Int. J. Fuzzy Syst. (2020) 22(1):22–34
https://doi.org/10.1007/s40815-019-00747-2

Training High-Order Takagi-Sugeno Fuzzy Systems Using Batch Least Squares and Particle Swarm Optimization

Krzysztof Wiktorowicz[1] · **Tomasz Krzeszowski**[1]

Received: 26 April 2019 / Revised: 17 September 2019 / Accepted: 24 September 2019 / Published online: 14 November 2019

Abstract This paper proposes two methods for training Takagi–Sugeno (T-S) fuzzy systems using batch least squares (BLS) and particle swarm optimization (PSO). The T-S system is considered with triangular and Gaussian membership functions in the antecedents and higher-order polynomials in the consequents of fuzzy rules. In the first method, the BLS determines the polynomials in a system in which the fuzzy sets are known. In the second method, the PSO algorithm determines the fuzzy sets, whereas the BLS determines the polynomials. In this paper, the ridge regression is used to stabilize the solution when the problem is close to the singularity. Thanks to this, the proposed methods can be applied when the number of observations is less than the number of predictors. Moreover, the leave-one-out cross-validation is used to avoid overfitting and this way to choose the structure of a fuzzy model. A method of obtaining piecewise linear regression by means of the zero-order T-S system is also presented.

Keywords Fuzzy systems · Least squares approximation · Particle swarm optimization

✉ Krzysztof Wiktorowicz
kwiktor@prz.edu.pl

Tomasz Krzeszowski
tkrzeszo@prz.edu.pl

[1] Faculty of Electrical and Computer Engineering, Rzeszow University of Technology, Al. Powstancow Warszawy 12, 35-959 Rzeszow, Poland

1 Introduction

One of the most commonly used models in artificial intelligence is the Takagi–Sugeno (T-S) [25] fuzzy system. Building T-S systems consists of two main tasks: structure identification and parameter estimation. The structure identification is mainly related with determining the number of fuzzy rules. The parameter estimation is related with determining the parameters of fuzzy sets and the coefficients of regression functions in the consequence part. These tasks can be achieved by various optimization techniques such as least squares [24, 26, 32], evolutionary algorithms [5, 8, 32] or particle swarm optimization. Particle swarm optimization (PSO) is a stochastic optimization method that was developed by Kennedy and Eberhart [9, 12]. The PSO is mainly inspired by the social behavior of organisms that live and interact within large groups, for example, swarms of bees or flocks of birds. The usefulness of the PSO in solving a wide range of optimization problems has been repeatedly confirmed [1, 13, 14, 22].

A group of papers concerns the algorithms in which the number of rules is constant. In [19], a multi-swarm cooperative PSO was proposed, where the population consists of one master swarm and several slave swarms. The algorithm was used to automatically design the fuzzy identifier and fuzzy controller in dynamical systems. The Takagi-Sugeno fuzzy systems with Gaussian membership functions in the antecedents and linear functions in the consequents of fuzzy rules were used. The approach described in [14] uses a hybrid-learning method for T-S fuzzy systems. This method combines the PSO and recursive least squares (RLSE) to obtain fuzzy approximation. The PSO is used to train the antecedent part of the T-S system (the parameters of Gaussian membership functions), and the consequent part (the coefficients of the linearly parameterized functions) is trained by the RLSE method. The paper [15] presents a self-learning complex neuro-fuzzy system that uses Gaussian complex fuzzy sets. The knowledge base of this system consists of the T-S fuzzy rules with complex fuzzy sets in the antecedent part and linear models in the consequent part. The PSO algorithm is used to train the antecedent parameters, and the recursive least squares method is used to train the consequent parameters. The paper [7] proposes a heterogeneous multi-swarm PSO to identify the first-order T-S fuzzy system. The T-S model uses linear regression models in several subspaces to describe a nonlinear system. The antecedent parameters and the consequent parameters of T-S models are encoded as particles, and they are obtained simultaneously in the training process. In [6], a model that replaces the function of inputs in the conclusion part of the TSK system with a wavelet neural network is proposed. The parameters of the model are obtained by a hybrid-learning method based on an improved PSO and gradient descent algorithm. An immune coevolution PSO with multi-strategy is proposed for T-S fuzzy systems in [17]. In the described method, the population consists of one elite subswarm and several normal subswarms. The parameters identified by the algorithm included the centers and widths of the membership functions (the antecedent parameters) and the coefficients of local models (the consequent parameters).

Another approach assumes the use of clustering algorithms to determine the structure of fuzzy systems. A two-phase swarm intelligence algorithm for zero-order Takagi–Sugeno–Kang (TSK) fuzzy systems was developed in [11]. In the first phase, the algorithm learns the fuzzy system structure and parameters by clustering-aided ant colony optimization. This phase is used to locate a good initial fuzzy rule base for further learning. The aim of the second phase is to optimize all of the free parameters in the fuzzy system using PSO. The paper [16] presents a learning method for TSK-type neuro-fuzzy networks that uses the self-clustering algorithm to partition the input space to create fuzzy rules. The method is based on the symbiotic evolution scheme in an immune algorithm and PSO to improve the mutation mechanism. The parameters of fuzzy sets and the parameters of the consequent part are optimized by this method. In [30], an approach for building a type-2 neural-fuzzy system from a set of input-output data was proposed. A fuzzy clustering method is used for partitioning the dataset into clusters. Then, a type-2 fuzzy TSK rule is derived from each cluster. A fuzzy neural network is constructed accordingly, and the parameters are refined using PSO and least-squares estimation. An approach for function approximation using robust fuzzy regression clustering algorithm and PSO was described in [31]. At first, the fuzzy regression is used to construct a first-order TSK fuzzy model. Next, PSO is applied for tuning the parameters of the obtained fuzzy model. An algorithm for fuzzy c-regression model clustering was presented in [21]. The method combines the advantages of the clustering algorithm and PSO algorithm. The fuzzy model used in this method is the T-S fuzzy system with local linear models. The orthogonal least squares method is applied to estimate the consequent parameters of the fuzzy rules. The paper [20] presents a learning algorithm based on a hierarchical PSO (HPSO) to train the parameters of a T-S fuzzy model. First, an unsupervised fuzzy clustering algorithm is applied for partitioning the data and identifying the antecedent parameters of the fuzzy system. Next, a self-adaptive HPSO algorithm is used to obtain the consequent parameters of the fuzzy system. An identification method for the Takagi–Sugeno fuzzy model was developed in [26]. First, the fuzzy c-means clustering is used to determine the optimal rule number. Next, the initial membership function and the consequent parameters are obtained by the PSO algorithm. A fuzzy c-regression model and orthogonal least squares methods are applied to obtain the final parameters.

In part of the papers, the PSO algorithm is used to determine the number of rules. An algorithm for automatically extracting T-S fuzzy models from data using PSO was described in [33]. The authors designed an improved version of the original PSO called the cooperative random learning PSO, where several subswarms search the space and exchange information. In their method, each fuzzy rule has a label which is used to decide whether the rule is included in the inference process or not. The antecedent parameters (the parameters of Gaussian functions), the consequent parameters (the coefficients of linear functions) and the rule labels are encoded in a particle. In [29], a self-constructing least-Wilcoxon generalized radial basis function neural-fuzzy system was proposed. A PSO is used for generating the antecedent parameters, and for the consequent parameters, instead of the traditional least squares estimation, the least-Wilcoxon norm is employed. A method that uses a hierarchical cluster-based multi-species PSO for building the TSK fuzzy system is presented in [18]. The method is used for a spatial analysis problem, where the area under study is divided into several subzones. For each subzone, the zero-order or the first-order fuzzy system is extracted from the set of patterns. In [4], a T-S model based on PSO and kernel ridge regression was presented. The developed method works in two main steps. In the first step, the clustering based on the PSO algorithm separates the input data into clusters and obtains the antecedent parameters. In the second step, the consequent parameters are calculated using a kernel ridge regression. The adaptive chaos PSO algorithm (ACPSO) for identification of the T-S fuzzy model parameters using weighted recursive least squares was proposed in [24]. This approach is a compromise between the chaos and adaptive PSO algorithms. The clustering criterion function was used as the objective function of ACPSO. The ACPSO is used to optimize the parameters of the model, then the obtained parameters are used to initialize the fuzzy c-regression model.

