Neurofuzzy Network Model Construction using Bézier Bernstein Polynomial Functions

C. J. Harris and X. Hong

Image, Speech and Intelligent Systems Group

Department of Electronics and Computer Science

University of Southampton, Southampton SO17 1BJ, UK

Email: cjh,xh@ecs.soton.ac.uk; WWW: http://www.isis.ecs.soton.ac.uk;
Tel: +44 (0)1703 596653; Fax: +44 (0)1703 594498

Short title — Bézier Bernstein Polynomial based network

Abstract — 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-splines and algebraic operators for inference *etc.* This paper introduces a new neurofuzzy model construction algorithm using Bézier Bernstein polynomial functions as basis functions. The new network maintains most of the properties of the B-spline expansion based neurofuzzy system, such as nonnegativity of the basis functions, unity of support but with the additional advantages of structural parsimony and Delaunay input space partitioning, avoiding the inherent computational problems of lattice networks. This new modelling network is based on the idea that an input vector can be mapped into barycentric coordinates with respect to a set of predetermined knots as vertices of a polygon (a set of tiled delaunay triangles) over the input space. The network is expressed as the Bézier Bernstein polynomial function of barycentric coordinates of the input vector. A new inverse de Casteljau procedure using backpropagation is developed to obtain the input vector's barycentric coordinates that form the basis functions. Extension of the Bézier Bernstein neurofuzzy algorithm to n-dimensional inputs is discussed followed by numerical examples 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 systems have emerged in recent years as researchers [1-4] have tried to combine the natural linguistic/symbolic transparency of fuzzy systems with the provable learning & representation capability of linear in the weights artificial neural networks (ANN). 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 [5, 6] to multi-sensor data fusion for helicopter guidance [7]. 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 the tensor product of univariate basis functions. As the 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 additive decomposition techniques can overcome the problem [8]. 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 [9-11]. Delaunay partition of the input space has been shown to be more appropriate than lattice partition [9] in obtaining a parsimonious and smooth piecewise linear model. Because a neurofuzzy network can also be interpreted as a locally piecewise *polynomial* model which naturally have a stronger approximation capability than locally 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 partition, whilst minimising the number of model free parameters.

In this paper, a new neurofuzzy network construction algorithm using the Bézier Bernstein polynomial function has been introduced for two dimensional input space problem. Whilst the restriction of this method to 2D input space may seem restrictive, future work will focus on the development of effective algorithms for the general n-dimensional input space modelling problem. In practice, many high input dimensional modelling problem can be decomposed into a sum of univariate and bivariate submodels via the analysis of variance (or Gabor-Kolmogrov) expansion as [8].

$$f(\mathbf{x}) = f_0 + \sum_{i=1}^{n} f_i(x_i) + \sum_{i=1}^{n} \sum_{j=i+1}^{n} f_{ij}(x_i, x_j) + \cdots \tag{1}$$

where $f(\mathbf{x})$ is a general multivariate nonlinear mapping for input $\mathbf{x} = [x_1, x_2, \cdots, x_n]^T \in \Re^n$. $f_i(x_i)$ and $f_{ij}(x_i, x_j)$ are univariate and bivariate nonlinear submodels of $f(\bullet)$. The expansion in Eq.(1) contains a series of terms with intersection order higher than 2, which may be neglected for most practical approximation problems. Based on Eq.(1), general extension to higher dimensional input space can be achieved with some increased complexity in the learning algorithm design. The neurofuzzy network based on the expansion of Bézier Bernstein polynomial function will maintain most of the properties held by the B-spline expansion, such as nonnegativity of the basis functions, unity of support with additional advantages of structural parsimony and a Delaunay input space partition. The construction of the Bézier-Bernstein curve and surface is based on the concept of barycentric coordinates and the de Casteljau algorithm which has been widely used in computer aided geometric design [12]. Since a neural network has to be constructed only from input/output data, the proposed neurofuzzy network construction algorithm based on the Bézier-Bernstein function is a new concept, which can be regarded as an inverse procedure of the Bézier surface geometric design. In the proposed algorithm, the Bézier Bernstein polynomial network basis, are formed by using an input vector's barycentric coordinates, which are obtained by using a new inverse de Casteljau procedure using backpropagation. The network weights are then trained by the least squares method used in other linear in the weights ANNs [4, 13]. The traditional backpropagation method has been extended in the de Casteljau inverse mapping algorithm, which can be viewed as a special network structuring algorithm. A simple matrix inversion for deriving barycentric coordinates from input data with respect to a triangle has been included in the backpropagation procedure [10, 14].

This paper is organised as follows. In Section 2, the problem is formulated by a brief description of the neurofuzzy modelling methodology and the motivation of the proposed algorithm.

Section 3 introduces the new concept and construction algorithm of neurofuzzy system using Bézier-Bernstein polynomial functions. Numerical examples are presented in Section 4 to demonstrate the effectiveness of the new algorithm. Section 5 is devoted to conclusions.

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 is significant in this context,

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

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

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

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

where ˆ denotes estimates. $\Omega = [\omega_1, \cdots, \omega_p]^T \in \Re^p$, $\mathbf{B}^T(k)$ 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 [13] 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 procedure.

