbf{b}^{(r-1)}_{i,j,k+1}(u,v,w) \tag{18}$$

for $r=1,2,\cdots,d$, where $i+j+k=d-r$, $\mathbf{b}^{(0)}_{i,j,k}(u,v,w)=\mathbf{b}_{i,j,k}$. It can be shown that, by using the recursive formula Eq.(15) and (18), the point on the resultant Bézier triangle patch $\mathbf{b}^{(d)}_{0,0,0}(u,v,w)$

(denoted as $\hat{\mathbf{x}}_\circ(u,v,w)$) can be written as [13]

$$\hat{\mathbf{x}}_\circ(u,v,w) = \mathbf{b}^{(d)}_{0,0,0}(u,v,w) = \sum_{i+j+k=d} \mathbf{b}_{i,j,k}(u,v,w) B^{(d)}_{i,j,k}(u,v,w) \tag{19}$$

Denote the range of resultant Bézier triangle patch formed using polygon S as $\tilde{S}$. The de Casteljau algorithm realises mapping $\Psi_\circ^{-1}: \{u,v\} \in triangle\{u=0, v=0, u+v=1\} \rightarrow \mathbf{x}_\circ(u,v) \in \tilde{S}$. An important property called endpoint interpolation exists for Bézier surface, that is, $\hat{\mathbf{x}}_\circ(1,0,0) = \mathbf{b}^{(d)}_{d,0,0}$, $\hat{\mathbf{x}}_\circ(0,1,0) = \mathbf{b}^{(d)}_{0,d,0}$, and $\hat{\mathbf{x}}_\circ(0,0,1) = \mathbf{b}^{(d)}_{0,0,d}$.

Definition 3 *Extended barycentric coordinates*

A natural extension of the definition 1 is to interprete $\{u,v,w\}$ as the the barycentric coordinates of $\hat{\mathbf{x}}_\circ \in S$ with respect to S, if the location of $\hat{\mathbf{x}}_\circ \in \tilde{S}$ can be expressed in terms of barycentric coordinates $\{u,v,w\}$ with respect to the polygon S with vertices $\mathbf{b}_{i,j,k}$, $i+j+k=d$ as

$$\hat{\mathbf{x}}_\circ(u,v,w) = \sum_{i+j+k=d} \mathbf{b}_{i,j,k}(s) B^{(d)}_{i,j,k}(u,v,w) \tag{20}$$

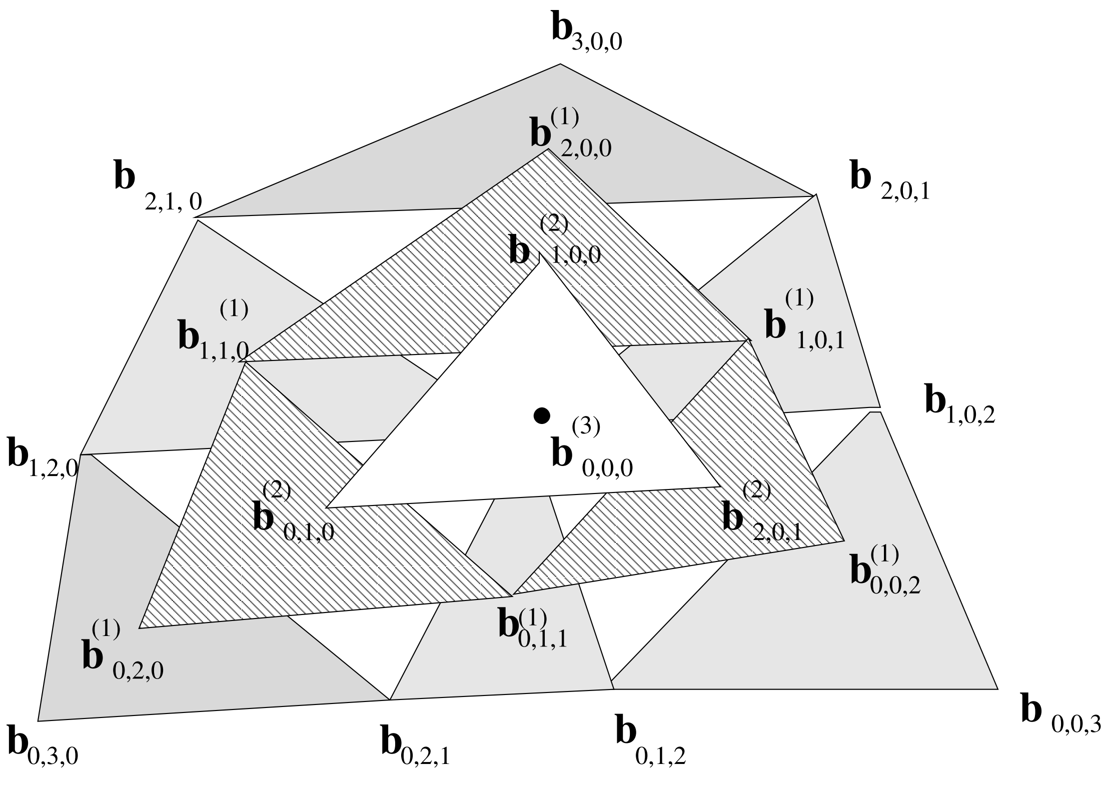


Figure 6: Bézier surface generation using de Casteljau algorithm

It follows from Definition 3 that in the neurofuzzy system Eq.(1), shown in Fig.1, the bivariate term $f_{k_1k_2}(\bullet)$ can be constructed based on Bernstein polynomial basis function as given in Eq.(16), if $u(\mathbf{x}_{k_1k_2}(t)), v(\mathbf{x}_{k_1k_2}(t))$ are barycentric coordinates of $\mathbf{x}_{k_1k_2}(t)$ with respect to S, a polygon which is formed using a set of predetermined knots $\mathbf{b}_{i,j,k}$. The endpoint interpolation property of a Bézier surface can be applied to predetermine the knots $\mathbf{b}_{i,j,k}$ using the input data range. The input data domain should be bounded within the Bézier triangle patch $\tilde{S}$, formed from S using the De Casteljau algorithm. It is convienient to predetermine the boundary knots such that S should be sufficiently larger than input data domain due to the endpoint interpolation property.