From the reviewed papers, it can be seen that only zero- or first-order systems are used with Gaussian functions. In this article, we propose the use of higher-order polynomials in the consequent part of rules, which can give greater flexibility in the selection of system parameters, e.g. when determining the number of rules. In addition to Gaussian sets, triangular sets will also be used that do not require determining the value of the exponential function but use simple linear functions. It is also worth noting that in the mentioned papers, the number of rules is fixed [6, 7, 14, 15, 17, 19], which raises the question of its selection, either selected by means of clustering [11, 16, 20, 21, 26, 30, 31] or a PSO method [4, 18, 24, 29, 33]. These methods require an additional mechanism in the learning algorithm to enable their selection. Hence, we suggest choosing the number of rules using a cross-validation method that is simple in implementation. Moreover, in papers in which regressions are used to determine system parameters, regularized regressions are not applied [21, 24, 26, 29, 30] (except for the work [4]). We propose here the use of ridge regression that allows us to consider ill-defined problems, e.g. those in which the number of observations is small.

Summarizing, the main contributions of this paper can be stated as:

- the use of the batch least squares (BLS) and PSO methods for high-order Takagi–Sugeno systems with triangular and Gaussian membership functions,
- the use of the leave-one-out cross-validation error to choose the structure (the number of rules) of fuzzy models,
- the use of a regularization method in the form of the ridge regression for ill-conditioned problems.

Moreover, in the article, it is shown how to realize a piecewise linear regression applying the zero-order T-S fuzzy system with triangular membership functions.

The structure of this article is as follows: Section 2 contains the description of the T-S fuzzy system with higher-order polynomials in the consequent part of fuzzy rules. Section 3 presents the training methods that use BLS and PSO. In this section, we also show how to obtain a piecewise linear regression using a zero-order T-S system. Section 4 contains the procedure for fuzzy models design with the BLS and PSO methods. The experimental results are presented in Sect. 5. Finally, the conclusions are given in Sect. 6.

2 High-Order Takagi–Sugeno Fuzzy System

We consider a Takagi–Sugeno fuzzy system [25] with one input x and one output y. The T-S system is described by r fuzzy inference rules with polynomial functions in the consequent part given by

$$R_j\text{: IF } x \in A_j(x) \text{ THEN } y = w_{mj}x^m + \cdots + w_{1j}x + w_{0j} \tag{1}$$

where $A_j(x)$ denotes a fuzzy set, $j = 1, 2, \ldots, r$, $m \geq 0$ is the polynomial degree, $w_{kj} \in \mathbb{R}$, and $k = 0, 1, \ldots, m$. The following definition extends the concept of the T-S system in which zero or first-order polynomials are used.

Definition 1 The T-S fuzzy system with rules (1) is called:

– *zero-order* if the consequent functions are crisp constants, i.e., $y = w_{0j}$ [25],
– *first-order* if the consequent functions are linear, i.e., $y = w_{1j}x + w_{0j}$ [25],
– *high-order* if the consequent functions are nonlinear, that occurs for $m \geq 2$.

In this paper, we apply triangular membership functions with the peaks placed on points p_j (as shown in Fig. 1) defined as

$$\begin{aligned} A_j(x) &= \mathrm{triang}(x; p_{j-1}, p_j, p_{j+1}) \\ &= \max\left(0, \min\left(\frac{x - p_{j-1}}{p_j - p_{j-1}}, \frac{p_{j+1} - x}{p_{j+1} - p_j}\right)\right) \end{aligned} \tag{2}$$

where x is the argument of the function, p_{j-1}, p_j, p_{j+1} are the parameters, and $p_{j-1} < p_j < p_{j+1}$. In (2), for the set $A_1(x)$ we take $p_{j-1} = p_1 - \epsilon$, and for the set $A_r(x)$ we take $p_{j+1} = p_r + \epsilon$, where $\epsilon > 0$. The peaks p_j are written as the vector $\boldsymbol{p} = [p_j] = [p_1, \ldots, p_r]$. The fuzzy sets A_j can be unevenly spaced in the interval $\mathbb{X} = [p_1, p_r]$, but it is

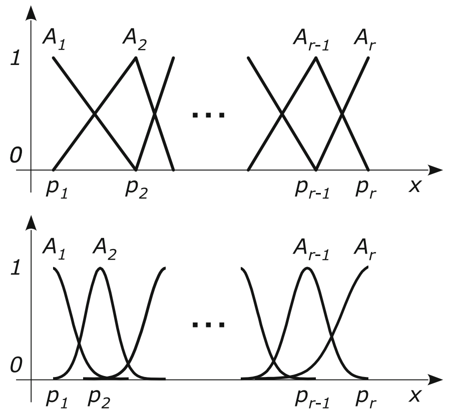

Fig. 1 Triangular (above) and Gaussian (below) membership functions

assumed that the sum of the membership grades for any argument x is equal to unity. We also consider Gaussian membership functions (as shown in Fig. 1) given by

$$A_j(x) = \operatorname{gauss}(x; p_j, \sigma_j) = \exp\left(-\frac{1}{2}\left(\frac{x-p_j}{\sigma_j}\right)^2\right) \tag{3}$$

where p_j is the peak of the function and $\sigma_j > 0$ is its width. The widths σ_j are written as the vector $\boldsymbol{\sigma} = [\sigma_j] = [\sigma_1, \ldots, \sigma_r]$.

The output of the T-S system is calculated from all rules outputs as

$$y = \frac{\sum_{j=1}^{r} A_j(x)\left(w_{mj}x^m + \cdots + w_{1j}x + w_{0j}\right)}{\sum_{j=1}^{r} A_j(x)} \tag{4}$$

Introducing the notion of a fuzzy basis function, the formula (4) can be written in a compact form.

Definition 2 [27] The *fuzzy basis function* (FBF) for the jth rule is the function $\xi_j(x)$ given by

$$\xi_j(x) = \frac{A_j(x)}{\sum_{j=1}^{r} A_j(x)} \tag{5}$$

Taking into account the definition of the FBF, the output of the system can be written as

$$y = \sum_{j=1}^{r} \xi_j(x)\left(w_{mj}x^m + \cdots + w_{1j}x + w_{0j}\right) = \sum_{j=1}^{r} w_{mj}\xi_j(x)x^m + \cdots + w_{1j}\xi_j(x)x + w_{0j}\xi_j(x) \tag{6}$$

Because in (6) the FBFs are multiplied by x^k, we define a modified fuzzy basis function.

Definition 3 The *modified FBF* (MFBF) for the jth rule is the function $h_{kj}(x)$ given by

$$h_{kj}(x) = \xi_j(x)x^k \tag{7}$$

where $k = 0, 1, \ldots, m$.

Applying the definition of the MFBF, we obtain a linearly parameterized function

$$y = \sum_{j=1}^{r} w_{mj}h_{mj}(x) + \cdots + w_{1j}h_{1j}(x) + w_{0j}h_{0j}(x) \tag{8}$$

Introducing vectors $\boldsymbol{h}_j(x)$ and $\boldsymbol{w}_j$ defined by

$$\boldsymbol{h}_j(x) = [h_{mj}(x), \ldots, h_{1j}(x), h_{0j}(x)] \tag{9}$$

$$\boldsymbol{w}_j = [w_{mj}, \ldots, w_{1j}, w_{0j}]^T \tag{10}$$

where $\dim(\boldsymbol{h}_j) = \dim(\boldsymbol{w}_j) = m + 1$, the output of the T-S system can be written as

$$y = [\boldsymbol{h}_1(x), \ldots, \boldsymbol{h}_r(x)] \begin{bmatrix} \boldsymbol{w}_1 \\ \vdots \\ \boldsymbol{w}_r \end{bmatrix} = \boldsymbol{h}(x)\boldsymbol{w} \tag{11}$$

where

$$\boldsymbol{h}(x) = [\boldsymbol{h}_1(x), \ldots, \boldsymbol{h}_r(x)] \tag{12}$$

$$\boldsymbol{w} = [\boldsymbol{w}_1, \ldots, \boldsymbol{w}_r]^T \tag{13}$$

In (11), the MFBFs are the elements of the vector $\boldsymbol{h}(x)$, and $\boldsymbol{w}$ is the vector of the model parameters with the dimension equal to $r(m+1)$.

