KYBERNETIKA — VOLUME *54* (2018), NUMBER 1, PAGES 79–95

A NOVEL ALGORITHM FOR THE MODELING OF COMPLEX PROCESSES

José de Jesús Rubio, Edwin Lughofer, Plamen Angelov, Juan Francisco Novoa and Jesús A. Meda-Campaña

In this investigation, a new algorithm is developed for the updating of a neural network. It is concentrated in a fuzzy transition between the recursive least square and extended Kalman filter algorithms with the purpose to get a bounded gain such that a satisfactory modeling could be maintained. The advised algorithm has the advantage compared with the mentioned methods that it eludes the excessive increasing or decreasing of its gain. The gain of the recommended algorithm is uniformly stable and its convergence is found. The new algorithm is employed for the modeling of two synthetic examples.

Keywords: recursive least square, Kalman filter, modeling, complex processes

Classification: 93A30

1. INTRODUCTION

In recent years, the recursive least square and extended Kalman filter algorithms have been highly utilized in the modeling issue. The recursive least square technique is an adaptive filter which recursively finds coefficients that minimize a weighted cost function relating to input signals, and it shows extremely fast convergence [4, 20]. The Kalman filter strategy is an algorithm that employes a series of measurements observed over time, containing statistical noise and other inaccuracies, and it estimates unknown variables. In the estimation theory, the extended Kalman filter is the nonlinear version of the Kalman filter which is the linearization about an estimate of the current mean and covariance [5, 16].

There is some research about recursive least square algorithms. In [24], the least square and backpropagation are combined. The least square method is addressed in [9]. In [17], fuzzy least squares are commented. The recursive fuzzily weighted least square is employed for updating consequent parameters in evolving fuzzy systems [21] and [30], which is extended to a generalized form in [26]. The characteristic of this algorithm is that its gain could converge through the time to a small value. The problem is that the gain could be too small; therefore, the quality of the modeling could become low.

DOI: 10.14736/kyb-2018-1-0079

There is some research about extended Kalman filter algorithms. In [1, 2, 3], and [31], several Kalman filter algorithms of neural networks are developed. An extended Kalman filter of a wavelet neural network is explored in [13]. In [8] and [33], the programming with Kalman filters is expressed. An observer-type of Kalman filtering algorithm is addressed in [12]. In [11], the Kalman filter of nonlinear processes is commented. Single-pass active modeling filters are employed in [22, 23] for the semi-supervised drift detection. The characteristic of this algorithm is that its gain could grow through the time to a big value. The problem is that the gain could be too big; therefore, the quality of the modeling could become low.

In this research, a new algorithm is employed for the updating of a feedforward neural network with one hidden layer. Compared with the mentioned methods, the advised algorithm is a combination between the recursive least square and extended Kalman filter such as it is concentrated in a fuzzy transition between both algorithms with the purpose to get a bounded gain, maintaining a satisfactory modeling.

Furthermore, the Lyapunov technique is employed to ensure the uniform stability and convergence of the gain in the recommended algorithm. Stability is a method to analyze whether the inputs, outputs, and parameters remain bounded through the time [6, 10, 27, 32, 37]. The uniform stability is stronger than the common stability due to the first is satisfied for any initial time, while the second is satisfied only for a zero initial time.

Finally, the new algorithm is compared with the recursive least square and extended Kalman filter for the modeling of two complex processes. The complex adaptive processes issue has been explored as a well established research area [14, 15, 25].

The paper is organized. The neural network, recursive least square, extended Kalman filter, and new algorithm are explained in Section 2. The advised technique is summarized in Section 3. The recommended method is applied for the modeling of two synthetic examples in Section 4. Conclusions and future research are explained in Section 5.

2. UPDATING ALGORITHMS OF A NEURAL NETWORK

In this part of the article: first, the neural network will be explained, second, the recursive least square, extended Kalman filter, and new algorithm will be explained for the updating of a neural network, and third, the stability and convergence of the gain in the advised algorithm will be analyzed.

2.1. The neural network

Take into account the next unknown complex process:

$$y(k) = f\left[x(k)\right] \tag{1}$$

with

$$\begin{aligned} x(k) &= [x_1(k), \ldots, x_i(k), \ldots, x_N(k)]^T \\ &= [y(k-1), \ldots, y(k-n), u(k), \ldots, u(k-m)]^T \in \Re^{N\times 1} \end{aligned}$$

$N = n + m$ is the process input, $y(k) = \upsilon \in \Re$ is the process output, and f is the unknown behavior of the complex process, $f \in C^{\infty}$.