4.3 Inverse de Casteljau algorithm using backpropagation

The de Casteljau algorithm is a recursive algorithm which realises the mapping from $\{u, v\} \in triangle\{u = 0, v = 0, u + v = 1\} \rightarrow \mathbf{x}_{k_1k_2} \in \tilde{S}$, with respect to the known supporting knots $\mathbf{b}_{i,j,k}$. The neurofuzzy network construction algorithm based on Bézier-Bernstein function is different from the de Casteljau algorithm based surface design method in that the objective is to learn the system characteristics only from input/output data, so that according to the basic idea given in Section 4.1, an *inverse* procedure of de Casteljau algorithm is needed to find the barycentric coordinates. So in the proposed algorithm, the Bézier Bernstein polynomial function based network basis, are formed by using the barycentric coordinates of input subvector, which in turn are obtained by using an inverse de Casteljau procedure based on backpropagation. The network weights can then trained by the least squares methods as in other linear in the weights artificial networks [5,14]. The principle of the new inverse de Casteljau algorithm is as follows: given input subvector data $\mathbf{x}_{k_1k_2} \in \tilde{S}$, the proposed inverse de Casteljau algorithm attempts to construct the mapping $\Psi : \mathbf{x}_{k_1k_2} \in \tilde{S} \rightarrow \{u, v\} \in triangle\{u = 0, v = 0, u + v = 1\}$ using backpropagation, where $\{u, v, 1 - u - v\}$ is the barycentric coordinates of input subvector $\mathbf{x}_{k_1k_2}$ with respect to the known supporting knots. Note that the de Casteljau algorithm can be viewed as a special multi-layer network structure, illustrated by Fig.6 and the recursive formula Eq.(18). For an input subvector $\mathbf{x}_{k_1k_2}(t)$, a method of obtaining a desired $\{u, v, 1-u-v\}$ is by the minimisation the loss function $\|\mathbf{x}_{k_1k_2}(t) - \hat{\mathbf{x}}_\circ(u, v)\|$ via backpropagation. In the backpropagation procedure, the traditional backpropagation method can be extended here for this special multi-layer network structure by utilising a matrix inversion [11, 17] which derives barycentric coordinates of an input data point with respect to a triangle (within which the input data is bounded). It is advantageous that the output error $\|\mathbf{x}_{k_1k_2}(t) - \hat{\mathbf{x}}_\circ(u, v)\|$ can be easily propagated to dynamically adjust the values for the desired $\{u, v, 1 - u - v\}$, so as the mapping $\Psi : \mathbf{x}_{k_1k_2} \in \tilde{S} \rightarrow \{u, v\} \in triangle\{u = 0, v = 0, u + v = 1\}$ can be achieved. Denote the iteration step in the minimisation procedure as superscript (m).

1. Initial set $m = 1$, $\{u, v\}$ as random number $\{u^{(m)}, v^{(m)}\}$, satisfying the constraints $0 < u^{(m)} < 1$, $0 < v^{(m)} < 1$ and $0 < u^{(m)} + v^{(m)} < 1$.

2. Calculate the corresponding point on the Bézier triangle patch using the de Casteljau algorithm using the recursive formula Eq.(18) until $r = d$.

$$
\begin{aligned}
\hat{\mathbf{x}}_\circ(u^{(m)}, v^{(m)}, w^{(m)}) \quad = \quad & u^{(m)}\mathbf{b}_{1,0,0}^{(d-1)}(u^{(m)}, v^{(m)}, w^{(m)}) + v^{(m)}\mathbf{b}_{0,1,0}^{(d-1)}(u^{(m)}, v^{(m)}, w^{(m)}) \\
& + w^{(m)}\mathbf{b}_{0,0,1}^{(d-1)}(u^{(m)}, v^{(m)}, w^{(m)})
\end{aligned}
\tag{21}
$$

3. A new point is generated by using backpropagation rule through

$$\tilde{\mathbf{x}}_{k_1k_2}(u^{(m)}, v^{(m)}, w^{(m)}) = \hat{\mathbf{x}}_\circ(u^{(m)}, v^{(m)}, w^{(m)}) + \alpha[\mathbf{x}_{k_1k_2}(t) - \hat{\mathbf{x}}_\circ(u^{(m)}, v^{(m)}, w^{(m)})] \tag{22}$$

where α is a very small positive number representing learning rate $0 < \alpha \ll 1$. Note that α should be sufficiently small such that $\tilde{\mathbf{x}}_{k_1k_2}(u^{(m)}, v^{(m)}, w^{(m)})$ is bounded within the triangle with vertices $\{\mathbf{b}_{1,0,0}^{(d-1)}(u^{(m)}, v^{(m)}, w^{(m)}),\ \mathbf{b}_{0,1,0}^{(d-1)}(u^{(m)}, v^{(m)}, w^{(m)}),\ \mathbf{b}_{0,0,1}^{(d-1)}(u^{(m)}, v^{(m)}, w^{(m)})\}$.