3 Training Methods for Takagi–Sugeno Systems

In this section, we propose two methods for training high-order T-S systems based on the observed data. In the training process, two techniques of machine learning are used:

– BLS for obtaining polynomial coefficients in the consequent part of the rules,
– PSO for obtaining parameters of fuzzy sets in the antecedent part of the rules.

In the first method (BLS), the fuzzy sets parameters are chosen manually, whereas in the second method (PSO-BLS), they are obtained using a PSO algorithm.

3.1 BLS Regression

We assume that we have a set of training data containing n observations in the form of pairs (x_i, y_i), where $i = 1, \ldots, n$. These observations are written as vectors $\boldsymbol{x} = [x_i]^T = [x_1, \ldots, x_n]^T$ and $\boldsymbol{y} = [y_i]^T = [y_1, \ldots, y_n]^T$. To build a regression model, we introduce the regression matrix

$$\underset{n\times r(m+1)}{\boldsymbol{X}} = \begin{bmatrix} \boldsymbol{h}_1(x_1), \ldots, \boldsymbol{h}_r(x_1) \\ \boldsymbol{h}_1(x_2), \ldots, \boldsymbol{h}_r(x_2) \\ \vdots \\ \boldsymbol{h}_1(x_n), \ldots, \boldsymbol{h}_r(x_n) \end{bmatrix} \tag{14}$$

where $\boldsymbol{h}_j(x_i)$ is given by (9). The cost function to be minimized is the sum of squared errors (SSE) defined by

$$\mathrm{SSE} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} (y_i - \boldsymbol{h}(x_i)\boldsymbol{w})^2 \tag{15}$$

where $\hat{y}_i = \boldsymbol{h}(x_i)\boldsymbol{w}$ is the estimated output of the system (see (11)) for the i th observation. The vector $\boldsymbol{w}$ contains the system parameters to be determined. The number of these parameters is equal to $p = \dim(\boldsymbol{w}) = r(m+1)$.

The optimal solution for the regression problem is given by [3]

$$\boldsymbol{w} = \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\boldsymbol{y} \tag{16}$$

where $\boldsymbol{y} = [y_1, \ldots, y_n]^T$. This is a BLS because we compute the model parameters directly from all data contained in $\boldsymbol{X}$ and $\boldsymbol{y}$.

In the case of an ill-conditioned problem, that is when the matrix $\boldsymbol{X}^T\boldsymbol{X}$ is close to singular, we propose to use ridge regression [10]:

$$\boldsymbol{w} = \left(\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I}\right)^{-1}\boldsymbol{X}^T\boldsymbol{y} \tag{17}$$

where $\lambda > 0$ is a regularization parameter, and $\boldsymbol{I}$ is the identity matrix. In this case, the cost function has the form of the penalized sum of squared errors (PSSE):

$$\mathrm{PSSE} = \sum_{i=1}^{n}\left(y_i - \boldsymbol{h}(x_i)\boldsymbol{w}\right)^2 + \lambda\boldsymbol{w}^T\boldsymbol{w} \tag{18}$$

The BLS is a one-pass regression method, and therefore it is very fast.

3.2 Zero-Order T-S System as a Piecewise Linear Regression

In [28], it was shown that the zero-order T-S fuzzy system with triangular membership functions defined as in Section 2 realizes a piecewise linear function in the interval $[p_1, p_r]$. This fact can be used in our problem to build a piecewise linear or segmented regression. The graph of this regression passes through the points $(p_1, w_{01}), (p_2, w_{02}), \ldots, (p_r, p_{0r})$, and the breakpoints are in $p_2, \ldots, p_{r-1}$.

In particular, using the BLS regression for the T-S system with two rules ($r = 2$), we obtain a linear regression in the form of $y = ax + b$.

In the interval $[p_1, p_2]$ we define two fuzzy sets with the peaks in p_1, p_2:

$$\begin{aligned} A_1(x) &= \mathrm{triang}(x; p_1 - \epsilon, p_1, p_2) \\ A_2(x) &= \mathrm{triang}(x; p_1, p_2, p_2 + \epsilon) \end{aligned} \tag{19}$$

where $\epsilon > 0$. The fuzzy inference rules for a zero-order T-S system have the form

$$\begin{aligned} &R_1\text{:IF x} \in \mathrm{A}_1 \text{ THEN y} = \mathrm{w}_{01} \\ &R_2\text{:IF x} \in \mathrm{A}_2 \text{ THEN y} = \mathrm{w}_{02} \end{aligned} \tag{20}$$

where w_{01} and w_{02} are real numbers. For this system, the regression matrix (14) is of the form

$$\underset{n\times 2}{\boldsymbol{X}} = \begin{bmatrix} A_1(x_1), A_2(x_1) \\ A_1(x_2), A_2(x_2) \\ \vdots \\ A_1(x_n), A_2(x_n) \end{bmatrix} \tag{21}$$

where

$$A_1(x) = \frac{p_2 - x}{p_2 - p_1}, \quad A_2(x) = \frac{x - p_1}{p_2 - p_1} \tag{22}$$

In this case, the solution (16) has the form $\boldsymbol{w} = [w_{01}, w_{02}]^T$. This way, we obtain a linear function $y = ax + b$, where

$$a = \frac{w_{02} - w_{01}}{p_2 - p_1}, \quad b = \frac{p_2 w_{01} - p_1 w_{02}}{p_2 - p_1} \tag{23}$$

The regression line passes through the points (p_1, w_{01}) and (p_2, w_{02}).

3.3 PSO Algorithm

The particle swarm model consists of a group of particles that are dispersed in the d-dimensional search space. During the optimization process, the particles explore this space and exchange information to find the optimal solution. Each kth particle has its position $\boldsymbol{x}_k$, velocity $\boldsymbol{v}_k$, and best local position $\mathbf{pbest}_k$. Moreover, each particle has access to the best global position $\mathbf{gbest}$ found by the swarm. Selection of the best global position and the best position for kth particle is based on the objective function.

In this paper, we use a PSO algorithm [9], where the velocity and position of particles are computed from the equations

$$\boldsymbol{v}_k^{l+1} = \chi[\boldsymbol{v}_k^l + c_1\boldsymbol{r}_1(\mathbf{pbest}_k^l - \boldsymbol{x}_k^l) + c_2\boldsymbol{r}_2(\mathbf{gbest}^l - \boldsymbol{x}_k^l)] \tag{24}$$

$$\boldsymbol{x}_k^{l+1} = \boldsymbol{x}_k^l + \boldsymbol{v}_k^{l+1} \tag{25}$$

where $\boldsymbol{r}_1$, $\boldsymbol{r}_2$ are vectors with uniformly distributed random numbers in the interval [0, 1] and l is the current iteration number. The parameters c_1, c_2 are positive constants equal to 2.05 and χ is equal to 0.7298.

In (24), three components can be distinguished—inertia, cognitive, and social. The first component models the particle's tendency to continue moving in the same direction. The second component attracts particles towards their personal best positions. The third component moves particles towards the best position found earlier by any particle.

3.4 PSO-BLS Algorithm

In the proposed hybrid method (as shown in Fig. 2), the PSO algorithm is used to construct fuzzy sets for the input of the model. Each particle in the swarm contains hypothetical parameters of these sets for which the consequents of fuzzy rules are determined by the BLS method. In subsequent iterations of the PSO algorithm, particles move in the search space generating a new set of candidate solutions. The best solution is obtained by minimizing the objective function defined as the root of the mean square error [14, 23]:

$$\mathrm{RMSE_T} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{26}$$

where $y_i - \hat{y}_i$ is an error between the observation and the output of the fuzzy model.