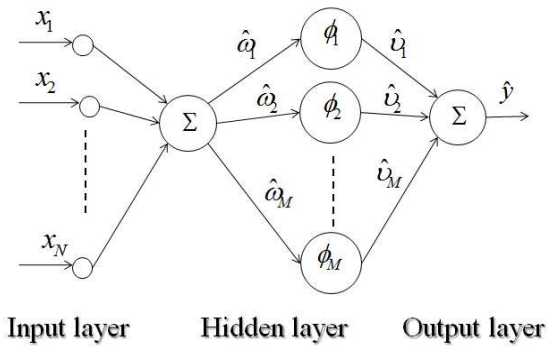

Fig. 1. The neural network with one hidden layer.

In this study, a special neural network is utilized which only has one hidden layer. It could be extended to a general multilayer neural network; however, this research is concentrated in a smaller neural network.

The structure of the neural network with one hidden layer of this study is shown in Figure 1.

The next neural network with the input, hidden, and output layers is written:

$$\begin{aligned} \widehat{y}(k) &= \widehat{\upsilon}(k)\phi(k) = \sum_{j=1}^{M} \widehat{\upsilon}_j(k)\phi_j(k) \\ \phi(k) &= [\phi_1(k), \ldots, \phi_j(k), \ldots, \phi_M(k)]^T \\ \phi_j(k) &= \tanh\left\{\widehat{\omega}_j(k) \sum_{i=1}^{N} x_i(k)\right\} \end{aligned} \tag{2}$$

with $i = 1, \ldots, N$, $j = 1, \ldots, M$, $x(k) \in \Re^{N\times 1}$ is the neural network input expressed in (1), $x_i(k) \in \Re$, $\widehat{y}(k) \in \Re$ is the neural network output, $\widehat{\omega}(k) \in \Re^{1\times M}$ and $\widehat{\upsilon}(k) \in \Re^{1\times M}$ are the hidden layer and output weights, $\widehat{\omega}_j(k) \in \Re$, $\widehat{\upsilon}_j(k) \in \Re$, $\phi_j(k) \in \Re$, $\phi(k) \in \Re^{M\times 1}$.

The next modeling error $e(k) \in \Re$ is expressed:

$$e(k) = \widehat{y}(k) - y(k) \tag{3}$$

with $y(k)$ and $\widehat{y}(k)$ being expressed in (1) and (2).

In the next three subsections, three alternative strategies for the updating of the neural network will be explained: the recursive least square, extended Kalman filter, and a new algorithm.

2.2. The recursive least square algorithm

The recursive least square algorithm is expressed in this subsection for the updating of a neural network. The characteristic of this algorithm is that its gain G_k could converge through the time to a small value. The problem is that the gain could be too small; consequently, the quality of the modeling could become low.

The next recursive least square algorithm utilized as the adapting law of the neural network (2) for the modeling of the complex process (1) [28]:

$$\begin{aligned} \widehat{\psi}(k+1) &= \widehat{\psi}(k) - \tfrac{1}{q(k)} G_{k+1} a(k) e(k) \\ G_{k+1} &= G_k - \tfrac{1}{r(k)} G_k a(k) a^T(k) G_k \end{aligned} \tag{4}$$

with

$$\begin{aligned} a^T(k) &= [\sigma_1(k), \ldots, \sigma_M(k), \phi_1(k), \ldots, \phi_M(k)] \in \Re^{1\times 2M} \\ \widehat{\psi}(k) &= [\widehat{\omega}_1(k), \ldots, \widehat{\omega}_M(k), \widehat{v}_1(k), \ldots, \widehat{v}_M(k)]^T \in \Re^{2M\times 1} \\ r(k) &= q(k) + a^T(k) G_k a(k) \\ q(k) &= r_2 + a^T(k) G_k a(k) \in \Re \\ \sigma_j(k) &= \widehat{v}_j(k) sech^2 \left\{ \widehat{\omega}_j(k) \sum_{i=1}^{N} x_i(k) \right\} \sum_{i=1}^{N} x_i(k) \end{aligned}$$

$0 < r_2 \in \Re$ is a forgetting factor, $e(k)$ is the modeling error of (3), $\phi_j(k)$ is expressed in (2), $\sigma_j(k)$ is the partial derivative of $\phi_j(k)$ with respect to $\widehat{\omega}_j(k)$. $G_{k+1} \in \Re^{2M\times 2M}$ is the algorithm gain which is a positive definite covariance matrix, $G_1 = g_1 I$ is the initial algorithm gain, and $g_1 > 0$ is a scalar constant usually big enough to ensure an acceptable convergence, and $I \in \Re^{2M\times 2M}$ is the identity matrix.