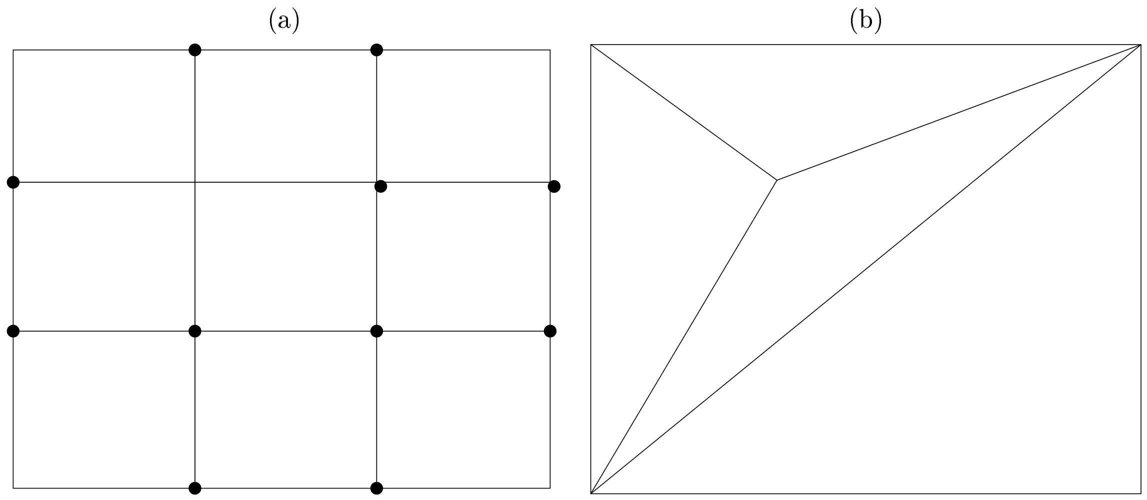


Figure 1: An illustrative example of hidden nodes required in two modelling approaches; (a). lattice partition, where • denotes redundant nodes redundancy compared with Delaunay partitioning approach (b)

The choice of membership function to represent system inputs is important for the modelling capability of a neurofuzzy model. Desired properties include compactness, 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 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 [8]. 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 [9-11]. Delaunay partition of the input space has been shown to be more appropriate than the conventional lattice partition [9] 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.1. Also piecewise local linear models based on Delaunay input space partition can overcome the discontinuity problem associated with lattice partitioned local piecewise linear models, this is simply illustrated in Fig.2, where $z = f(x, y)$ is modelled by two versions of local piecewise linear models. Because a neurofuzzy network can also be interpreted as a locally piecewise polynomial model which has stronger approximation ability 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 that all the advantages associated with the neurofuzzy network can be extended to the case of Delaunay input space partitioning. In this work, the Bézier Bernstein polynomial function is introduced as an alternative choice of fuzzy

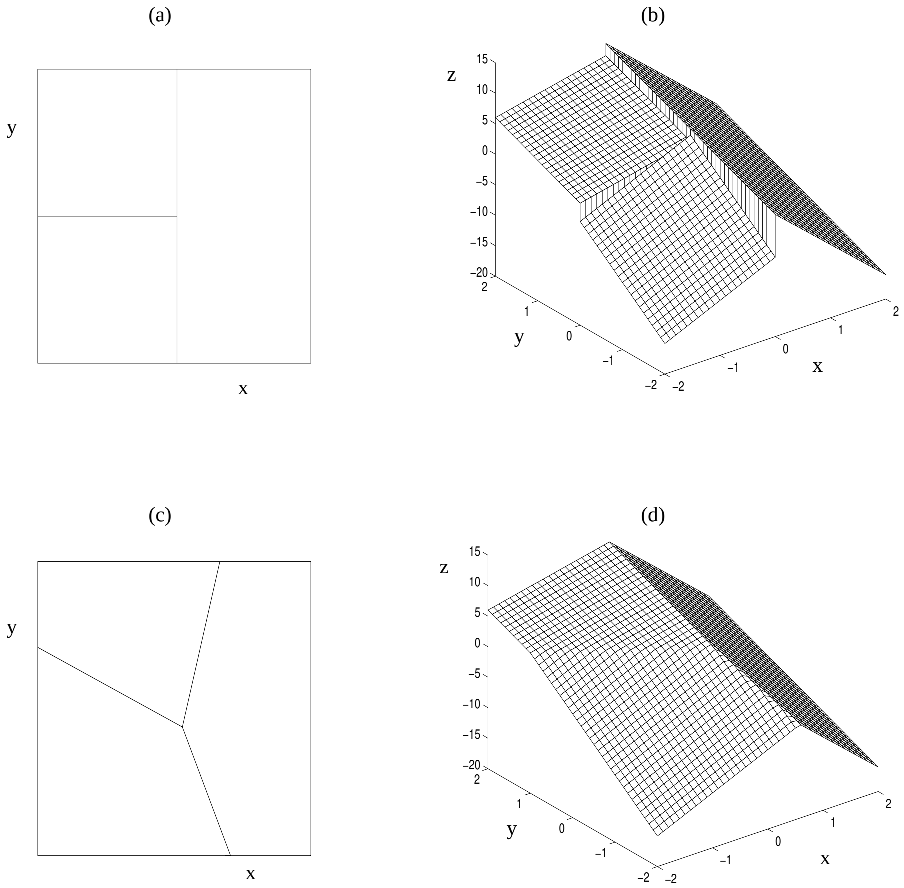


Figure 2: Input space partitions and the corresponding piecewise linear surfaces in modelling $z = f(x, y)$; (a). Orthogonal/lattice decomposition; (b). resultant locally piecewise linear models of (a); (c). Delaunay partitioning; (d). resultant locally piecewise linear models of (c).

function in the expansion of a general unknown nonlinear function. The neurofuzzy network based on the expansion of the Bézier Bernstein polynomial function will maintain most properties held by the B-spline expansion, such as nonnegativity of the basis functions, unity of support with an additional advantage of structural parsimony and Delaunay input space partition.