In the case of triangular fuzzy sets (2), a particle contains the peaks of membership functions and has the following structure:

$$\begin{array}{c|c|c|c|c} p_2 & p_3 & \ldots & p_{r-2} & p_{r-1} \end{array} \tag{27}$$

In our proposition, we assume that the peaks p_1 and p_r are known and not optimized by the PSO; hence the number of parameters is equal to $r-2$.

In the case of Gaussian fuzzy sets (3), a particle contains the peaks as in (27), and additionally the widths of all membership functions:

$$\begin{array}{c|c|c|c|c|c|c|c|c|c} p_2 & p_3 & \ldots & p_{r-2} & p_{r-1} & \sigma_1 & \sigma_2 & \ldots & \sigma_{r-2} & \sigma_r \end{array} \tag{28}$$

The number of parameters to be optimized is equal to $2r-2$.

3.5 The Criterion of Model Quality

In this paper, the best T-S model is chosen using the *leave-one-out cross-validation* (LOOCV) method [2], in which the number of tests is equal to the number of observations, and one pair creates a testing dataset. The LOOCV method is used because in our case, the dataset is small. For larger datasets, the *k*-fold cross-validation may be used.

The quality of a model in the LOOCV method is evaluated by the root of the mean square error defined as

$$\mathrm{RMSE_{CV}} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_{-i})^2} \tag{29}$$

where $\hat{y}_{-i}$ denotes the output of the model obtained in the ith step of the validation process. The subscript '$-i$' means that in the ith step the pair (x_i, y_i) is removed from the training data. In each step of the cross-validation loop, the polynomial coefficients in the THEN-part of fuzzy rules are determined using the BLS method based on training data without the pair (x_i, y_i).

Cross-validation is a model validation method, which allows us to assess how the model makes predictions on unknown data. The main purpose of cross-validation is to avoid overfitting, which may occur when the model is to complex.

4 Procedure for Designing Fuzzy Models Using the BLS and PSO-BLS Methods

The design procedures with the use of the BLS and PSO-BLS methods are given below.

4.1 Design with the BLS Method

We assume that the T-S system is described by r fuzzy rules (1) where the polynomial degree is equal to m.

Step 1. Based on the observations x_i, determine the universal set $\mathbb{X} = [a, b]$, where

$$a = \min_i(x_i), \quad b = \max_i(x_i) \tag{30}$$

and $a < b$, $i = 1, 2, \ldots, n$.

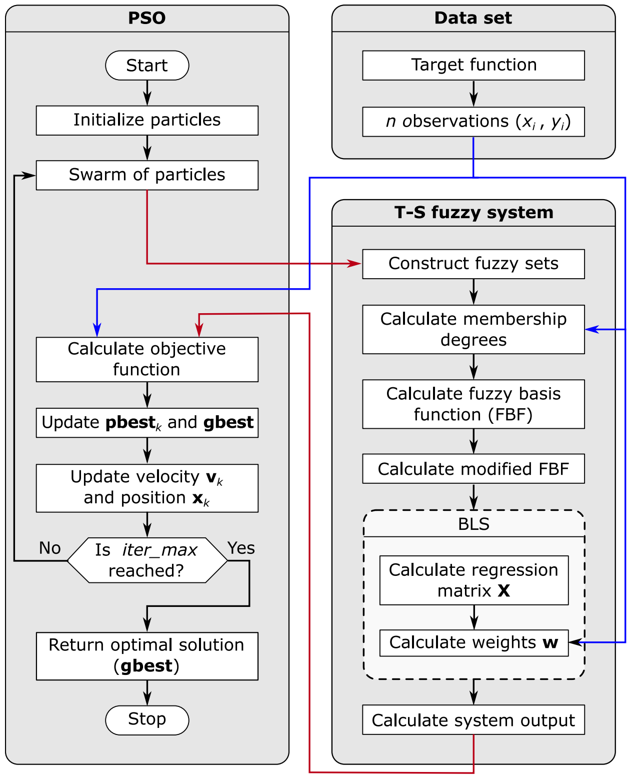
PSO
Start
Initialize particles
Swarm of particles
Calculate objective function
Update **pbest**$_k$ and **gbest**
Update velocity $\mathbf{v}_k$ and position $\mathbf{x}_k$
Is *iter_max* reached?
No
Yes
Return optimal solution (**gbest**)
Stop
Data set
Target function
n observations (x_i, y_i)
T-S fuzzy system
Construct fuzzy sets
Calculate membership degrees
Calculate fuzzy basis function (FBF)
Calculate modified FBF
BLS
Calculate regression matrix **X**
Calculate weights **w**
Calculate system output

Fig. 2 The idea of the PSO-BLS method

Step 2. Define r triangular (2) or Gaussian (3) fuzzy sets where the number of sets is at least two. These sets can be chosen manually or distributed evenly using the formula

$$p_j = a + (j-1)\delta \tag{31}$$

where $j = 1, 2, \ldots, r$ and $\delta = (b-a)/(r-1)$. The widths σ_j of Gaussian sets can be chosen in such a way that the cross-point of two adjacent sets is equal to 0.5. Based on (3), the widths σ_j can be expressed as

$$\sigma_j = \delta / \left(2\sqrt{-2\ln(0.5)}\right) \tag{32}$$

Step 3. From (12) determine for all x_i the vectors $\boldsymbol{h}(x_i)$ containing the modified fuzzy basis functions (7).

Step 4. Construct the regression matrix $\boldsymbol{X}$ given by (14).

Step 5. Determine polynomial coefficients in the fuzzy rules from (16). If the matrix $\boldsymbol{X}^T\boldsymbol{X}$ is close to singular, take $\lambda > 0$ and apply the ridge regression (17).

4.2 Design with the PSO-BLS Method

Before starting the PSO algorithm, the number of particles and the number of iterations $\mathrm{iter}_{\max}$ have to be chosen.

Step 1. Initialize all swarm particles where each of them represents the parameters of triangular (27) or Gaussian fuzzy sets (28). For triangular and Gaussian sets, determine the initial position of the peaks (p_j) from (31). Change $p_2, \ldots, p_{r-1}$ according to the formula

$$p_j := \max(\min(p_j + d_p \cdot \mathrm{rand} - d_p/2, p_r), p_1) \tag{33}$$

where rand is a random number in the interval [0, 1] and d_p is the width of initialization range for the peaks. For Gaussian sets, initialize the widths σ_j using the formula

$$\sigma_j := \max(\min(d_\sigma \cdot \mathrm{rand}, \sigma_{\max}), \sigma_{\min}) \tag{34}$$

where $j \in 1, \ldots, r$, d_σ is the width of initialization range for σ_j and $0 < \sigma_{\min} < \sigma_{\max}$.

Step 2. For each particle, determine the value of the objective function. To do this, follow Step 3 to Step 5 from Sect. 4.1 and calculate the error (26). For particles that have repeated peaks apply a penalty by assigning them a very high value of the objective function.

Step 3. Update $\mathbf{pbest}_k$ and $\mathbf{gbest}$ based on the objective function determined in the previous step.

Step 4. Update the velocity $\boldsymbol{v}_k$ and the position $\boldsymbol{x}_k$ of the particles according to (24) and (25), respectively. Limit the peaks p_j to the interval $[p_1, p_r]$ and the widths σ_j of Gaussian sets to the interval $[\sigma_{min}, \sigma_{max}]$.

Step 5. If the maximum number of iterations $iter_{max}$ is not reached, go to Step 2.

Step 6. Stop, return **gbest** as the solution.

4.3 Choosing the Best Model

In both methods, the best model can be chosen using the fitness error $\mathrm{RMSE_T}$ (26) or the cross-validation error $\mathrm{RMSE_{CV}}$ (29). In the case of cross-validation, which is the main criterion in our method, the fuzzy sets proposed by the user or by the PSO algorithm are used by the BLS regression while constructing fuzzy models in the validation loop. In the BLS method, one proposition is analyzed. In the PSO-BLS method, 10 propositions are generated, then the validation error for each proposition is determined, and finally, a proposition with the smallest error is chosen.