4. The desired solution of $\{u, v\}$ at iteration step $(m+1)$ is determined as the barycentric coordinates of $\tilde{\mathbf{x}}_{k_1k_2}(u^{(m)}, v^{(m)}, w^{(m)})$ with respect to the triangle with vertices $\{\mathbf{b}_{1,0,0}^{(d-1)}(u^{(m)}, v^{(m)}, w^{(m)}),$ $\mathbf{b}_{0,1,0}^{(d-1)}(u^{(m)}, v^{(m)}, w^{(m)}),\ \mathbf{b}_{0,0,1}^{(d-1)}(u^{(m)}, v^{(m)}, w^{(m)})\}$. Applying Eq.(17) (by definition) with w replaced by $1 - u - v$, yields

$$\begin{aligned}\tilde{\mathbf{x}}_{k_1k_2}(u^{(m)}, v^{(m)}, w^{(m)}) &= u\mathbf{b}_{1,0,0}^{(d-1)}(u^{(m)}, v^{(m)}, w^{(m)}) + v\mathbf{b}_{0,1,0}^{(d-1)}(u^{(m)}, v^{(m)}, w^{(m)}) \\ &\quad +(1-u-v)\mathbf{b}_{0,0,1}^{(d-1)}(u^{(m)}, v^{(m)}, w^{(m)})\end{aligned} \tag{23}$$

The solution $\{u, v\}$ can be solved [11, 17] from Eq.(23) as

$$\left|\begin{array}{c} u^{(m+1)} \\ v^{(m+1)} \end{array}\right| = \tilde{B}_{(m)}^{-1}\left[\tilde{\mathbf{x}}_{k_1k_2}(u^{(m)}, v^{(m)}, w^{(m)}) - \mathbf{b}_{0,0,1}^{(d-1)}(u^{(m)}, v^{(m)}, w^{(m)})\right]$$

where

$$\tilde{B}_{(m)} = \left[\mathbf{b}_{1,0,0}^{(d-1)}(u^{(m)}, v^{(m)}, w^{(m)}) - \mathbf{b}_{0,0,1}^{(d-1)}(u^{(m)}, v^{(m)}, w^{(m)}), \mathbf{b}_{0,1,0}^{(d-1)}(u^{(m)}, v^{(m)}, w^{(m)}) - \mathbf{b}_{0,0,1}^{(d-1)}(u^{(m)}, v^{(m)}, w^{(m)})\right] \tag{24}$$

5. The procedure continues while the cost function $||\mathbf{x}_{k_1k_2}(t) - \hat{\mathbf{x}}_\circ(u, v)||$ is monitored to see if $||\mathbf{x}_{k_1k_2}(t) - \hat{\mathbf{x}}_\circ(u^{(m)}, v^{(m)})|| \le \varepsilon$, where ε is a positive number close to zero. If $||\mathbf{x}_{k_1k_2}(t) - \hat{\mathbf{x}}_\circ(u, v)|| \le \varepsilon$, then set $u = u^{(m)}$ and $v = v^{(m)}$ and the iteration loop stops. Otherwise set $m = m + 1$, goto step 2.

5 A summary of the construction algorithm

The neurofuzzy modelling using Bézier bernstein polynomial function via additive decompostion, shown in Fig.1, is summarised as an offline version as below. Given input/output data sets $\{\mathbf{x}(t), y(t)\}_{t=1}^N$. Predetermine a polynomial degree as d.

1. For each $x_k(t)$, the kth component of the input vector $\mathbf{x}(t)$, predetermine $b_0, \cdots, b_d$, first components of $(d+1)$ knots as shown in Fig.4, under the constraints that $[\min(x_k(t)), \max(x_k(t))] \in [b_0, b_d]$.

2. For all t, applying the univariate inverse de Casteljau algorithm as introduced in Section 3 to find out the solution $s(t)$ for $x_k(t)$.

3. Applying Eq.(4) and (6) form a basis subvector $\mathbf{B}(x_k(t)) \in \Re^{d+1}$. The objective of procedures $1 \sim 3$ is to form the basis functions in the univariate term $f_k(\bullet)$. The total number of the univariate basis functions is therefore $n(d+1)$.

4. For every combination using $x_{k_1}(t)$ and $x_{k_2}(t)$, $(k_1 \neq k_2)$ two distinctive components of the input vector $\mathbf{x}(t)$, predetermine the position of $\frac{1}{2}(d+1)(d+2)$ knots $\mathbf{b}_{i,j,k}$ ranked to form the polygon S shaped as plotted in Fig.6. This can be done by graphically inputing via mouse in a graph window in which the input data range had been plotted. The polygon S should be sufficiently larger than the input range to ensure all the input data are bounded within the generated d polynomial degree Bézier surface which formed using a series of triangles formed by $\mathbf{b}_{i,j,k}$'s.

5. For all t, applying the bivariate inverse de Casteljau algorithm as described in Section 4.3 to find the $u(t)$, $v(t)$, and $w(t) = 1 - u(t) - v(t)$ from the input subvector $[x_{k_1}(t), x_{k_2}(t)]$.

6. Applying Eq.(14) and (16) to form the basis vector $\mathbf{B}(\mathbf{x}_{k_1k_2}(t)) \in \Re^{\frac{1}{2}(d+1)(d+2)}$ for each input subvector $[x_{k_1}(t), x_{k_2}(t)]$. The objective of procedures 4 $\sim$ 6 is to from the basis function in the bivariate term $f_{k_1k_2}(\bullet)$. The total number of the bivariate basis functions is therefore $\frac{1}{4}n(n-1)(d+1)(d+2)$.

