Exploiting the Functional Training Approach in Takagi-Sugeno Neuro-fuzzy Systems

Cristiano L. Cabrita[1], António E. Ruano[2,*], Pedro M. Ferreira[3], and László T. Kóczy[4]

[1] University of Algarve, Portugal
ccabrita@ualg.pt
[2] Centre for Intelligent Systems, IDMEC, IST and the University of Algarve, Portugal
aruano@ualg.pt
[3] Algarve STP – Algarve Science & Technology Park, Portugal
pfrazao@ualg.pt
[4] Faculty of Engineering Sciences, Széchenyi István University, Gyõr, Hungary
koczy@sze.hu

Abstract. When used for function approximation purposes, neural networks and neuro-fuzzy systems belong to a class of models whose parameters can be separated into linear and nonlinear, according to their influence in the model output. This concept of parameter separability can also be applied when the training problem is formulated as the minimization of the integral of the (functional) squared error, over the input domain. Using this approach, the computation of the derivatives involves terms that are dependent only on the model and the input domain, and terms which are the projection of the target function on the basis functions and on their derivatives with respect to the nonlinear parameters, over the input domain. These later terms can be numerically computed with the data.

The use of the functional approach is introduced here for Takagi-Sugeno models. An example shows that this approach obtains better results than the standard, discrete technique, as the performance surface employed is more similar to the one obtained with the function underlying the data. In some cases, as shown in the example, a complete analytical solution can be found.

Keywords: Takagi-Sugeno models, Functional training, Parameter separability.

1 Introduction

One of the crucial problems of fuzzy rule based modelling is how to find an optimal or at least a quasi-optimal rule base for a certain system. If there is a human expert who can tell the rules explicitly or if there is a linguistic description of the modelled

* The second author would like to acknowledge the support of FCT grant SFRH/BSAB/1229/2012.

system available then the fuzzy rule base can be easily constructed. However, in most applications, none is available. In this case some automatic method to determine the fuzzy rule base must be employed. Some of these methods were inspired by the evolutionary processes can be found in nature. Apart from the evolutionary methods, in the area of neural networks there are training algorithms known and these can be applied to fuzzy systems as well. These are useful when there are only numerical data about the problem.

Provided some assumptions are met in the rule base and in the membership functions, a fuzzy system, and in particular, a Takagi-Sugeno (TS) fuzzy system, can be seen as a system whose parameters can be separated, according to their influence on the output, as linear and nonlinear. This is shown in Section 2.

Having in mind that the ultimate goal of modelling is to obtain a "good" approximation of the function behind the data, and not to the data in itself, the modelling problem can be formulated as the minimization of the integral of the (functional) squared error, along the input domain, and not as the usual sum of the square of the errors. This new formulation, denoted as functional approach, is discussed in Section 3.

Section 4 applies this new formulation, employing the separability property, to TS systems. This is achieved using a simple example. In the functional approach, the training algorithms employ two types of terms. i) terms dependent only on the model and the input domain, and independent on the target function, and ii) other terms which are dependent on the model, the target function and the input domain. As the latter are integrals, which have to be approximated, in Section 5 we compare the performance of different numerical integration techniques. Conclusions and future work directions are drawn in Section 6.

2 TS Fuzzy Systems

It is assumed that the model consists of a rule base:

$$R = \{R^{(i)}\} \tag{1}$$

where, for the SISO case:

$$R^{(i)} ::= \text{if}\,(x \text{ is } X^{(i)}) \text{ then } \mathrm{y}^{(i)} = u_0^{(i)} + u_1^{(i)} x; \quad i = 1, \ldots, r \tag{2}$$

r being the number of rules.

Given a crisp input datum x, assuming that for implementing logic connectives such as the conjunction and implication, the *t-norm* used is the algebraic product, the output, y , of this model is:

$$y = \frac{\sum_{i=1}^{r} \mu_{X^{(i)}}(x) y^{(i)}}{\sum_{i=1}^{r} \mu_{X^{(i)}}(x)}, \tag{3}$$

where $\mu_{X^{(i)}}$ is the membership function for the input linguistic term $X^{(i)}$. Assuming that the membership functions form a Ruspini partition, the denominator of (3) is unitary. Therefore, (3) can be written as $y = \boldsymbol{\varphi}^T \mathbf{u}$, with

$$\boldsymbol{\varphi}^T = \left[\mu_{X^{(1)}}(x) \quad \mu_{X^{(2)}}(x) \quad \dots \quad \mu_{X^{(r)}}(x) \quad \mu_{X^{(1)}}(x)x \quad \mu_{X^{(2)}}(x)x \quad \dots \quad \mu_{X^{(r)}}(x)x \right] \quad (4)$$

and

$$\mathbf{u}^T = \left[u_0^{(1)} \quad u_0^{(2)} \quad \dots \quad u_0^{(r)} \quad u_1^{(1)} \quad u_1^{(2)} \quad \dots \quad u_1^{(r)} \right] \quad (5)$$

The generalisation for the multi-input, single output case is straightforward. Assuming k inputs, n_i linguistic terms for the i^{th} input, and denoting by

$$\mathbf{u}_{(i)} = \left[u_i^{(1)} \quad u_i^{(2)} \quad \dots \quad u_i^{(r)} \right] \quad (6)$$

and by

$$\boldsymbol{\mu}(\mathbf{x}) = \left[\mu_{X^{(1)}}(\mathbf{x}) \quad \mu_{X^{(1)}}(\mathbf{x}) \quad \dots \quad \mu_{X^{\left(\prod_{i=1}^{k} n_i\right)}}(\mathbf{x}) \right], \quad (7)$$

eqs (4) and (5) can be transformed to:

$$\boldsymbol{\varphi}^T = \left[\boldsymbol{\mu}(\mathbf{x}) \quad \boldsymbol{\mu}(\mathbf{x})\mathbf{I}\mathbf{x}_1 \quad \dots \quad \boldsymbol{\mu}(\mathbf{x})\mathbf{I}\mathbf{x}_k \right] \quad (8)$$

and

$$\mathbf{u}^T = \left[\mathbf{u}_{(0)} \quad \mathbf{u}_{(1)} \quad \dots \quad \mathbf{u}_{(k)} \right] \quad (9)$$