5 Experimental Results

To verify the developed method, experiments were carried out involving the construction of a fuzzy approximator for selected nonlinear functions. The functions from the paper [14] were used because the methods presented there have parameters similar to our method. Thanks to this, it was possible to compare the results. In Experiment 1, a benchmark function proposed by the authors in [14] was used. In Experiment 2, a function was used, which can be found, for example, in [14, 23, 29]. The experiments were carried out on a mobile computer equipped with Intel(R) Core(TM) i7-3740QM and 32GB RAM.

5.1 Experiment 1

We consider the following nonlinear function [14]

$$y = 0.08\left(1.2(x-1)\right)\cos(3x) + \left(x - (x-1)\cos(3x)\right)\sin(x) \tag{35}$$

where $x \in \mathbb{X} = [3, 7]$. According to (35), $n = 25$ data points (x_i, y_i) are equally spaced in the interval $\mathbb{X}$. The fuzzy models are considered with the number of rules $r \in \{2, 3, \ldots, 30\}$ and the polynomial degree $m \in \{0, 1, 2\}$. All models are determined using the ridge regression (17) with $\lambda = 1\mathrm{e}{-8}$.

5.1.1 Design with the BLS Method

The fitness and cross-validation errors are given in Table 1. For triangular functions, the smallest fitness error, equal to 6.102e−10, was obtained for $r = 23$ and $m = 2$. For Gaussian functions, the smallest fitness error, equal to 6.405e−10, was obtained for $r = 24$ and $m = 2$. The smallest validation error ($\mathrm{RMSE}_{\mathrm{CV}}$ = 1.661e−02) for triangular functions was obtained with $r = 22$ and $m = 0$ (as shown in Fig. 3). The smallest validation error ($\mathrm{RMSE}_{\mathrm{CV}}$ = 8.119e−02) for Gaussian functions was obtained with $r = 4$ and $m = 2$. The calculation time of training high-order T-S fuzzy systems was about 0.03 s.

5.1.2 Zero-Order T-S System as a Linear Regression

For the considered example, we can construct in the interval [3, 7] a linear model using the zero-order T-S system (see Sect. 3.2). Taking triangular fuzzy sets (19) in the form of

$$\begin{aligned} A_1(x) &= \mathrm{triang}(x; 2, 3, 7) \\ A_2(x) &= \mathrm{triang}(x; 3, 7, 8) \end{aligned} \tag{36}$$

and using the BLS method (16) we obtain the following fuzzy rules:

$$\begin{aligned} &R_1\text{: IF } x \in A_1(x) \text{ THEN } y = w_{01} = -3.288 \\ &R_2\text{: IF } x \in A_2(x) \text{ THEN } y = w_{02} = -0.306 \end{aligned} \tag{37}$$

Based on (23), the regression function has the form $y = ax + b$, where $a = 0.7456$ and $b = -5.525$.

5.1.3 Design with the PSO-BLS Method

The PSO-BLS models were determined using the PSO algorithm with the number of particles equal to 60 and the maximum number of iterations $\mathrm{iter}_{\max} = 500$. The remaining parameters were as follows: $d_p = 3$, $d_\sigma = 5.0$, $\sigma_{min} = 0.1$, and $\sigma_{max} = 5.0$.

The fitness and cross-validation errors are given in Table 1. For triangular functions, the smallest fitness error, equal to 4.332e−11, was obtained for $r = 16$ and $m = 2$.

Table 1 Experiment 1: Fitness error, cross-validation error, number of rules and the polynomial degree ($n = 25$)

Algorithm	$\mathrm{RMSE}_{\mathrm{T}}$	r	m	$\mathrm{RMSE}_{\mathrm{CV}}$	r	m
BLS (triang)	6.102e−11	23	2	1.642e−02	22	0
BLS (gauss)	6.404e−11	24	2	8.699e−02	4	2
PSO-BLS (triang)	4.332e−11	16	2	**4.255e−07**	**16**	**0**
PSO-BLS (gauss)	1.103e−09	25	2	1.761e−04	9	2

Bold indicates the best result in the column

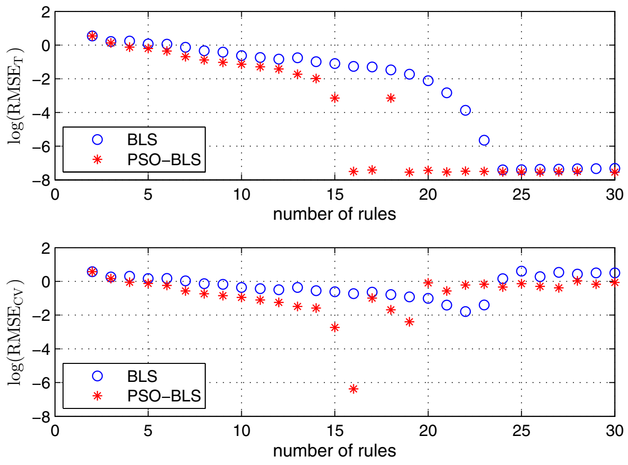

Fig. 3 Experiment 1: Fitness and cross-validation errors (in logarithmic scale) for triangular membership functions and $m = 0$

For Gaussian functions, the smallest fitness error, equal to 1.103e−09, was obtained for $r = 25$ and $m = 2$. The smallest cross-validation error for triangular functions is equal to 4.255e−07, and it was obtained for the zero-order model with 16 rules (as shown in Fig. 3). For Gaussian functions, this error is equal to 1.761e−04, and it was obtained for the second-order model with nine rules. The model with the smallest $\mathrm{RMSE}_{\mathrm{CV}}$ error was selected as the final solution, the parameters of which are presented in Table 2. The time consumed to train T-S fuzzy systems was about 10 s.

Examples of function approximation by the BLS and PSO-BLS models are shown in Fig. 4.

Table 2 Experiment 1: Parameters of the model with the smallest cross-validation error in Table 1

Rule	p	w_0
R_1	3.000	0.5056
R_2	3.227	− 0.6451
R_3	3.437	− 1.732
R_4	3.714	− 1.851
R_5	4.120	− 0.4552
R_6	4.385	− 1.039
R_7	4.567	− 2.820
R_8	4.945	− 7.725
R_9	5.151	− 8.751
R_{10}	5.412	− 7.837
R_{11}	5.534	− 5.872
R_{12}	5.926	− 0.7716
R_{13}	6.255	0.7111
R_{14}	6.418	0.3960
R_{15}	6.741	2.283
R_{16}	7.000	6.443