7. Substitute Eq.(6) and (16) into Eq.(1) and (3). The total number of the basis function excluding the constant term is $p = n(d+1) + \frac{1}{4}n(n-1)(d+1)(d+2)$ (Compared with the L^n basis functions of a conventional n -dimensional neurofuzzy model constructed using tensor product approach with L univariate basis functions). For instance, if $n = 3$, $d = 3$, $L = 5$, then there is 42 basis functions in the new approach compared with 125 in conventional approach, and if $n = 4$, $d = 3$, $L = 5$, then there are 76 basis functions in the new approach compared with 625 in conventional approach. Least squares is then used, via the minimisation of $\sum_{t=1}^{N}(y(t) - \hat{y}(t))^2$, to obtain the weights vector Ω in Eq.(3).

6 Numerical examples

Example 1. Consider the non-linear system

$$y(t) = \frac{2.5y(t-1)y(t-2)(1+u_s(t-1))}{1+y(t-1)^2+y(t-2)^2} + 0.3\cos\left[0.5(y(t-1)+y(t-2))\right] + 1.2u_s(t-1) \quad (25)$$

where system input is $u_s(t) = sin(\pi t/20) + sin(\pi t/50)$. Here the subscript "$_s$" is used to distinguish the system input with $u(t)$, one of the barycentric coordinates. 500 data points were generated and the system output is added with a noise sequence with mean zero and variance 0.1^2. The system output with noise is used as estimation data set. The system input vector is set as $\mathbf{x}(t) = [y(t-1), y(t-2), u_s(t-1)]^T$. Set polynomial degree d=4 for both univariate basis function and bivariate basis function. Corresponding to the additive expansion based on Eq.(1). The system input vector is decomposed into 6 terms based on $y(t-1)$, $y(t-2)$, $u_s(t-1)$, $[y(t-1), u_s(t-1)]$, $[y(t-1), u_s(t-1)]$ and $[y(t-1), u_s(t-1)]$ respectively. For each of these terms, the knots were input to a graph window as shown in Fig.7. Note that the knots are subject to constraints, for instance, for univariate term, the input region $[\min(x_k(t)), \max(x_k(t))] \in [b_0, b_d]$, and for the bivariate terms, a polygon S formed by the knots $\mathbf{b}_{i,j,k}$'s should sufficiently cover the input region.

For all data samples, the univariate inverse de Casteljau algorithm was applied to find the $s(t)$ from 3 input components $y(t-1)$, $y(t-2)$ and $u_s(t-1)$ respectively. Each basis subvector $\mathbf{B}(x_k(t)) \in \Re^{d+1}$ was formed by applying Eq.(4) and (6). The number of basis function in univariate terms is therefore $3(d+1) = 15$.

Then for all data samples, the bivariate inverse de Casteljau algorithm was applied to find the $u(t), v(t)$, and $w(t) = 1 - u(t) - v(t)$ from 3 input subvector $[y(t-1), u(t-1)]$, $[y(t-1), u(t-1)]$ and $[y(t-1), u(t-1)]$ respectively. By applying Eq.(14) and (16), the bivariate basis vector $\mathbf{B}(\mathbf{x}_{k_1k_2}(t)) \in \Re^{\frac{1}{2}(d+1)(d+2)}$ for each subvector, $[y(t-1), y(t-2)]$, $[y(t-1), u_s(t-1)]$ and $[y(t-1), u_s(t-1)]$ was formed. The total number of the bivariate basis function is therefore $\frac{3}{2}(d+1)(d+2) = 45$.

The total number of basis function in this example is $p = 60$ excluding the constant term, compared with the $L^n = 125$ basis functions of a typical example of conventional n-dimensional neurofuzzy model using the tensor product approach with $L = 5$ univariate basis functions.

The weight vector Ω were finally obtained by using least squares. The mean square error (MSE) of the network was obtained as $\sigma^2 = 0.106^2$. The autocorelation coefficients of the model residual, plotted in Fig8(a), is used for model validation, showing that the model residual is uncorrelated. A comparison of the actual system output with one-step ahead prediction using the obtained network is demonstrated in Fig.8(b) to indicate the excellent system performance. The results from this example demonstrate the excellent modelling properties of the Bézier Bernstein polynomial function based neurofuzzy system based neurofuzzy system.

Example 2. Consider non-linear dynamical system given by

$$y(t) \quad = \quad \frac{y^2(t-1) + u_s(t-1)}{1 + y^2(t-1) + y(t-1)u_s(t-2)} + 0.3e(t-1) + e(t) \tag{26}$$

where the system input sequence $u_s(t)$ was a uniformly distributed random excitation signal within range $[-1, 1]$. and the noise sequence $e(t) \sim N(0, 0.1^2)$. 500 data sample were generated for estimation set. Suppose that system structure is known to have a linear noise term

$$y(t) = f(\mathbf{x}(t)) + ce(t-1) + e(t) \tag{27}$$

where the input vector $\mathbf{x} = [y(t-1), u_s(t-1), u_s(t-2)]$. Set polynomial degree d=4 for both univariate basis function and bivariate basis function. Corresponding to the additive expansion based on Eq.(1). The system input vector is decomposed into 6 terms based on $y(t-1)$, $u_s(t-1)$, $u_s(t-2)$, $[y(t-1), u_s(t-1)]$, $[y(t-1), u_s(t-2)]$ and $[u_s(t-1), u_s(t-2)]$ respectively. For each of these terms, the knots were input to a graph window as shown in Fig.9. The knots are subject to constriants such that, for univariate term, the input region $[\min(x_k(t)), \max(x_k(t))] \in [b_0, b_d]$, and for the bivariate terms, a polygon S formed by the knots $\mathbf{b}_{i,j,k}$'s should sufficiently cover the input region. In this example the knots for univariate terms $f(y(t-1))$ is plotted as in Fig9(a), the knots for univariate terms $f(u_s(t-1))$ and $f(u_s(t-2))$ (set to be the same) was plotted as in Fig9(b). The knots for bivariate terms $f(y(t-1), u_s(t-1))$, $f(y(t-1), u_s(t-2))$ and $f(u_s(t-1), u_s(t-2))$