In this work trapezoidal membership functions will be used. The j^{th} (out of p_i) interval on the i^{th} input is defined as:

$$I_{i,j} = \begin{cases} [\lambda_{i,j-1} \; \lambda_{i,j}[& j = 1, \dots, p_i - 1 \\ [\lambda_{i,j-1} \; \lambda_{i,j}] & j = p_i \end{cases} \quad (10)$$

The j^{th} membership in the i^{th} dimension is defined as:

$$\mu_{i,j}(x_i) = \begin{cases} 1, & x_i \in I_{i,2j-1} \\ \dfrac{x_i - \lambda_{i,2j-3}}{\lambda_{i,2j-2} - \lambda_{i,2j-3}}, & x_i \in I_{i,2j-2} \\ \dfrac{\lambda_{i,2j} - x_i}{\lambda_{i,2j} - \lambda_{i,2j-1}}, & x_i \in I_{i,2j} \\ 0, & otherwise \end{cases} \quad (11)$$

As it can be seen, from the previous equations, the u parameters appear linearly in the output, and the λ parameters nonlinearly.

In this work, the model parameters are estimated with data, a process known in the neural network jargon as training. Considering m patterns, $\mathbf{y}$ is the output of the model, a vector with m elements, and $\mathbf{X}$ is the input matrix, with dimensions $m*n$ (n is the number of inputs). It is assumed that the model has n_u linear parameters and n_v nonlinear parameters[1]. $\mathbf{\Gamma}$ is the matrix of the basis functions (dimensions $m * n_u$), so that the output can be seen as:

$$\mathbf{y}(\mathbf{X},\mathbf{v},\mathbf{u}) = \mathbf{\Gamma}(\mathbf{X},\mathbf{v})\mathbf{u} \tag{12}$$

The training criterion normally employed is

$$\Omega_d(\mathbf{X},\mathbf{v},\mathbf{u},\mathbf{t}) = \frac{\left\|\mathbf{t}-\mathbf{\Gamma}(\mathbf{X},\mathbf{v})\mathbf{u}\right\|_2^2}{2} = \frac{\left\|\mathbf{e}(\mathbf{X},\mathbf{v},\mathbf{u},\mathbf{t})\right\|_2^2}{2}, \tag{13}$$

where $\mathbf{t}$ is the target vector, $\mathbf{e}$ is the error vector and $\| \ \|_2$ is the Euclidean norm. As the model parameters can be decomposed into linear and nonlinear ones, we can determine the optimal value of the linear parameters with respect to (wrt) the nonlinear ones:

$$\hat{\mathbf{u}}_d(\mathbf{X},\mathbf{v},\mathbf{t}) = \left(\mathbf{\Gamma}^T\mathbf{\Gamma}\right)^{-1}\mathbf{\Gamma}^T\mathbf{t} = \mathbf{\Gamma}^+\mathbf{t} \tag{14}$$

where the symbol '+' denotes a pseudo-inverse. We can incorporate this value in the usual criterion and introduce a new criterion (independent on the linear parameters):

$$\Psi_d(\mathbf{X},\mathbf{v},\mathbf{t}) = \frac{\left\|\mathbf{t}-\mathbf{\Gamma}\hat{\mathbf{u}}_d\right\|_2^2}{2} = \frac{\left\|\mathbf{t}-\mathbf{\Gamma}\mathbf{\Gamma}^+\mathbf{t}\right\|_2^2}{2} = \frac{\left\|\mathbf{P}_{\Gamma_\perp}\mathbf{t}\right\|_2^2}{2} \tag{15}$$

Where $\mathbf{P}_{\Gamma(\mathbf{x},\mathbf{v})_\perp} = \mathbf{I}-\mathbf{\Gamma}\mathbf{\Gamma}^+$ is the projector on the orthogonal complement of the space spanned by the basis functions.

The concept of separability of parameters in general nonlinear least-squares problems has been introduced in [1]. A good review on the applications of this concept can be found in [2]. In the context of neural networks, it has been introduced in [3], for Multilayer Perceptrons, in [4] for Radial Basis Functions and in [5] for B-splines and neuro-fuzzy systems.

3 The Functional Approach

Let us assume that the function to approximate was known, and let it be denoted by $t(x)$. In this case, (12) and (13) can be replaced by:

[1] From now on, we shall denote, for compatibility with previous works, the nonlinear parameters as v.

$$\mathrm{y}(\mathrm{x},\mathbf{v},\mathbf{u}) = \boldsymbol{\varphi}^{\mathrm{T}}(\mathrm{x},\mathbf{v})\mathbf{u} \tag{16}$$

and

$$\begin{aligned}\boldsymbol{\Omega}_f(\mathrm{x}_{\min},\mathrm{x}_{MAX},t,\mathbf{u},\mathbf{v}) &= \frac{\int_{\mathrm{x}_{\min}}^{\mathrm{x}_{MAX}} \left(t(\mathrm{x}) - \boldsymbol{\varphi}^T(\mathrm{x},\mathbf{v})\mathbf{u}\right)^2 d\mathrm{x}}{2} = \\ \frac{\int_{x_{\min}}^{\mathrm{x}_{MAX}} \left(t - \mathbf{u}^T\boldsymbol{\varphi}\right)^2 dx}{2} &= \frac{\int_{\mathrm{x}_{\min}}^{\mathrm{x}_{MAX}} e^2(\mathrm{x},t,\mathbf{v},\mathbf{u})d\mathrm{x}}{2}\end{aligned} \tag{17}$$