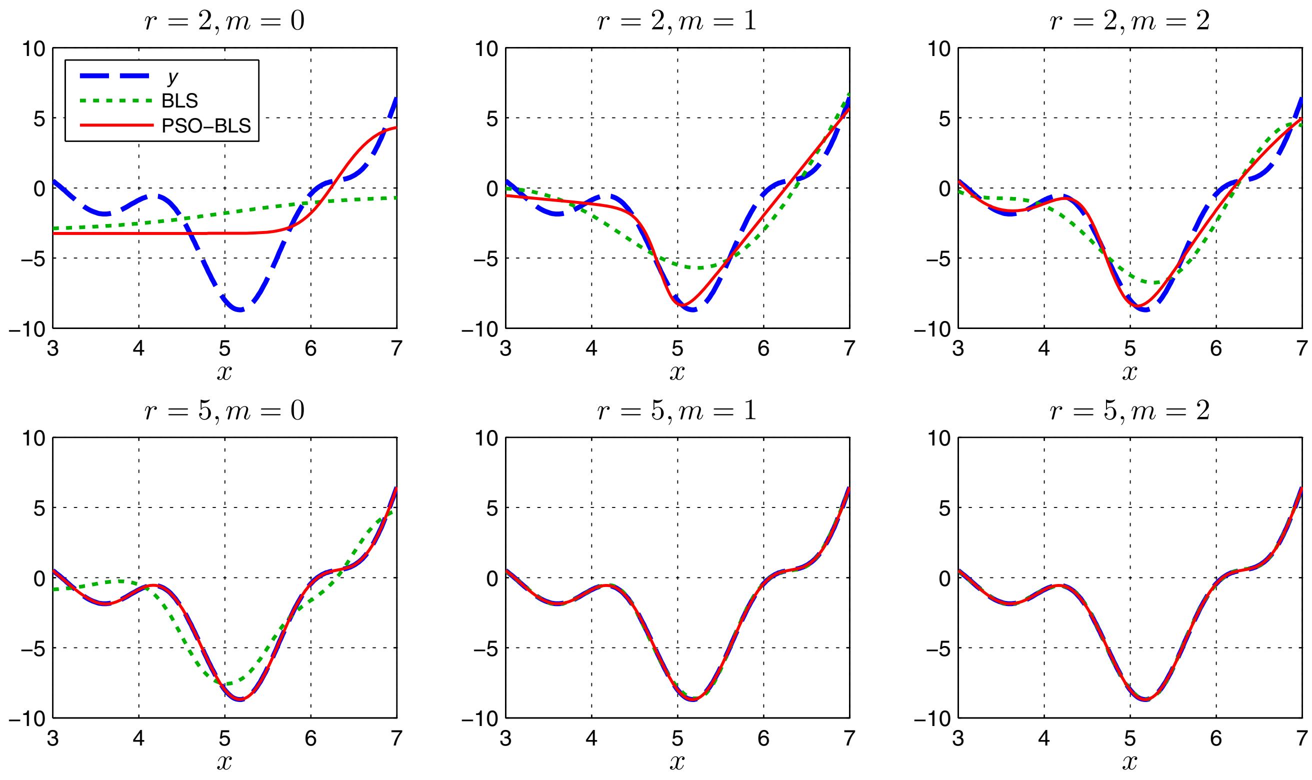

Fig. 4 Experiment 1: Examples of function approximation by the BLS and PSO-BLS models with Gaussian membership functions

5.1.4 Comparison with the Work by Li, Wu [14]

To compare the developed algorithm, the fitness error was determined for the BLS and PSO-BLS models with parameters similar to those used in [14] for the PSO and PSO-RLSE models. The number of observations was 100, and the number of rules was nine.

For the BLS models, nine triangular and Gaussian membership functions were defined with the peaks in points $p_j = 3.0, 3.5, \ldots, 7.0$. Using (32), the width of all Gaussian sets was determined as $\sigma_j = 0.2123$. The polynomials in the THEN-part of the fuzzy rules was chosen as second-order $(m = 2)$, and this way, the number of parameters was $p = r(m+1) = 27$ (see Table 3). The fitness errors for the BLS models are shown in Table 3.

The PSO-BLS models were obtained using 60 particles and 2000 iterations. The number of parameters in the antecedent part was equal to seven for triangular functions, and 16 for Gaussian functions. The number of parameters in the consequent part was the same as for BLS models and equal to 27. The models were chosen as the best from 10 trials. The performance comparison is shown in Table 3, in which the proposed training method (PSO-BLS) with Gaussian membership functions has the smallest fitness error $\mathrm{RMSE_T} = 3.136\mathrm{e}{-05}$, better than the fitness error of [14]. The methods by [14] have the errors of $\mathrm{RMSE_T} = 3.051\mathrm{e}{-02}$ and $\mathrm{RMSE_T} = 8.920\mathrm{e}{-04}$. The parameters of this model are shown in Table 4, whereas the function approximation is shown in Fig. 5.

Table 3 Experiment 1: Fitness error comparison and the number of parameters $(n = 100)$

Algorithm	$\mathrm{RMSE_T}$	Antecedent part	Consequent part
PSO [14]	3.015e−02	12	27
PSO-RLSE [14]	8.920e−04	12	27
BLS (triang)	6.243e−03	–	27
BLS (gauss)	9.667e−03	–	27
PSO-BLS (triang)	2.835e−03	7	27
PSO-BLS (gauss)	**3.136e−05**	16	27

Bold indicates the best result in the column

5.2 Experiment 2

We consider the function [14, 23, 29]

$$y = \frac{(x-2)(2x-1)}{1+x^2} \qquad (38)$$

where $x \in \mathbb{X} = [-8, 12]$. The number of observations is equal to 50, and they are equally spaced in the interval $\mathbb{X}$. The fuzzy models are considered with the number of rules $r \in \{0, 1, \ldots, 50\}$ and the polynomial degree

Table 4 Experiment 1: Parameters of the model with the smallest fitness error in Table 3

Rule	p	σ	w_2	w_1	w_0
R_1	3.000	3.910	6.676	− 7.798	− 5.645
R_2	3.431	0.6768	2.204	− 6.281	− 4.437
R_3	4.891	0.5978	4.904	− 19.11	− 7.087
R_4	3.828	0.4976	0.7342	− 13.01	− 4.906
R_5	6.045	0.9458	− 15.51	16.07	5.631
R_6	6.021	3.220	− 3.027	0.3070	− 1.754
R_7	6.056	0.6413	3.439	40.14	12.57
R_8	4.341	0.4936	− 6.976	32.45	13.02
R_9	7.000	0.6176	10.16	− 36.10	− 10.62

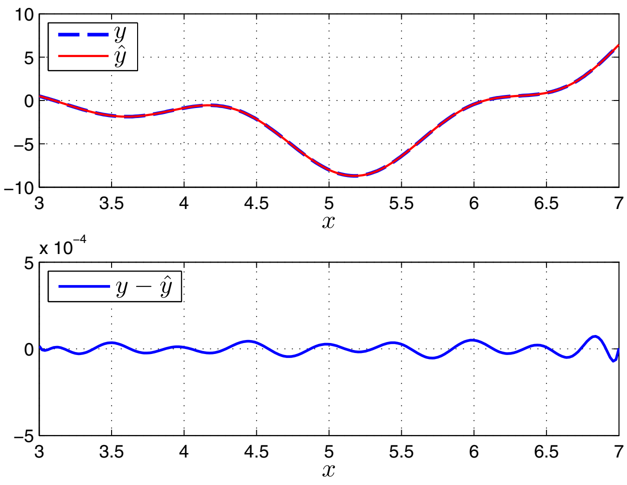

Fig. 5 Experiment 1: Function approximation and the error between the target function y and the estimator $\hat{y}$ for the PSO-BLS model in Table 4

$m \in \{0, 1, 2, 3\}$. The models are determined with the regularization parameter $\lambda = 1e{-}8$.

5.2.1 Design with the BLS Method

The smallest fitness error equal to 5.765e−09 for triangular sets was obtained for 45 rules and the third-order polynomial (see Table 5). For Gaussian sets, this error was equal to 5.581e−09, and it was obtained for the third-order system with 38 rules. In turn, for triangular sets, the smallest cross-validation error equal to 1.805e−02 was obtained for 21 rules and the second-order polynomial. This error for Gaussian sets was equal to 1.353e−02, and it was obtained for the second-order system with 18 rules. The calculation time of training T-S fuzzy systems was about 0.03 s.

Table 5 Experiment 2: Fitness error, cross-validation error, number of rules and polynomial degree

Algorithm	RMSE_T	r	m	RMSE_CV	r	m
BLS (triang)	5.765e−09	45	3	1.805e−02	21	2
BLS (gauss)	5.581e−09	38	3	1.353e−02	18	2
PSO-BLS (triang)	3.559e−09	44	2	3.774e−04	10	2
PSO-BLS (gauss)	2.093e−08	48	3	**1.983e−05**	**9**	**3**

Bold indicates the best result in the column

5.2.2 Design with the PSO-BLS Method

In the PSO-BLS method, the following parameters were applied: the number of particles equal to 60, the number of iterations equal to 500, $d_p = 12$, $d_\sigma = 8$, $\sigma_{\min} = 0.5$ and $\sigma_{\max} = 10.0$. For triangular functions, the smallest fitness error equal to 3.559e−09 was obtained for $r = 44$ and $m = 2$ (see Table 5). For Gaussian sets, this error was obtained for $r = 48$ and $m = 3$, and it was equal to 2.093e−08. The smallest validation error for triangular sets was equal to 3.774e−04, and it was obtained for 10 rules and the second-order system. For Gaussian sets, this error was equal to 1.983e−05, and it was obtained for nine rules and the third-order system. The parameters of the best model are shown in Table 6. The time consumed to train T-S fuzzy systems was about 36 s.