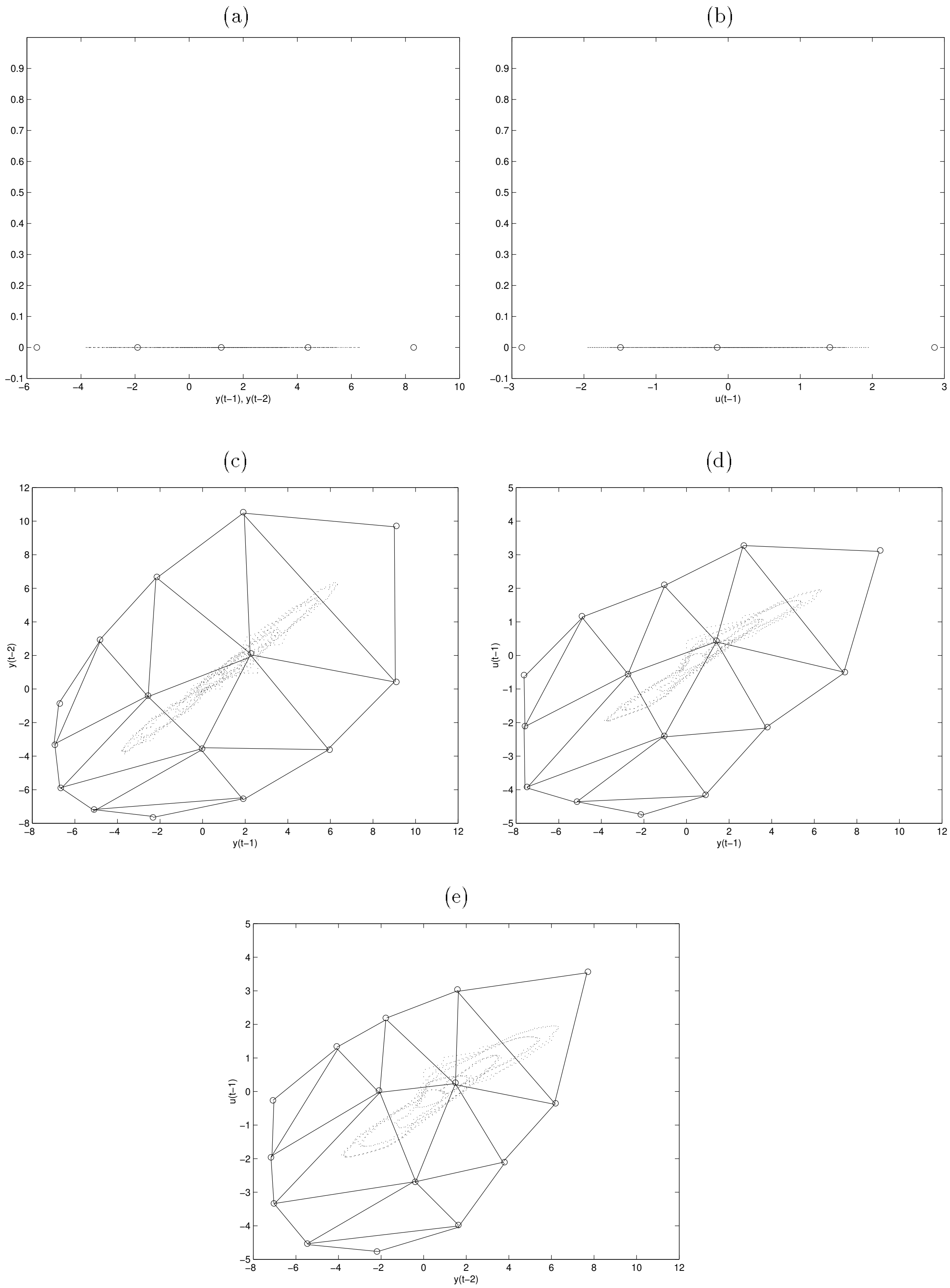


Figure 7: Input space partition using predetermined knots in example 1; (a) Knots for univariate term $f(y(t-1))$ and $f(y(t-2))$; (b) Knots for univariate term $f(u_s(t-1))$; (c) Knots for bivariate term $f(y(t-1), y(t-2))$; (d) Knots for bivariate term $f(y(t-1), u_s(t-1))$; and (e) Knots for bivariate term $f(y(t-2), u_s(t-1))$

(a)

(b)

