A new T-S fuzzy model predictive control for nonlinear processes

Boulkaibet I [a,b], Belarbi K [b], Bououden S [c], Marwala T [a] and Chadli M [d].

[a] Institute for Intelligent Systems, University of Johannesburg, South Africa.
[b] Department of Electronics Engineering, University of Constantine 1, Algeria.
[c] Faculty of Sciences and Technology, University Abbes Laghrour, Khenchela, Algeria.
[d] University of Picardie Jules Verne Amiens, France.

Email: ilyes@aims.ac.za (I. Boulkaibet), kbelarbi@yahoo.com (K. Belarbi), ss_bououden@yahoo.fr (S. Bououden), tmarwala@uj.ac.za (T. Marwala), mohammed.chadli@u-picardie.fr (M. Chadli)

Abstract— In this paper, a novel fuzzy Generalized Predictive Control (GPC) is proposed for discrete-time nonlinear systems via Takagi-Sugeno system based Kernel Ridge Regression (TS-KRR). The TS-KRR strategy approximates the unknown nonlinear systems by learning the Takagi-Sugeno (TS) fuzzy parameters from the input-output data. Two main steps are required to construct the TS-KRR: the first step is to use a clustering algorithm such as the clustering based Particle Swarm Optimization (PSO) algorithm that separates the input data into clusters and obtains the antecedent TS fuzzy model parameters. In the second step, the consequent TS fuzzy parameters are obtained using a Kernel ridge regression algorithm. Furthermore, the TS based predictive control is created by integrating the TS-KRR into the Generalized Predictive Controller. Next, an adaptive, online, version of TS-KRR is proposed and integrated with the GPC controller resulting an efficient adaptive fuzzy generalized predictive control methodology that can deal with most of the industrial plants and has the ability to deal with disturbances and variations of the model parameters. In the adaptive TS-KRR algorithm, the antecedent parameters are initialized with a simple K-means algorithm and updated using a simple gradient algorithm. Then, the consequent parameters are obtained using the sliding-window Kernel Recursive Least squares (KRLS) algorithm. Finally, two nonlinear systems: A surge tank and Continuous Stirred Tank Reactor (CSTR) systems were used to investigate the performance of the new adaptive TS-KRR GPC controller. Furthermore, the results obtained by the adaptive TS-KRR GPC controller were compared with two other controllers. The numerical results demonstrate the reliability of the proposed adaptive TS-KRR GPC method for discrete-time nonlinear systems.

***Index Terms*— Generalized Predictive Control; Takagi-Sugeno fuzzy system; Kernel ridge regression; clustering algorithm; Particle Swarm Optimization; Takagi-Sugeno system based Kernel ridge regression; Sliding-window Kernel Recursive Least squares.**

I. INTRODUCTION

The Model Predictive Control (MPC) approaches represent one of the most significant control developments in the last thirty years (Prett, & Garcia, 1988). The features of the predictive controller such as model structures, prediction horizon and optimization criteria allow for the modification and adjustment of the MPC to suit a large range of engineering applications. The predictive control was first introduced by Richalet et al. (Richalet, Rault, Testud, & Papon, 1978; Richalet, 1993) where their algorithmic formulations have benefited from the recent advancement in digital computers and became more practical. Another predictive control formulation that suits well the open-loop stable processes: the dynamic matrix control (DMC). The DMC method became famous due to its simplicity as well as its exploitation of the step response models which can be easily obtained (Marchetti, Mellichamp, & Seborg, 1983; Brujin, & Verbruggen, 1984). Moreover, the Generalized Predictive Control (GPC), which has been introduced by Clark et al. (Clarke, Mothadi, & Tuffs, 1989; Clarke, & Mohtadi, 1989), has offered virtuous results in handling unstable systems with a wider range of non-minimum phase. The GPC strategy uses mostly polynomial models which limit the number of parameters that describes the process, and help obtaining effective and solid algorithms (Clarke, & Mohtadi, 1989). The GPC algorithm has been applied many times to a wide class of industrial plants and showed decent results (Richalet, Rault, Testud, & Papon, 1978). However, most of the systems controlled by the GPC were linear systems since the quadratic optimization in the GPC algorithm can only be solved for linear predictions.

The idea of developing efficient Nonlinear GPC (NGPC) algorithms to control nonlinear process was attracted by many researchers, and many papers were published in the NGPC field. The simplest strategy for using GPC to control nonlinear plants is to linearize the nonlinear model of the plants (Zhu, Warwick, & Douce, 1991). However, this approach has performed poorly since the operating point may change. Various strategies have been developed for NGPC such as the stabilizing predictive control with nonlinear ARX models which was presented by Nicolao et al. (Nicolao, Magi, & Scattolini, 1997) to control nonlinear discrete-time systems. Kanev et al. (Kanev, & Verhaegen, 2000) combines a multiple model estimator and the GPC algorithm for controller reconfiguration of nonlinear systems. Chen et al. (Chen, Balance, Gawthrop, Gribble, & O'Reilly, 1999) controlled nonlinear plants using a class of nonlinear PID controllers that have been derived from a nonlinear GPC approach. An

automatic differentiation approach is used by Cao in (Cao, 2005) to formulate a nonlinear model GPC.

Generally, the models used by all predictive controllers (including the GPC) are assumed to be accurate. This can be a serious problem since a wide range of plants are complex and cannot be mathematically modelled. Moreover, sometimes these plants have large uncertainties and strong nonlinearities. In the case that no mathematical model is available to describe a system, approximation methods, such as fuzzy logic (Zadeh, 1973; Driankov et al., 1993; Chen et al., 2013; Flores et al., 2005; Sáez et al., 2007; Babuska, 1998) and neural networks (NNs) (Chen, & Billings, 1992), present a good alternative. The use of neural networks to approximate functions has shown practical results and has been applied successfully by many researchers (Tsai et al., 2002; Zamarreno, & Vega, 1999; Palos et al., 2001; Huang, & Lewis, 2003; Lu, & Tsai, 2004; Lu, & Tsai, 2004; Eski, & Temürlenk, 2013) in modelling complex processes. The results in (Tsai et al., 2002; Zamarreno, & Vega, 1999; Palos et al., 2001; Huang, & Lewis, 2003; Lu, & Tsai, 2004; Lu, & Tsai, 2004; Eski, & Temürlenk, 2013) demonstrated the abilities of the neural predictive control techniques for nonlinear dynamic systems. On the other hand, Takagi–Sugeno (TS) (Takagi, & Sugeno, 1985) fuzzy model has been established as an efficient approximation model for nonlinear GPC. The Takagi–Sugeno (TS) fuzzy model has the ability to accurately approximate complex nonlinear systems by using data along with a prior knowledge of processes (Mollov, Babuska, Abonyi, & Verbruggen, 2004; Bououden, Chadli, & Karimi, 2015). The studies presented in (Mollov, Babusˇka, Abonyi, & Verbruggen, 2004; Sousa, 2000; Mahfouf, Linkens, & Abbod, 2000; Sousa, & Kaymak, 2001; Ali, 2003; Flores, Sáez, Araya, Berenguel, & Cipriano, 2005; Bououden, Chadli, & Karimi, 2015a; Bououden, Chadli, & Karimi, 2015b) reported many successful applications of NGPC using fuzzy models. Chang et al. (Chang; Tsai, 2013) proposed an adaptive Takagi-Sugeno-Kang (ATSK) to model nonlinear processes. In this method, the membership functions were selected as a triangular functions and initialised using the training data, while the consequent parameters were identified using the recursive least squares algorithm. Then, an adaptive fuzzy model adaptive stable generalized predictive control for nonlinear discrete-time systems was constructed by integrating the ATSK algorithm with the GPC algorithm. Jang (Jang, 1991; Jang, 1993) proposed an adaptive neuro-fuzzy inference system (ANFIS) method that combines the capabilities of the artificial neural network in modelling nonlinear processes and the fuzzy reasoning in handling uncertainties. The ANFIS Algorithm was used many times to construct a NGPC (Zhang, Chai, Wang, & Fu, 2010; Abghari, Sadi, 2014).

Generally, there are two main approaches to obtain TS fuzzy models: the off-line and adaptive TS fuzzy algorithms. In the presence of input-output data collected from plants, the TS fuzzy model can be obtained by implementing the following procedure: First, the antecedent TS fuzzy model parameters (which includes: rules number, antecedent membership functions, and a set of rules) are obtained by partitioning the data into subsets (or clusters). This can be done using clustering algorithms. Then, the number of clusters, the centroid vectors and variance (width) of the clusters are used to describe the TS fuzzy model antecedent parameters. The second step is to identify the consequent TS fuzzy model parameters which can be done using optimization algorithms. Unfortunately, the datasets collected from plants are usually limited and cannot accurately describe all operating areas of the plant. Moreover, the behaviour of the plant may change over time. These limitations can seriously diminish the accuracy of the approximations made by a TS fuzzy model. On the other hand, introducing adaptive capabilities to the TS fuzzy model may improve the accuracy of the approximations made by the TS model. Several papers described the use of adaptive TS fuzzy models for system identification and control. Li et al. (Li, Zhou, Xiang, Li, & An, 2009) introduced an adaptive fuzzy-modelling approach that can automatically determine the right number of rules. In this algorithm, the premise parameters are obtained by using a fuzzy c-regression model clustering algorithm, while exploiting an orthogonal least squares algorithm to identify the consequent parameters. Rastegar et al. (Rastegar, Araújo, & Mendes, 2014) proposed a new online evolving Takagi–Sugeno (TS) fuzzy model identification method based on an unsupervised fuzzy clustering algorithm (NUFCA). Then, the proposed method was integrated with a GPC algorithm resulting in an adaptive predictive process control methodology. In this algorithm, the input-output data were partitioned to identify the antecedent parameters of the fuzzy system, while a recursive least squares algorithm (RLS) was applied to update the consequent parameters. Mondes et al. (Mendes, Araújo, & Souza, 2013) proposed an adaptive identification for industrial applications where a hierarchical genetic algorithm (HGA) was utilized to approximate the unknown nonlinear processes in the presence of input-output data.

Recently, the integration of regression methods based on kernel machine in fuzzy modelling has been attracted by many researches (Chiang, & Hao, 2004; Lin, Liang, Yeh, & Fan, 2005; Juang, & Hsieh, 2009; Guo, & Guan, 2015). Kernel regression methods, such as Kernel ridge regression (Saunders, Gammerman, & Vovk, 1998) and support vector regression (SVR) (Cortes, & Vapnik, 1995; Girosi, 1998), perform nonlinear input data mapping to a high-dimensional feature space using the properties of kernel functions. This property gives these methods a high generalization ability and strong capacity to deal with nonlinearities in modelling and system identification.

Chiang et al. (Chiang, & Hao, 2004) exploited the properties of support vector regression and proposed a fuzzy modelling network based on the SVR. In this approach, the fuzzy basis function of the fuzzy model is considered as a kernel function in a SVR and the antecedent part of the fuzzy system is then generated based on the obtained support vectors. The main advantage of this method is that the number of rules is generated automatically since the number of rules is equal to the number of support vectors. However, this approach is computationally expensive since the number of support vectors in SVR is usually large. Another method that uses the properties of SVR to identify the antecedent part of fuzzy models was proposed by Lin et al. (Lin, Liang, Yeh, &

Fan, 2005). In this method, A SVR-based forward neural network (FNN) is introduced where the number of Support vectors is equal to the initial number of rules. Then, the size of the model (number of rules) is reduced by eliminating the irrelevant rules. However, the reduction procedure degrades the performance of the original fuzzy model. Juang et al. (Juang, & Hsieh, 2009) proposed another approach where the SVR is used to identify the consequent parameters of the TS fuzzy model, while a simple clustering algorithm is used to define the antecedent parameters. However, this method uses complex kernel functions and the obtained fuzzy model is too complex to be implemented for an adaptive TS identification and control. Inspired by Juang, & Hsieh, 2009 work, this paper uses a Kernel ridge regress to identify the consequent TS fuzzy model parameters; however the obtained TS fuzzy model is simple and can be easily converted to an adaptive TS fuzzy model for system identification and control.

In this paper, two main objectives will be discussed. The first objective is to introduce the new TS fuzzy model for the offline and the online system identifications. In the case of an offline TS model, the model is constructed by separating input-output data using clustering based Particle Swarm Optimization (PSO) algorithm. Then, the antecedent TS fuzzy model parameters will be identified. However, the real novelty of this approach is to exploit the properties of Kernel functions and use a Kernel ridge regression approach to identify the consequent TS fuzzy parameters. For the adaptive TS fuzzy model algorithm, the structure of the proposed fuzzy model is very simple and the consequent parameters can be easily updated using a recursive least squares algorithm. However, in this paper a different approach is adopted where the consequent TS fuzzy model parameters are obtained using a modified Kernel Recursive Least squares (KRLS) algorithm called the sliding-window KRLS algorithm (Van Vaerenbergh, Vía, & Santamaria, 2006). Then, the antecedent TS fuzzy model parameters are initialized with a simple clustering (K-means) algorithm and updated using a simple gradient algorithm. In the sliding-window KRLS algorithm, a window of the last *M* data is stored as its dictionary where in each step the new data is added to the dictionary while the oldest data is discarded. This can lead to a sliding-window approach and reduce the computational time (instead of using all data) to execute a single real time step. The second objective of this paper is to integrate the proposed TS method into GPC to construct a TS fuzzy generalized predictive control. By introducing the concept of the dictionary in the adaptive TS fuzzy GPC, more values of the previous input-output data will be involved in the adaptation procedure. This can help obtaining more accurate results, and the proposed adaptive TS fuzzy GPC can be used to control nonlinear plants with time-varying processes, disturbances or nonlinear plants with varying operating regions. The performance of the proposed adaptive controller is highlighted by comparing the adaptive TS-KRR GPC with two different fuzzy predictive controllers: The ANFIS GPC and the ATSK GPC controllers.

The rest of this paper is organized as follows: Section II presents the basic mathematical model of the Takagi-Sugeno system based Kernel ridge regression (TS-KRR). The theory and the mathematical formulations of the TS-KRR as well as the clustering based PSO algorithm will be presented in details in Section III. Moreover, the adaptive TS-KRR will be discussed in Section III. The predictive control law is derived in Section IV. Section V discusses both: offline and adaptive identification results, and the online/offline TS-KRR GPC control results for a simple nonlinear system (surge tank system). In this paper, more attentions were given to the proposed adaptive algorithm where the disturbances capabilities of the adaptive TS-KRR GPC controller are tested in Section V by adding disturbances to the surge tank system. Section VI presents the offline/online TS-KRR identification results as well as the online/offline TS-KRR GPC control results for a CSTR nonlinear system. In section VI, more investigations were made for the adaptive TS-KRR GPC controller where this controller is tested under the presence of the disturbances. Furthermore, the performance of the TS-KRR GPC controller was investigated in the case where the reference signal is sinusoidal function and in the presence of disturbances. In Section VII, we conclude this paper.

II. THE TAKAGI-SUGENO FUZZY MODEL

In this section, the Takagi-Sugeno fuzzy model based Kernel ridge regression (TS-KRR) will be discussed. First, the TS-KRR algorithm is based on the "*IF-THEN*" Takagi-Sugeno fuzzy rules. Similar to the classical Takagi-Sugeno fuzzy rules, the *i*-th rule in a TS-KRR is presented as follows:

$$\begin{aligned} R_i: \quad & \text{IF } x_1(k) \text{ is } A_1^i, \text{and } \dots, \text{and } x_n(k) \text{ is } A_n^i \\ & \text{THEN } y_i(k) = a_1^i x_1(k) + \cdots + a_n^i x_n(k) \\ & i = 1, \dots, N \end{aligned} \tag{1}$$

where R_i $(i = 1, 2, \dots, N)$ is the *i*-th fuzzy rule and N is the number of rules. The input variables are: $x_1(k), \dots, x_n(k)$, where k is the time increment, and $y_i(k)$ is the system output of the *i*-th fuzzy rule. $A_j^i, i = 1, 2, \dots, N, j = 1, 2, \dots, n$ are the linguistic terms where these terms are characterized by the fuzzy membership functions $\mu_{A_j^i}(x_j)$, $i = 1, 2, \dots, N, j = 1, 2, \dots, n$, and each term describes a local operating region of the nonlinear plant. Figure (1) demonstrates the structure of the proposed fuzzy system. The proposed TS-KRR has five layers. In this section, the mathematical functions for each layer are presented in details.

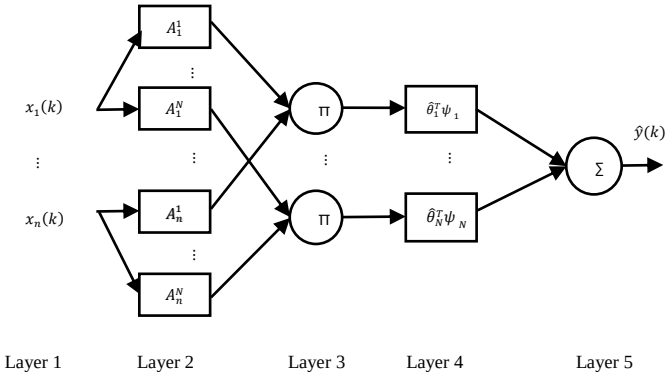


Figure 1: The configuration of the TS-KRR structure

Layer 1: represents the input variables of the model where these input signals are transmitted to layer 2. In layer 2 (or the *Fuzzification procedure*), each A_j^i represents a membership value of the *j*-th input variable $x_j(k)$ in rule *i*. The value of the *j*-th input variable that satisfies the quantity A_j^i is defined by the membership function $\mu_{A_j^i}(x_j)$. The output of each node in layer 3 (the rule layer) represents the product of all input signals of the node. In layer 3, each node represents an "*IF*" part of "*IF-THEN"* rule obtained by fuzzy logic operation "*AND*". The results obtained from node *i* in layer 3 gives the firing strength function $\mu_i(\boldsymbol{x})$. In layer 4, the output of each node *i* has the form of $\widehat{\boldsymbol{\theta}}_i\psi_i$ where ψ_i is the firing strength of node *i* in layer 3 multiplied by an input vector, and $\widehat{\boldsymbol{\theta}}_i$ is the consequent part of role *i*. Layer 4 can be seen as the layer that computes the consequent values of each node *i*. Finally, layer 5, which represents the *Defuzzification Operation*, computes the summation of all incoming signals from layer 4 and gives the estimated output $\hat{y}(k)$ of the nonlinear plant. As it appears, the Defuzzification Operation in the proposed TS-KRR model does not perform any normalization operations, and in the next section we will prove that the proposed fuzzy model does not need any kind of normalizations when the Kernel ridge regression algorithm is implemented to identify the consequent parameters $(\widehat{\boldsymbol{\theta}}_i,\ i = 1, \dots, N)$.