Notice that in (16) x is now a multi-dimensional real variable:

$$\mathrm{x} = x_1,\ldots,x_{i,}\ldots,x_n \tag{18}$$

and that in (17)

$$\int_{\mathrm{x}_{\min}}^{\mathrm{x}_{MAX}} f(\mathrm{x},.)d\mathrm{x} = \int_{x_{1\min}}^{x_{1MAX}} \cdots \int_{x_{k\min}}^{x_{kMAX}} \cdots \int_{x_{n\min}}^{x_{nMAX}} f(x_1,\ldots,x_k,\ldots,x_n,.)dx_1\ldots dx_k,\ldots,dx_n \tag{19}$$

t and e are also real functions. $\boldsymbol{\varphi}$ is a vector of basis functions, and not a matrix as in the discrete case. The Jacobian is given as:

$$\begin{aligned}\mathbf{j}(\mathrm{x},\mathbf{v},\mathbf{u}) &= \frac{\partial \mathbf{y}(\mathrm{x},\mathbf{v},\mathbf{u})}{\partial\left[\mathbf{u}^T \mid \mathbf{v}^T\right]} = \left[\mathbf{j}_{\mathbf{u}}(\mathrm{x},\mathbf{v}) \mid \mathbf{j}_{\mathrm{v}}(\mathrm{x},\mathbf{v},\mathbf{u})\right] = \\ &= \left[\boldsymbol{\varphi}^T(\mathrm{x},\mathbf{v}) \mid \left[\frac{\partial\boldsymbol{\varphi}(\mathrm{x},\mathbf{v})}{\partial\mathbf{v}^T}\right]^T \mathbf{u}\right]\end{aligned} \tag{20}$$

and the gradient as:

$$\begin{aligned}\mathbf{g}_{\Omega_{f_{\mathbf{u}}}}(\mathrm{x}_{\min},\mathrm{x}_{MAX},t,\mathbf{v}) &= \frac{\partial\Omega_f}{\partial\mathbf{u}^T} = -\int_{x_{\min}}^{x_{MAX}} \mathbf{j}_{\mathbf{u}}^T e d\mathrm{x} \\ \mathbf{g}_{\Omega_{f_{\mathbf{v}}}}(t,\mathbf{v},\mathrm{x}_{\min},\mathrm{x}_{MAX}) &= \frac{\partial\Omega_f}{\partial\mathbf{v}^T} = -\int_{\mathrm{x}_{\min}}^{\mathrm{x}_{MAX}} \mathbf{j}_{\mathbf{v}}^T e d\mathrm{x}\end{aligned} \tag{21}$$

Equating the first eq.of (21) to **0**, we have:

$$\begin{aligned}&-\int_{\mathrm{x}_{\min}}^{\mathrm{x}_{MAX}} \boldsymbol{\varphi}\left(t - \boldsymbol{\varphi}^T\mathbf{u}_f\right)d\mathrm{x} = 0 \Leftrightarrow \left[\int_{\mathrm{x}_{\min}}^{\mathrm{x}_{MAX}} \boldsymbol{\varphi}\boldsymbol{\varphi}^T d\mathrm{x}\right]\hat{\mathbf{u}}_f = \int_{\mathrm{x}_{\min}}^{\mathrm{x}_{MAX}} \boldsymbol{\varphi}t d\mathrm{x} \Leftrightarrow \\ &\Leftrightarrow \hat{\mathbf{u}}_f(t,\mathbf{v},\mathrm{x}_{\min},\mathrm{x}_{MAX}) = \left[\int_{\mathrm{x}_{\min}}^{\mathrm{x}_{MAX}} \boldsymbol{\varphi}\boldsymbol{\varphi}^T d\mathrm{x}\right]^{-1} \int_{\mathrm{x}_{\min}}^{\mathrm{x}_{MAX}} \boldsymbol{\varphi}t d\mathrm{x}\end{aligned} \tag{22}$$

As in the discrete case, we can incorporate this value in criterion (17), and have a new criterion, independent of the linear parameters:

$$\Psi_f\left(\mathrm{x},\mathbf{v},\mathrm{x}_{\min},\mathrm{x}_{MAX}\right)=\frac{\int\limits_{\mathrm{x}_{\min}}^{\mathrm{x}_{MAX}}\left(t-\boldsymbol{\varphi}^T\left[\int\limits_{\mathrm{x}_{\min}}^{\mathrm{x}_{MAX}}\boldsymbol{\varphi}\boldsymbol{\varphi}^T d\mathrm{x}\right]^{-1}\int\limits_{\mathrm{x}_{\min}}^{\mathrm{x}_{MAX}}\boldsymbol{\varphi}t d\mathrm{x}\right)^2 d\mathrm{x}}{2} \tag{23}$$

3.1 Training Algorithms

If we want to apply the Back-Propagation (BP) algorithm [6] to train the model:

$$\begin{aligned}\mathbf{v}[k+1]&=\mathbf{v}[k]+\mathbf{s}[k]\\ \mathbf{s}[k]&=-\eta\mathbf{g}_{\Psi_f}[k]\end{aligned}, \tag{24}$$

We need to compute the gradient of criterion (23). It can be proved (please see [7] for details) that it can be obtained as:

$$\mathbf{g}_{\Psi_f}=-\int\limits_{\mathrm{x}_{\min}}^{\mathrm{x}_{MAX}} t\frac{\partial\boldsymbol{\varphi}^T}{\partial\mathbf{v}^T}d\mathrm{x}\hat{\mathbf{u}}_f+\hat{\mathbf{u}}_f^T\frac{\partial\int\limits_{\mathrm{x}_{\min}}^{\mathrm{x}_{MAX}}\boldsymbol{\varphi}\boldsymbol{\varphi}^T d\mathrm{x}}{2\partial\mathbf{v}^T}\hat{\mathbf{u}}_f \tag{25}$$

Therefore, to use this methodology, we need to have available:

$$\left[\int\limits_{\mathrm{x}_{\min}}^{\mathrm{x}_{MAX}}\boldsymbol{\varphi}\boldsymbol{\varphi}^T d\mathrm{x}\right]^{-1} \tag{26}$$

$$\left[\int\limits_{\mathrm{x}_{\min}}^{\mathrm{x}_{MAX}}\boldsymbol{\varphi}\boldsymbol{\varphi}^T d\mathrm{x}\right]^{-1}\frac{\partial\int\limits_{\mathrm{x}_{\min}}^{\mathrm{x}_{MAX}}\boldsymbol{\varphi}\boldsymbol{\varphi}^T d\mathrm{x}}{2\partial\mathbf{v}^T}\left[\int\limits_{\mathrm{x}_{\min}}^{\mathrm{x}_{MAX}}\boldsymbol{\varphi}\boldsymbol{\varphi}^T d\mathrm{x}\right]^{-1} \tag{27}$$

which are <u>independent on the function to approximate</u>, and can be obtained analytically for the model at hand.