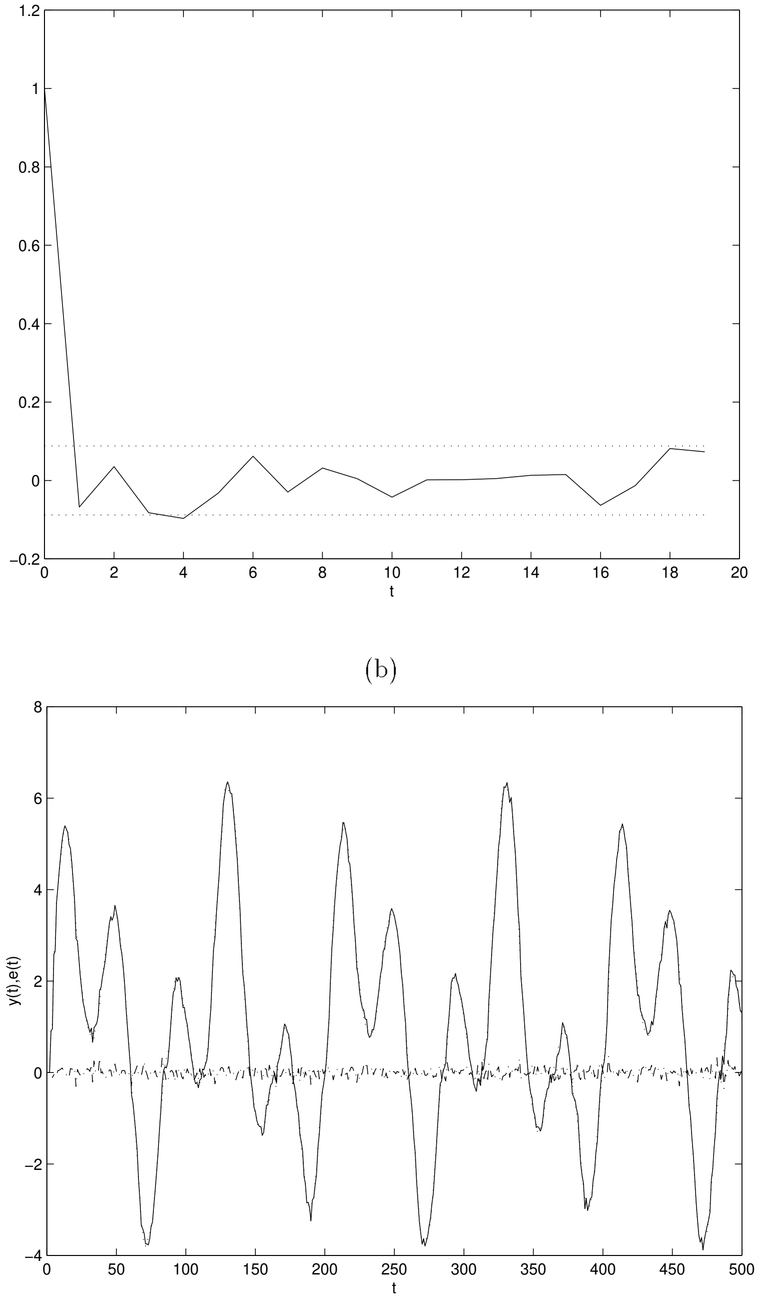


Figure 8: Simulation results in example 1; (a)Model validation test using autocorrelation coefficients of the residual sequence; (b) System actual output (Solid Line), model one-step ahead predictions (Dotted Line) and model residual (Dashdotted Line)

were plotted in Fig9(c),(d),(e), respectively.

For all data samples, the univariate inverse de Casteljau algorithm was applied to find the $s(t)$ from 3 input components $y(t-1)$, $u_s(t-1)$ and $u_s(t-2)$ respectively. Each basis subvector $\mathbf{B}(x_k(t)) \in \Re^{d+1}$ was formed by applying Eq.(4) and (6). The number of basis function in univariate terms is therefore $3(d+1) = 15$.

Then for all data sample, the bivariate inverse de Casteljau algorithm was applied to find the $u(t), v(t)$, and $w(t) = 1 - u(t) - v(t)$ from 3 input subvector $[y(t-1), u_s(t-1)]$, $[y(t-1), u_s(t-2)]$, and $[u_s(t-1), u_s(t-2)]$ respectively. By applying Eq.(14) and (16), the bivariate basis vector $\mathbf{B}(\mathbf{x}_{k_1k_2}(t)) \in \Re^{\frac{1}{2}(d+1)(d+2)}$ for each subvector, $[y(t-1), u_s(t-1)]$, $[y(t-1), u_s(t-2)]$, and $[u_s(t-1), u_s(t-2)]$ was formed. The total number of the basis functions in bivariate terms is therefore $\frac{3}{2}(d+1)(d+2) = 45$.

For parameter estimation of Eq.(26), the weight vector Ω were finally obtained by using least squares. Here the extended least squares was used due to the noise term in the model structure of Eq.(26) and the term $e(t-1)$ can be substituted by $(y(t-1) - \hat{y}(t-1|t-2))$, where $\hat{y}(t-1|t-2)$ was model predictions via several times of iterations. The mean square error (MSE) of the network was obtained as $\sigma^2 = 0.098^2$. The autocorelation coefficients of the model residual, plotted in Fig10(a), is used for model validation, showing that the model residual is uncorrelated. A comparison of the actual system output with one-step ahead prediction using the obtained network is demonstrated in Fig.10(b) to indicate the excellent system performance. It has been shown again by this complicated function approximation example that Bézier Bernstein polynomial function based neurofuzzy system possesses excellent modelling capabilities.

7 Conclusions

This paper generalises a new neurofuzzy system model construction algorithm over n-dimensional input space by using Bézier Bernstein polynomial functions based on an additive decomposition approach. This new modelling network is based on the additive decomposition approach to expand the sytem model into both univariate terms and bivariate terms, with two separate basis function formation approaches for both univariate and bivariate Bézier Bernstein polynomial functions construction, the latter is based on previous work [12] where Bézier Bernstein polynomial function are introduced as basis functions for two-dimensional input space problem. In order to model the univariate term, a new construction algorithm for univariate Bézier Bernstein polynomial functions is developed for the completeness of the generalised procedure. The overall weights are then learnt using least squares methods. Like B-spline expansion based neurofuzzy system, Bézier Bernstein polynomial functions based neurofuzzy networks hold desirable properties such as nonnegativity of the basis functions, unity of support, interpretability of basis function as fuzzy membership function. It is also additionally advantageous in terms of structural parsimony and Delaunay input space partition. Numerical examples are included to demonstrate the effectiveness of this