2.3. The extended Kalman filter algorithm

The extended Kalman filter algorithm is expressed in this subsection for the updating of a neural network. The characteristic of this algorithm is that its gain G_k could grow through the time to a big value. The problem is that the gain could be too big; consequently, the quality of the modeling could become low.

The next extended Kalman filter algorithm utilized as the adapting law of the neural network (2) for the modeling of the complex process (1) [29]:

$$\begin{aligned} \widehat{\psi}(k+1) &= \widehat{\psi}(k) - \tfrac{1}{q(k)} G_{k+1} a(k) e(k) \\ G_{k+1} &= G_k - \tfrac{1}{r(k)} G_k a(k) a^T(k) G_k + R_1 \end{aligned} \tag{5}$$

with

$$\begin{aligned} a^T(k) &= [\sigma_1(k), \ldots, \sigma_M(k), \phi_1(k), \ldots, \phi_M(k)] \in \Re^{1\times 2M} \\ \widehat{\psi}(k) &= [\widehat{\omega}_1(k), \ldots, \widehat{\omega}_M(k), \widehat{v}_1(k), \ldots, \widehat{v}_M(k)]^T \in \Re^{2M\times 1} \\ r(k) &= q(k) + a^T(k) G_k a(k) \\ q(k) &= r_2 + a^T(k) G_k a(k) \in \Re \\ \sigma_j(k) &= \widehat{v}_j(k) sech^2 \left\{ \widehat{\omega}_j(k) \sum_{i=1}^{N} x_i(k) \right\} \sum_{i=1}^{N} x_i(k) \end{aligned}$$

$0 < r_2 \in \Re$ is a forgetting factor, $e(k)$ is the modeling error of (3), $\phi_j(k)$ is expressed in (2), $\sigma_j(k)$ is the partial derivative of $\phi_j(k)$ with respect to $\widehat{\omega}_j(k)$. $G_{k+1} \in \Re^{2M\times 2M}$ is the algorithm gain which is a positive definite covariance matrix, $G_1 = g_1 I$ is the initial algorithm gain, and $g_1 > 0$ is a scalar constant usually big enough to ensure an acceptable convergence, and $I \in \Re^{2M\times 2M}$ is the identity matrix. $R_1 = r_1 I$, $0 < r_1 \in \Re$.

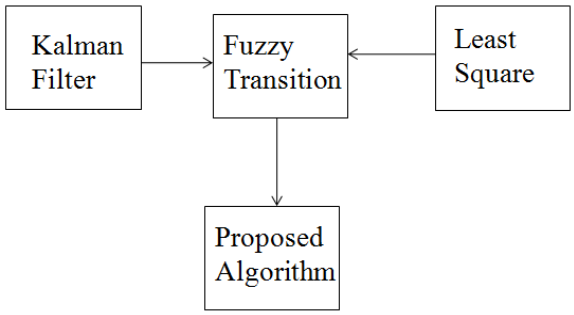

Fig. 2. The new algorithm.

2.4. The new algorithm

The new algorithm is expressed in this subsection for the updating of a neural network. It is concentrated in a fuzzy transition between the recursive least square and extended Kalman filter algorithms with the purpose to get a bounded gain G_k, maintaining a satisfactory modeling. Figure 2 shows the advised algorithm.

The next recommended algorithm utilized as the adapting law of the neural network (2) for the modeling of the complex process (1):

$$\begin{aligned} \widehat{\psi}(k+1) &= \widehat{\psi}(k) - \tfrac{1}{q(k)} G_{k+1} a(k) e(k) \\ G_{k+1} &= G_k - \tfrac{1}{r(k)} G_k a(k) a^T(k) G_k + R_1 \end{aligned} \tag{6}$$

with

$$\begin{aligned} a^T(k) &= [\sigma_1(k), \ldots, \sigma_M(k), \phi_1(k), \ldots, \phi_M(k)] \in \Re^{1\times 2M} \\ \widehat{\psi}(k) &= [\widehat{\omega}_1(k), \ldots, \widehat{\omega}_M(k), \widehat{v}_1(k), \ldots, \widehat{v}_M(k)]^T \in \Re^{2M\times 1} \end{aligned}$$

$$\begin{aligned} r(k) &= q(k) + a^T(k) G_k a(k) \\ q(k) &= r_2 + a^T(k) G_k a(k) \in \Re \end{aligned}$$

$$\sigma_j(k) = \widehat{v}_j(k) sech^2 \left\{ \widehat{\omega}_j(k) \sum_{i=1}^{N} x_i(k) \right\} \sum_{i=1}^{N} x_i(k)$$