Examples of function approximation by the BLS and PSO-BLS models are shown in Fig. 6.

5.2.3 Comparison with Other Methods

The proposed algorithms were compared with the methods used in [14, 23, 29]. The number of observations was 50, the number of rules was nine, and the polynomials were second-order. The PSO-BLS models were obtained using 60 particles and 500 iterations. The comparison of the fitness error for the considered models is given in Table 7. It is seen that the proposed training method (PSO-BLS) with Gaussian membership functions has the smallest fitness error $\text{RMSE}_\text{T} = 9.072e{-}06$, better than the fitness error of other methods [14, 23, 29]. The methods by [23] have the errors of $\text{RMSE}_\text{T} = 8.430e{-}02$ and $\text{RMSE}_\text{T} = 8.700e{-}03$. The method by [14] has the error of $\text{RMSE}_\text{T} = 5.290e{-}04$, while the method by [29] has the error of $\text{RMSE}_\text{T} = 3.320e{-}02$. The parameters of the best model are shown in Table 8.

Table 6 Experiment 2: Parameters of the model with the smallest cross-validation error in Table 5

Rule	p	σ	w_3	w_2	w_1	w_0
R_1	− 8.000	2.976	0.007742	0.09625	− 0.09317	0.3063
R_2	− 2.213	1.037	− 0.05236	− 0.5003	− 1.559	2.027
R_3	− 7.209	4.096	− 0.01995	− 0.3618	− 1.634	1.815
R_4	− 0.5277	0.5000	− 0.5533	− 1.111	0.03736	6.372
R_5	0.5228	0.5941	− 0.7274	2.288	− 2.247	− 1.507
R_6	0.4927	1.159	− 0.1921	1.393	− 3.400	2.343
R_7	7.691	4.000	− 0.0006450	− 0.09154	0.06841	0.7992
R_8	10.46	5.053	− 0.01265	0.2316	− 0.02034	0.5944
R_9	12.00	2.238	− 0.003798	0.07565	− 0.3725	− 0.07597

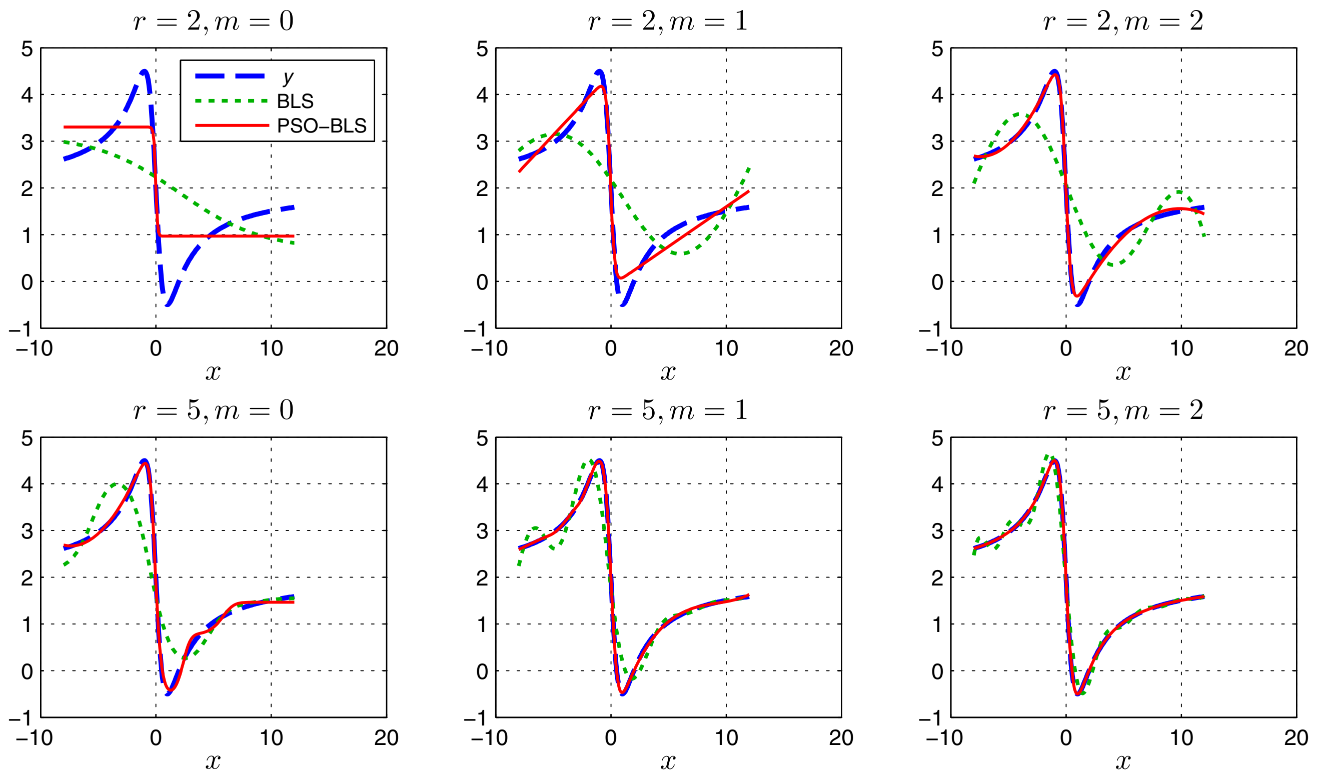

Fig. 6 Experiment 2: Examples of function approximation by the BLS and PSO-BLS models with Gaussian membership functions

Table 7 Experiment 2: Fitness error comparison and number of parameters

Algorithm	RMSE$_T$	Antecedent part	Consequent part
LS-GRBFNFIS [23]	8.430e−02	–	–
LW-GRBFNFIS [23]	8.700e−03	–	–
PSO-RLSE [14]	5.290e−04	12	27
LW-GRBFNFS [29]	3.320e−02	–	–
BLS (triang)	5.516e−02	–	27
BLS (gauss)	2.931e−02	–	27
PSO-BLS (triang)	1.098e−04	7	27
PSO-BLS (gauss)	**9.072e−06**	16	27

Bold indicates the best result in the column

Table 8 Experiment 2: Parameters of the model with the smallest fitness error in Table 7

Rule	p	σ	w_2	w_1	w_0
R_1	− 8.000	2.356	881.9	666.6	342.6
R_2	− 5.906	1.485	5.720	1.948	0.6479
R_3	− 5.767	2.446	− 24,350	14,560	14,270
R_4	− 2.361	0.9911	166.3	51.78	15.92
R_5	0.5223	0.5000	− 147.8	− 88.37	− 41.91
R_6	− 0.3668	0.5536	− 0.9968	− 0.3998	− 0.1548
R_7	− 1.589	0.5000	− 2.150e−10	− 7.299e−11	− 2.475e−11
R_8	10.59	3.996	− 71.42	710.9	− 7.621
R_9	12.00	7.829	92.41	− 1231.1	2898

6 Conclusions

In this paper, two methods for training high-order Takagi–Sugeno systems using BLS and PSO has been proposed. The considered methods can be used for triangular or Gaussian membership functions in the antecedents and high-order polynomials in the consequents of fuzzy rules. The fuzzy sets can be chosen manually or by a PSO algorithm, whereas the polynomials are determined by the least squares method. In the case where the system is close to the singularity, we proposed to regularize the solution using the ridge regression. To avoid overfitting, we used the leave-one-out cross-validation method. Moreover, we showed how to realize piecewise linear regression using the zero-order Takagi–Sugeno system with triangular functions. The proposed PSO-BLS fuzzy approach has been successfully applied for two examples. From the experimental results and performance comparison, the approach has shown excellent performance in training high-order Takagi–Sugeno fuzzy systems.

Future work will be devoted to the generalization of the presented methods for systems with many input variables.