3 Neurofuzzy modelling using Bézier Bernstein Polynomial Function

The construction of the Bézier-Bernstein curve and surface based on a concept of barycentric coordinates and de Casteljau algorithm has been widely used in computer aided geometric design [12]. Since a neural network has to be constructed only from input/output data, the proposed neurofuzzy network construction algorithm based on Bézier-Bernstein polynomial functions is a new concept, which can be regarded as an inverse procedure of a Bézier surface construction that is applied in geometric design. In the proposed algorithm, the Bézier Bernstein polynomial function based network basis, are initially formed by using the barycentric coordinates of the input vector, which are obtained by using a new inverse de Casteljau procedure using backpropagation (see Section 3.3).

3.1 Fundamentals

Definition 1 *Bernstein polynomials.*

Bernstein polynomials $B^{(d)}_{i,j,k}(u,v,w)$ are the expansion of $(u+v+w)^d$, defined by

$$B^{(d)}_{i,j,k}(u,v,w) = \frac{d!}{i!j!k!}u^i v^j w^k, \qquad i+j+k=d \tag{4}$$

where i, j, k are nonnegative integer number. The total number of $B^{(d)}_{i,j,k}(u,v,w)$ is $p=\frac{1}{2}(d+1)(d+2)$. It can be shown [12] that Bernstein polynomials satisfy the following recursion:

$$B^{(d)}_{i,j,k}(u,v,w) = uB^{(d-1)}_{i-1,j,k}(u,v,w) + vB^{(d-1)}_{i,j-1,k}(u,v,w) + wB^{(d-1)}_{i,j,k-1}(u,v,w) \tag{5}$$

If u, v, w satisfies the constraints: $0 \le u,\ v,\ w \le 1$, and $u+v+w=1$, then 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 $\mu_A(\bullet)$) and $\sum_{i+j+k=d} B^{(d)}_{i,j,k}(u,v,w) = 1$ (a unity of support). Consider a two-dimensional input nonlinear system $y(t) = f(\mathbf{x}(t))$, $t = 1, 2, \cdots, N$, $\mathbf{x}(t) \in \Re^2$, where t is the labelling index of data samples, and N is the number of data samples, modelled using a neurofuzzy system with Bernstein polynomials $B^{(d)}_{i,j,k}(u,v,w)$ as basis functions as

$$\begin{aligned} \hat{y}(t) &= \mathbf{B}^T(\mathbf{x}(t))\Omega \\ &= \sum_{i+j+k=d} \omega_{i,j,k} B^{(d)}_{i,j,k}\left[u(\mathbf{x}(t)), v(\mathbf{x}(t)), w(\mathbf{x}(t))\right] \end{aligned} \tag{6}$$

where $\Omega = \{\omega_{i,j,k}\} \in \Re^p$ is the unknown weight parameter vector, $\mathbf{B}^T(\mathbf{x}(t)) \in \Re^p$ is basis function vector. $u(\mathbf{x}(t)) + v(\mathbf{x}(t)) + w(\mathbf{x}(t)) = 1$, and $0 \leq u(\mathbf{x}(t)),\ v(\mathbf{x}(t)),\ w(\mathbf{x}(t)) \leq 1$. The total number of terms in Eq.(6) is $p = \frac{1}{2}(d+1)(d+2)$.

It follows that in order to construct a Bernstein polynomial function based two input dimensional neurofuzzy network which can be trained using linear-in-the-parameter based methods, it is necessary to find a one-to-one mapping: $\Psi : \mathbf{x}(t) \in S \in \Re^2 \rightarrow \{u(\mathbf{x}(t)), v(\mathbf{x}(t))\} \in triangle\{u = 0, v = 0, u + v = 1\}$, or it is necessary to relate system $\mathbf{x}(t)$ with the parameters $\{u(t), v(t), w(t)\}$. Based on this basic idea, a new algorithm is introduced in Section 3.3 which aims to construct the function mapping $\Psi : \mathbf{x} \in \tilde{S} \rightarrow \{u, v\} \in triangle\{u = 0, v = 0, u + v = 1\}$. This new network is based on the concept of barycentric coordinates and the de Casteljau algorithm which has been widely used in computer aided geometric design [12]. For the purpose of exposition of this modelling algorithm, some background about barycentric coordinates and the de Casteljau algorithm is initially described in Section 3.2.

3.2 Barycentric coordinates and de Casteljau algorithm for triangular blossoms

Fundamental to the Bézier-Bernstein curve and surface design are the concept of barycentric coordinates and the de Casteljau algorithm.