$0 < r_2 \in \Re$ is a forgetting factor, $e(k)$ is the modeling error of (3), $\phi_j(k)$ is expressed in (2), $\sigma_j(k)$ is the partial derivative of $\phi_j(k)$ with respect to $\widehat{\omega}_j(k)$. $G_{k+1} \in \Re^{2M\times 2M}$ is the algorithm gain which is a positive definite covariance matrix, $G_1 = g_1 I$ is the initial algorithm gain, and $g_1 > 0$ is a scalar constant usually big enough to ensure an acceptable convergence, and $I \in \Re^{2M\times 2M}$ is the identity matrix. The next 5 rules are employed to get R_1 as a fuzzy transition between the recursive least square $R_1 = 0I$ and

the extended Kalman filter $R_1 = r_1 I$:

$$\begin{aligned}
&\text{if } 0 \leq \|G_k\| \leq \tfrac{1}{4} g_1, \text{ then } R_1 = r_1 I \\
&\text{if } \tfrac{1}{4} g_1 < \|G_k\| \leq \tfrac{1}{2} g_1, \text{ then } R_1 = \tfrac{3}{4} r_1 I \\
&\text{if } \tfrac{1}{2} g_1 < \|G_k\| \leq \tfrac{3}{4} g_1, \text{ then } R_1 = \tfrac{1}{2} r_1 I \\
&\text{if } \tfrac{3}{4} g_1 < \|G_k\| \leq g_1, \text{ then } R_1 = \tfrac{1}{4} r_1 I \\
&\text{if } g_1 < \|G_k\|, \text{ then } R_1 = 0 I
\end{aligned} \tag{7}$$

with $0 < r_1 \in \Re$.

The stability and convergence of the gain in the recommended algorithm are analyzed by the next Theorem.

Theorem 2.1. The gain of the new algorithm (6) (7) for the updating of the neural network (2), (3) is uniformly stable and the next convergence is satisfied:

$$\limsup_{T \to \infty} \frac{1}{T} \sum_{k=1}^{T} b^T \frac{1}{r(k)} G_k a(k) a^T(k) G_k b = r_1 \tag{8}$$

with $b = [1, 1 \ldots, 1]^T$, G_k is the gain of the advised algorithm expressed in (6), $a(k)$ and $r(k)$ are expressed in (6), and r_1 is expressed in (7).

P r o o f. Chose the next Lyapunov function:

$$L_k = b^T G_k b \tag{9}$$

getting ΔL_k:

$$\Delta L_k = L_{k+1} - L_k = b^T G_{k+1} b - b^T G_k b \tag{10}$$

five cases are commented. a) when $0 \leq \|G_k\| \leq \frac{1}{4} g_1$, taking into account (6) and that $b^T b = 1$:

$$\begin{aligned}
\Delta L_k &= b^T G_{k+1} b - b^T G_k b \\
&= b^T \left[G_k - \tfrac{1}{r(k)} G_k a(k) a^T(k) G_k + r_1 \right] b - b^T G_k b \\
&= b^T G_k b - b^T \tfrac{1}{r(k)} G_k a(k) a^T(k) G_k b + b^T r_1 b - b^T G_k b \\
\Delta L_k &= -b^T \tfrac{1}{r(k)} G_k a(k) a^T(k) G_k b + r_1
\end{aligned} \tag{11}$$

The next result is gotten:

$$\Delta L_k = -b^T \frac{1}{r(k)} G_k a(k) a^T(k) G_k b + r_1 \tag{12}$$

since $b^T \frac{1}{r(k)} G_k a(k) a^T(k) G_k b \geq 0$ and r_1 is small and positive, the gain of the algorithm is uniformly stable. b) when $\frac{1}{4} g_1 < \|G_k\| \leq \frac{1}{2} g_1$, taking into account (6) and that