The only terms involving the function to approximate are:

$$\int\limits_{\mathrm{x}_{\min}}^{\mathrm{x}_{MAX}}\boldsymbol{\varphi}t d\mathrm{x} \tag{28}$$

$$\int\limits_{\mathrm{x}_{\min}}^{\mathrm{x}_{MAX}} t\frac{\partial\boldsymbol{\varphi}^T}{\partial\mathbf{v}^T}d\mathrm{x} \tag{29}$$

In a practical application the underlying function is not known (otherwise it should be used). The integrals (28) and (29) can be numerically approximated using the training data.

4 An Example

We are going to use a very simple example to illustrate how the functional approach can be applied to TS systems. Our aim will be to approximate the function $t(x)=\frac{Sin(10x)}{x}$, over the domain $x\in[-1,1]$ with a TS system, with only 2 rules. For this domain, $\int_{\mathrm{x}_{\min}}^{\mathrm{x}_{MAX}} \varphi\varphi^T d\mathrm{x}$ can be given as:

$$\begin{bmatrix} \frac{3+2\lambda_1+\lambda_2}{3} & \frac{\lambda_2-\lambda_1}{6} & \frac{-6+3\lambda_1^2+2\lambda_1\lambda_2+\lambda_2^2}{12} & \frac{\lambda_2^2-\lambda_1^2}{12} \\ & \frac{3-\lambda_1-2\lambda_2}{3} & \frac{\lambda_2^2-\lambda_1^2}{12} & \frac{6-\lambda_1^2-2\lambda_1\lambda_2-3\lambda_2^2}{12} \\ & & \frac{10+4\lambda_1^3+3\lambda_1^2\lambda_2+2\lambda_1\lambda_2^2+\lambda_2^3}{30} & \frac{-3\lambda_1^3-\lambda_1^2\lambda_2+\lambda_1\lambda_2^2+3\lambda_2^3}{60} \\ & & & \frac{10-\lambda_1^3-2\lambda_1^2\lambda_2-3\lambda_1\lambda_2^2-4\lambda_2^3}{30} \end{bmatrix} \tag{30}$$

Notice that, as the matrix is symmetric, only the upper-diagonal elements are shown. Notice also that, due to lack of space, we could not shown (26), the inverse of (30). For the same reason, we can not show (27). Instead, we will illustrate $\frac{\partial \int_{\mathrm{x}_{\min}}^{\mathrm{x}_{MAX}} \varphi\varphi^T d\mathrm{x}}{\partial \mathbf{v}^T}$.

$$\frac{\partial \int_{\mathrm{x}_{\min}}^{\mathrm{x}_{MAX}} \varphi\varphi^T d\mathrm{x}}{\partial \lambda_1} = \begin{bmatrix} \frac{2}{3} & -\frac{1}{6} & \frac{3\lambda_1+\lambda_2}{6} & -\frac{\lambda_1}{6} \\ & -\frac{1}{3} & -\frac{\lambda_1}{6} & -\frac{\lambda_1+\lambda_2}{6} \\ & & \frac{6\lambda_1^2-3\lambda_1\lambda_2+\lambda_2^2}{15} & \frac{-9\lambda_1^2-2\lambda_1\lambda_2+\lambda_2^2}{60} \\ & & & \frac{-3\lambda_1^2-4\lambda_1\lambda_2-3\lambda_2^2}{30} \end{bmatrix} \tag{31}$$

$$\frac{\partial \int_{\mathrm{x}_{\min}}^{\mathrm{x}_{MAX}} \varphi\varphi^T d\mathrm{x}}{\partial \lambda_2} = \begin{bmatrix} \frac{1}{3} & \frac{1}{6} & \frac{\lambda_1+\lambda_2}{6} & \frac{\lambda_2}{6} \\ & -\frac{2}{3} & \frac{\lambda_2}{6} & -\frac{\lambda_1+3\lambda_2}{6} \\ & & \frac{3\lambda_1^2+4\lambda_1\lambda_2+3\lambda_2^2}{30} & \frac{-\lambda_1^2+2\lambda_1\lambda_2+9\lambda_2^2}{60} \\ & & & \frac{-\lambda_1^2-3\lambda_1\lambda_2-6\lambda_2^2}{15} \end{bmatrix} \tag{32}$$

We can train the TS model analytically. Equation (24) will be used, with $\eta=0.02$, for 5 different starting points. The training will be stopped whether when a maximum number of 200 iterations is reached, or when the following criteria are simultaneously satisfied:

$$\begin{aligned} &\Psi[k-1]-\Psi[k]<\beta[k] \\ &\left\|\mathbf{v}[k-1]-\mathbf{v}[k]\right\|_2<\sqrt{\tau}\left(1+\left\|\mathbf{v}[k]\right\|_2\right), \\ &\left\|\mathbf{g}[k]\right\|_2\leq\sqrt[3]{\tau}\left(1+\left|\Psi[k]\right|\right) \end{aligned} \tag{33}$$

where

$$\beta[k]=\tau\left(1+\left|\Psi[k]\right|\right) \tag{34}$$

and τ is a measure of the desired number of correct digits in the objective function. This is a criterion typically used in unconstrained optimization [8]. In all examples, $\tau=10^{-6}$ is used.