Definition 2 *barycentric coordinates.*

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

$$\mathbf{x} = u\mathbf{b}_1 + v\mathbf{b}_2 + w\mathbf{b}_3, \quad with \quad u + v + w = 1, \quad 0 \leq u, v, w \leq 1. \tag{7}$$

Definition 3 *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 $\mathbf{b}_{ijk}$, with $i + j + k = d$. In the cubic case ($d = 3$), as shown in Fig.3, the control net consists of vertices

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

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

$$\mathbf{b}^{(r)}_{i,j,k}(u,v,w) = u\mathbf{b}^{(r-1)}_{i+1,j,k}(u,v,w) + v\mathbf{b}^{(r-1)}_{i,j+1,k}(u,v,w) + w\mathbf{b}^{(r-1)}_{i,j,k+1}(u,v,w) \tag{8}$$

for $r = 1, 2, \cdots, d$, where $i + j + k = d - r$, $\mathbf{b}^{(0)}_{i,j,k} = \mathbf{b}_{i,j,k}$. It can be shown [12] that, by using the recursive formula Eq.(5) and (8), the point on the resultant Bézier triangle patch $\mathbf{b}^{(d)}_{0,0,0}(u,v,w)$ (denoted as $\hat{\mathbf{x}}(u,v,w)$) can be written as

$$\hat{\mathbf{x}}(u,v,w) = \mathbf{b}^{(d)}_{0,0,0}(u,v,w) = \sum_{i+j+k=d} \mathbf{b}_{i,j,k} B^{(d)}_{i,j,k}(u,v,w) \tag{9}$$

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

Definition 4 *Extended barycentric coordinates.*

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

$$\hat{\mathbf{x}}(u,v,w) = \sum_{i+j+k=d} \mathbf{b}_{i,j,k} B^{(d)}_{i,j,k}(u,v,w) \tag{10}$$

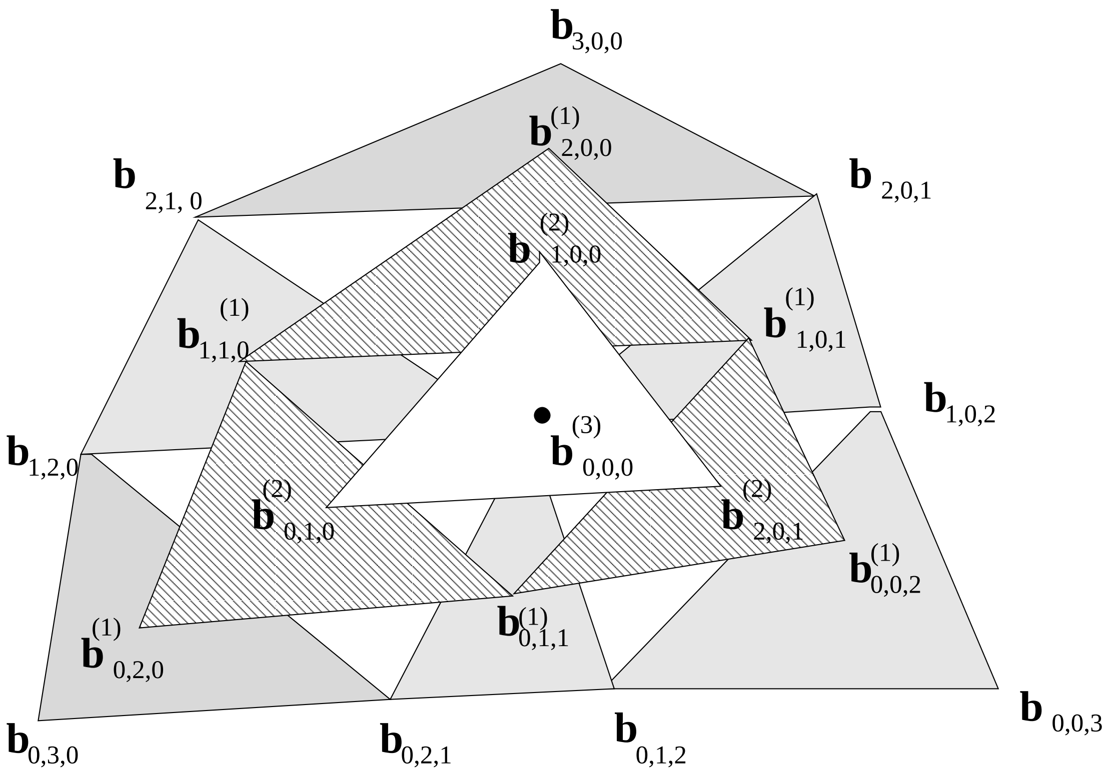


Figure 3: Bézier surface generation using de Casteljau algorithm

It follows from Definition 4 that a given data sample set $\{y(t), \mathbf{x}(t)\}$, $t = 1, 2, \cdots, N$, a Bernstein polynomial basis function based neurofuzzy system, defined by Eq.(6) as shown in Fig.4, can be constructed, if $u(\mathbf{x}(t)), v(\mathbf{x}(t))$ are barycentric coordinates of the input $\mathbf{x}(t)$ with respect to a set of predetermined knots $\mathbf{b}_{i,j,k}$. The endpoint interpolation property of the 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. So 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.