Acknowledgements This work has been partially supported by the Polish Ministry of Science and Higher Education under Grant No. DS/M.EA.17.002.

References

1. Alfi, A., Fateh, M.M.: Intelligent identification and control using improved fuzzy particle swarm optimization. Expert Syst. Appl. **38**(10), 12312–12317 (2011)
2. Arlot, S., Celisse, A.: A survey of cross-validation procedures for model selection. Stat. Surv. **4**, 40–79 (2010)
3. Bishop, C.M.: Pattern recognition and machine learning. Information science and statistics. Springer-Verlag, Inc, New York (2006)
4. Boulkaibet, I., Belarbi, K., Bououden, S., Marwala, T., Chadli, M.: A new T-S fuzzy model predictive control for nonlinear processes. Expert Syst. Appl. **88**, 132–151 (2017). https://doi.org/10.1016/j.eswa.2017.06.039
5. Chen, C., Liu, Y.: Enhanced ant colony optimization with dynamic mutation and ad hoc initialization for improving the design of TSK-type fuzzy system. Comput. Int. Neurosci. **2018**, 1–15 (2018)
6. Cheng, R., Bai, Y.: A novel approach to fuzzy wavelet neural network modeling and optimization. Int. J. Electr. Power Energy Syst. **64**, 671–678 (2015)
7. Cheung, N.J., Ding, X.M., Shen, H.B.: Optifel: a convergent heterogeneous particle swarm optimization algorithm for Takagi-Sugeno fuzzy modeling. IEEE Trans. Fuzzy Syst. **22**(4), 919–933 (2014)
8. Du, H., Zhang, N.: Application of evolving Takagi-Sugeno fuzzy model to nonlinear system identification. Appl. Soft Comput. **8**(1), 676–686 (2008)
9. Eberhart, R.C., Shi, Y.: Comparing inertia weights and constriction factors in particle swarm optimization. In: Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No.00TH8512), vol. 1, pp. 84–88 (2000)
10. Hoerl, A.E., Kennard, R.W.: Ridge regression: biased estimation for nonorthogonal problems. Technometrics **12**(1), 55–67 (1970)
11. Juang, C.F., Lo, C.: Zero-order TSK-type fuzzy system learning using a two-phase swarm intelligence algorithm. Fuzzy Sets Syst. **159**(21), 2910–2926 (2008)
12. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proc. of IEEE Int. Conf. on Neural Networks, vol. 4, pp. 1942–1948. IEEE Press, Piscataway, NJ (1995)
13. Krzeszowski, T., Przednowek, K., Wiktorowicz, K., Iskra, J.: Estimation of hurdle clearance parameters using a monocular human motion tracking method. Comput. Methods Biomech. Biomed. Eng. **19**(12), 1319–1329 (2016). PMID: 26838547
14. Li, C., Wu, T.: Adaptive fuzzy approach to function approximation with PSO and RLSE. Expert Syst. Appl. **38**(10), 13266–13273 (2011)
15. Li, C., Wu, T., Chan, F.T.: Self-learning complex neuro-fuzzy system with complex fuzzy sets and its application to adaptive image noise canceling. Neurocomputing **94**, 121–139 (2012)
16. Lin, C.J.: An efficient immune-based symbiotic particle swarm optimization learning algorithm for TSK-type neuro-fuzzy networks design. Fuzzy Sets Syst. **159**(21), 2890–2909 (2008)
17. Lin, G., Zhao, K., Wan, Q.: Takagi-Sugeno fuzzy model identification using coevolution particle swarm optimization with multi-strategy. Appl. Intell. **45**(1), 187–197 (2016)

18. Martino, F.D., Loia, V., Sessa, S.: Multi-species PSO and fuzzy systems of Takagi-Sugeno-Kang type. Inf. Sci. **267**(Supplement C), 240–251 (2014)
19. Niu, B., Zhu, Y., He, X., Shen, H.: A multi-swarm optimizer based fuzzy modeling approach for dynamic systems processing. Neurocomputing **71**(7–9), 1436–1448 (2008)
20. Rastegar, S., Araujo, R., Mendes, J.: Online identification of Takagi-Sugeno fuzzy models based on self-adaptive hierarchical particle swarm optimization algorithm. Appl. Math. Model. **45**((Supplement C)), 606–620 (2017)
21. Soltani, M., Chaari, A., Ben Hmida, F.: A novel fuzzy c-regression model algorithm using a new error measure and particle swarm optimization. Int. J. Appl. Math. Comput. Sci. **22**(3), 617–628 (2012)
22. Srinivasan, D., Loo, W.H., Cheu, R.L.: Traffic incident detection using particle swarm optimization. In: Swarm Intelligence Symposium. SIS '03. Proceedings of the IEEE, pp. 144–151 (2003)
23. Sun, T.Y., Tsai, S.J., Tsai, C.H., Huo, C.L., Liu, C.C.: Nonlinear function approximation based on least Wilcoxon Takagi-Sugeno fuzzy model. In: 2008 Eighth International Conference on Intelligent Systems Design and Applications, vol. 1, pp. 312–317 (2008)
24. Taieb, A., Soltani, M., Chaari, A.: A fuzzy C-regression model algorithm using a new PSO algorithm. Int. J. Adapt. Control Signal Process. **32**(1), 115–133 (2018)
25. Takagi, T., Sugeno, M.: Fuzzy identification of systems and its applications to modeling and control. IEEE Trans. Syst. Man Cybern. **SMC–15**(1), 116–132 (1985)
26. Tsai, S.H., Chen, Y.W.: A novel identification method for Takagi-Sugeno fuzzy model. Fuzzy Sets Syst. **338**, 117–135 (2018)
27. Wang, L., Mendel, J.M.: Fuzzy basis functions, universal approximation, and orthogonal least-squares learning. IEEE Trans. Neural Netw. **3**(5), 807–814 (1992)
28. Wiktorowicz, K.: Output feedback direct adaptive fuzzy controller based on frequency-domain methods. IEEE Trans. Fuzzy Syst. **24**(3), 622–634 (2016)
29. Yang, Y.K., Sun, T.Y., Huo, C.L., Yu, Y.H., Liu, C.C., Tsai, C.H.: A novel self-constructing radial basis function neural-fuzzy system. Appl. Soft Comput. **13**(5), 2390–2404 (2013)
30. Yeh, C.Y., Jeng, W.H.R., Lee, S.J.: Data-based system modeling using a type-2 fuzzy neural network with a hybrid learning algorithm. IEEE Trans. Neural Netw. **22**(12), 2296–2309 (2011)
31. Ying, K.C., Lin, S.W., Lee, Z.J., Lee, I.L.: A novel function approximation based on robust fuzzy regression algorithm model and particle swarm optimization. Appl. Soft Comput. **11**(2), 1820–1826 (2011)
32. Youssef, K.H., Yousef, H.A., Sebakhy, O.A., Wahba, M.A.: Adaptive fuzzy APSO based inverse tracking-controller with an application to DC motors. Expert Syst. Appl. **36**(2, Part 2), 3454–3458 (2009)
33. Zhao, L., Qian, F., Yang, Y., Zeng, Y., Su, H.: Automatically extracting T-S fuzzy models using cooperative random learning particle swarm optimization. Appl. Soft Comput. **10**(3), 938–944 (2010)

Krzysztof Wiktorowicz received the M.Sc. (Eng.) degree from the Rzeszow University of Technology, Poland, in 1993 and the Ph.D. degree from the Technical University of Wroclaw, Poland, in 2001. He is an Assistant Professor in the Faculty of Electrical and Computer Engineering, Rzeszow University of Technology, Poland. His research interests include fuzzy logic control, artificial intelligence and machine learning.

Tomasz Krzeszowski received the M.Sc. (Eng.) degree in Computer Science from the Rzeszow University of Technology in 2009. In 2013, he received his Ph.D. in Computer Science at the Silesian University of Technology. He is currently an assistant professor in the Faculty of Electrical and Computer Engineering at the Rzeszow University of Technology. His areas of interest lie in computer vision, human motion tracking, machine learning, and particle swarm optimization algorithms.