As an order relation must be preserved for the nonlinear parameters (i.e. $\lambda_2>\lambda_1$), the standard BP algorithm needs to be modified. In order to maintain the same search direction, the update vector ($s[k]=-\eta\mathbf{g}[k]$), when a violation is detected ($\boldsymbol{\lambda}_{i+1}[k]<\boldsymbol{\lambda}_i[k]$), is reduced of a factor g so that the position of the $(i+1)^{\text{th}}$ knot is located half-way between the previous distance of the two corresponding knots:

$$g=\frac{\boldsymbol{\lambda}_{i+1}[k-1]-\boldsymbol{\lambda}_i[k-1]}{2\left(\mathbf{s}_{i+1}[k]-\mathbf{s}_i[k]\right)} \tag{35}$$

Therefore, when there is a violation of the order relation, the learning rate for the BP algorithm is updated at that iteration.

Figure 1 shows the analytical performance surface, together with the results of five different trainings.

As it can be seen, even with a very simple example, several local minima exist. The surface is symmetric, and there are two global minima, located at [-0.357,-0.116] and [0.116, 0.357], with a value of 4.4.for criterion (23).

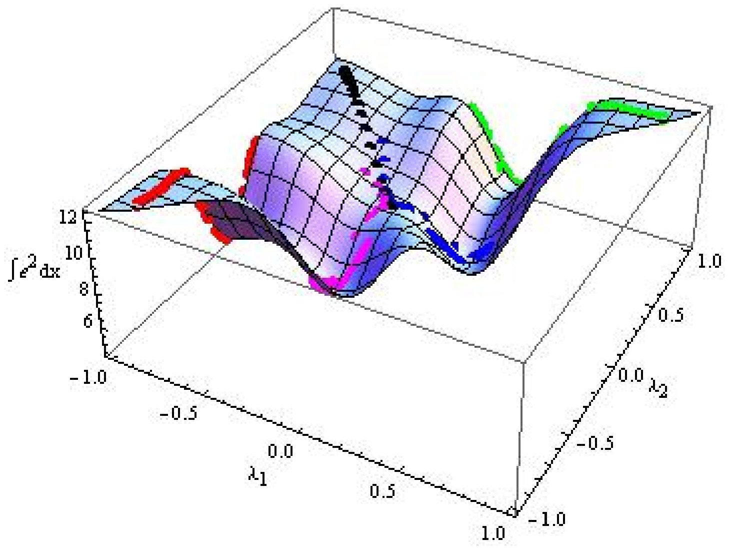

Fig. 1. Analytical performance surface

The flowing table illustrates the different trainings.

Table 1. Analytical trainings

$\mathbf{v}_a[1]$	[-0.90,-0.85]	[-0.90,0.90]	[0.85,0.90]	[-0.5,0.6]	[-0.5,0.4]
n	55	26	55	200	200
$\Psi_a[n]$	5.22	7.01	5.22	4.82	6.15
$\mathbf{V}_a[n]$	[-1.0,-0.087]	[-0.28,0.28]	[0.088,1.0]	[0.182,0.395]	[-0.28,0.06]

The first line shows the starting points, and *n* is the number of iterations taken by the training. Notice that no training attains the global minima. This is due to the setting of the maximum number of iterations (200). The two last evolutions would attain the global minima with a few more iterations.

If the function is not known, but only data is available, a same methodology can be employed, this time approximating the integrals (28) and (29). In a first case, we shall use a Gaussian quadrature algorithm. In this technique, the integral is approximated as:

$$\int_{x_{\min}}^{x_{\max}} f(x)dx \approx \sum_{i=1}^{m} f(x_i)\,w_i \tag{36}$$

Notice that the abscissas x_i are chosen by the method. The following figure and table illustrate the results obtained with *m*=25.

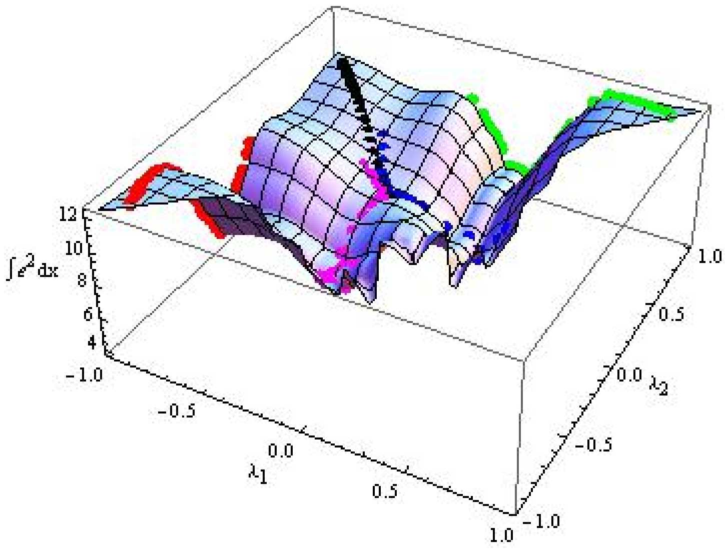

Fig. 2. Gaussian quadrature performance surface

The performance surface is very similar with the analytical one. The different evolutions go to the same minima, as it can be also seen in Table 2.

Table 2. Gaussian quadrature trainings

vG[1]	[-0.90,-0.85]	[-0.90,0.90]	[0.85,0.90]	[-0.5,0.6]	[-0.5,0.4]
n	102	39	55	29	32
$\Psi_G[n]$	5.02	7.01	4.99	4.27	4.15
vG[n]	[-1.0,-0.118]	[-0.29,0.29]	[0.121,1.0]	[0.12,0.41]	[-0.31,-0.16]
$\Psi_a\|_{v_a[n]}$	5.34	7.01	5.36	4.49	4.49

The last line in the table shows the value of the analytical criterion, evaluated for the last parameters values obtained by the training performed with the data, using Gaussian quadrature. As it can be seen, comparing with Table 1, similar values were obtained.

4.1 Different Integration Techniques

As pointed above, although the last training was performed with data, the training inputs were obtained with Gaussian quadrature. In practice, we do not usually have control over the training data. For this reason, we conducted afterwards a series of trainings, where the input data was randomly generated. Different integration techniques were also compared. In all experiments, 25 input patterns were used, In order to compare the different methods, we fixed the extrema to -1 and +1. Therefore, only 23 of the 25 values were randomly generated.