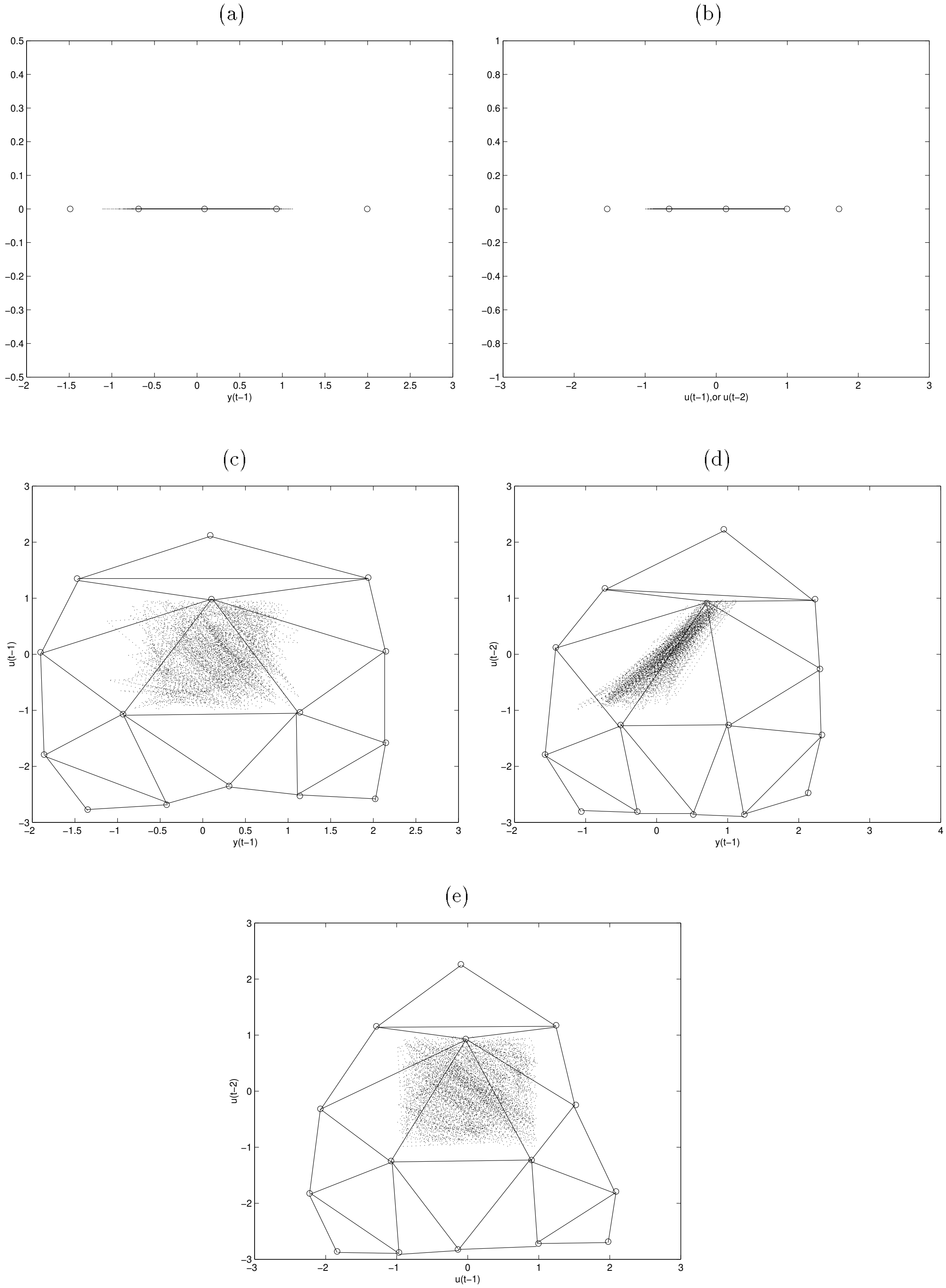
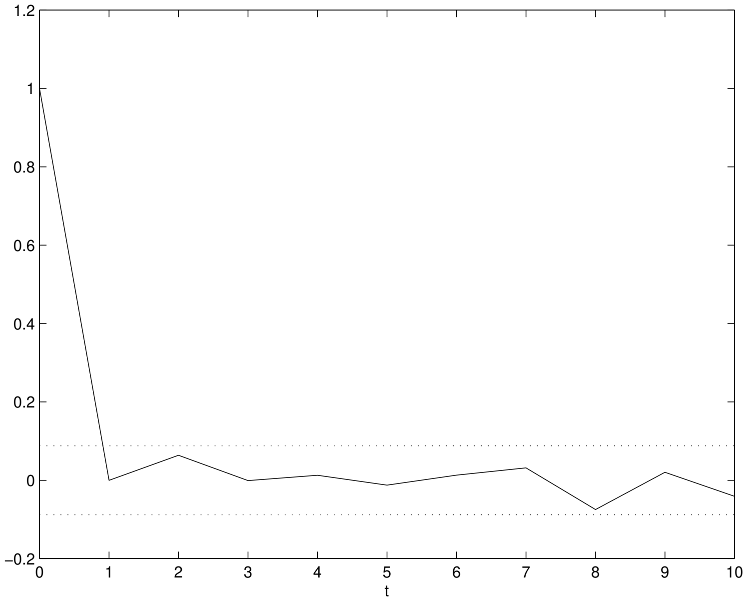