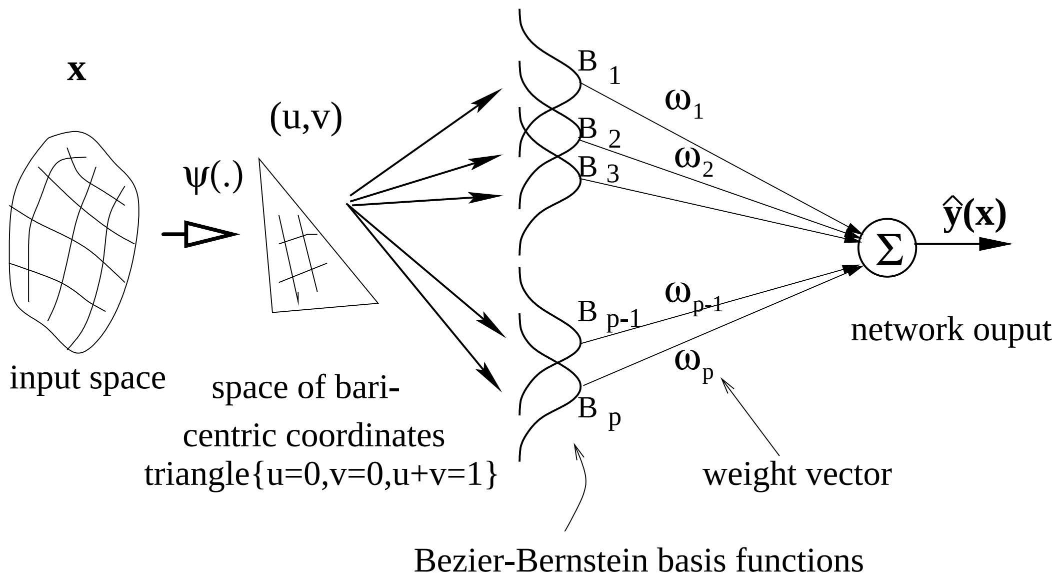


Figure 4: Bézier Bernstein polynomial function based neurofuzzy network

3.3 The new inverse de Casteljau algorithm using backpropagation

The de Casteljau algorithm is a recursive algorithm which realises the mapping from $\{u, v\} \in$ *triangle*$\{u = 0, v = 0, u + v = 1\} \rightarrow \mathbf{x} \in \tilde{S}$, 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 3.1, 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 vector, 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 [4, 13]. The principle of the new inverse de Casteljau algorithm is as follows: given input data $\mathbf{x} \in \tilde{S}$, the proposed inverse de Casteljau algorithm attempts to construct the mapping $\Psi : \mathbf{x} \in \tilde{S} \rightarrow \{u, v\} \in$ *triangle*$\{u = 0, v = 0, u + v = 1\}$ using backpropagation, where $\{u, v, 1 - u - v\}$ is the barycentric coordinates of $\mathbf{x}$

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.3 and the recursive formula Eq.(8). For an input vector $\mathbf{x}(t)$, a method of obtaining a desired $\{u, v, 1-u-v\}$ is by the minimisation the loss function $\|\mathbf{x}(t) - \hat{\mathbf{x}}(u, v)\|$ 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 [10, 14] 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 $\|\mathbf{x}(t) - \hat{\mathbf{x}}(u, v)\|$ can be easily propagated to dynamically adjust the values for the desired $\{u, v, 1-u-v\}$, so as the mapping $\Psi : \mathbf{x} \in \tilde{S} \to \{u, v\} \in triangle\{u = 0, v = 0, u + v = 1\}$ 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.(8) until $r = d$.

$$\begin{aligned}\hat{\mathbf{x}}(u^{(m)}, v^{(m)}, w^{(m)}) &= u^{(m)}\mathbf{b}_{1,0,0}^{(d-1)}(u^{(m)}, v^{(m)}, w^{(m)}) + v^{(m)}\mathbf{b}_{0,1,0}^{(d-1)}(u^{(m)}, v^{(m)}, w^{(m)}) \\ &\quad + w^{(m)}\mathbf{b}_{0,0,1}^{(d-1)}(u^{(m)}, v^{(m)}, w^{(m)}) \end{aligned} \tag{11}$$

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

$$\tilde{\mathbf{x}}(u^{(m)}, v^{(m)}, w^{(m)}) = \hat{\mathbf{x}}(u^{(m)}, v^{(m)}, w^{(m)}) + \alpha[\mathbf{x}(t) - \hat{\mathbf{x}}(u^{(m)}, v^{(m)}, w^{(m)})] \tag{12}$$

where α is a very small positive number representing learning rate $0 < \alpha \ll 1$. Note that α should be sufficiently small such that $\tilde{\mathbf{x}}(u^{(m)}, v^{(m)}, w^{(m)})$ 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 $\tilde{\mathbf{x}}(u^{(m)}, v^{(m)}, w^{(m)})$ 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.(7) (by definition) with w replaced by $1 - u - v$, yields

$$\begin{aligned}\tilde{\mathbf{x}}(u^{(m)}, v^{(m)}, w^{(m)}) &= u\mathbf{b}_{1,0,0}^{(d-1)}(u^{(m)}, v^{(m)}, w^{(m)}) + v\mathbf{b}_{0,1,0}^{(d-1)}(u^{(m)}, v^{(m)}, w^{(m)}) \\ &\quad + (1-u-v)\mathbf{b}_{0,0,1}^{(d-1)}(u^{(m)}, v^{(m)}, w^{(m)}) \end{aligned} \tag{13}$$

The solution $\{u, v\}$ can be solved [10, 13] from Eq.(13) as

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

where

$$\tilde{B}_{(m)} = \left[\mathbf{b}_{1,0,0}^{(d-1)}(u^{(m)}, v^{(m)}, w^{(m)}) - \mathbf{b}_{0,0,1}^{(d-1)}(u^{(m)}, v^{(m)}, w^{(m)}), \mathbf{b}_{0,1,0}^{(d-1)}(u^{(m)}, v^{(m)}, w^{(m)}) - \mathbf{b}_{0,0,1}^{(d-1)}(u^{(m)}, v^{(m)}, w^{(m)})\right] \tag{14}$$

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

3.4 A summary of the construction algorithm

The neurofuzzy modelling using Bézier bernstein polynomial function is summarised as an offline version as below. Given input/output data sets $\{\mathbf{x}(t), y(t)\}_{t=1}^{N}$.