Besides comparing the different integration techniques, we also compared the functional approach with the standard, discrete training, minimizing (15). The following figure and table show the results obtained.

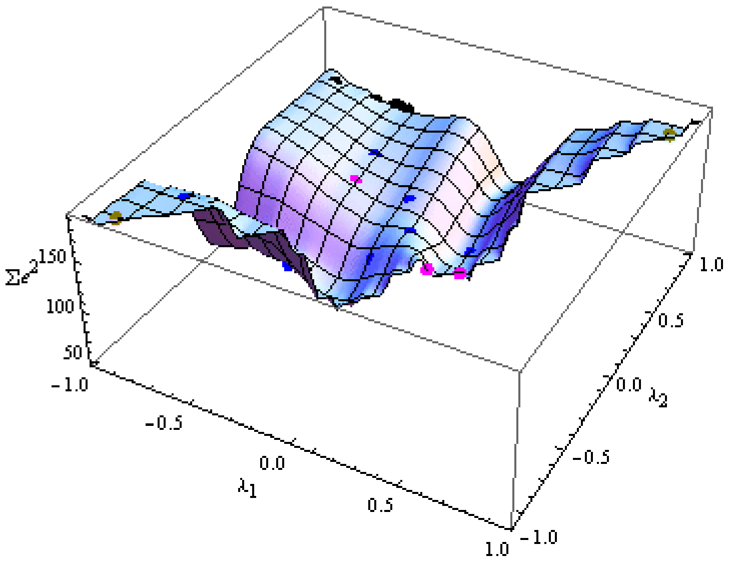

Fig. 3. Discrete performance surface

Table 3. Discrete trainings

$v_d[1]$	[-0.90,-0.85]	[-0.90,0.90]	[0.85,0.90]	[-0.5,0.6]	[-0.5,0.4]
n	2	200	2	3	4
$\Psi_d[n]$	181.31	112-81	172.82	63.43	72.54
$v_d[n]$	[-0.90,-0.85]	[-0.58,1]	[0.85,0.90]	[0.12,0.39]	[0.17,0.18]
$\Psi_a\|_{v_G[n]}$	12.21	9.00	12.21	4.65	5.35

Comparing these results with the analytical results, and the Gaussian quadrature ones, we can conclude that for 3 of the 5 initial points, very different (worse) results were obtained.

Subsequently, with the same random data, the functional methodology was experimented, with different integration techniques.

The first one used was the forward integration. The integral is approximated as:

$$\int_{x_{\min}}^{x_{\max}} f(x)dx \approx \sum_{i=1}^{m-1} f(x_i)(x_{i+1} - x_i) \tag{37}$$

The results obtained are shown in the fig. and table below.

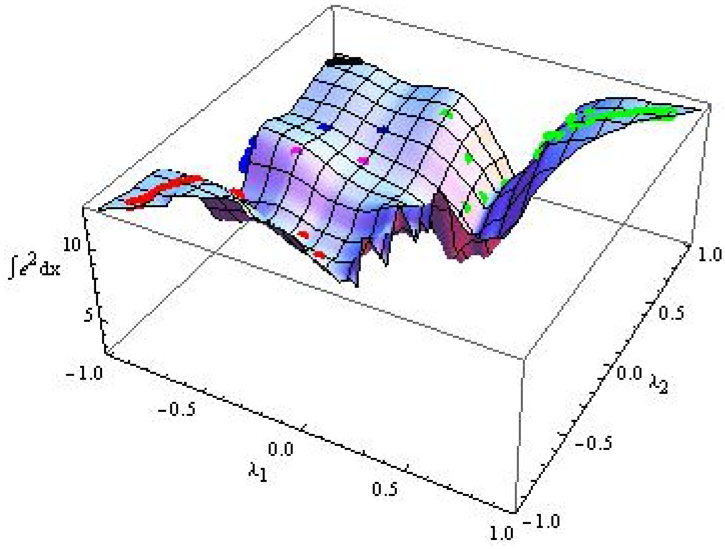

Fig. 4. Forward integration

Table 4. Forward integration trainings

$v_f[1]$	[-0.90,-0.85]	[-0.90,0.90]	[0.85,0.90]	[-0.5,0.6]	[-0.5,0.4]
n	15	7	42	46	3
$\Psi_f[n]$	4.82	9.54	4.69	7.02	5.21
$v_f[n]$	[-0.43,-0.14]	[-0.85,0.95]	[0.08,0.81]	[-0.99,0.12]	[-0.87,-0.04]
$\Psi_a\|_{v_f[n]}$	4.70	9.75	5.71	7.85	5.67

The results obtained are better than the discrete training for 2 out of the 5 initial points. The 2nd and the 5th trainings are similar and only in the 4th evolution the discrete version is clearly better.

The next integration method tested was the backward approximation:

$$\int_{x_{\min}}^{x_{\max}} f(x)dx \approx \sum_{i=1}^{m-1} f(x_{i+1})(x_{i+1}-x_i) \tag{38}$$

The results obtained are summarized in Fig. 5 and Table 5.