To compute the output $\hat{y}(k)$ of the proposed TS-KRR, the membership functions are defined as a Gaussian function:

$$\mu_{A_j^i}(x_j) = exp\left\{-\frac{(x_j - m_{ij})^2}{2{\sigma_{ij}}^2}\right\}, i = 1, \dots, N \text{ and } j = 1, \dots, n \quad (2)$$

where m_{ij} and σ_{ij} are the centre and the width of the membership function, respectively. The firing strength of each node in layer 3 is given as:

$$\mu_i(\boldsymbol{x}) = \prod_{j=1}^{n} \mu_{A_j^i}(x_j) = exp\left\{-\sum_{j=1}^{n} \frac{(x_j - m_{ij})^2}{2{\sigma_{ij}}^2}\right\} = exp\left\{-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{m}_i)^T \Sigma (\boldsymbol{x} - \boldsymbol{m}_i)\right\}, i = 1, \dots, N \quad (3)$$

where $\Sigma = \begin{bmatrix} {\sigma_{i1}}^2 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & {\sigma_{in}}^2 \end{bmatrix}^{-1}$. In layer 4, the function ψ_i is defined as $\psi_i = \boldsymbol{x}.\mu_i(\boldsymbol{x})$. Then, the output of the TS-KRR model is the following:

$$\hat{y}(k) = \sum_{i=1}^{N}\left(\widehat{\boldsymbol{\theta}}_i^T.\boldsymbol{x}\right).\mu_i(\boldsymbol{x}) = \sum_{i=1}^{N}\left(\widehat{\boldsymbol{\theta}}_i^T.\boldsymbol{x}\right).exp\left\{-\sum_{j=1}^{n}\left(\frac{(x_j - m_{ij})^2}{2{\sigma_{ij}}^2}\right)\right\} \quad (4)$$

Next section, the procedure of identifying the consequent vectors $\widehat{\boldsymbol{\theta}}_i, i = 1, \dots, N$ of the TS-KRR system as well as the values of the centre and the width of the membership functions are discussed in details.

III. IDENTIFICATION OF THE ANTECEDENT AND CONSEQUENT PARAMETERS OF THE TAKAGI-SUGENO FUZZY SYSTEM BASED KERNEL RIDGE REGRESSION

In this section, the kernel ridge regression will be applied to identify the TS-KRR consequent parameters. First, several definitions of the Kernel methods will be presented and then the algorithm will be described in detail. Next, the clustering based Particle Swarm Optimization (PSO) algorithm is presented to identify the antecedent parameters. Then, the proposed adaptive version of the Takagi-Sugeno system based Kernel ridge regression (TS-KRR) is introduced.

A. Basics of Kernel ridge regression

The kernel ridge regression (KRR) is a very recognized regression method in the area of nonlinear regressions. The idea of this regression method is to perform a linear regression in very high-dimensional spaces in an efficient way by exploiting the kernel trick. It is equivalent to performing nonlinear regression in the original input space.

Definition 1*:* A kernel function is a function $\kappa : \chi \to \mathbb{R}$ that for all $\boldsymbol{x}, \boldsymbol{z}$ from a nonempty set χ satisfies:

$$\kappa(\boldsymbol{x}, \boldsymbol{z}) = \langle \boldsymbol{\phi}(\boldsymbol{x}), \boldsymbol{\phi}(\boldsymbol{z}) \rangle \quad (5)$$

where $\boldsymbol{\phi}$ is a mapping from the set χ to a Hilbert space $\mathcal{F}$ (also called the feature space) and $\langle ,.,\rangle$ is the inner product operation. To verify that a function κ is a valid kernel, the properties of positive semi-definite kernel function need to be satisfied.

Theorem 1*:* a function $\kappa(\boldsymbol{x}, \boldsymbol{z})$ of a two points (or vectors) $\boldsymbol{x}, \boldsymbol{z}$ defined on χ^2 is a positive semi-definite kernel if it satisfies the Mercer's theorem (Mercer, 1909):

$$\sum_{i,j=1}^{N} \tau_i \tau_j \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 0 \quad (6)$$

for any numbers τ_i, τ_j and any points $\boldsymbol{x}_i, \boldsymbol{x}_j$ where $i.j = 1, \dots, N_D$, and N_D represents the size of training data**.** In this paper, the sum of kernel functions will be needed to define our model and to identify the consequent parameters. This leads to the following theorem:

Theorem 2*:* Let κ_1 and κ_2 be valid kernel functions, then $\kappa_1 + \kappa_2$ is a valid kernel function.

Proof*:*

It is easy to prove this using the Mercer's theorem in Eq. (6):

$$\sum_{i,j=1}^{N_D} \tau_i \tau_j \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{i,j=1}^{N_D} \tau_i \tau_j \left(\kappa_1(\boldsymbol{x}_i, \boldsymbol{x}_j) + \kappa_2(\boldsymbol{x}_i, \boldsymbol{x}_j)\right) = \sum_{i,j=1}^{N_D} \tau_i \tau_j \kappa_1(\boldsymbol{x}_i, \boldsymbol{x}_j) + \sum_{i,j=1}^{N_D} \tau_i \tau_j \kappa_2(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 0$$

Another important definition that needs to be mentioned in this section is the Representer Theorem (Schölkopf, Herbrich, & Smola, 2001). This theorem suggested that for a large class of optimization problems in reproducing kernel Hilbert space (RKHS), the solutions can be expressed as kernel expansions in terms of the training data only. According to Representer Theorem, the objective of kernel-based learning methods is to find a nonlinear relationship $f : \chi \to \mathbb{R}$ expressed as the following kernel expansion:

$$f(\boldsymbol{x}) = \sum_{i=1}^{N_D} \alpha_i \kappa(\boldsymbol{x}, \boldsymbol{x}_i) \tag{7}$$

where N_D is the number of available training data, $\alpha_i \in \mathbb{R}, i = 1, \dots, N_D$ are the expansion coefficients. Next, the regularized problem for kernel ridge regression is defined as (Saunders, Gammerman, & Vovk, 1998):

$$\min_{f \in \mathcal{F}} \sum_{i=1}^{N_D} \big(y_i - f(\boldsymbol{x}_i)\big)^2 + \lambda \|f\|_{\mathcal{F}}^2 \tag{8}$$

where $\lambda > 0$ and $f(\boldsymbol{x})$ is given in Eq. (7). Let define the vectors: $\boldsymbol{y} = \big(y_1, \dots, y_{N_D}\big)^T$, $\boldsymbol{\alpha} = \big(\alpha_1, \dots, \alpha_{N_D}\big)^T$ and $\boldsymbol{K} = \begin{pmatrix} \kappa(\boldsymbol{x}_1, \boldsymbol{x}_1) & \dots & \kappa\big(\boldsymbol{x}_1, \boldsymbol{x}_{N_D}\big) \\ \vdots & \dots & \vdots \\ \kappa(\boldsymbol{x}_N, \boldsymbol{x}_1) & \dots & \kappa\big(\boldsymbol{x}_N, \boldsymbol{x}_{N_D}\big) \end{pmatrix}$, then the problem in Eq. (8) can be expressed as:

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^T} (\boldsymbol{y} - \boldsymbol{K}\boldsymbol{\alpha})^T (\boldsymbol{y} - \boldsymbol{K}\boldsymbol{\alpha}) + \lambda \boldsymbol{\alpha}^T \boldsymbol{K} \boldsymbol{\alpha} \tag{9}$$

The minimisation of the quadratic problem in Eq. (9) is simple and the resulted expansion coefficients are given by:

$$\boldsymbol{\alpha} = (\boldsymbol{K} + \lambda \boldsymbol{I})^{-1} \boldsymbol{y} \tag{10}$$

And the matrix $\boldsymbol{I}$ is a $N_D \times N_D$ identity matrix.

B. Identification of the consequent parameters of the proposed TS-KRR

To obtain the TS-KRR formulation presented in Eq. (4), two choices has to be made: the number of kernel functions needed for the TS-KRR, and the form of the mapping $\boldsymbol{\phi}_l$ to define each kernel function κ_l (as in definition 1). The main idea of constructing the TS-KRR is to propose the form of the mapping and then the kernel functions. In this paper, the kernel function is defined such as:

$$\kappa(\boldsymbol{x}, \boldsymbol{z}) = \sum_{l=1}^{N} \kappa_l(\boldsymbol{x}, \boldsymbol{z}) = \sum_{l=1}^{N} \langle \boldsymbol{\phi}_l(\boldsymbol{x}), \boldsymbol{\phi}_l(\boldsymbol{z}) \rangle \tag{11}$$

where N is the number of rules for the Takagi-Sugeno system based Kernel ridge regression (TS-KRR), and the mapping $\boldsymbol{\phi}_l, l = 1, \dots, N$ are defined as:

$$\boldsymbol{\phi}_l(\boldsymbol{x}) = (\boldsymbol{x}).\mu_l(\boldsymbol{x}) = (\boldsymbol{x}).exp\left\{-\sum_{j=1}^{n} \frac{(x_j - m_{lj})^2}{2{\sigma_{lj}}^2}\right\}$$

$$= (\boldsymbol{x}).exp\left\{-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{m}_l)^T.\Sigma.(\boldsymbol{x} - \boldsymbol{m}_l)\right\}, l = 1, \dots, N \tag{12}$$

In Eq. (12), the vectors: $\boldsymbol{x} = (x_1, \dots, x_n)^T$ and $\boldsymbol{m}_l = (m_{1l}, \dots, m_{nl})^T$ are the input and the centroid vectors, respectively. The width vector is $\boldsymbol{\sigma}_l = (\sigma_{1l}, \dots, \sigma_{nl})^T$ and the matrix $\Sigma = \begin{bmatrix} {\sigma_{l1}}^2 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & {\sigma_{ln}}^2 \end{bmatrix}^{-1}$. As seen in Eq. (12), the mapping function $\boldsymbol{\phi}_l(\boldsymbol{x})$ is defined as a vector that has the same size of the input vector $\boldsymbol{x}$. Then, the result of the mapping inner product is: $\langle \boldsymbol{\phi}_l(\boldsymbol{x}), \boldsymbol{\phi}_l(\boldsymbol{z}) \rangle = \boldsymbol{\phi}_l(\boldsymbol{x})^T.\boldsymbol{\phi}_l(\boldsymbol{z})$. Moreover, it is clear that the functions $\kappa_l(x, z) = \langle \boldsymbol{\phi}_l(x), \boldsymbol{\phi}_l(z) \rangle, l = 1, \dots, N$ are valid kernel functions.

Proof:

$$\sum_{i,j=1}^{N_D} \tau_i \tau_j \kappa_l\big(\boldsymbol{x}_i, \boldsymbol{x}_j\big) = \sum_{i,j=1}^{N_D} \tau_i \tau_j \langle \boldsymbol{\phi}_l(\boldsymbol{x}_i), \boldsymbol{\phi}_l(\boldsymbol{x}_j) \rangle = \sum_{i,j=1}^{N_D} \langle \tau_i \boldsymbol{\phi}_l(\boldsymbol{x}_i), \tau_j \boldsymbol{\phi}_l(\boldsymbol{x}_j) \rangle$$

By using the properties of the inner product operation, it is obvious that:

$$\sum_{i,j=1}^{N_D} \tau_i \tau_j \kappa_l\big(\boldsymbol{x}_i, \boldsymbol{x}_j\big) = \langle \sum_{i=1}^{N_D} \tau_i \boldsymbol{\phi}_l(\boldsymbol{x}_i), \sum_{j=1}^{N_D} \tau_j \boldsymbol{\phi}_l(\boldsymbol{x}_j) \rangle = \left\| \sum_{i=1}^{N_D} \tau_i \boldsymbol{\phi}_l(\boldsymbol{x}_i) \right\|^2 \geq 0.$$

Then, $\kappa_l(\boldsymbol{x}, \boldsymbol{z}) = \langle \boldsymbol{\phi}_l(\boldsymbol{x}), \boldsymbol{\phi}_l(\boldsymbol{z}) \rangle, l = 1, \dots, N$ are valid kernel functions (κ_l was proved to be a valid kernel function without even replacing $\boldsymbol{\phi}_l(\boldsymbol{x})$ with its proposed form), and according to theorem 2, the function $\kappa(\boldsymbol{x}, \boldsymbol{z}) = \sum_{l=1}^{N} \kappa_l(\boldsymbol{x}, \boldsymbol{z})$ is a valid Kernel function as well.

By replacing Eq. (11) in Eq. (7), we obtain the following:

$$\begin{aligned} f(x) &= \sum_{i=1}^{N_D} \alpha_i \kappa(\boldsymbol{x}, \boldsymbol{x_i}) = \sum_{i=1}^{N_D} \alpha_i . \left(\sum_{l=1}^{N} \kappa_l(\boldsymbol{x}, \boldsymbol{x}_i) \right) \\ &= \sum_{i=1}^{N_D} \alpha_i \left(\sum_{l=1}^{N} \langle \boldsymbol{\phi}_l(\boldsymbol{x}), \boldsymbol{\phi}_l(\boldsymbol{x}_i) \rangle \right) \\ &= \sum_{i=1}^{N_D} \alpha_i . \left(\sum_{l=1}^{N} \boldsymbol{\phi}_l(\boldsymbol{x}_i)^T \boldsymbol{\phi}_l(\boldsymbol{x}) \right) \\ &= \sum_{l=1}^{N} \left(\sum_{i=1}^{N_D} \alpha_i . \boldsymbol{\phi}_l(\boldsymbol{x}_i)^T \right) \boldsymbol{\phi}_l(\boldsymbol{x}) \\ &= \sum_{l=1}^{N} \left(\sum_{i=1}^{N_D} \alpha_i . \boldsymbol{x}_i^T \mu_l(\boldsymbol{x}_i) \right) \boldsymbol{x} . \mu_l(\boldsymbol{x}) \end{aligned}$$

It is obvious that by setting $\widehat{\boldsymbol{\theta}}_l = \big(\sum_{i=1}^{N_D} \alpha_i \boldsymbol{\phi}_l(\boldsymbol{x}_i)\big) = \sum_{i=1}^{N_D} \alpha_i \boldsymbol{x}_i \mu_l(\boldsymbol{x}_i)$, then the Takagi-Sugeno system based Kernel ridge regression (TS-KRR) defined in Eq. (4) is attained where the consequent parameters are:

$$\widehat{\boldsymbol{\theta}}_l = \sum_{i=1}^{N_D} \alpha_i \boldsymbol{x}_i . \mu_l(\boldsymbol{x}_i) = \sum_{i=1}^{N_D} \alpha_i \boldsymbol{x}_i . exp\left\{ - \sum_{j=1}^{n} \frac{(x_j - m_{lj})^2}{2\sigma_{lj}{}^2} \right\} \quad (13)$$

and the model is:

$$f(x) = \sum_{l=1}^{N} \widehat{\boldsymbol{\theta}}_l^T \boldsymbol{\phi}_l(\boldsymbol{x}) = \sum_{l=1}^{N} \widehat{\boldsymbol{\theta}}_l^T \boldsymbol{x} . \mu_l(\boldsymbol{x}) = \sum_{l=1}^{N} \left(\widehat{\boldsymbol{\theta}}_l^T . \boldsymbol{x}\right) . exp\left\{ - \sum_{j=1}^{n} \frac{(x_j - m_{lj})^2}{2\sigma_{lj}{}^2} \right\} \quad (14)$$

Clearly, for a certain training input-output dataset, the consequent parameters of the proposed TS fuzzy model are identified using expansion coefficients $\boldsymbol{\alpha} = \left(\alpha_1, \dots, \alpha_{N_D}\right)^T$ which can be easily obtained from the quadratic problem in Eq. (9). The only missing part of the TS-KRR is the antecedent parameters (number of rules, $\boldsymbol{m}_l$ and $\boldsymbol{\sigma}_l$). In this paper, a clustering algorithm will be used to identify the antecedent parameters. In the next subsection, a clustering based PSO algorithm number is discussed where the number of rules will be defined as the number of clusters, while $\boldsymbol{m}_l$ and $\boldsymbol{\sigma}_l$ are the centroid and the width of each cluster l, respectively.

C. Identification of the antecedent parameters of the proposed TS-KRR

In this subsection, the antecedent parameters of the TS-KRR fuzzy model are identified using a clustering algorithm, where the centroid and the width of the clusters will represent the vectors $\boldsymbol{m}_l$ and $\boldsymbol{\sigma}_l$ of the kernel functions. There are many possibilities to choose a clustering algorithm. In the case of an adaptive TS-KRR fuzzy model (which will be presented later), a simple K-means clustering (Forgy, 1965) algorithm could be sufficient to initialise the algorithm while the vectors $\boldsymbol{m}_l$ and $\boldsymbol{\sigma}_l$ are adjusted in each iteration. However, for offline TS-KRR model an accurate clustering algorithm helps getting better approximations.

In this paper, a clustering based PSO algorithm is used for the offline TS-KRR model. This algorithm is proposed by Van der Merwe et al. (Van der Merwe, & Engelbrecht, 2003). In this algorithm, first a K-means clustering (Forgy, 1965) algorithm is used to initialise the initial swarm. Then, the PSO algorithm refines the clusters formed by the K-means.

The K-means algorithm is a very recognized tool for data clustering. This algorithm attempts to assemble data vectors into a predefined number of clusters, where the Euclidean distance is used as a similarity measure. As a result, the data vectors within the same cluster have small Euclidean distances from one another. Each cluster is then associated with one centroid vector, which represents the "midpoint" of that cluster (the centroid vector is the mean of the data that belongs to the same cluster). Furthermore, the width of each cluster is defined by the variance of the data belongs to the same cluster.

The Particle swarm optimization (PSO) is a population-based optimization algorithm that has been introduced by Kennedy and Eberhart (Kennedy, & Eberhart, 1995). This algorithm was inspired from the behaviour of flocks of birds (particles) in nature. In the standard PSO algorithm, particles (or individuals) represent potential solutions to the optimization problem where each particle i is composed of three vectors; its position in the search space, which is given by $\breve{\boldsymbol{x}}_{\boldsymbol{i}} = (\breve{x}_i^1, \dots, \breve{x}_i^d)$, the best position that has been found by the individual (or particle) $\boldsymbol{p}_{\boldsymbol{i}} = (p_i^1, \dots, p_i^d)$ and the velocity $\boldsymbol{v}_{\boldsymbol{i}} = (v_i^1, \dots, v_i^d)$ of the individual i. In this algorithm, both: positions and the velocities of the particles are randomly initialized in the search space. During the implementation of the PSO algorithm, each particle moves around the search space by updating its velocity and position vector. Various equations can be used to update the velocity of particles. However, the version with the inertia weight will be used in this paper. The velocity equation that updates particle i is given by:

$$\boldsymbol{v}_{\boldsymbol{i}} = w\boldsymbol{v}_{\boldsymbol{i}} + c_1 r_1 (\boldsymbol{p}_{\boldsymbol{i}} - \breve{\boldsymbol{x}}_{\boldsymbol{i}}) + c_2 r_2 \left(\boldsymbol{p}_{\boldsymbol{i}}^{\boldsymbol{g}} - \breve{\boldsymbol{x}}_{\boldsymbol{i}}\right) \quad (15)$$

where the influence of the previous velocities in Eq. (15) is adjusted by introducing an inertia weight w. r_1 and r_2 are random values. c_1 and c_2 are the local and global influence factors, respectively. The simulations inertia weight is reduced during the algorithm evaluation to allow local search, where $w = \frac{iter\_\max - iter}{iter\_max} \times w_0$, where w_0 is the initial weight, $iter$ is the current iteration of the algorithm while the total number of iterations is defined by $iter\_max$.