1. For a predetermined polynomial degree d, predetermine the position of $p = \frac{1}{2}(d+1)(d+2)$ knots $\mathbf{b}_{i,j,k}$ ranked to form the polygon S shaped as plotted in Fig.3. 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 (see eg Fig.5) 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.

2. For all t, applying the inverse de Casteljau algorithm to find the $u(t), v(t)$, and $w(t) = 1 - u(t) - v(t)$ from the input vector $\mathbf{x}(t)$.

3. Applying Eq.(4) and (6) to form the basis vector $\mathbf{B}(\mathbf{x}(t)) \in \Re^{p}$ for each data sample $\mathbf{x}(t)$.

4. Least squares is then used, via the minimisation of $\sum_{t=1}^{N}(y(t) - \hat{y}(t))^2$, to obtain the weight vector Ω in Eq.(6).

Remarks:

(i). The two input dimensional Bézier Bernstein polynomial function based neurofuzzy network is different from the B-spline function based neurofuzzy network in that the basis function is not based on the tensor product of univariate basis functions, but by positioning the knots as vertices of a series of triangles covering the input space. This can be interpreted as a Delaunay input space partitioning rather than the conventional lattice structure or orthogonal network. The resultant network is inherently a parsimonious structure as the number of basis functions (and weights) of the resultant network is predetermined through $p = \frac{1}{2}(d+1)(d+2)$, where d is the predetermined polynomial degree (typically d=2 or 3).

(ii). The knot positions can be easily predetermined or adjusted via the input function via a graphical window display, like ‘ ginput’ in MATLAB. The region of the polygon S should be sufficiently larger than the input domain to ensure that the input data range is bounded within the resultant Bézier surface $\tilde{S}$ (see Fig.5).

(iii). Whilst the restriction of this method to 2D input space may seem restrictive, in practice, many high input dimensional modelling problems can be decomposed into a sum of univariate and bivariate submodels via the analysis of variance (or Gabor-Kolmogrov) expansion [8]. For instance, this algorithm could be extended to higher dimensional input space by developing the algorithm to realise the transformation of n-dimensional input vector to n-dimensional barycentric coordinates with respect to $(n+1)$-vertices simplexes [15], but increased complexity in the learning algorithm design is expected due to the high dimensional knots assignment requirement. Future work will be focused on the extension or development of effective algorithms for general modelling problems.

4 Numerical examples

Example 1. Consider a functional approximation problem where

$$y(x_1, x_2) = \frac{\sin(10[(x_1-10)^2 + (x_2-10)^2)]}{10[(x_1-10)^2 + (x_2-10)^2]} \tag{15}$$

is modelled with the input range $x_1 \in [9.75, 10.25]$, $x_2 \in [9.75, 10.25]$. 2500 data sample points (labelled as $t = 1 \sim 2500$) were evenly latticed generated using Eq.(15). Set d=4; The number of the knots (number of basis functions) is therefore $p = \frac{1}{2}(d+1)(d+2) = 15$. The knots $\mathbf{b}_{i,j,k}$'s is an input to a graph window to form the polygon S shaped as Fig.3, for this example, this is plotted in Fig.5, where the input data is also shown. For all data samples, applying the inverse de Casteljau algorithm to find the $u(t), v(t)$, and $w(t) = 1 - u(t) - v(t)$ from the input vector $\mathbf{x}(t)$. The basis vector $\mathbf{B}(\mathbf{x}(t)) \in \Re^p$ for each data sample $\mathbf{x}(t)$ is then formed by applying Eq.(4) and (6). The weights vector Ω are finally obtained by using least squares. The actual system output and the predicted output $\hat{y}(x_1, x_2)$ using the trained B$\acute{e}$zier Bernstein polynomial function based neurofuzzy network with respect to input variable $[x_1, x_2]$ are plotted in Fig.6 (a) and (b), respectively. The mean square error (MSE) of the derived network is $\sigma^2 = 1.2027 \times 10^{-5}$.

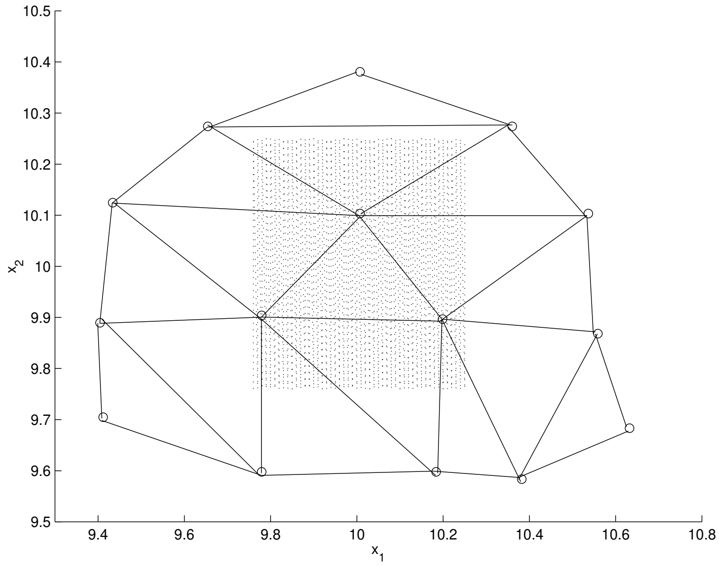


Figure 5: The Delaunay input space partition using predetermined knots.