Figure 9: Input space partition using predetermined knots in example 2; (a) Knots for univariate term $f(y(t-1))$; (b) Knots for univariate term $f(u_s(t-1)), f(u_s(t-2))$; (c) Knots for bivariate term $f(y(t-1), u_s(t-1))$; (d) Knots for bivariate term $f(y(t-1), u_s(t-2))$; and (e) Knots for bivariate term $f(u_s(t-1), u_s(t-2))$

(a)



(b)

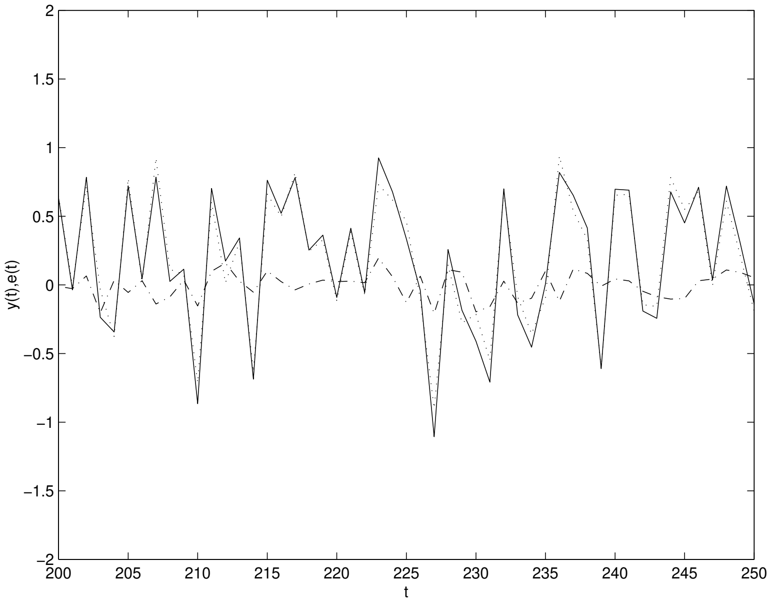


Figure 10: Simulation results in example 2; (a)Model validation test using autocorrelation coefficients of the residual sequence; (b) System actual output (Solid Line), model one-step ahead predictions (Dotted Line) and model residual (Dashdotted Line)

new approach. This new algorithm is applicable to a wide range of nonlinear signal processing problems.

8 Acknowledgements

The authors gratefully acknowledge the finacial support of EPSRC and RACAL.

References

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

[2] Wang, L. X. (1994). *Adaptive Fuzzy Systems and Control: design and stability analysis.* Prentice Hall, NJ.

[3] Jang, J. S. R, C. T. Sun, and Mizutani, E. (1997). *Neurofuzzy and Soft Computing.* Prentice Hall, NJ.

[4] Juditsky, A., Hjalmarsson, H., Benveniste, A., Delyon, B., Liung, L., Sjoberg, J. and Zhang, Q. (1995). Nonlinear black-box models in system identification: mathematical foundations. *Automatica.*, Vol. 31, No.12, pp1725-1750.

[5] Harris, C. J., M. Brown, K. M. Bossley, D. J. Mills and Feng, M. (1996). Advances in neurofuzzy algorithms for real time modelling and Control. *J. Engineering Applications. AI.*. 9(1). pp1-16.

[6] Wu, Z. Q. and Harris, C. J. (1997). A neurofuzzy network structure for modelling and state estimation for unknown nonlinear systems. *Int. J. Systems Science.* Vol. 28(4), pp335-345.

[7] Hong, X., C. J. Harris and P. A. Wilson (1999). Neurofuzzy state identification using prefiltering. *Proc IEE Part. D. Control Theory and Applications,* Vol. 146, No. 2, pp234-240.

[8] Doyle, R. S. and Harris, C. J. (1996). Multi-sensor data fusion for helicopter guidance using neurofuzzy estimation algorithms. *Royal Aeronautical Society Journal.* June/July, Vol.102. pp241-251.

[9] Bossley, K. M. (1997). *Neurofuzzy Modelling Approaches in System Identification.* Ph.D Thesis, Dept of Electronics and Computer Science, University of Southampton.

[10] Wu, Z. Q. and Harris, C. J. (1998). A mixture expert network for piecewise linear modelling. ISIS Technical Report, Dept of Electronics and Computer Science, University of Southampton.

[11] Harris, C. J., X. Hong and Feng, M. (1999). Optimal piecewise locally linear modelling. *SPIE Aero Sense'99, Applications and Science of Computational Intelligence II*, Orlando, USA.

[12] Harris, C. J. and X. Hong (1999). Neurofuzzy network model construction using Bézier Bernstein polynomial functions. ISIS report, Dept of Electronics and Computer Science, University of Southampton. *submitted for publication.*

[13] Gerald Farin (1994). *Curves and Surfaces for Comnputer-aided Geometric Design: a Practical Guide*. Academic Press, Boston.

[14] Goodwin G. C. and Sin, K. S. (1984). *Adaptive Filtering Prediction and Control*. Information and System Science Series, Prentice Hall, Englewood Cliffs, New Jersey.

[15] Kavli, T. (1993). ASMOD - an algorithm for adaptive spline modelling of obervation data. *Int. J. of Control*. Vol. 58, No. 4, pp947-967.

[16] Nelles, O., S. Sinsel and Isermann, R. (1996). Local basis function networks for identificaiton of a turbogenerator. *IEE. UKACC Control Int. Conf.*. Vol. 2, pp7-12.

[17] Hong, X., C. J. Harris and P. A. Wilson (1999). Autonomous ship collision free trajectory navigation and control algorithms. *7th IEEE international Conference on Emerging Technology and Factory Automation, ETFA'99*, Oct, 1999, Barcelona.