The position vector is then updated using:

$$\breve{\boldsymbol{x}}_{\boldsymbol{i}} = \breve{\boldsymbol{x}}_{\boldsymbol{i}} + \boldsymbol{v}_{\boldsymbol{i}} \quad (16)$$

The PSO algorithm is terminated after running Eqs. (15) and (16) for a specified number of iterations $iter\_max$.

To adjust the PSO algorithm for clustering, a particle $\breve{\boldsymbol{x}}_{\boldsymbol{i}}$ contains all centroid of the clusters such as $\breve{\boldsymbol{x}}_{\boldsymbol{i}} = (\boldsymbol{m}_i^1, \dots, \boldsymbol{m}_i^N)$, where all particles represent a possible solution for an accurate centroids. The fitness function is then defined by:

$$J = \frac{\sum_{i=1}^{N} \left(\sum_{\forall\, \boldsymbol{z}_p \in C_i} \frac{d(\boldsymbol{z}_p, \boldsymbol{m}_i)}{|C_i|} \right)}{N} \quad (17)$$

where C_i represents the cluster i and $|C_i|$ is the number of data vectors belongs to the cluster C_i. The distance is defined as: $d(\boldsymbol{z}_p, \boldsymbol{m}_i) = \sqrt{\sum_{j=1}^{n} (z_j - m_{ij})^2}$, and for each cluster, the centroid is calculated as $\boldsymbol{m}_{\boldsymbol{i}} = \frac{\boldsymbol{1}}{|C_i|} \sum_{\forall\, \boldsymbol{z}_p \in C_i} \boldsymbol{z}_p$ and the widths of clusters are: $\sigma_{ij} = \frac{\boldsymbol{1}}{|C_i|} \left(\sum_{\forall\, \boldsymbol{z}_{\boldsymbol{p}} \in C_i} (z_{pj} - m_{ij})^{\boldsymbol{2}} \right)^{\frac{1}{2}}$. Note that $\boldsymbol{z}_p$ represents particle $\breve{\boldsymbol{x}}_p$ that belongs to the cluster C_i.
The clustering algorithm is summarized as follows:

Algorithm 1 (PSO clustering):

1) Initialise each particle to contain N centroids. The particles are randomly initialised except for one particle which has the results of the K-means algorithm.

2) For i=1: $iter\_max$
   a) For each particle i do
      i) For each data vector $\boldsymbol{z}_p$ do
         (1) The distance to all centroids is calculated.
         (2) Assign the vector $\boldsymbol{z}_p$ to the cluster that has minimum distance
         (3) Calculate the fitness using Eq. (17)
      ii) Updates the global and local best positions using Eqs. (15) and (16).
      iii) Update the clusters centroids and widths.

Finally, to perform an identification procedure using the offline TS-KRR, the following steps must be executed:

Algorithm 2: (offline TS-KRR)

1) Initialize the input size, number of clusters N, the PSO parameters (w, c_1, c_2, $iter\_max$ and the number of particles).
2) Perform Algorithm 1.
3) Obtain the expansion coefficients using Eq. (10).
4) Finally, obtain the estimated $\widehat{\boldsymbol{\theta}}_l$ from Eq. (13).
5) Obtain the model in Eq. (4) (The same model in Eq. (14)).

D. The adaptive TS-KRR algorithm

In the previous 2 subsections, the off-line TS-KRR was introduced where the offline TS-KRR model can be easily obtained from the input-output data that belongs to certain plants. However, the main disadvantage of the off-line algorithms is that the collected data is limited and the model obtained may not be sufficiently accurate. This inspires the implementation of the adaptive (online) version of the TS-KRR algorithm for identifications and control systems. Adaptive techniques are designed to learn from one data instance at a time. They are typically used in real-time scenarios, such as control systems or tracking problems, where the data arrive sequentially and instant decisions must be made. The adaptive TS-KRR algorithm consists of two steps. The first step is to adjust the antecedent parameters ($\boldsymbol{m}_l$ and $\boldsymbol{\sigma}_l$, $l = 1, \dots, N$) where this can be done by a simple back-propagation learning algorithm. The second step is to use an online regression method to update the consequent parameters. The easiest way is to use the famous RLS algorithm to update the $\widehat{\boldsymbol{\theta}}_l, l = 1, \dots, N$. However, the RLS algorithm has been used many times. Also, the accuracy of this algorithm is not ensured since the proposed TS-KRR model does not require normalization in the defuzzification Operation. In this paper, an alternative scenario is investigated where a modified version of the Kernel RLS (sliding-window kernel RLS) algorithm is exploited to update these parameters. The idea is then to recursively update the expansion coefficients, and then use these coefficients to obtain the $\widehat{\boldsymbol{\theta}}_l, l = 1, \dots, N$.

To update the parameters m_{ij} and σ_{ij} of the TS-KRR in Eq. (4), the following error function is introduced:

$$e(k) = \frac{1}{2}(y(k) - \hat{y}(k))^2 \tag{18}$$

where $y(k)$ is the actual output and $\hat{y}(k)$ is the approximated output obtained by the TS-KRR at the instant k. By applying the backpropagation learning algorithm, the updated values of m_{ij} and σ_{ij} at the instant $k+1$ are given by:

$$\begin{aligned} m_{ij}(k+1) &= m_{ij}(k) - \eta \frac{\partial e(k)}{\partial m_{ij}} \\ &= m_{ij}(k) + \eta(y(k) - \hat{y}(k)) \frac{\partial \hat{y}(k)}{\partial m_{ij}} \end{aligned} \tag{19}$$

$$\begin{aligned} \sigma_{ij}(k+1) &= \sigma_{ij}(k) - \eta \frac{\partial e(k)}{\partial \sigma_{ij}} \\ &= \sigma_{ij}(k) + \eta(y(k) - \hat{y}(k)) \frac{\partial \hat{y}(k)}{\partial \sigma_{ij}} \end{aligned} \tag{20}$$

where η is a positive learning rate, and:

$$\begin{aligned} \frac{\partial \hat{y}(k)}{\partial m_{ij}} &= \frac{\partial \hat{y}(k)}{\partial \phi_l(x)} . \frac{\partial \phi_l(x)}{\partial \mu_{A_j^i}(x_j)} . \frac{\partial \mu_{A_j^i}(x_j)}{\partial m_{ij}} \\ &= \boldsymbol{\phi}_l{}^T(\boldsymbol{x}) . \left(\boldsymbol{\phi}_l(\boldsymbol{x}) . \left(\sum_{l=1}^{M} \alpha_i\right) + \widehat{\boldsymbol{\theta}}_i\right) . \frac{(x_j - m_{ij})}{\sigma_{ij}{}^2} \end{aligned} \tag{21}$$

$$\begin{aligned} \frac{\partial \hat{y}(k)}{\partial \sigma_{ij}} &= \frac{\partial \hat{y}(k)}{\partial \phi_l(x)} . \frac{\partial \phi_l(x)}{\partial \mu_{A_j^i}(x_j)} . \frac{\partial \mu_{A_j^i}(x_j)}{\partial \sigma_{ij}} \\ &= \boldsymbol{\phi}_l{}^T(\boldsymbol{x}) . \left(\boldsymbol{\phi}_l(\boldsymbol{x}) . \left(\sum_{l=1}^{M} \alpha_i\right) + \widehat{\boldsymbol{\theta}}_i\right) . \frac{(x_j - m_{ij})^2}{\sigma_{ij}{}^3} \end{aligned} \tag{22}$$

and M is the size of the dictionary used by the sliding-window kernel RLS algorithm. To update the expansion coefficients, the kernel RLS method (Engel, Mannor, & Meir, 2004) is implemented. In addition, the modifications made in the sliding-window kernel RLS are presented. Suppose that $k-1$ data have been received and processed at the $(k-1)$-th iteration. The regression solution from Eq. (10) can be expressed as:

$$\boldsymbol{\alpha}_{k-1} = \acute{\boldsymbol{K}}_{k-1}^{-1} \boldsymbol{y}_{k-1} \tag{23}$$

The matrix $\acute{\boldsymbol{K}}$ in Eq. (23) is the regularized kernel matrix $\acute{\boldsymbol{K}} = \boldsymbol{K} + \lambda \boldsymbol{I}$. When a new data pair $(\boldsymbol{x}(k), y(k))$ is arrived, several steps are required to update Eq. (23) recursively. First, the predicted output will be calculated such as: $\hat{y}(k) = \boldsymbol{\kappa}_k^T \boldsymbol{\alpha}_{l-1}$, where $\boldsymbol{\kappa}_k = (\kappa(\boldsymbol{x}(k), \boldsymbol{x}(1)), \dots, \kappa(\boldsymbol{x}(k), \boldsymbol{x}(k-1)))^T$, and the error between the real and the predicted output is $e(k) = y(k) - \hat{y}(k)$. The kernel matrix at the instance k is then given by:

$$\acute{\boldsymbol{K}}_k = \begin{pmatrix} \acute{\boldsymbol{K}}_{k-1} & \boldsymbol{\kappa}_k \\ \boldsymbol{\kappa}_k^T & \kappa\big(\boldsymbol{x}(k), \boldsymbol{x}(k)\big) + \lambda \end{pmatrix} \tag{24}$$

By introducing the new variables: $\boldsymbol{h}_k = \acute{\boldsymbol{K}}_{k-1}^{-1} \boldsymbol{\kappa}_k$ and $\gamma_k = \kappa\big(\boldsymbol{x}(k), \boldsymbol{x}(k)\big) + \lambda - \boldsymbol{\kappa}_k^T \boldsymbol{a}_k$, the new inverse matrix is defined by:

$$\acute{\boldsymbol{K}}_k^{-1} = \frac{1}{\gamma_k}\begin{pmatrix}\gamma_k \acute{\boldsymbol{K}}_{k-1}^{-1} + \boldsymbol{h}_k \boldsymbol{h}_k^T & -\boldsymbol{h}_k \\ -\boldsymbol{h}_k & 1\end{pmatrix} \quad (25)$$

Finally, the updated values of the expansion coefficients are:

$$\boldsymbol{\alpha}_k = \begin{pmatrix}\boldsymbol{\alpha}_{k-1} - \frac{\boldsymbol{h}_k e(k)}{\gamma_k} \\ \frac{e(k)}{\gamma_k}\end{pmatrix} \quad (26)$$

The size of the Kernel inverse matrix increases with time, which is difficult to be implemented in real time applications. To deal with the size problem, a modified version with a fixed dictionary size, the sliding-window kernel RLS algorithm, is implemented to update the expansion coefficients. The idea of this algorithm is to store a fixed size of the last M data as its dictionary. In each step, the algorithm adds the new arrived data pair $(\boldsymbol{x}(k), y(k))$ and removes the oldest one from its dictionary (which is obvious since the latest data are more relevant than the old one), and this will lead to a sliding-window approach.

Given the kernel matrix $\acute{\boldsymbol{K}}_{k-1}$, the new regularized kernel $\acute{\boldsymbol{K}}_k$ at the instance k is constructed by removing the first row and column of $\acute{\boldsymbol{K}}_{k-1}$ **(***downsizing* the kernel matrix step**)**, and the resulting matrix is denoted as $\overline{\boldsymbol{K}}_{k-1}$. Then, kernels of the new arrived data are added to $\overline{\boldsymbol{K}}_{k-1}$as the last row and column of this matrix:

$$\acute{\boldsymbol{K}}_k = \begin{pmatrix}\overline{\boldsymbol{K}}_{k-1} & \boldsymbol{\kappa}_k \\ \boldsymbol{\kappa}_k^T & \kappa\big(\boldsymbol{x}(k), \boldsymbol{x}(k)\big) + \lambda\end{pmatrix} \quad (27)$$

the vector $\boldsymbol{\kappa}_k = (\kappa(\boldsymbol{x}(k), \boldsymbol{x}(k-M)), \dots, \kappa(\boldsymbol{x}(k), \boldsymbol{x}(k-1)))^T$. The inverse of this matrix can be easily obtained using the same steps used earlier for matrix in Eq. (24). Then, the inverse matrix $\acute{\boldsymbol{K}}_k^{-1}$ is described as:

$$\acute{\boldsymbol{K}}_k^{-1} = \frac{1}{\bar{\gamma}_k}\begin{pmatrix}\bar{\gamma}_k \overline{\boldsymbol{K}}_{k-1}^{-1} + \bar{\boldsymbol{h}}_k \bar{\boldsymbol{h}}_k^T & -\bar{\boldsymbol{h}}_k \\ -\bar{\boldsymbol{h}}_k & 1\end{pmatrix} \quad (28)$$

where $\bar{\boldsymbol{h}}_k = \overline{\boldsymbol{K}}_{k-1}^{-1}\boldsymbol{\kappa}_k$ and $\bar{\gamma}_k = \kappa\big(\boldsymbol{x}(k), \boldsymbol{x}(k)\big) + \lambda - \boldsymbol{\kappa}_k^T \bar{\boldsymbol{h}}_k$, and the inverse matrix $\overline{\boldsymbol{K}}_{k-1}^{-1}$ can be obtained from the matrix $\acute{\boldsymbol{K}}_{k-1}^{-1}$ (which is already recursively obtained in the previous instant or iteration). First, the inverse matrix $\acute{\boldsymbol{K}}_{k-1}^{-1}$ is divided as: $\acute{\boldsymbol{K}}_{k-1}^{-1} = \begin{pmatrix} s & \boldsymbol{p}^T \\ \boldsymbol{p} & \boldsymbol{U}\end{pmatrix}$, and we already know that:

$$\acute{\boldsymbol{K}}_{k-1} = \begin{pmatrix}\kappa\big(\boldsymbol{x}(k-M), \boldsymbol{x}(k-M)\big) + \lambda & \hat{\boldsymbol{\kappa}}_{k-1} \\ \hat{\boldsymbol{\kappa}}_{k-1}^T & \overline{\boldsymbol{K}}_{k-1}\end{pmatrix} \quad (29)$$

where M is the size of dictionary and $\hat{\boldsymbol{\kappa}}_{k-1} = (\kappa(\boldsymbol{x}(k-1), \boldsymbol{x}(k-M+1)), \dots, \kappa(\boldsymbol{x}(k-1), \boldsymbol{x}(k-2)))^T$. Then, the inverse matrix $\overline{\boldsymbol{K}}_{k-1}^{-1}$ is obtained as:

$$\overline{\boldsymbol{K}}_{k-1}^{-1} = \boldsymbol{U} - \frac{\boldsymbol{p}\boldsymbol{p}^T}{s} \quad (30)$$

The sliding-window kernel RLS algorithm is summarized as follows:

Algorithm 3 (sliding-window kernel RLS algorithm):

1. Compute the error $e(k)$
2. Update inverse matrix in Eq. (25)
3. : If the dictionary size is more than *M*
   (1) Calculate the inverse matrix in Eq. (30)
   (2) Calculate the matrix in Eq. (27)
   (3) Calculate the inverse matrix in Eq. (28)
   (4) Update expansion coefficients $\boldsymbol{\alpha}_k = \acute{\boldsymbol{K}}_k^{-1}\boldsymbol{y}_k$
4. Else :
   (1) Update expansion coefficients in Eq. (26)

After obtaining expansion coefficients, the consequent parameters $\widehat{\boldsymbol{\theta}}_l, l = 1, \dots, N$ are updated using only M pairs of the input-output data (the size of dictionary) such as:

$$\widehat{\boldsymbol{\theta}}_l = \sum_{i=1}^{M} \alpha_i . (\boldsymbol{x}_i) . exp\left\{-\sum_{j=1}^{n} \frac{(x_j - m_{lj})^2}{2\sigma_{lj}{}^2}\right\} \quad (31)$$

The adaptive TS-KRR is summarized as follows:

Algorithm 4: (online TS-KRR steps)

1. Initialize the input size, number of clusters N)
2. Execute K-means algorithm to initialize the antecedent parameters (an offline procedure).
3. Set the size of the dictionary *M*
4. Measure the output signal
5. Update m_{ij}, σ_{ij} in Eq.(19) and (20).
6. Run Algorithm 3 to obtain the updated expansion coefficients.
7. Obtain the estimated variables $\widehat{\boldsymbol{\theta}}_l$ from Eq. (31), and obtain the model in Eq. (4)
8. Go to 4

Note that, only K-means algorithm is performed in the adaptive TS fuzzy system since the antecedent parameters are going to be updated further at each instant. Also, the updated value of the learning rate η will be introduced later in the simulation example.