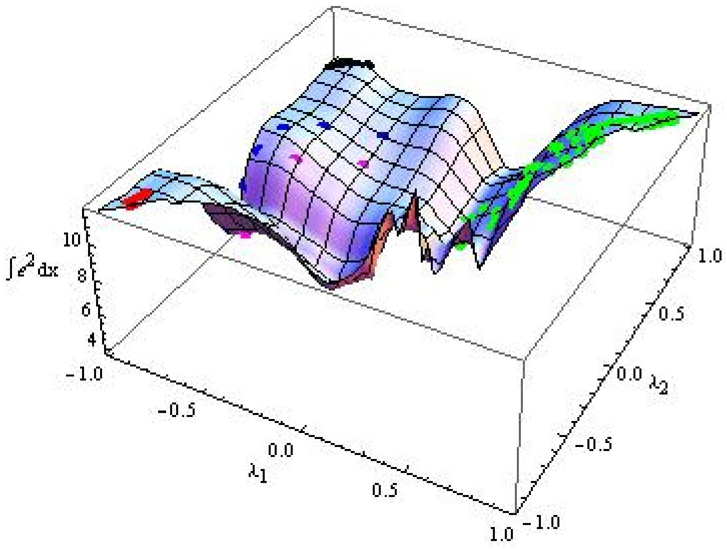

Fig. 5. Backward integration

Table 5. Backward integration trainings

$v_b[1]$	[-0.90,-0.85]	[-0.90,0.90]	[0.85,0.90]	[-0.5,0.6]	[-0.5,0.4]
n	7	9	59	7	3
$\Psi_b[n]$	11.60	9.16	5.14	4.70	5.09
$v_b[n]$	[-0.88,-0.75]	[-0.80,0.98]	[0.08,0.32]	[-0.99,-0.05]	[-0.87,-0.09]
$\Psi_a\|_{v_b[n]}$	12.16	9.64	4.58	5.32	5.51

Subsequently, a trapezoidal technique was experimented. In the trapezoidal, or Tustin method, the approximation employed is:

$$\int_{x_{\min}}^{x_{\max}} f(x)dx \approx \frac{1}{2}\sum_{i=1}^{m-1}\left(f(x_{i+1})+f(x_i)\right)(x_{i+1}-x_i) \tag{39}$$

The results obtained are summarized below.

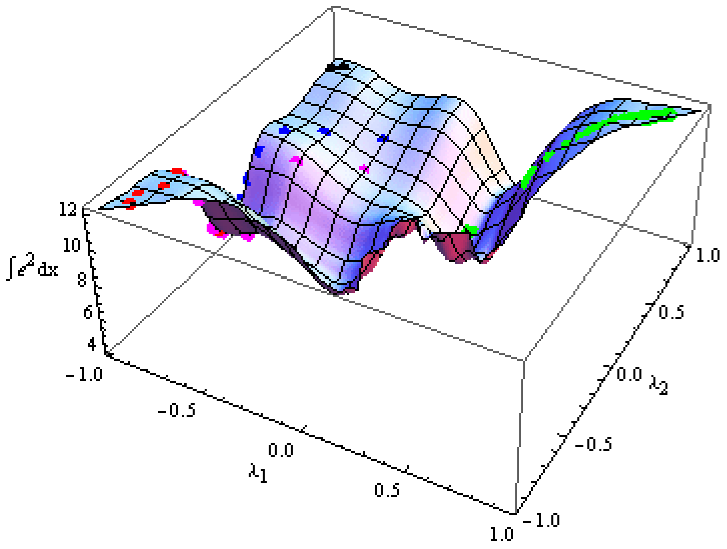

Fig. 6. Trapezoidal integration

Table 6. Trapzoidal integration trainings

$v_t[1]$	[-0.90,-0.85]	[-0.90,0.90]	[0.85,0.90]	[-0.5,0.6]	[-0.5,0.4]
n	6	2	24	22	12
$\Psi_t[n]$	5.59	9.43	4.74	5.11	5.80
$v_t[n]$	[-0.98,-0.16]	[-0.95,0.86]	[0.03,0.49]	[-0.99,-0.02]	[-0.99,0.18]
$\Psi_a\big\|_{v_t[n]}$	5.81	9.77	5.08	5.60	6.06

The results are more consistent across the different initial points. The last training technique experimented was a polynomial interpolation algorithm, available in Mathematica©. The order 3 was experimented.

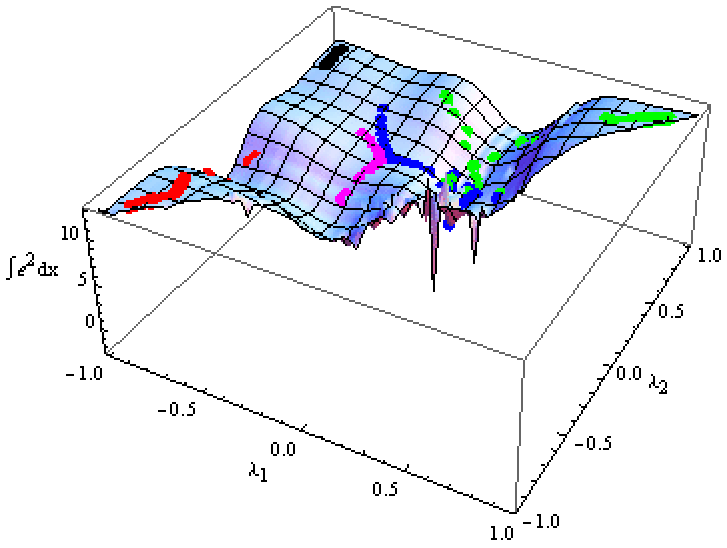

Fig. 7. 3rd order polynomial integration

Table 7. 3rd order polynomial integration trainings

$v_p[1]$	[-0.90,-0.85]	[-0.90,0.90]	[0.85,0.90]	[-0.5,0.6]	[-0.5,0.4]
n	35	12	45	35	15
$\Psi_p[n]$	6.46	9.77	4.31	3.88	5.66
$v_p[n]$	[-0.91,0.03]	[-0.91,0.74]	[0.071,0.36]	[0.08,0.19]	[-0.39,0.03]
$\Psi_a\big\vert_{v_p[n]}$	6.34	9.51	4.65	5.59	5.32

Finally, Table 8 compares the different methods tested, together with the analytical and the discrete methods.

On bold we highlight the best results obtained with the random data. The discrete technique obtains the best values, for 3 out of the 5 starting points. In the other 2 points very bad results were obtained, compared with the other functional approaches

In a final test, 20 experiments were performed for each method, for each one of the 5 starting points. The results are presented in terms of mean (μ) and standard deviation (σ). Table 9 presents these statistics for the optimal values of the criteria optimized, and Table 10 shows the same statistics, but translated to the analytical criterion, evaluated at the final points for each different optimization.