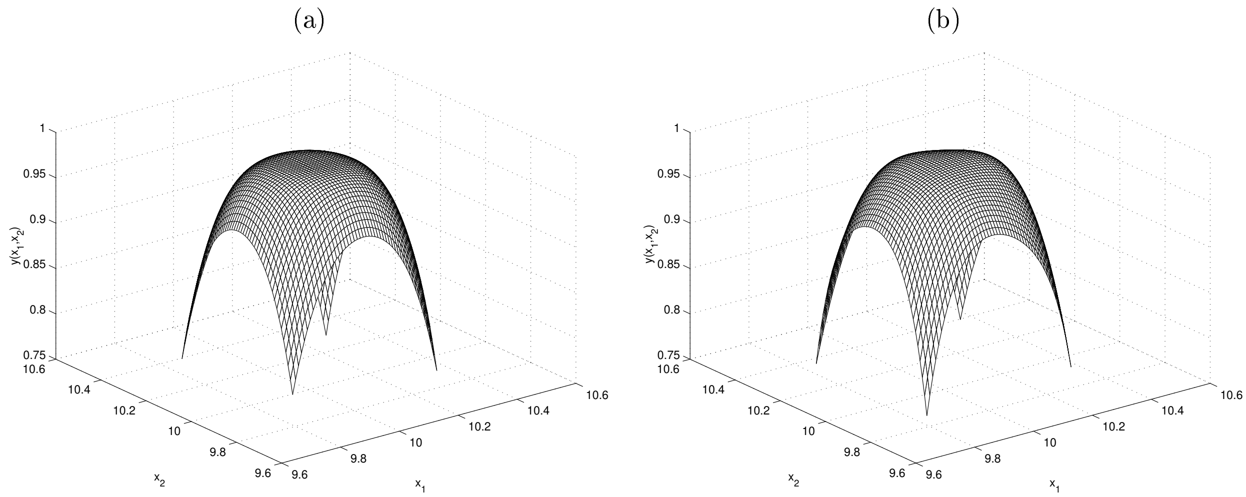


Figure 6: The actual system output and model predictions in example 1; (a) actual system output and (b) model prediction using Bézier Bernstein polynomial function based neurofuzzy network.

Example 2. Consider the benchmark nonlinear autoregressive (NAR) time series [16]:

$$\begin{aligned} y(t) &= (0.8-0.5\exp(-y^2(t-1)))y(t-1)-(0.3+0.9\exp(-y^2(t-1)))y(t-2) \\ &\quad +0.1\sin(3.1415926y(t-1))+e(t) \end{aligned} \tag{16}$$

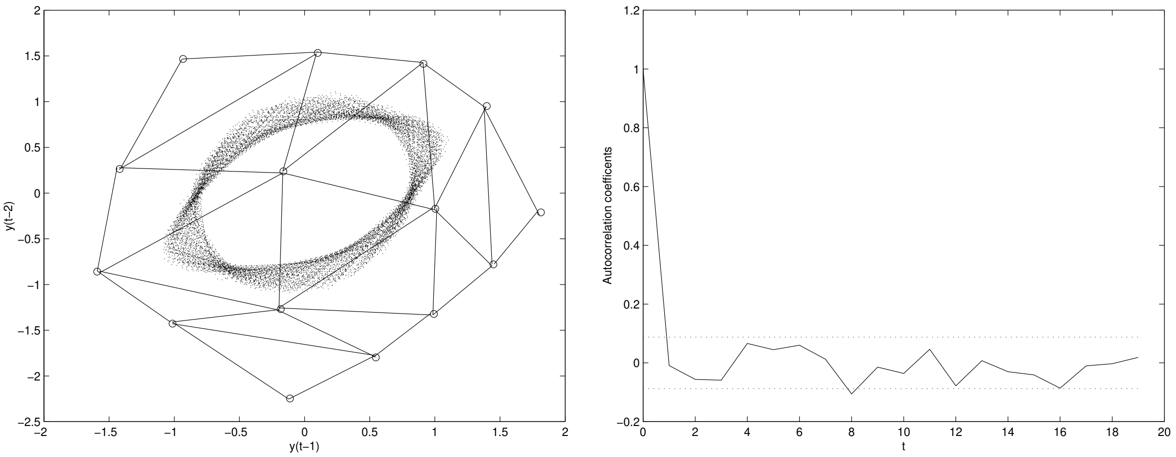
where the noise $e(t)$ is a gaussian white sequence with mean zero and variance 0.02^2. 500 data points were generated as an estimation data set. The input vector is set as $\mathbf{x}(t) = [y(t-1), y(t-2)]^T$. Set d=4; the number of the knots (number of basis functions) is $p = \frac{1}{2}(d+1)(d+2) = 15$. The knots $\mathbf{b}_{i,j,k}$'s was input to a graph window to form the polygon S and plotted in Fig.7a, where the input data $y(t-1)$ with respect to $y(t-2)$ is also shown and covered by the polygon. For all data samples, the inverse de Casteljau algorithm was applied to find the $u(t), v(t)$, and $w(t) = 1-u(t)-v(t)$ from the input vector $\mathbf{x}(t)$. Then the basis vector $\mathbf{B}(\mathbf{x}(t)) \in \Re^p$ for each data sample $\mathbf{x}(t)$ was formed by applying Eq.(4) and (6). The weight vector Ω were finally obtained by using least squares. The mean square error (MSE) of the network was obtained as $\sigma^2 = 0.0207^2$. The autocorelation coefficients of the model residual, plotted in Fig7(b), 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 over $t = 200 \sim 300$ is demonstrated in Fig.7(c) to indicate the system performance. The results from the benchmark example demonstrate the excellent modelling properties of the Bézier Bernstein polynomial function based neurofuzzy system.