IV. DERIVATION OF THE GENERALIZED PREDICTIVE CONTROL LAW

In this section, a generalized predictive control law is described. The GPC controller is implemented by removing the nonlinear effects of the system, which is done by replacing the original nonlinear plants with the TS-KRR when the control signal is computed. A nonlinear plant is considered to have the following general nonlinear structure:

$$y(k) = f\Big(y(k-1), y(k-2), \dots, y\big(k-n_y\big), u(k-d), u(k-d-1), \dots, u(k-n_u)\Big) \quad (32)$$

where y and u are the output and the control signals of the nonlinear system, respectively. The function f represents a nonlinear mapping that describes the relation between the output and the control signals, where the form of this mapping function is assumed to be unknown. n_u and n_y are the orders of the control signal and the output respectively, and d is the time delay of the system. The nonlinear mapping $f(.)$ is approximated using several local affine models where the system in Eq. (32) can be described by the following TS fuzzy rules (similar to the TS rules in Eq. (1)):

$$\begin{aligned} R_i: \quad & \text{IF } x_1(k) \text{ is } A_1^i, \text{and } \dots, \text{and } x_n(k) \text{ is } A_n^i \\ & \text{THEN } y_i(k) = a_i(z^{-1})y(k-1) + b_i(z^{-1})u(k-d-1) \\ & i = 1, \dots, N \end{aligned} \quad (33)$$

where the input vector: $x(k) = \big(x_1(k), \dots, x_n(k)\big) = \big(y(k-1), \dots, y(k-n_a), u(k-d-1), \dots, u(k-n_b)\big)$, $a_i(z^{-1})$ and $b_i(z^{-1})$ are linear polynomials with $n_a \le n_y$ and $n_b \le n_u$ such as:

$$\begin{aligned} a_i(z^{-1}) &= a_i^1 z^{-1} + \cdots + a_i^{n_a} z^{-n_a} \\ b_i(z^{-1}) &= b_i^0 + b_i^1 z^{-1} + \cdots + b_i^{n_b} z^{-n_b} \end{aligned} \quad (34)$$

From Eq. (33), the estimated TS-KRR model is defined as:

$$\hat{y}(k) = \sum_{i=1}^{N} \big(a_i(z^{-1})y(k-1) + b_i(z^{-1})u(k-d-1)\big).exp\left\{-\sum_{j=1}^{n}\left(\frac{(x_j - m_{ij})^2}{2\sigma_{ij}{}^2}\right)\right\} \quad (35)$$

where $\widehat{\boldsymbol{\theta}}_i = (a_i,\ b_i)\ i = 1, \dots, N$ is obtained by Eq. (13) (or Eq. (31) in the case of an adaptive control). To obtain the control signal, Eq. (35) is written as follows:

$$\bar{A}(z^{-1})y(k) = \bar{B}(z^{-1})u(k-d-1) \quad (36)$$

with $\bar{A}(z^{-1}) = 1 - \bar{a}^1 z^{-1} - \cdots - \bar{a}^{n_a} z^{-n_a}$ and $\bar{B}(z^{-1}) = \bar{b}^0 + \bar{b}^1 z^{-1} + \cdots + \bar{b}^{n_b} z^{-n_b}$, and:

$$\begin{aligned} \bar{a}^j &= \sum_{i=1}^{N} a_i^j.exp\left\{-\sum_{j=1}^{n}\left(\frac{(x_j - m_{ij})^2}{2\sigma_{ij}{}^2}\right)\right\} \\ \bar{b}^j &= \sum_{i=1}^{N} b_i^j.exp\left\{-\sum_{j=1}^{n}\left(\frac{(x_j - m_{ij})^2}{2\sigma_{ij}{}^2}\right)\right\} \end{aligned} \quad (37)$$

The control law of the GPC algorithm is obtained to minimize the following cost function:

$$\begin{aligned} J(k) = & \sum_{j=d+1}^{N_p} \big(\hat{y}(k+j|k) - w(k+j)\big)^2 \\ & + \sum_{j=d+1}^{N_u+d-1} \vartheta(\mathrm{z}^{-1})\Delta u(k+j-d-1|k)^2 \end{aligned} \quad (38)$$

where $\hat{y}(k+j|k)$ is an optimum j step ahead prediction of the system on time k. N_p and N_u are the output and control signal horizons, respectively. $w(k+j)$ is the reference trajectory. $\vartheta(\mathrm{z}^{-1}) = \vartheta_0 + \vartheta_1 \mathrm{z}^{-1} + \cdots + \vartheta_{N_p+n_b-1} \mathrm{z}^{-(N_p+n_b-1)}$ is the weighting polynomial, and $\Delta = 1 - \mathrm{z}^{-1}$. First, we consider the following Diophantine equation:

$$1 = E_j(\mathrm{z}^{-1})\Delta\bar{\mathrm{A}}(\mathrm{z}^{-1}) + \mathrm{z}^{-j}F_j(\mathrm{z}^{-1}) \quad (39)$$

The degrees of the polynomials $E_j(\mathrm{z}^{-1})$ and $F_j(\mathrm{z}^{-1})$ are $j-1$ and n_a, respectively. To simplify the Diophantine equation, $\tilde{A}(\mathrm{z}^{-1})$ is introduced such as $\tilde{A}(\mathrm{z}^{-1}) = \Delta\bar{\mathrm{A}}(\mathrm{z}^{-1})$. Both $E_j(\mathrm{z}^{-1})$ and $F_j(\mathrm{z}^{-1})$ are obtained when 1 is divided by $\tilde{A}(\mathrm{z}^{-1})$. If Eq. (36) is multiplied by $\Delta E_j(\mathrm{z}^{-1})\mathrm{z}^j$ and the Eq. (39) is used for simplification, then the best prediction of $y(k+j|k)$ is:

$$y(k+j|k) = F_j(\mathrm{z}^{-1})y(k) + E_j(\mathrm{z}^{-1})\bar{B}(z^{-1})u(k+j-d-1) \quad (40)$$

where $G_j(\mathrm{z}^{-1}) = E_j(\mathrm{z}^{-1})\bar{B}(z^{-1})$. The polynomials $E_j(\mathrm{z}^{-1})$ and $F_j(\mathrm{z}^{-1})$ expressed in Eq. (41) are obtained recursively (see Clarke et al., 1989 for more details).

$$\begin{aligned} E_j(\mathrm{z}^{-1}) &= 1 + e_{j,1}\mathrm{z}^{-1} + \cdots + e_{j,j-1}\mathrm{z}^{-(j-1)} \\ F_j(\mathrm{z}^{-1}) &= f_{j,0} + f_{j,1}\mathrm{z}^{-1} + \cdots + f_{j,n_a}\mathrm{z}^{-n_a} \end{aligned} \quad (41)$$

Given $E_j(\mathrm{z}^{-1})$, the polynomial $E_{j+1}(\mathrm{z}^{-1})$ is obtained as follows:

$$E_{j+1}(\mathrm{z}^{-1}) = E_j(\mathrm{z}^{-1}) + \mathrm{z}^{-j}e_{j+1,j} \quad (42)$$

where $e_{j+1,j} = f_{j,0}$. The coefficient of $F_j(\mathrm{z}^{-1})$ are obtained as:

$$f_{j+1,i} = f_{j,i+1} - f_{j,0}\tilde{a}_{i+1}, i = 0, \dots, n_a - 1 \quad (43)$$

where $f_{j,n_a} = 0$. The coefficients of $G_j(\mathrm{z}^{-1})$ are also obtained recursively where the first j coefficients of $G_{j+1}(\mathrm{z}^{-1})$ are equal to $G_j(\mathrm{z}^{-1})$ coefficients. The rest of the coefficients are obtained such as:

$$g_{j+1,j+i} = g_{j,j+i} - f_{j,0}b_i, i = 0, \dots, n_b \quad (44)$$

From Eq. (40), we can write the following:

$$\boldsymbol{y}(k) = \boldsymbol{G}\boldsymbol{u}(k) + \boldsymbol{F}(\mathrm{z}^{-1})y(k) + \boldsymbol{L}(\mathrm{z}^{-1}) \quad (45)$$

where

$$\boldsymbol{y}(k) == \begin{bmatrix} y(k+d+1) \\ y(k+d+2) \\ \vdots \\ y(k+N_p) \end{bmatrix}, \quad \boldsymbol{u}(k) = \begin{bmatrix} \Delta u(k) \\ \Delta u(k+1) \\ \vdots \\ \Delta \mathrm{u}(k+N_u-1) \end{bmatrix} \quad (46)$$

$$\boldsymbol{F}(\mathrm{z}^{-1}) = \begin{bmatrix} F_{d+1}(\mathrm{z}^{-1}) \\ F_{d+2}(\mathrm{z}^{-1}) \\ \vdots \\ F_{N_p}(\mathrm{z}^{-1}) \end{bmatrix}, \quad \boldsymbol{G} = \begin{bmatrix} g_{1,0} & 0 & \dots & 0 \\ g_{2,1} & g_{2,0} & \dots & 0 \\ \vdots & \vdots & \ddots & 0 \\ g_{N_p,N_p-1} & g_{N_p,N_p-2} & \cdots & g_{N_p,N_p-N_u} \end{bmatrix} \quad (47)$$

$$\boldsymbol{L}(\mathrm{z}^{-1}) = \begin{bmatrix} \left(G_{d+1}(\mathrm{z}^{-1}) - \tilde{G}_{d+1}(\mathrm{z}^{-1})\right)\mathrm{z}\Delta u(k-1) \\ \left(G_{d+1}(\mathrm{z}^{-1}) - \tilde{G}_{d+1}(\mathrm{z}^{-1})\right)\mathrm{z}^{2}\Delta u(k-1) \\ \vdots \\ \left(G_{N_p}(\mathrm{z}^{-1}) - \tilde{G}_{N_p}(\mathrm{z}^{-1})\right)\mathrm{z}^{N_p}\Delta u(k-1) \end{bmatrix} \quad (48)$$

and $\tilde{G}_j(\mathrm{z}^{-1}) = g_{j,0} + g_{j,1}\mathrm{z}^{-1} + \cdots + g_{j,j-d-1}\mathrm{z}^{d+1-j}$. Next, the weighting polynomial is set to be constant: $\vartheta(\mathrm{z}^{-1}) = \vartheta$, and Eq. (45) is used to simply the cost function described in Eq. (38). Then, the new form of the cost function is given by:

$$J(k) = (\boldsymbol{G}\boldsymbol{u}(k) + \boldsymbol{F}(\mathrm{z}^{-1})y(k) + \boldsymbol{L}(\mathrm{z}^{-1}) - \boldsymbol{W})^{T}(\boldsymbol{G}\boldsymbol{u}(k) + \boldsymbol{F}(\mathrm{z}^{-1})y(k) + \boldsymbol{L}(\mathrm{z}^{-1}) - \boldsymbol{W}) + \vartheta.\boldsymbol{u}(k)^{T}\boldsymbol{u}(k) \quad (49)$$

where the vector $\boldsymbol{W} = \left(w(k+d+1), \ldots, w(k+N_p)\right)^{T}$. By minimizing the Eq. (49) ($\frac{\partial J(k)}{\partial u(k)} = 0$), the obtained solution is given by:

$$\boldsymbol{u}(k) = (\boldsymbol{G}^{T}\boldsymbol{G} + \vartheta\boldsymbol{I})^{-1}\boldsymbol{G}^{T}(\boldsymbol{W} - \boldsymbol{F}(\mathrm{z}^{-1})y(k) - \boldsymbol{L}(\mathrm{z}^{-1})) \quad (50)$$

where $\boldsymbol{I}$ is an identity matrix. The control signal sent to the process is only the first element of the vector $\boldsymbol{u}(k)$. In this case, the increment of the control signal is:

$$\Delta u(k) = \boldsymbol{\Gamma}(\boldsymbol{W} - \boldsymbol{F}(\mathrm{z}^{-1})y(k) - \boldsymbol{L}(\mathrm{z}^{-1})) \quad (51)$$

where $\boldsymbol{\Gamma}$ is the first row of the matrix $(\boldsymbol{G}^{T}\boldsymbol{G} + \vartheta\boldsymbol{I})^{-1}\boldsymbol{G}^{T}$.

V. ILLUSTRATIVE EXAMPLE 1: IDENTIFICATION AND CONTROL OF A SURGE TANK SYSTEM

In this subsection, the surge tank system (Eski, & Temürlenk, 2013) represented in Figure (2) is considered for system identification using the TS-KRR algorithm (and control using TS-KRR GPC controller). The mathematical model of this nonlinear system is given by:

$$\frac{dh_s(t)}{dt} = \frac{-c_s\sqrt{2g_s h_s(t)}}{A(h_s(t))} + \frac{1}{A(h_s(t))}u(t) \quad (52)$$

where $u(t)$ is the control input (the input flow of the system), $h_s(t)$ is the liquid level in the tank and the constants $c_s = 1$ and $g_s = 9.8$. The cross section area of the tank is defined as: $A\left(h_s(t)\right) = a_s\left(h_s(t)\right)^{2} + b_s$. The constants: $a_s = 0.01$ and $b_s = 0.2$.

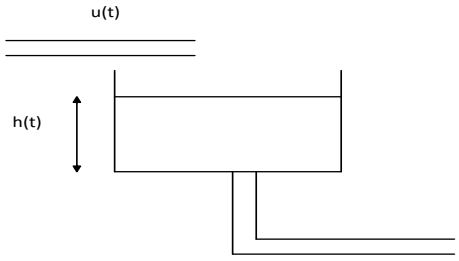


Figure 2: The surge tank system

A. *The offline Identification*

To obtain the offline TS-KRR system of the surge tank system, the sampling time is set to $t_s = 0.1\,sec$ and the nonlinear mathematical model presented in Eq. (52) is used to generate $N_S = 900$ samples; the first 400 samples were used to train the TS-KRR system, while the rest of these samples were used to validate the proposed TS-KRR method. The control signal represented in Figure (4b) was used to generate these 900 samples. The size of the input variable vector is set to 4 ($n_a = 3$ and $n_b = 1$).

Generally, there are several methods to identify the right number of clusters from the training data. Mendes et al. (Mendes, Araújo, Souza, 2013) used the number of the operating zones (or the hyper-planes) of the system described by the training data as the number of clusters. This method is not very accurate since one operating zone might be repeated many times in the training data. In this paper, the number of clusters is identified as follows: First, the number of clusters is initialized (equal to the number of the operating zones). Then, the K-means algorithm is applied to obtain the centers of the clusters. Next, the similarities among these clusters are eliminated since two clusters (or more) may describe a similar hyper-plane. The number of clusters will be reduced if similarities are detected. This process will be repeated till no similarities between clusters are excited.

Due to the simplicity of the training data, the previous approach can be easily applied. Thus, the obtained number of clusters (number of fuzzy rules) is $N = 7$. The number of clusters can also be identified using Dunn index (Dunn, 1973) method. To apply this method, the number of clusters is set to be changing from 2 to 16. Then, the k-means algorithm is performed for all possible numbers of clusters and the Dunn index is calculated. The right number of clusters is selected according to the highest value of the Dunn index (see the upper value of the index in Figure 3). It is clear that the Dunn index method obtained the same number of clusters.

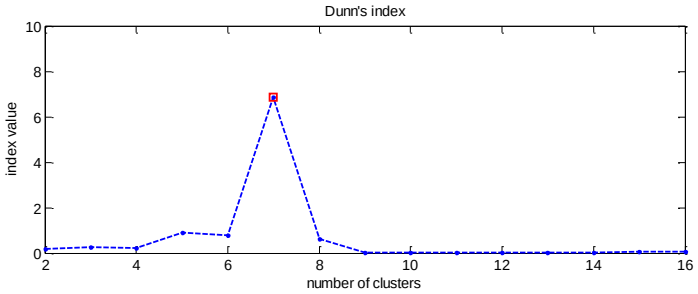
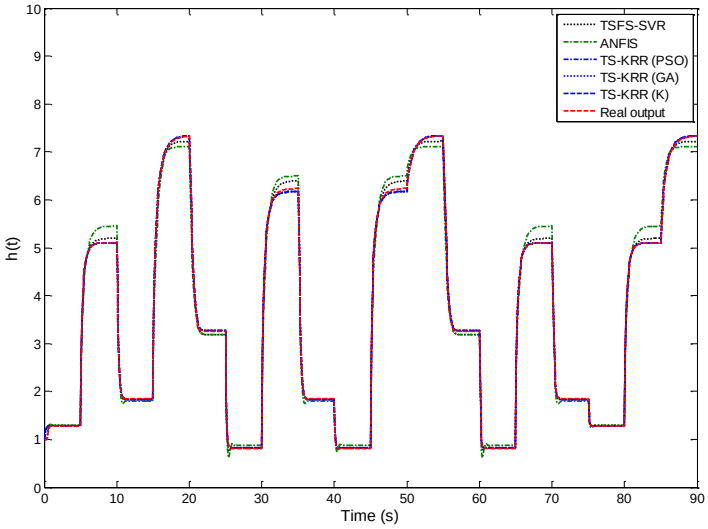


Figure 3: The variation of Dunn index

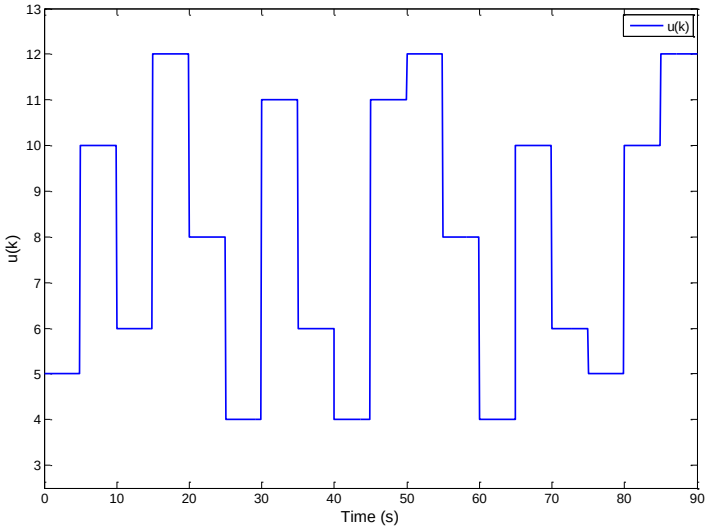
Next, the clustering based PSO algorithm is used to initialise the centroid and the width of the clusters. The number of particles used in the PSO algorithm is set to 16 while the PSO parameters w_0, c_1, c_2, $iter\_max$ are set to: $0.65, 1.7, 1.7$ and 100, respectively. To initialize the 16 particles, first the K-means algorithm is performed and the results obtained will be considered as the first particles (store it in $\tilde{\boldsymbol{x}}_1$). The rest of the particles are randomly selected where the maximum and minimum boundaries of these particles are defined as: $\tilde{\boldsymbol{x}}_{max} = \tilde{\boldsymbol{x}}_1 + 0.25 \times \tilde{\boldsymbol{x}}_1$ and $\tilde{\boldsymbol{x}}_{min} = \tilde{\boldsymbol{x}}_1 -$

$0.25 \times \breve{\boldsymbol{x}}_1$, respectively. To perform the quadratic problem in Eq. (9), the constant λ is set to 0.001. After running the offline TS-KRR steps in Algorithm 2, the identification results of the liquid level $h_s(t)$ are presented in Figure (4a).

The performance of the clustering based PSO algorithm is evaluated by comparing its offline identification results with the results obtained by two different clustering algorithms: the standard K-means algorithm and clustering based Genetic Algorithm (GA) algorithm. The clustering based GA algorithm parameters are: the population size is equal to 16, the mutation rate is 0.2, the selection rate is 0.5, the crossover rate is 0.7 and the same PSO maximum and minimum boundaries are used to initialise the GA population. To perform the clustering based GA algorithm, the same steps in Algorithm 1 are executed except step 3 where the global and local best positions are updated using the following GA operators: selection, mutation and crossover.



(a)



(b)

Figure 4: Identification of the nonlinear system: (a) the modelling performance of the proposed algorithm TR-KRR. (b) The control signal.

Figure (4a) displays the TS-KRR results when three different clustering algorithms are used to compute the antecedent parameters. The obtained results are referred to as: TS-KRR (PSO), TS-KRR (GA) and TS-KRR (K-means). Additionally, two different algorithms: the Takagi–Sugeno Fuzzy System-based Support Vector Regression (TSFS-SVR) (Juang, & Hsieh, 2009) and the generalized neural networks based fuzzy inference system (GNN-FIS) (Jang, 1991) are also are also presented in Figure (4a).

The results show that the TS-KRR algorithm (in all cases) performed well and the accuracy of the modelling is better than both: GNN-FIS and TSFS-SVR algorithms. Moreover, the TS-KRR (PSO), TS-KRR (GA) and TS-KRR (K-means) appear to have similar results which have been expected due to the simplicity of the training data.