Table 8. Comparison of different methods

	[-0.90,-0.85]	[-0.90,0.90]	[0.85,0.90]	[-0.5,0.6]	[-0.5,0.4]
$\Psi_a[n]$	5.22	7.01	5.22	4.82	6.15
$\Psi_a\big\|_{v_G[n]}$	5.34	7.01	5.36	4.49	4.49
$\Psi_a\big\|_{v_d[n]}$	12.21	**9.00**	12.21	**4.65**	**5.35**
$\Psi_a\big\|_{v_f[n]}$	**4.70**	9.75	5.71	7.85	5.67
$\Psi_a\big\|_{v_b[n]}$	12.16	9.64	**4.58**	5.32	5.51
$\Psi_a\big\|_{v_t[n]}$	5.81	9.77	5.08	5.60	6.06
$\Psi_a\big\|_{v_f[n]}$	6.34	9.51	4.65	5.59	5.32

Table 9. Mean and standard deviation of the criteria optimized

	[-0.90,-0.85]	[-0.90,0.90]	[0.85,0.90]	[-0.5,0.6]	[-0.5,0.4]
$\mu_d \pm \sigma_d$	119±56	71±28	136±37	56±13	54±14
$\mu_f \pm \sigma_f$	5.6±2.6	9.5±1.5	7.8±4.3	5.2±2.1	4.2±1.9
$\mu_b \pm \sigma_b$	9.7±3.0	7.7±3.7	3.5±2.4	4.1±3.1	4.0±2.5
$\mu_t \pm \sigma_t$	10.9±3.2	9.0±1.5	4.4±1.1	5.0±1.1	4.9±1.5
$\mu_p \pm \sigma_p$	10.2±3.5	7.5±2.7	9.1±3.3	3.1±2.3	3.2±2.5

Table 10. Mean and standard deviation of the analytical criteria, for the optima found for each criterion

	[-0.90,-0.85]	[-0.90,0.90]	[0.85,0.90]	[-0.5,0.6]	[-0.5,0.4]
$\mu_a \pm \sigma_a\big\|_{v_d[n]}$	11.8±1.7	**7.5±2.2**	12.2±0.03	5.7±1.3	5.5±0.84
$\mu_a \pm \sigma_a\big\|_{v_f[n]}$	**6.0±2.1**	9.3±1.0	8.3±3.3	6.0±1.2	5.4±0.2
$\mu_a \pm \sigma_a\big\|_{v_b[n]}$	11.6±1.2	9.7±0.1	5.4±0.5	5.9±1.1	5.8±0.6
$\mu_a \pm \sigma_a\big\|_{v_t[n]}$	11.5±2.1	9.7±0.1	**4.9±0.4**	5.8±0.9	5.4±0.2
$\mu_a \pm \sigma_a\big\|_{v_p[n]}$	9.4±3.5	7.7±2.4	8.8±3.5	**5.3±1.0**	**5.2±1.6**

The discrete criterion obtains the better results only for 1 out of the 5 initial points, and even for this case, with results very similar to the polynomial version of the functional approach. Comparing the different integration schemes, we can conclude that, for this particular example, the polynomial integration technique is the best method, as it obtains 2 best results overall, and 3 out of 5, comparing only the functional approaches.

5 Conclusions

In this paper we have shown how to apply the functional methodology for TS fuzzy models, with trapezoidal membership functions, that form a Ruspini partition of the input space.

The functional approach offers important advantages over the standard, discrete approach. We have not mentioned that it achieves important savings in computational time, if the training data is large (for results please see [9]). We have shown, previously and in this paper that, in most of the cases, it obtains a better approximation over the whole input domain than the discrete training algorithms. This is due to the fact that the functional performance surface is closer to the analytical performance surface than the discrete one. Another important point of this methodology is that it allows determining the local minima and the performance of a specified model, if the function generating the data is known.

As the performance of the functional approach is strongly related with the numerical integral approximation used, we have compared different alternatives and concluded that, for the example considered, a polynomial interpolation algorithm of order 3 should be chosen. The results, however, are not conclusive and further work on this topic should be done, and extended for the multi-dimensional case.

References

1. Golub, G.H., Pereyra, V.: Differentiation of pseudo-inverses and nonlinear least-squares problems whose varables separate. Siam Journal on Numerical Analysis 10(2), 413–432 (1973)
2. Golub, G., Pereyra, V.: Separable nonlinear least squares: the variable projection method and its applications. Inverse Problems 19(2), R1–R26 (2003)
3. Ruano, A.E.B., Jones, D.I., Fleming, P.J.: A New Formulation of the Learning Problem for a Neural Network Controller. In: 30th IEEE Conference on Decision and Control, Brighton, UK, December 11-13, pp. 865–866 (1991)
4. Ferreira, P.M., Ruano, A.E.: Exploiting the separability of linear and nonlinear parameters in radial basis function networks. In: Adaptive Systems for Signal Processing, Communications, and Control Symposium (AS-SPCC), Lake Louise, Canada, pp. 321–326 (October 2000)
5. Ruano, A.E., Cabrita, C., Oliveira, J.V., Koczy, L.T.: Supervised training algorithms for B-Spline neural networks and neuro-fuzzy systems. Int. J. Syst. Sci. 33(8), 689–711 (2002), doi:10.1080/00207720210155062

6. Werbos, P.: Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences. PhD, Harvard University (1974)
7. Ruano, A.E., Cabrita, C.L., Ferreira, P.M.: Towards a More Analytical Training of Neural Networks and Neuro-Fuzzy Systems. In: IEEE 7th International Symposium on Intelligent Signal Processing (WISP 2011), Floriana, Malta, September 19-21 (2011)
8. Gill, P.E., Murray, W., Wright, M.H.: Practical Optimization. Academic Press, Inc. (1981)
9. Cabrita, C.L., Ruano, A.E., Ferreira, P.M., Koczy, L.T.: Extending the functional training approach for B-Splines. In: 2012 IEEE International Joint Conference on Neural Networks (IJCNN), Brisbane, Australia, June 10-15, pp. 2702–2709. IEEE (2012)