5 Conclusions

In this paper, a new neurofuzzy system has been introduced by using Bézier Bernstein polynomial function as basis functions for a two-dimensional input space problem. Based on the idea that

(a)

(b)



(c)

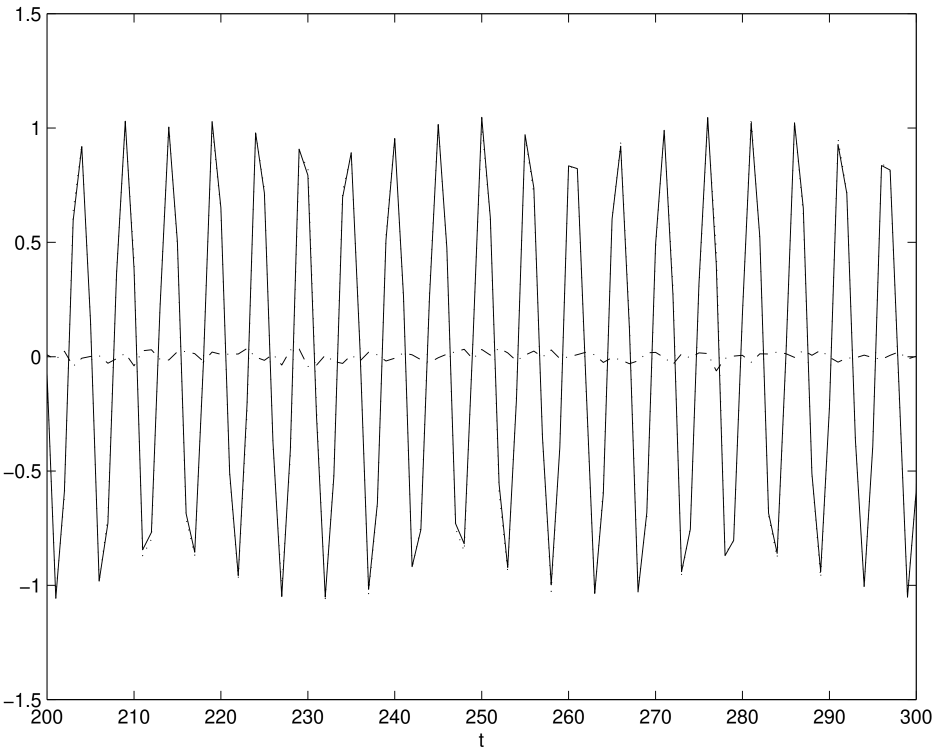


Figure 7: Simulation results in example 2; (a)The Delaunay input space partition using predetermined knots; (b)Model validation test using autocorrelation coefficients of the residual sequence; (c) System actual output (Solid Line), model one-step ahead predictions (Dotted Line) and model residual (Dashdotted Line)

an input vector can be initially mapped into a barycentric coordinates with respect to a set of predetermine knots as vertices of a polygon (or a set of tiled Delaunay triangles) over input space, the basis function in the form of Bézier Bernstein polynomial were then formed based on the barycentric coordinates. The construction algorithm involves the introduction of a new inverse de Casteljau procedure using backpropagation to realise the mapping from the input vector into the barycentric coordinates. The new neurofuzzy network maintains most properties held by B-spline expansion based neurofuzzy, such as nonnegativity of the basis functions, the unity of support with additional adanvantages of structural parsimony and Delaunay input space partition. The effectiveness of the new approach has been demonstrated using numerical examples. This new algorithm is applicable to a wide range of nonlinear signal processing problems.

6 Acknowledgements

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

References

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

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

[3] Jang, J. S. R, Sun, C. T. and Mizutani, E. : '*Neurofuzzy and Softcomputing*' (Prentice Hall, NJ, 1997).

[4] 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.

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

[6] Hong, X., Harris, C. J. and Wilson, P. A.: 'Neurofuzzy network state space identification algorithms using prefiltering for non-linear kalman filtering,' *Proc IEE Part. D. Control Theory and Applications*, 1998, Vol.146, No.2, pp234-240.

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

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

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

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

[11] Brown, M. and Ullrich. T. : 'Comparison of node insertion algorithms for delaunay networks ', *IMACS 2nd Mathmod*, 1997, Vienna, Austria, pp775-780.

[12] Farin, G.: '*Curves and Surfaces for Computer-aided Geometric Design: a Practical Guide*' (Academic Press, Boston, 1993)

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

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

[15] Stoer, J and Witzgall, C: '*Convexity and Optimization in Finite Dimensions I*' (Springer-Verlag, 1970).

[16] Chen, S and Billings, S. A. :'Recursive hybrid algorithm for non-linear system identification using radial basis function networks,' *Int. J. Control*. 1992, Vol.55, No 5, pp1051-1070.