To have an efficient comparison between the TS-KRR (PSO), TS-KRR (GA), TS-KRR (K-means), GNN-FIS and TSFS-SVR algorithms, the Root Mean Squared Error (RMSE), the Mean Absolute Error (MAE), the Mean Absolute Percentage Error (MAPE) and the symmetric Mean Absolute Percentage Error (sMAPE) are computed for all algorithms. The RMSE is described by:

$$RMSE = \sqrt{\frac{1}{N_S}\sum_{k=1}^{N_S}(y(k) - \hat{y}(k))^2} \tag{53}$$

where $y(k)$ is the actual (or real) output, $\hat{y}(k)$ is the predicted output obtained by the TS-KRR and N_S is the number of samples. The MAE, MAPE and the sMAPE (Shcherbakov et al., 2013) are given by Eq. (54), Eq. (55) and Eq. (56), respectively:

$$MAE = \frac{1}{N_S}\sum_{k=1}^{N_S}|y(k) - \hat{y}(k)| \tag{54}$$

$$MAPE = \frac{100}{N_S}\sum_{k=1}^{N_S}\left|\frac{y(k)-\hat{y}(k)}{y(k)}\right| \tag{55}$$

$$sMAPE = \frac{200}{N_S}\sum_{k=1}^{N_S}\frac{|y(k)-\hat{y}(k)|}{|y(k)|+|\hat{y}(k)|} \tag{56}$$

where $|*|$ represents the absolute value of $*$. The comparison results are presented in Table (1).

Table 1: Comparison results for the surge tank system

Methods	Rules	Number of inputs	RMSE	MAE	MAPE (%)	sMAPE (%)	Simulation time (s)
TS-KRR (PSO)	07	$n_a = 3$, $n_b = 1$	0.0039194	0.0018413	0.04609	0.04621	12.9901
TS-KRR (GA)	07	$n_a = 3$, $n_b = 1$	0.0039194	0.0018413	0.04609	0.04621	13.4612
TS-KRR (k-means)	07	$n_a = 3$, $n_b = 1$	0.0039219	0.0018427	0.04611	0.04628	07.1240
GNN-FIS	20	$n_a = 2$, $n_b = 2$	0.1812871	0.0947221	3.0451	3.1132	00.8104
TSFS-SVR	14	$n_a = 3$, $n_b = 1$	0.0787312	0.0354782	0.8247	0.8098	52.0758

The TS-KRR (PSO) and (GA) have produced the same error values (RMSE, MAE, MAPE and sMAPE) while the values obtained by the TS-KRR (K-means) are slightly

different (see Table 1). This indicates that the K-means algorithm efficiently clusters the data, and can be used to compute the centroid and the width of the clusters without any modifications. In this example, the improvements made by the PSO and GA algorithms are minor due to simplicity of the data and the K-means algorithm is sufficient to compute the antecedent parameters. On the other hand, the proposed TS-KRR approach outperforms both GNN-FIS and TSFS-SVR algorithms. This can be seen in Table (1) where the TS-KRR requires less fuzzy rules (only 7 rules for the TS-KRR) than the other two algorithms. Also, the values of errors (the values of RMSE, MAE, MAPE and sMAPE) obtained by the TS-KRR are smaller than those obtained by GNN-FIS and TSFS-SVR algorithms. Despite that the GNN-FIS had relatively a poor performance (RMSE=0.1813, MAE= 0.0947, MAPE= 3.0451% and sMAPE = 3.1132%), this algorithm is faster than both TS-KRR and TSFS-SVR algorithms where the GNN-FIS simulation time was less than 1 sec (the time of performing clustering and obtaining the consequent parameters).

B. Fuzzy predictive Control of the surge tank system

Usually, the offline TS-KRR algorithm could be enough to approximate nonlinear systems when sufficient data about these systems are available. In this case, the GPC controller is then implemented by replacing the original nonlinear plants with the offline TS-KRR model when the control signal is computed. To implement the TS-KRR GPC controller with offline identification, several parameters are selected by the designer such as: the size of the input vector which is set to 4 ($n_a = 3$ and $n_b = 1$) and the parameters of the GPC algorithm which are: $N_p = 7$, $N_u = 2$ and $\vartheta = 30$. The number of fuzzy rules is equal to $N = 7$ and the rest of the TS-KRR parameters are similar to those discussed earlier (PSO parameters and λ). The TS-KRR GPC (with offline identification) procedure is summarized as follows:

Algorithm 5: (TS fuzzy GPC with offline identification)

1. The reference signal $W(k)$ is selected.
2. The identification parameters such as: number of rules, the PSO algorithm parameters, and the control parameters (N_p, N_u and ϑ) are selected.
3. Perform Algorithm 2 to obtain the antecedent and consequent parameters, and then obtain $\bar{A}(z^{-1})$ and $\bar{B}(z^{-1})$.
4. The control signal increment $\Delta u(k)$ is computed using Eq. (51).
5. The new control signal $\Delta u(k) + u(k-1)$ is applied to the surge tank system.
6. Repeat steps 4 and 5.

By running the above algorithm, the system output and the applied control signal are illustrated in Figures (5) and (6) respectively. Again, the performance of the TS-KRR GPC controller is investigated when another two different clustering algorithms are used to compute the centroid and the width of the clusters. The simulation results are included in Figures (5). The results show that the proposed controller was successfully able to control the system output at the desired reference signal $W(k)$. Furthermore, the proposed controller response is fast since it moves from the initial values to the desired reference signal (as well as moving from one reference level to another) in a reasonable amount of time. As expected, due to the simplicity of the data, the offline TS-KRR GPC with the three clustering methods gave similar results. This concludes that the K-means algorithm is sufficient to compute the antecedent parameters and control the surge tank system.

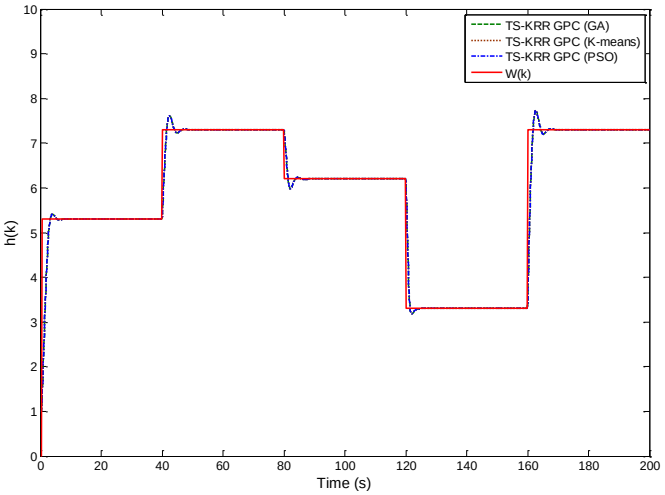


Figure 5: Results of the proposed offline TS-KRR GPC when three different clustering algorithms are used to compute the antecedent parameters.

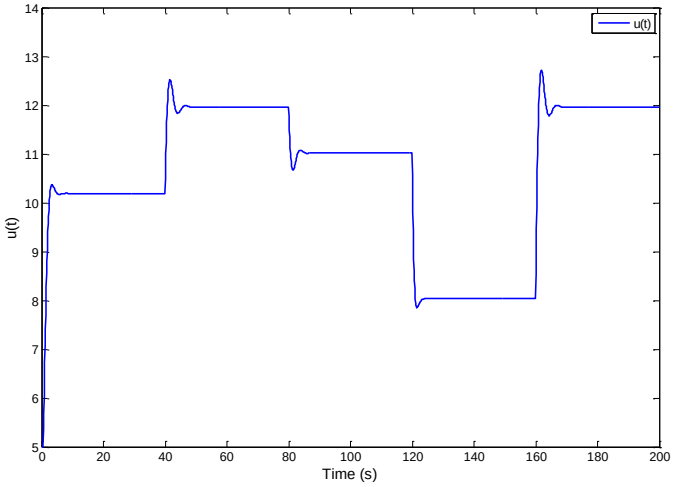


Figure 6: The applied control signal obtained by the TS-KRR GPC when the clustering based PSO algorithm is used to compute the antecedent parameters.

In general, the precision of the offline fuzzy models is limited due to the uncertainties associated with plants. Also, the collected data is incomplete and cannot describe all operating zones of the plants. Furthermore, the behaviour of the plants may change over time. In this case, introducing the adaptive capabilities is considered more practical since the TS-KRR parameters can be refined to improve its approximations. In the next subsection, the adaptive (or online) TS-KRR algorithm is investigated when the same system is controlled. The K-means algorithm will be used to initialise the antecedent parameters of the adaptive TS

algorithm since the improvements made by PSO algorithm are minor. Moreover, the antecedent parameters are updated in each iteration according to Eqs. (19) and (20).

C. The Online Identification

Again, the same 900 samples obtained earlier from Eq. (52) are used to test the adaptive TS-KRR performance. The first 400 samples were used to initialize the online TS-KRR while the rest of samples were used to validate the proposed adaptive fuzzy method. The same size of the input variables was chosen and the fuzzy rule is equal to $N = 7$. The dictionary size is set to $M = 50$ and the constant λ is set to 0.001. A K-means clustering algorithm is used to initialise the antecedent parameters. After running the adaptive TS-KRR steps in Algorithm 4, the liquid level $h_s(t)$ of the surge tank system is presented in Figure (7).

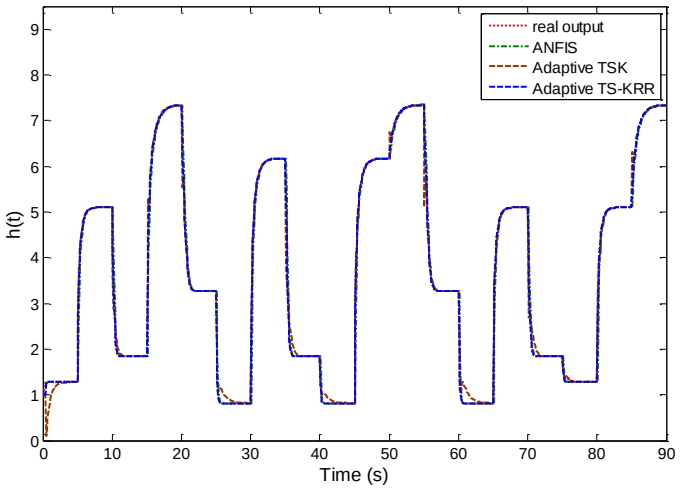


Figure 7: The online modelling performance of the adaptive TR-KRR, ANFIS and ATSK algorithms.

Figure (7) also contains the original signal of the liquid level $h_s(t)$, the signal predicted by the ANFIS algorithm and the signal obtained by the adaptive Takagi-Sugeno-Kang (TSK) method (Chang; Tsai, 2013). As illustrated in Figure (7), the three algorithms performed well and the accuracy of their modelling is relatively virtuous. Table 2 illustrates the accuracy results for the three algorithms where the RMSE, MAE, MAPE and sMAPE error measurements are calculated for the three algorithms. Clearly, the adaptive TS-KRR algorithm outperformed both the ANFIS and the adaptive TSK algorithms where the adaptive TS-KRR requires less fuzzy rules (only 7 rules) than the ANFIS and adaptive TSK algorithms. Furthermore, the error values obtained by the adaptive TS-KRR are smaller than those obtained by the ANFIS and the ATSK algorithms (see Table 2).

Table 2: Comparison results for the surge tank system

Methods	Rules	Number of inputs	RMSE	MAE	MAPE (%)	sMAPE (%)
Adaptive TS-KRR	07	$n_a = 3$, $n_b = 1$	0.0347	0.0130	0.9030	0.8848
ANFIS	20	$n_a = 2$, $n_b = 2$	0.09864	0.0396	2.7151	2.7062
Adaptive TSK	20	$n_a = 2$, $n_b = 2$	0.18332	0.0752	5.1174	5.2018

D. Fuzzy adaptive predictive control

In this subsection, the adaptive TS-KRR GPC is investigated. Again, the same size of the input vector ($n_a = 3$ and $n_b = 1$) was chosen as well as the fuzzy rule number (equal to $N = 7$). The parameters of the GPC algorithm are: $N_p = 7$, $N_u = 2$, $\vartheta = 30$, the dictionary size is set to $M = 50$ and the K-means algorithm is used to initialize the antecedent parameters. The adaptive learning rate η (the same form of η in (Lu, & Tsai, 2007) was chosen) is defined as:

$\eta = \frac{\beta}{\Sigma_{j=1}^{N}\Sigma_{i=1}^{n}\left(\left(\frac{\partial\hat{y}(k)}{\partial m_{ij}}\right)^2+\left(\frac{\partial\hat{y}(k)}{\partial \sigma_{ij}}\right)^2\right)}$, where $\beta = 0.09$ and the adaptive fuzzy GPC procedure is summarized as following:

Algorithm 6: (adaptive TS-KRR GPC)

1. The reference signal W(k) is selected
2. The identification parameters such as: number of rules, and the control parameters (N_p, N_u and ϑ) are selected.
3. Perform the K-means algorithm to initialize the antecedent parameters.
4. Measure the output signal
5. Update m_{ij}, σ_{ij} and the learning rate η.
6. Apply steps: 6 and 7 in Algorithm 4 to obtain the updated expansion coefficients and then update the $\widehat{\boldsymbol{\theta}}$ in Eq. (31), and then calculate $\bar{A}(z^{-1})$ and $\bar{B}(z^{-1})$.
7. Obtain the control increment $\Delta u(k)$ using Eq. (51)
8. Obtain control signal $u(k) = \Delta u(k) + u(k-1)$ and apply it to the nonlinear system.
9. Repeat steps 4 - 8.

The system output and the control signal are illustrated in Figures (9) and (10) respectively. In addition, the system output obtained by an ANFIS GPC (the ANFIS GPC parameters are: $N = 20$, $N_p = 7$, $N_u = 2$, $\vartheta = 30$ and the rule number $N = 20$) and an Adaptive TSK (ATSK) GPC (the ATSK GPC parameters are: $N = 20$, $N_p = 7$, $N_u = 2$, $\vartheta = 30$ and $N = 20$) are also illustrated in Figure (8).

The obtained results show that the proposed adaptive TS-KRR GPC controller effectively controls the system output at the desired reference $W(k)$. The same comments can be made for the ANFIS GPC and ATSK GPC controllers. However, the proposed adaptive TS-KRR GPC controller relatively displayed less overshoots (and undershoots) than the ANFIS GPC and the ATSK GPC controllers. The mean value of the execution time for one sample or iteration (the execution of steps 4, 5, 6, 7 and 8 in Algorithm 6) for a window (or dictionary size) of $M = 50$ is $0.003552s$, which is considered as a short period of time and suits most of the industrial systems (in this example $0.003552s \ll t_s = 0.1\,sec$). On the other hand, the ANFIS GPC controller has smaller execution time (0.000713s) since its algorithm does not require a large number of previous data. The same comment can be made for the ATSK GPC algorithm where the execution time for one sample is relatively small (0.000689s).

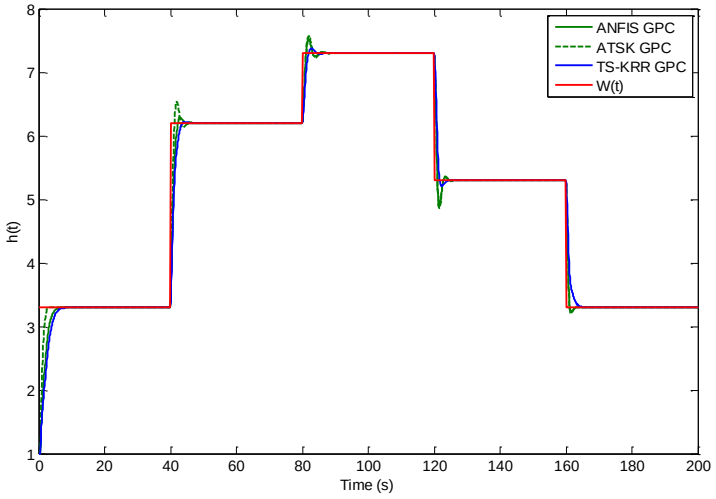


Figure 8: Results of the adaptive TS-KRR GPC, ANFIS GPC and ATSK GPC controllers.

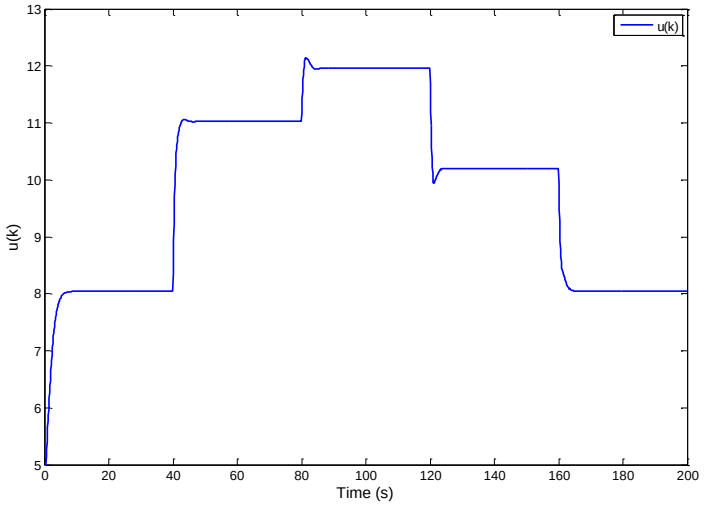


Figure 9: the applied control signal obtained by the adaptive TS-KRR GPC

Next, the disturbance rejection capabilities of the proposed adaptive TS-KRR GPC controller are investigated. A disturbance with amplitude of 1 was applied to the system in samples 200-600 (the time interval $20 \leq k \leq 60$ s). Next, a disturbance of amplitude -1 was applied again to the system at the interval time $60 < k \leq 110$s (in samples 600-1100). Then, a disturbance with amplitude of 1 was applied again to the system in samples 1400-2000 (the time interval $140 < k \leq 200$ s). The simulation results are presented in Figures (10) and (11). The results show that the proposed adaptive TS-KRR GPC controller have good disturbance rejection capabilities where the disturbances were eliminated in a short period of time. Moreover, the proposed controller exhibits smaller overshoots (and undershoots) than the controllers based on the ANFIS and ATSK algorithms. In addition, the ATSK GPC controller exhibits relatively large overshoots (and undershoots) than the ANFIS GPC controller.

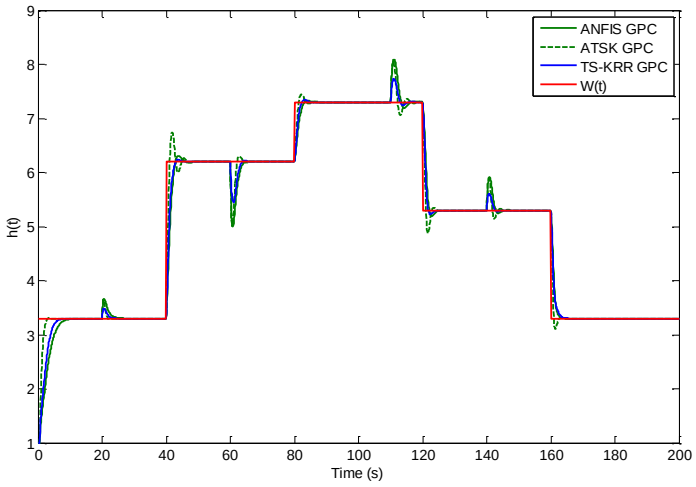


Figure 10: Results of the adaptive TS-KRR GPC, ANFIS GPC and ATSK GPC controllers.

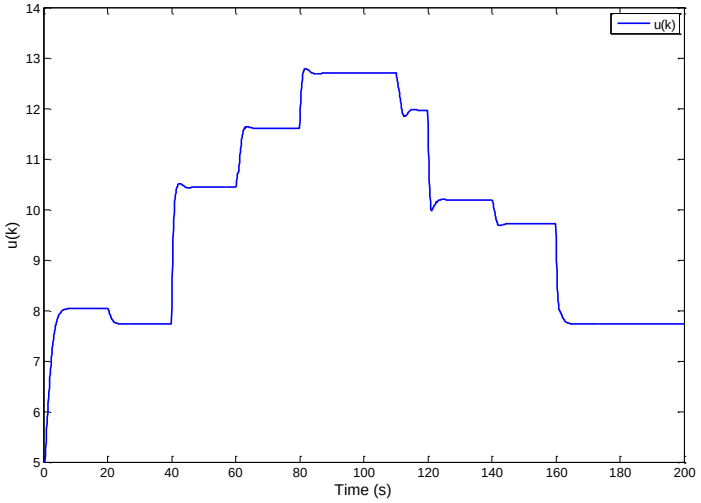


Figure 11: the applied control signal obtained by the adaptive TS-KRR GPC

VI. ILLUSTRATIVE EXAMPLE 2: IDENTIFICATION AND CONTROL OF A CONTINUOUS-STIRRED TANK REACTOR (CSTR)

As a second example, a continuous-stirred tank reactor plant (CSTR) (Oviedo, Vandewalle, & Wertz, 2006) presented in Figure (12) is used to validate the TS-KRR performance in system identification, and when the TS-KRR is integrated with the generalized predictive controller.

The nonlinear model is described by the following differential equations:

$$\begin{aligned} \dot{C}_a(t) &= \frac{q}{v}\left(C_{a0} - C_a(t)\right) - k_0 C_a(t) e^{-\frac{E}{RT(t)}} \\ \dot{T}(t) &= \frac{q}{v}\left(T_0 - T(t)\right) + k_1 C_a(t) e^{-\frac{E}{RT(t)}} \\ &+ k_2 q_c(t)\left(1 - e^{-\frac{k_3}{q_c(t)}}\right)\left(T_{c0} - T(t)\right) \end{aligned} \tag{57}$$

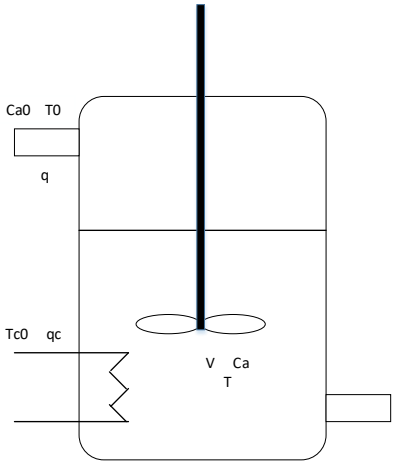


Figure 12: continuous-stirred tank reactor

This system describes the process of converting the product A_a into a new product B_b. The concentration $C_a(t)$ is the concentration of product A_a, while $T(t)$ is the temperature of the mixture. The reaction described by the nonlinear system is exothermic and the coolant flow rate $q_c(t)$ is used to control this reaction. When the coolant flow is adjusted (or controlled), the temperature will be controlled and so the concentration. The constant C_{a0} is the inlet feed concentration, and q is a constant that represents the process flow rate. The inlet feed and coolant temperatures are assumed to be constant and defined by T_0 and T_{c0}, respectively. The rest of the thermodynamic and chemical constants are given in Table (3), and the parameters k_1, k_2 and k_3 are:

$k_1 = \frac{\Delta H k_0}{\rho C_p}$, $k_2 = \frac{\rho_c C_{pc}}{\rho C_p v}$ and $k_3 = \frac{h_a}{\rho_c C_{pc}}$.

To have a concentration of $C_a = 0.1\ mol/l$, the nominal conditions for the temperature and coolant flow are $438.54\ K$ and $0.1\ l/min$, respectively.

Table 3: Nominal parameters of the CSTR nonlinear system

Parameters	Explanation	Nominal value
q	Process flow-rate	$100\ l/min$
k_0	Reaction rate constant	$7.2 \times 10^{10}\ min^{-1}$
v	Volume of the Reactor	$100\ l$
T_0	Feed temperature	$350\ K$
E/R	Activation energy	$1 \times 10^4\ K$
T_{c0}	Inlet coolant temperature	$350\ K$
ΔH	Reaction heat	$2 \times 10^5\ cal/mol$
ρ, ρ_c	Liquid densities	$1 \times 10^3\ g/l$
C_p, C_{pc}	Specific heats	$1\ cal\ /g/K$
C_{a0}	Inlet feed concentration	$1\ mol/\ l$
h_a	Coefficient of heat transfer	$7 \times 10^5\ cal\ /min/K$

A. The offline Identification

To model the CSTR plant using the TS-KRR, the sampling time is set to $t_s = 6\ sec$ ($0.1\ min$) and the nonlinear system presented in Eq. (57) is used to generate $N_S = 900$ samples; the first 400 samples were used to train the TS-KRR, while the last 500 were used to validate the proposed fuzzy method. The samples were obtained by applying the control signal (the coolant flow rate $q_c(t)$) represented in Figure (14b). The size of the input variable vector is set to 8 ($n_a = 5$ and $n_b = 3$), and the number of clusters is usually equal to the number of the operating zones described by the training data. The number of clusters is identified with the same technique described earlier in the previous example.

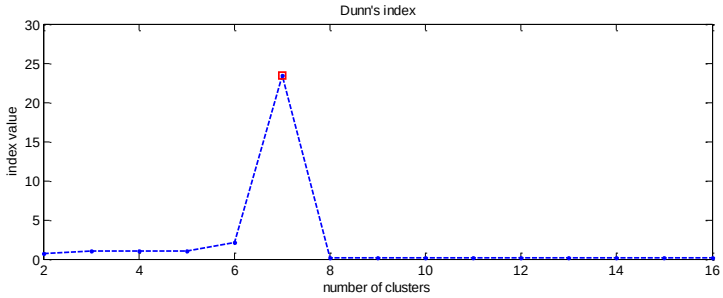
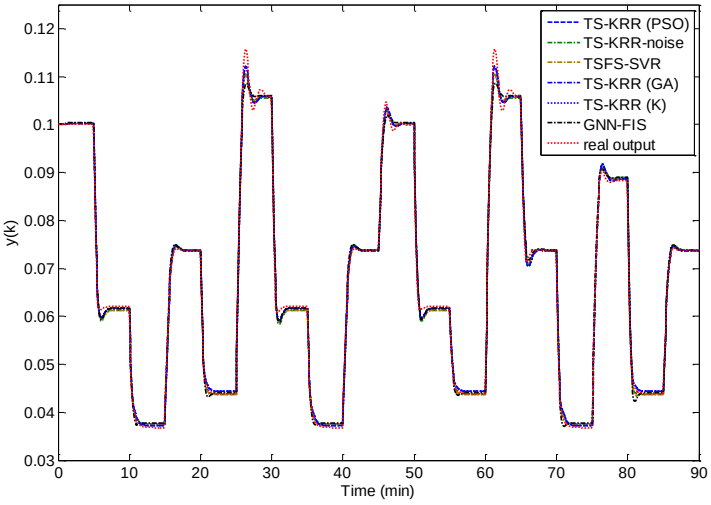


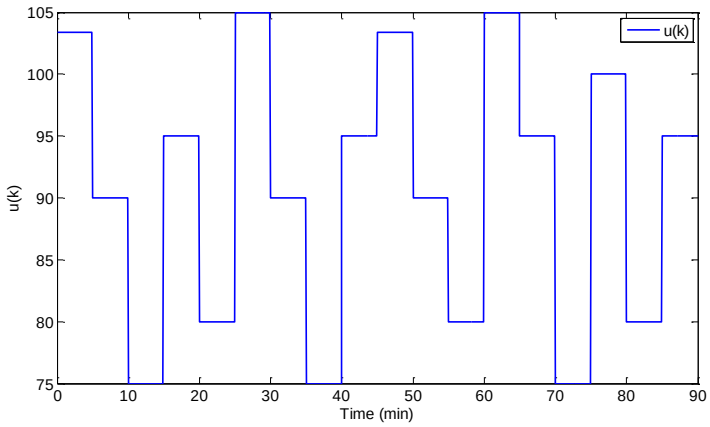
Figure 13: The variation of Dunn index

As a result, the obtained number of clusters (number of fuzzy rules) is $N = 7$. To validate this approach, the Dunn index method is used once more to find the number of clusters where the correct number of clusters always has the highest value of the Dunn index (see the upper value in Figure 13). To cluster the data, the number of particles used in the PSO algorithm is set to 12 while the PSO parameters w_0, c_1, c_2, $iter\_max$ are set to $0.7, 1.6, 1.6$ and 100, respectively. As mentioned in the first example, the K-means algorithm is performed and the resulted centroids are stored in $\breve{\boldsymbol{x}}_1$. The rest of the particles are randomly selected where the particles maximum and minimum boundaries are defined as: $\breve{\boldsymbol{x}}_{max} = \breve{\boldsymbol{x}}_1 + 0.25 \times \breve{\boldsymbol{x}}_1$ and $\breve{\boldsymbol{x}}_{min} = \breve{\boldsymbol{x}}_1 - 0.25 \times \breve{\boldsymbol{x}}_1$, respectively. To perform the quadratic problem in Eq. (9), the constant λ is set to 0.0005. After running the TS-KRR steps in Algorithm 2, the identification results of the CSTR output $C_a(t)$ are presented in Figure (14a).

To investigate the performance of the clustering based PSO algorithm, the offline identification results obtained by this algorithm are compared with the results obtained by the standard K-means and the clustering based GA algorithms. To perform the clustering based GA algorithm, the mutation rate is set 0.2, population size is set to 12, the selection rate is set to 0.5, the crossover rate is equal to 0.7 and the same maximum and minimum boundaries are used to initialise the population. Figure (14a) contains the TS-KRR results when the antecedent parameters are computed by three different clustering methods: TS-KRR (PSO), TS-KRR (GA) and TS-KRR (K-means). Besides, the result of the TS-KRR (PSO), when a disturbance of 6% is contaminating the training data, is presented in Figure (14a). Furthermore, Figure (14a) includes the GNN-FIS and the TSFS-SVR prediction results.



(a)



(b)

Figure 14: Identification of the nonlinear system: (a) The modelling performance of the proposed algorithm TR-KRR (b) control signal.

As expected, the TS-KRR algorithm performs well (with and without noise) and provided good results (see Figure 14a). In addition, the improvements made by clustering based PSO algorithm are minor (see Table 4), where the accuracy tests show that the results obtained by TS-KRR (PSO) are slightly better than the results obtained by TS-KRR with other two clustering algorithms.

Table 4: Comparison results for the surge tank system

Methods	Rules	Number of inputs	RMSE	MAE	MAPE (%)	sMAPE (%)	Simulation time (s)
TS-KRR (PSO)	07	$n_a = 5$, $n_b = 3$	3.132132 $\times 10^{-5}$	1.754412 $\times 10^{-5}$	2.662045 $\times 10^{-4}$	2.664721 $\times 10^{-4}$	14.1892
TS-KRR (GA)	07	$n_a = 5$, $n_b = 3$	3.139146 $\times 10^{-5}$	1.754966 $\times 10^{-5}$	2.671203 $\times 10^{-4}$	2.681160 $\times 10^{-4}$	14.6214
TS-KRR (K-means)	07	$n_a = 5$, $n_b = 3$	3.159611 $\times 10^{-5}$	1.778136 $\times 10^{-5}$	2.696238 $\times 10^{-4}$	2.714248 $\times 10^{-4}$	05.1732
TS-KRR (PSO) (6%)	07	$n_a = 5$, $n_b = 3$	4.415412 $\times 10^{-5}$	2.184041 $\times 10^{-5}$	3.167274 $\times 10^{-4}$	3.186142 $\times 10^{-4}$	14.1892
GNN-FIS	20	$n_a = 5$, $n_b = 3$	8.782313 $\times 10^{-4}$	6.024566 $\times 10^{-4}$	0.009402	0.009378	0.9874
TSFS-SVR	14	$n_a = 5$, $n_b = 3$	6.527324 $\times 10^{-4}$	4.212638 $\times 10^{-4}$	0.006614	0.006616	55.4168

In the first case where no disturbances (noise) are included, the obtained errors were: $RMSE = 3.132 \times 10^{-5}$, $MAE = 1.754 \times 10^{-5}$, $MAPE = 2.662 \times 10^{-4}\%$ and $sMAPE = 2.665 \times 10^{-4}\%$. These values indicate that the TS-KRR (PSO) performs well and accurately predicts the signal output $C_a(t)$. The same comments can be made in the second case when a noise of 6% is applied to the learning data, and the obtained errors remain low ($RMSE = 4.415 \times 10^{-5}$, $MAE = 2.184 \times 10^{-5}$, $MAPE = 3.167 \times 10^{-4}\%$, $sMAPE = 3.186 \times 10^{-4}$). The TS-KRR method outperforms the TSFS-SVR and GNN-FIS methods where only 7 fuzzy rules were used in the identification process while the TSFS-SVR and GNN-FIS algorithms used 14 and 20 rules, respectively. Again, the TS-KRR method gave better accuracy results than the TSFS-SVR and GNN-FIS algorithms (see the error values in Table 4) while only less number of rules were used by the TS-KRR predictor.

The TS-KRR (GA) and TS-KRR (K-means) algorithms give results slightly different from the TS-KRR (PSO) which indicates that the K-means algorithm efficiently clusters the data, and the improvements made by the PSO algorithms were minor.

B. Fuzzy predictive Control of the CSTR plant

In this subsection, an offline TS-KRR GPC controller is used to control the CSTR plant. First, an offline TS-KRR is used to model the CSTR plant where the size of the input vector is kept the same which is 8 ($n_a = 5$ and $n_b = 3$) while the fuzzy rule number is set to $N = 7$. The parameters of the GPC algorithm are: $N_p = 10$, $N_u = 2$, $\vartheta = 0.0008$ and the parameters of the TS-KRR identification are similar to those discussed earlier. The TS-RR GPC (with offline identification) procedure is summarized earlier in Algorithm 5. The system output and the applied control signal are illustrated in Figures (15) and (16) respectively. The results obtained by the TS-KRR GPC controllers when the K-means and the clustering based GA algorithms are used to compute the antecedent parameters are also presented in Figure (15). The obtained results show that the proposed controller was successfully able to control the system output at the desired reference signal $W(k)$. Furthermore, the proposed controller response is fast since it moves from the initial values to the desired reference signal (as well as moving from one reference level to another) in a reasonable amount of time. Furthermore, the results obtained by all versions of the TS-KRR GPC controllers are almost the same (the TS-KRR GPC controller based on the K-means algorithm is slightly slower than the other two controllers). As a conclusion, the k-means algorithm appears to perform well when the training data is relatively simple. Thus, the k-means algorithm will be used to initialise the antecedent parameters in the adaptive version of the TS-KRR method. Next, the performance of the adaptive TS-KRR method is investigated by modelling and controlling the CSTR plant.

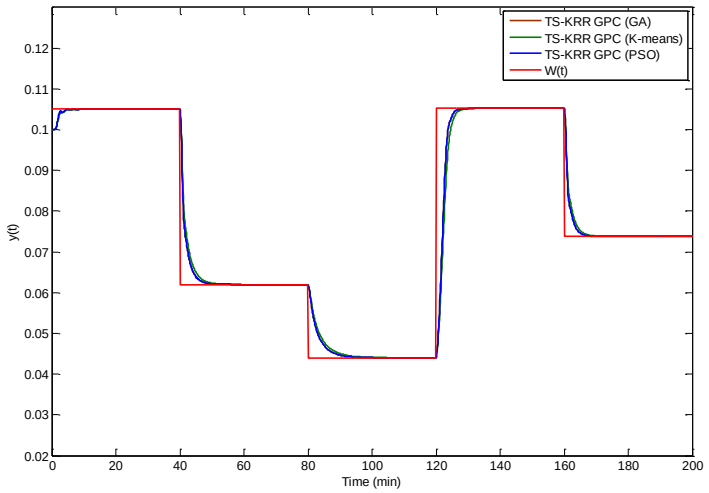


Figure 15: Results of the proposed offline TS-KRR GPC when three different clustering algorithms are used to compute the antecedent parameters.

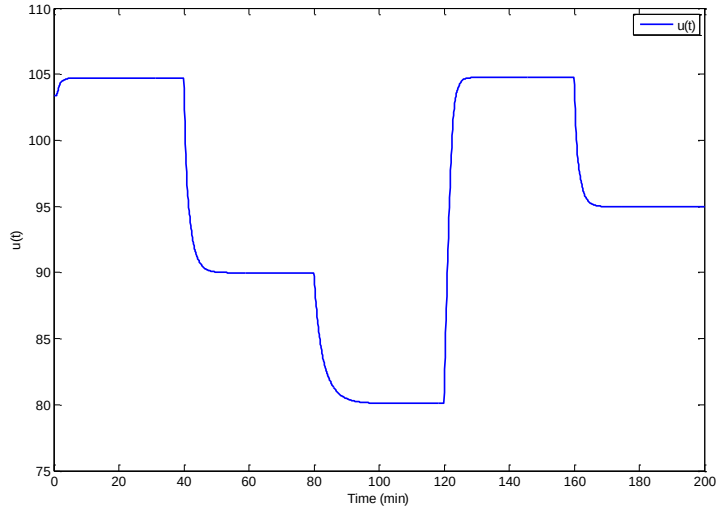


Figure 16: the applied control signal obtained by the TS-KRR GPC when the clustering based PSO algorithm is used to compute the antecedent parameters.

C. *Online Identification*

In this subsection, the adaptive TS-KRR method is considered for the identification of the CSTR plant. The previous 900 samples were used to test the adaptive TS-KRR. The first 400 samples were used to initialize the online (adaptive) TS-KRR while the rest were used for validation. The same size of the input variables and fuzzy rules number were used to model the system. The dictionary size is set to $M = 50$ and the constant λ is set to 0.0005. A K-means clustering algorithm is used to initialize the antecedent parameters. After running the adaptive TS-KRR steps in Algorithm 4, the CSTR output $C_a(t)$ is presented in Figure (17). The obtained results are compared with those attained by the ANFIS and ATSK algorithms. The Comparison results are presented in Table (5).

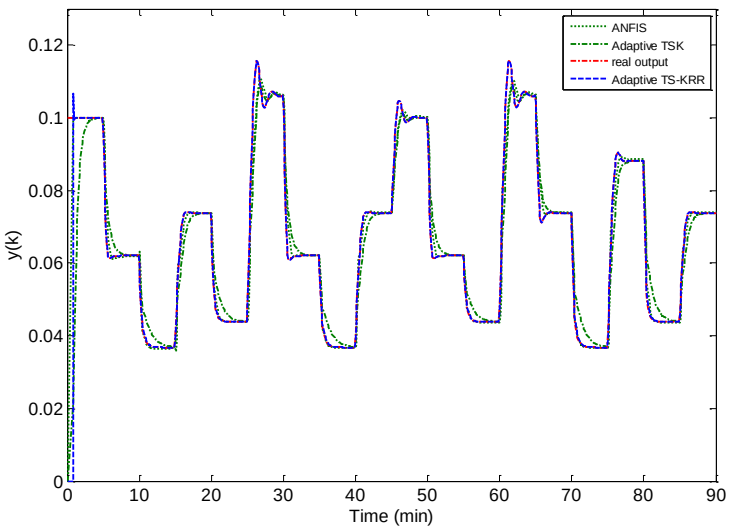


Figure 17: The online modelling performance of the proposed TR-KRR algorithm.

The adaptive TS-KRR generates an error values of $RMSE = 1.023 \times 10^{-3}$, $MAE = 4.030 \times 10^{-3}$, $MAPE = 6.097 \times 10^{-3}\%$ and $sMAPE = 6.100 \times 10^{-3}\%$. The values obtained indicate that the adaptive TS-KRR effectively predicted the output signal $C_a(t)$ and gives better accuracy than both ANFIS and ATSK algorithms (see Table 5). Also, the number of rules used in the TS-KRR is less than that used by the ANFIS and ATSK algorithms. The errors obtained by the ATSK algorithm were relatively high ($RMSE = 5.174 \times 10^{-2}$, $MAE = 2.061 \times 10^{-2}$, $MAPE = 0.3116\%$ and $sMAPE = 0.3147\%$) which might be related to the fixed values of the antecedent parameters (the Adapted TSK algorithm initialises the antecedent parameters without any adaptation during the online simulation).

Table 5: Comparison results for the surge tank system

Methods	Rules	Number of inputs	RMSE	MAE	MAPE (%)	sMAPE (%)
TS-KRR	07	$n_a = 5$, $n_b = 3$	1.023×10^{-3}	4.030×10^{-4}	6.097×10^{-3}	6.100×10^{-3}
ANFIS	20	$n_a = 5$, $n_b = 3$	2.381×10^{-2}	7.620×10^{-3}	0.1156	0.1167
ATSK	20	$n_a = 5$, $n_b = 3$	5.173×10^{-2}	2.061×10^{-2}	0.3116	0.3147

Next, the adaptive TS-KRR algorithm is integrated with the GPC and used to control the CSTR plant.

D. *Fuzzy adaptive predictive control of the CSTR plant*

Again, the previous identification parameters such as: the input vector and the number of fuzzy rules are retained in this section. The parameters of the GPC algorithm are similar to the previous subsection ($N_p = 10$, $N_u = 2$ and $\vartheta = 0.0008$). The dictionary size is kept equal to $M = 50$ and the K-means algorithm was used to initialize the antecedent parameters. The adaptive learning rate η is the same as the first example where $\beta = 0.3$. The adaptive TS-KRR GPC procedure

summarized in algorithm 6 was executed and the system output and the control signal are illustrated in Figures (18) and (19), respectively. The ANFIS GPC and The ATSK GPC controllers were also applied to control the CSTR plant where the control parameters for both algorithms are: $N = 20$, $N_p = 10$, $N_u = 2$, $\vartheta = 0.00076$. The output obtained by the ANFIS GPC and the ATSK GPC controllers are also illustrated in Figure (18).

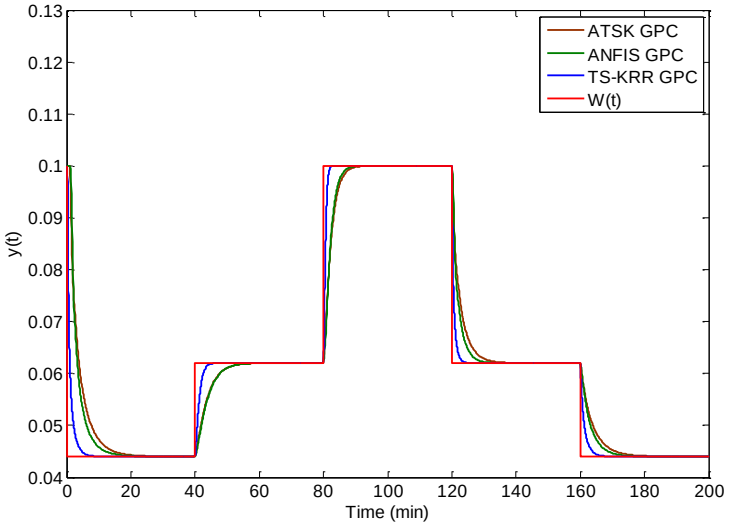


Figure 18: Results of the adaptive TS-KRR GPC, ANFIS GPC and ATSK GPC controllers

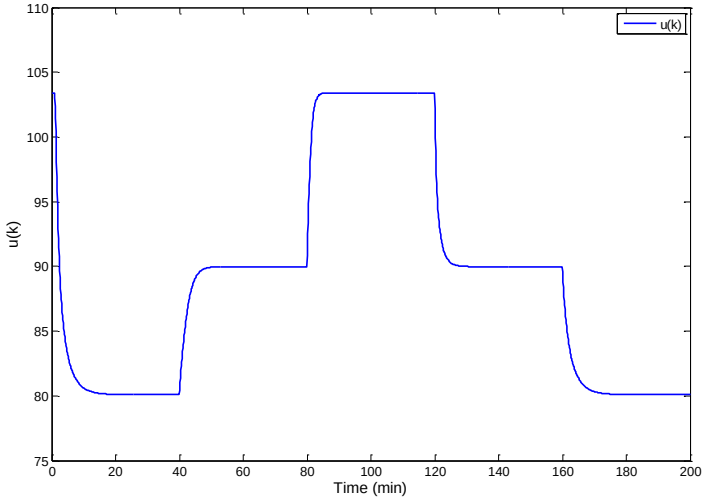


Figure 19: The applied control signal obtained by the adaptive TS-KRR GPC

The obtained results show that the proposed adaptive TS-KRR GPC controller effectively controls the system output at the desired reference $W(k)$. The mean value of the execution time for one sample or iteration (the execution of steps 4, 5, 6, 7 and 8 in Algorithm 5) for a window $M = 50$ is $0.00781s$, which suits most of the industrial systems (less than $t_s = 6\,sec$). The ANFIS GPC controller has relatively small execution time ($0.000912s$ for one iteration) since only few previous input-output data are required to determine the control signal. On the other hand, the ATSK GPC controller has the lowest exclusion time ($0.000796s$ for one iteration) which is obvious since this algorithm does not require an adaptation for its antecedent parameters.

Figure (20) represents the absolute error, the error between the output signal and the reference trajectory signal $e(t) = |y(t) - w(t)|$, produced by the three controllers . As expected, the adaptive TS-KRR GPC controller is fast and shows less error when it moves from a reference level to another. Moreover, the results indicate that the ATSK GPC controller produces relatively the largest errors when moving from between references which might be related to membership functions used in this algorithm.

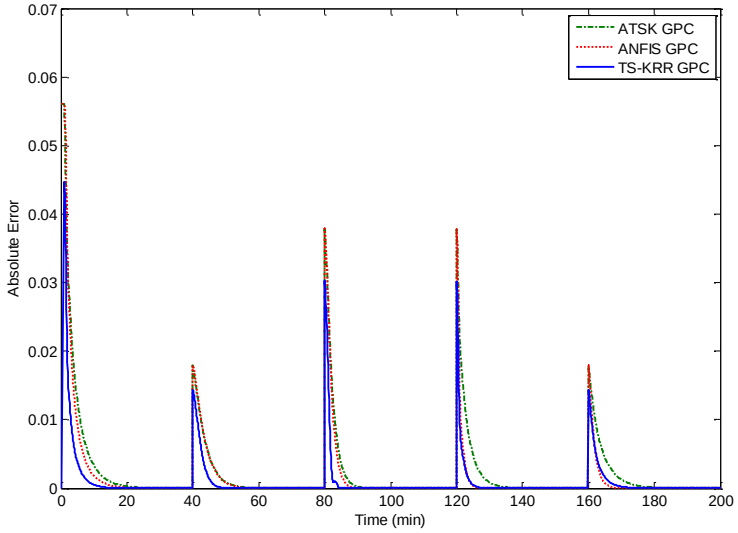


Figure 20: The absolute error resulted by the adaptive TS-KRR GPC, ANFIS GPC and ATSK GPC controllers

To investigate the performance of the TS-KRR GPC under disturbances, a disturbance with an amplitude of 0.012 was applied to the system in samples 600-1000 (at time interval $60 \leq k \leq 100$ min). Another disturbance of amplitude 0.02 was applied again to the system at the interval time $100 < k \leq 140$ min (in samples 1000-1400). The simulation results are presented in Figures (21) and (22).

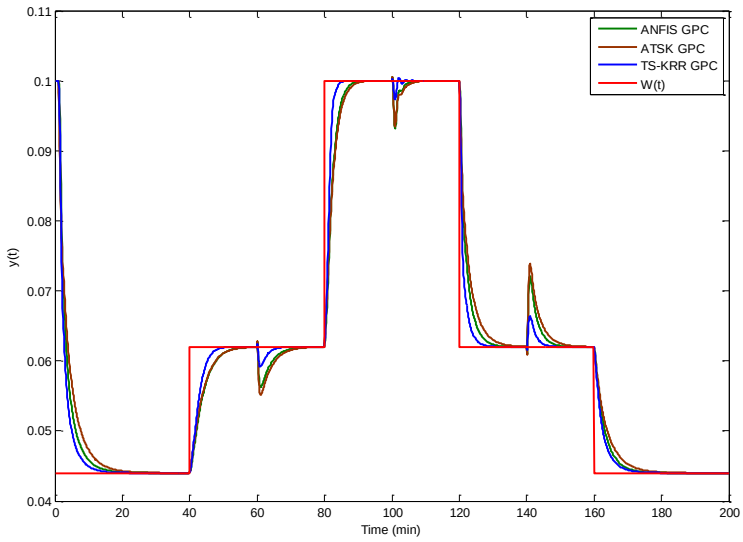


Figure 21: Results of the adaptive TS-KRR GPC, ANFIS GPC and ATSK GPC controllers

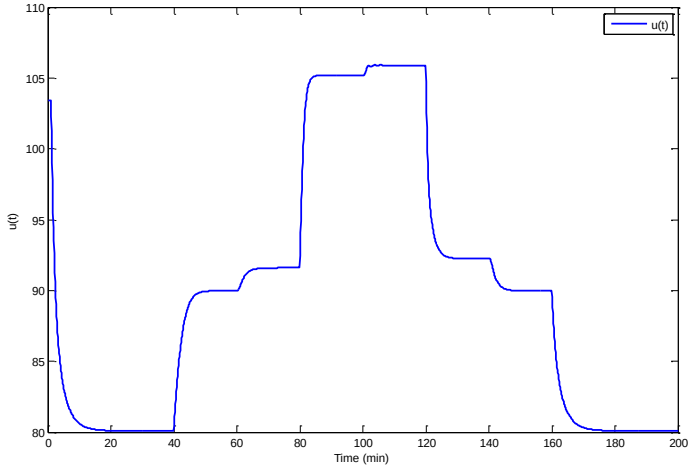


Figure 22: The applied control signal obtained by the adaptive TS-KRR GPC

The results show that the proposed adaptive controller has good disturbance rejection capabilities. Also, the overshoots (and undershoots) displayed by the TS-KRR GPC controller were smaller than those displayed by the ANFIS GPC and ATSK GPC controllers. This can be verified from Figure (23) where the absolute error of the adaptive TS-KRR GPC (when the disturbances are applied) is less than the absolute errors produced by the other two controllers. Again, the results in Figure (23) show that the ATSK GPC controller produces relatively the largest errors when the disturbances are applied to the system. It is clear that the triangular membership functions used by the ATSK system have a negative impact on the controller accuracy (the ATSK algorithm does not update the antecedent parameters during the controlling process).

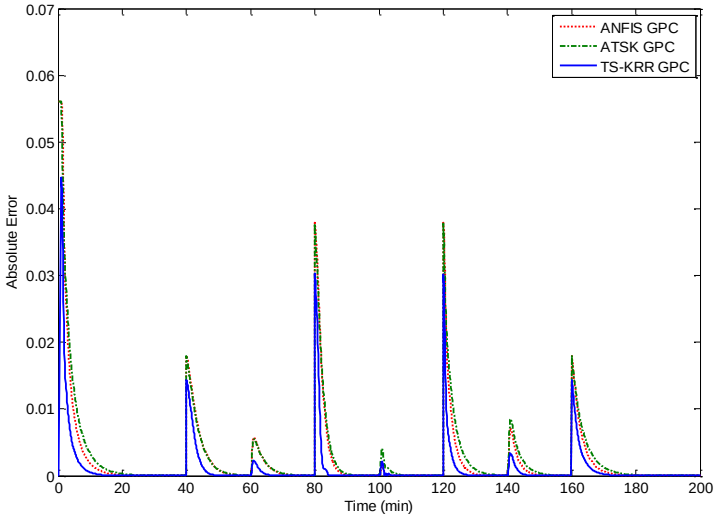


Figure 23: The absolute error resulted by the adaptive TS-KRR GPC, ANFIS GPC and ATSK GPC controllers

E. The adaptive predictive control of the CSTR plan: A sinusoidal reference trajectory

In this subsection, the adaptive TS-KRR GPC is investigated when the reference trajectory is varying during the simulation. The trajectory reference of the concentration $C_a(t)$ is defined as the following sinusoidal equation: $W(t) = 0.07 + 0.01\sin(0.02\pi t)$. The parameters of the TS-KRR GPC algorithm are: $N_p = 12$, $N_u = 2$, $\vartheta = 0.00025$ and the dictionary size is $M = 50$. Again, the K-means algorithm was used to initialize the antecedent parameters while the adaptive learning rate η is set to $\beta = 0.3$. The system output and the control signal are illustrated in Figures (24) and (25) respectively. The ANFIS GPC and the ATSK GPC controllers were also applied to control the same plants with the same desired sinusoidal reference signal, and the parameters for both controllers are: $N = 20$, $N_p = 12$, $N_u = 2$, $\vartheta = 0.00024$. The outputs obtained by the ANFIS GPC and the ATSK GPC controllers are also illustrated in Figure (24).

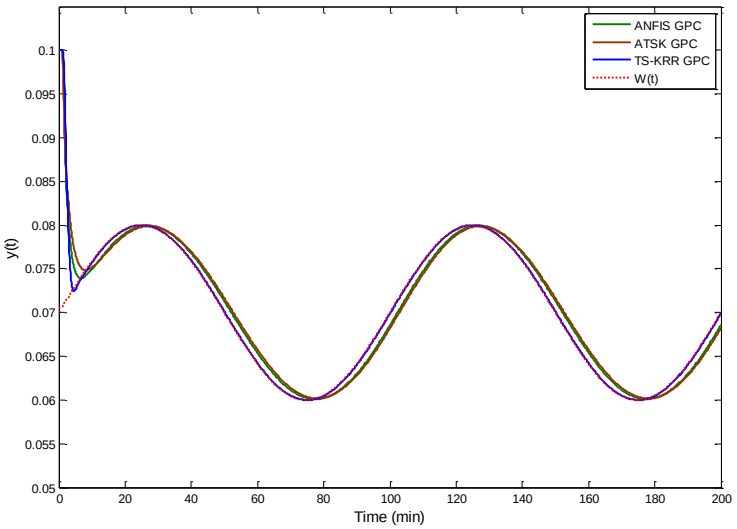


Figure 24: Results of the adaptive TS-KRR GPC, ANFIS GPC and ATSK GPC controllers

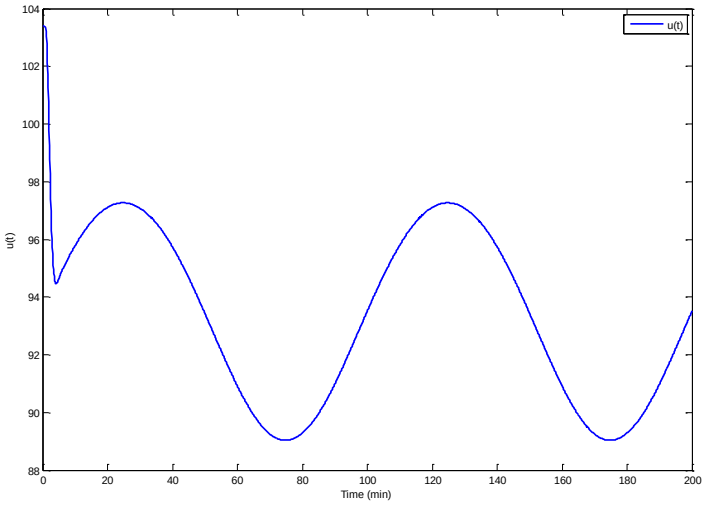


Figure 25: The applied control signal obtained by the adaptive TS-KRR GPC

The results show that the TS-KRR GPC controller effectively controls the nonlinear system at the desired reference $W(k)$. However, the ANFIS GPC and ATSK GPC controllers were displaying outputs with relatively large errors (see Figure 24). The efficiency of the adaptive TS-KRR GPC controller can also be verified from Figure (26) where the absolute error for the three controllers were plotted. As expected, the error obtained by the adaptive TS-KRR GPC controller is less than those obtained by the ANFIS GPC and the ATSK GPC controllers. Again, the results in Figure (26) confirm that the ATSK GPC controller produces relatively the less accurate results among the three algorithms.

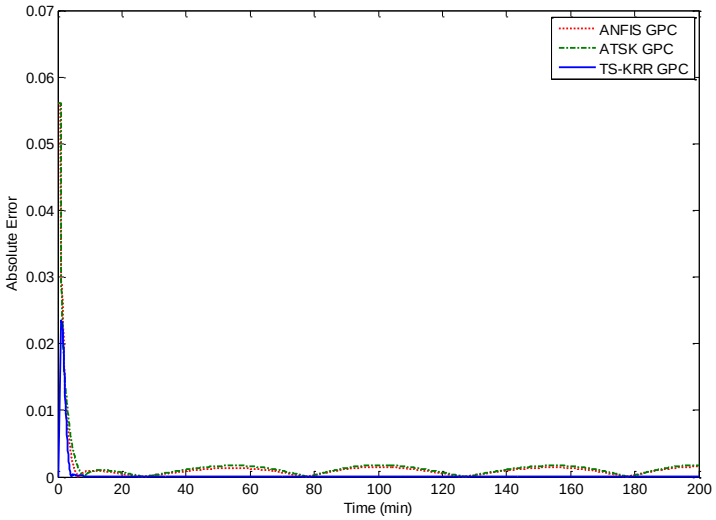


Figure 26: The absolute error resulted by the adaptive TS-KRR GPC, ANFIS GPC and ATSK GPC controllers

Again, the performance of the TS-KRR GPC controller in the presence of disturbances is investigated where a disturbance with an amplitude of 0.012 was applied to the system at time interval $60 \leq k \leq 100$ min) while another disturbance of amplitude 0.02 was applied to the system at the time interval $100 < k \leq 140$ min.

The output and control signals of the adaptive TS-KRR GPC controller are illustrated in Figures (27) and (28), respectively.

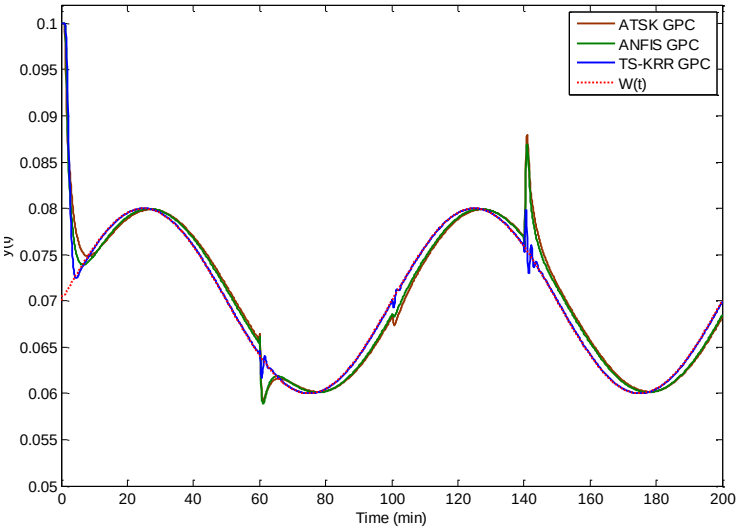


Figure 27: Results of the adaptive TS-KRR GPC, ANFIS GPC and ATSK GPC controllers

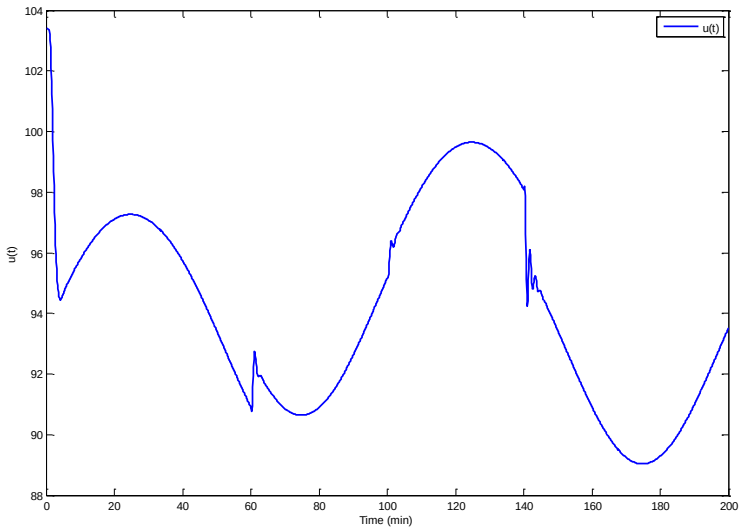


Figure 28: The applied control signal obtained by the adaptive TS-KRR GPC

Once again, the results show that the proposed adaptive TS-KRR GPC controller has good disturbance rejection capabilities. Also, the overshoots (and undershoots) displayed by the TS-KRR GPC controller were very small compared to the overshoots (and undershoots) displayed by the ANFIS GPC and ATSK GPC controllers. This can be verified from Figure (29) where the absolute error obtained by the adaptive TS-KRR GPC controller is less than those obtained by the ANFIS GPC and the ATSK GPC controllers when the disturbances were applied. Once more, the results in Figure (29) show that the ATSK GPC controller creates relatively the largest errors when the disturbances are applied to the system.

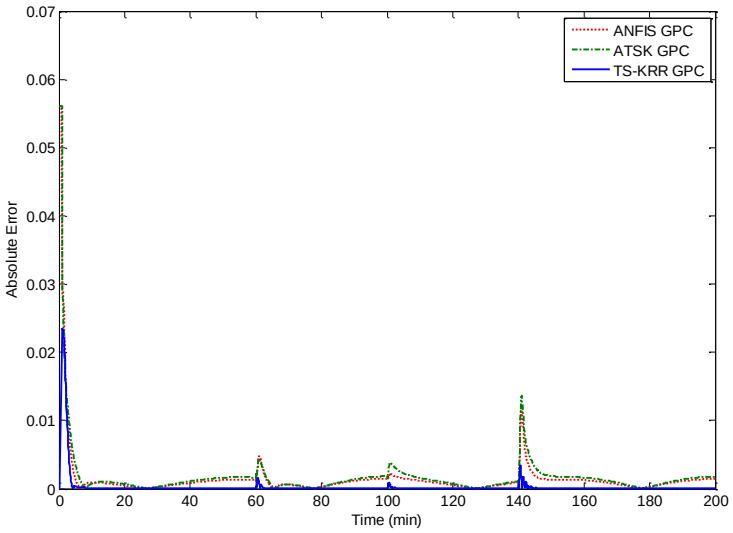


Figure 29: The absolute error resulted by the adaptive TS-KRR GPC, ANFIS GPC and ATSK GPC controllers

VII. CONCLUSION

In this paper, a Takagi-Sugeno system based Kernel ridge regression (TS-KRR) was proposed for nonlinear system identification and control. In this approach, the antecedent parameters of the TS-KRR fuzzy system were identified using a clustering algorithm while the consequent parameters are calculated using a Kernel ridge regression algorithm. The proposed TS-KRR model effectively used to identify two nonlinear systems: a surge tank and CSTR systems. Then, the offline TS-KRR was integrated with a generalized predictive controller to control these two nonlinear systems. The proposed TS-KRR GPC methodology used input-output data to learn model parameters and successfully controlled both nonlinear systems.

In the offline TS-KRR, the KRR method is used to perform a linear regression in very high-dimensional spaces in an efficient way by exploiting the properties of kernel function. This is equivalent to performing a nonlinear regression in the original input space. Thus, the proposed offline TS-KRR showed promising results in system identification and control system.

The clustering based PSO algorithm used to compute the antecedent parameters (centroid and the width of the clusters) provided minor improvements which has been expected due to the nature of the training data. However, more non-linear systems with relatively complex training data will be considered in future work. The offline TS-KRR method can be helpful to model systems when enough data about the systems are available, and the sampling times of these systems are very short. Thus, the offline TS-KRR reduces the computational costs since no adaptations are needed to update the antecedent and consequent parameters. Furthermore, more studies will be considered regarding the influence of the clustering algorithms when complex training data is available.

In this paper, more attentions were giving to the adaptive version of the TS-KRR method. The adaptive TS method is introduced to deal with real time applications. The proposed adaptive fuzzy model was investigated for both: system identification and control. The new adaptive methodology performed well in system identification and gave good predictions with less errors. Moreover, the adaptive TS-KRR GPC controller effectively controlled the two nonlinear systems. Furthermore, the disturbance rejection capabilities of the proposed adaptive TS-KRR GPC methodology were investigated by disturbing the nonlinear systems in preselected instants. The proposed adaptive TS-KRR GPC methodology successfully eliminated these disturbances. Finally, the adaptive TS-KRR GPC methodology was investigated when the reference signal varies as a sinusoidal function. The CSTR plant was controlled using the TS-KRR GPC and the results show that the adaptive TS-KRR GPC has a good performance. Again, the adaptive TS-KRR GPC controller successfully eliminated the disturbances under a sinusoidal reference signal.

As a conclusion, the proposed adaptive controller showed good results and was able to deal with disturbances. In future work, more generalized kernel approaches will be used to introduce more effective online/offline TS fuzzy systems for system identification and control.

REFERENCES

Abghari, S. Z., Sadi, M. (2014). Application of adaptive neuro-fuzzy inference system for the prediction of the yield distribution of the main products in the steam cracking of atmospheric gasoil. J Taiwan Inst Chem Eng, 44, 365-76.

Ali, E. (2003). Heuristic on-line tuning for nonlinear model predictive controllers using fuzzy logic. Journal of Process Control, 13, 383-396.

Babuska, R. (1998). Fuzzy Modeling for Control, Kluwer Academic Publishers.

Bououden, S., Chadli, M., Allouani, F., & Filali, S. (2013). A new approach for fuzzy predictive adaptive controller design using particle swarm optimization algorithm. International Journal of Innovative Computing, Information and Control, 9(9), 3741-3758.

Bououden, S., Chadli, M., & Karimi, H. R. (2015). An ant colony optimization-based fuzzy predictive control approach for nonlinear processes. Information Sciences, 299, 143-158.

Bououden, S., Chadli, M., & Karimi, H. R. (2015). Control of uncertain highly nonlinear biological process based on Takagi–Sugeno fuzzy models. Signal Processing, 108, 195-205.

Brujin, P. M., & Verbruggen, H. B. (1984). Model algorithmic control using impulse response model. Journal A, 25(2), 69-74.

Cao, Y. (2005). A formulation of nonlinear model predictive control using automatic differentiation. Journal of Process Control, 15, 851-858.

Chang, Y.L., & Tsai, C.C. (2013). Adaptive stable generalized predictive control using TSK fuzzy model for nonlinear discrete-time systems with time-delays. International Journal of Fuzzy Systems, 15(2), 133-141.

Chen, W. H., Balance, D. J., Gawthrop, P. J., Gribble, J. J., & O'Reilly, J. (1999). Nonlinear PID predictive controller. IEE Proceedings Control Theory and Applications, 146(6), 603-611.

Chen, S., & Billings, S. (1992). Neural networks for nonlinear dynamic system modelling and identification. Intl. J. Control, 56, 319-346.

Chen, D., Zhao, W., Sprott, J. C., & Ma, X. (2013). Application of Takagi-Sugeno fuzzy model to a class of chaotic synchronization and anti-synchronization. Nonlinear Dynamics, 73, 1495-1505.

Chiang, J. H., & Hao, P. Y. (2004). Support vector learning mechanism for fuzzy rule-based modeling: a new approach, IEEE Transactions on Fuzzy Systems, 12(1), 1-11.

Clarke, D. W., Mothadi, C., & Tuffs, P.S.C. (1989). Generalized predictive control - Part 1. The basic algorithm. *Automatica*, 23, 137-148.

Clarke, D. W., & Mohtadi, C. (1989). Properties of generalized predictive control. *Automatica*, 25(6), 859-875.

Cortes, C., & Vapnik, V. (1995). Support-Vector Networks. Machine Learning, 20(3), 273-297.

Driankov, D., Hellendoorn, H., & Reinfrank, M. (1993). An Introduction to Fuzzy Control, Springer-Verlag.

Dunn, J. C. (1973). A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters. Journal of Cybernetics. 3 (3): 32-57.

Engel, Y., Mannor, S., & Meir, R. (2004). The kernel recursive least squares algorithm. IEEE Transactions on Signal Processing, 52(8), 2275- 2285.

Eski, I., & Temürlenk, A. (2013). Design of neural network-based control systems for active steering system, Nonlinear Dynamic, 73, 1443-1454.

Flores, A., Sáez, D., Araya, J., Berenguel, M., & Cipriano, A. (2005). Fuzzy predictive control of a solar power plant. IEEE Transactions on Fuzzy Systems, 13(1), 58-68.

Forgy, E. (1965). Cluster Analysis of Multivariate Data: Efficiency versus Interpretability of Classification, Bio- metrics, 2, 768-769.

Girosi, F. (1998). An equivalence between Sparse Approximation and Support Vector Machines. Neural Computation, 10(6), 1455-1480.

Guo, Z., & Guan, X. (2015). Nonlinear generalized predictive control based on online least squares support vector machines, Nonlinear Dynamic, 79, 1163-1168.

Huang, J.Q., & Lewis, F.L. (2003). Neural-network predictive control for nonlinear dynamic systems with time-delay. IEEE Transactions on Neural Networks, 14(2), 377-389.

Jang, J. S. R. (1991). Fuzzy Modeling Using Generalized Neural Networks and Kalman Filter Algorithm, Proc. of the Ninth National Conf. on Artificial Intelligence (AAAI-91), 762-767.

Jang, J. S. R. (1993). ANFIS: Adaptive-Network-based Fuzzy Inference Systems, IEEE Transactions on Systems, Man, and Cybernetics, 23(3), 665-685.

Juang, C. F. & Hsieh, C. D. (2009). TS-fuzzy system-based support vector regression, Fuzzy Sets and Systems, 160, 2486-2504, 2009.

Kanev, S., & Verhaegen, M. (2000). Controller reconfiguration for non-linear systems, Control Engineering Practice 8: 1223-1235, 2000.

Kennedy, J., & Eberhart, R. C. (1995). Particle Swarm Optimization, Proceedings of the IEEE International Joint Conference on Neural Networks, 4, 1942-1948.

Li, C., Zhou, J., Xiang, X., Li, Q., & An, X. (2009). T-S fuzzy model identification based on a novel fuzzy C-regression model clustering algorithm. Engineering Applications of Artificial Intelligence, 22(4-5), 646-653.

Lin, C.T., Liang, S.F., Yeh, C.M., & Fan, K.W. (2005). Fuzzy neural network design using support vector regression for function approximation with outliers, in: Proc. IEEE Internat. Conf. on System, Man and Cybernetics, 3, 2763-2768.

Lu, C. H., & Tsai, C.C. (2004). Adaptive neural predictive control for industrial multivariable processes, Journal of Systems and Control Engineering, 218(7), 557-567.

Lu, C. H., & Tsai, C. C. (2007). Generalized predictive control using recurrent fuzzy neural networks for industrial processes. Journal of process control,17(1), 83-92.

Mahfouf, M., Linkens, D.A., & Abbod, M.F. (2000). Multi-objective genetic optimization of GPC and SOFLC tuning parameters using a fuzzybased ranking method, IEE Proceedings—Control Theory and Applications, 147(3), 344-354.

Marchetti, J. L., Mellichamp, D. A., & Seborg, D.E. (1983). Predictive control based on discrete convolution models. Industrial Engineering Chemical Process Design and Development, 22, 488-495.

Mendes, J., Araújo, R., & Souza, F. (2013). Adaptive fuzzy identification and predictive control for industrial processes. Expert Systems with Applications, 40(17), 6964-6975.

Mercer, J. (1909). Functions of positive and negative type, and their connection with the theory of integral equations. Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character, 209, 415-446.

Mollov, S., Babuska, R., Abonyi, J., & Verbruggen, H.B. (2004). Effective optimization for fuzzy model predictive control, IEEE Transactions on Fuzzy Systems, 12(5), 661-675.

Nicolao, G. D., Magi, L., & Scattolini, R. (1997). Stabilizing predictive control of nonlinear ARX Models, Automatica, 33(9), 1691-1697.

Oviedo, J. J. E., Vandewalle, J. P., & Wertz, V. (2006). Fuzzy logic, identification and predictive control. Springer Science & Business Media.

Palos, A.G., Parthasarathy, S., & Atiya, A.F. (2001). Neural-predictive process control using on-line controller adaptation, IEEE Transactions on Control System Technology, 9(5), 741-755.

Prett, D. M., & Garcia, C. E. (1988). Fundamental Process Control. Butterworths, Boston.

Rastegar, S., Araújo, R., & Mendes, J. (2014). A new approach for online TS fuzzy identification and model predictive control of nonlinear systems. Journal of Vibration and Control, 1077546314544894.

Richalet, J. (1993). Industrial applications of model based predictive control. *Automatica*, 29, 1251-1274.

Richalet, J., Rault, A., Testud, J. L., & Papon, J. (1978). Model predictive heuristic control: Applications to industrial processes. *Automatica* 14, 413-428.

Rossiter, J., Kouvaritakis, B., & Dunnett, R. (1991). Application of generalised predictive control to a boiler-turbine unit for electricity generation. IEE Proceedings Part D, 138(1), 59-67.

Sáez, D., Milla, F., & Vargas, L. (2007). Fuzzy Predictive Supervisory Control based on Genetic Algorithms for Gas Turbines of Combined Cycle Power Plants. IEEE Transaction on Energy Conversion, 22(3), 689-696.

Saunders, C., Gammerman, A., & Vovk, V. (1998). Ridge regression learning algorithm in dual variables, 15th International Conference on Machine Learning, 515-521, Madison, WI.

Schölkopf, B., Herbrich, R., & Smola, A. J. (2001). A generalized representer theorem. In Computational learning theory, 416-426, Springer.

Shcherbakov, M. V., Brebels, A., Shcherbakova, N. L., Tyukov, A. P., Janovsky, T. A., & Kamaev, V. A. E. (2013). A survey of forecast error measures. World Applied Sciences Journal, 24, 171-176.

Sousa, J.M. (2000). Optimization issues in predictive control with fuzzy objective functions, International Journal of Intelligent System 15, 879-899.

Sousa, J.M.D.C., & Kaymak, U. (2001). Model prediction control using fuzzy decision functions, IEEE Transactions on System, Man, and Cybernetics— part B: Cybernetics, 31(1), 54-65.

Takagi, T., & Sugeno, M. (1985). Fuzzy identification of systems and its application to modeling and control, IEEE Trans. Syst., Man, Cybern, 15, 116-132.

Tsai, P.F., Chu, J.Z., Jang, S.S., & Shieh, S.S. (2002). Developing a robust model predictive control architecture through regional knowledge analysis of artificial neural networks, Journal of Process Control, 13, 423-435.

Van der Merwe, D. W., & Engelbrecht, A. P. (2003). Data clustering using particle swarm optimization. In Evolutionary Computation, 2003 CEC'03. The 2003 Congress, 1, 215-220.

Van Vaerenbergh, S., Vía, J., & Santamaria, I. (2006). A sliding-window kernel RLS algorithm and its application to nonlinear channel identification. In 2006 IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP), 789-792, Toulouse, France.

Zadeh, L. (1973). Outline of a new approach to the analysis of complex systems and decision processes, IEEE Trans. Syst., Man, Cybern, Part B 3(1), 28-44.

Zamarreno, J.M., & Vega, P. (1999). Neural predictive control, application to a highly non-linear system, Engineering Applications of Artificial Intelligence, 12, 149-158.

Zhang, Y., Chai, T., Wang H., & Fu, J. (2010). Generalized predictive control method for a class of nonlinear systems using ANFIS and multiple models. In: 49th IEEE conference on decision and control, Atlanta, GA: 15-17.

Zhu, Q., Warwick, K., & Douce, J. (1991). Adaptive general predictive controller for nonlinear systems. IEE Proceedings Part D, 138(1), 33-40.