$b^T b = 1$:

$$\begin{aligned}\Delta L_k &= b^T G_{k+1} b - b^T G_k b \\ &= b^T \left[G_k - \tfrac{1}{r(k)} G_k a(k) a^T(k) G_k + \tfrac{3}{4} r_1 \right] b - b^T G_k b \\ &= b^T G_k b - b^T \tfrac{1}{r(k)} G_k a(k) a^T(k) G_k b - b^T G_k b + \tfrac{3}{4} r_1 b^T b \\ &= -b^T \tfrac{1}{r(k)} G_k a(k) a^T(k) G_k b + \tfrac{3}{4} r_1\end{aligned} \tag{13}$$

The next result is gotten:

$$\Delta L_k = -b^T \frac{1}{r(k)} G_k a(k) a^T(k) G_k b + \frac{3}{4} r_1 \tag{14}$$

since $b^T \frac{1}{r(k)} G_k a(k) a^T(k) G_k b \geq 0$ and $\frac{3}{4} r_1$ is small and positive, the gain of the algorithm is uniformly stable. c) when $\frac{1}{2} g_1 < \|G_k\| \leq \frac{3}{4} g_1$, taking into account (6) and that $b^T b = 1$:

$$\begin{aligned}\Delta L_k &= b^T G_{k+1} b - b^T G_k b \\ &= b^T \left[G_k - \tfrac{1}{r(k)} G_k a(k) a^T(k) G_k + \tfrac{1}{2} r_1 \right] b - b^T G_k b \\ &= b^T G_k b - b^T \tfrac{1}{r(k)} G_k a(k) a^T(k) G_k b - b^T G_k b + \tfrac{1}{2} r_1 b^T b \\ &= -b^T \tfrac{1}{r(k)} G_k a(k) a^T(k) G_k b + \tfrac{1}{2} r_1\end{aligned} \tag{15}$$

The next result is gotten:

$$\Delta L_k = -b^T \frac{1}{r(k)} G_k a(k) a^T(k) G_k b + \frac{1}{2} r_1 \tag{16}$$

since $b^T \frac{1}{r(k)} G_k a(k) a^T(k) G_k b \geq 0$ and $\frac{1}{2} r_1$ is small and positive, the gain of the algorithm is uniformly stable. d) when $\frac{3}{4} g_1 < \|G_k\| \leq g_1$, taking into account (6) and that $b^T b = 1$:

$$\begin{aligned}\Delta L_k &= b^T G_{k+1} b - b^T G_k b \\ &= b^T \left[G_k - \tfrac{1}{r(k)} G_k a(k) a^T(k) G_k + \tfrac{1}{4} r_1 \right] b - b^T G_k b \\ &= b^T G_k b - b^T \tfrac{1}{r(k)} G_k a(k) a^T(k) G_k b - b^T G_k b + \tfrac{1}{4} r_1 b^T b \\ &= -b^T \tfrac{1}{r(k)} G_k a(k) a^T(k) G_k b + \tfrac{1}{4} r_1\end{aligned} \tag{17}$$

The next result is gotten:

$$\Delta L_k = -b^T \frac{1}{r(k)} G_k a(k) a^T(k) G_k b + \frac{1}{4} r_1 \tag{18}$$

since $b^T \frac{1}{r(k)} G_k a(k) a^T(k) G_k b \geq 0$ and $\frac{1}{4} r_1$ is small and positive, the gain of the algorithm is uniformly stable. e) when $g_1 < \|G_k\|$, taking into account (6) and that $b^T b = 1$:

$$\begin{aligned}\Delta L_k &= b^T G_{k+1} b - b^T G_k b \\ &= b^T \left[G_k - \tfrac{1}{r(k)} G_k a(k) a^T(k) G_k \right] b - b^T G_k b \\ &= b^T G_k b - b^T \tfrac{1}{r(k)} G_k a(k) a^T(k) G_k b - b^T G_k b \\ &= -b^T \tfrac{1}{r(k)} G_k a(k) a^T(k) G_k b\end{aligned} \tag{19}$$

The next result is gotten:

$$\Delta L_k = -b^T \frac{1}{r(k)} G_k a(k) a^T(k) G_k b \tag{20}$$

since $b^T \frac{1}{r(k)} G_k a(k) a^T(k) G_k b \geq 0$, the gain of the algorithm is asymptotically stable. Since the uniform stability is weaker than the asymptotic stability [6, 10, 27, 32, 37], taking into account all cases, the gain of the new algorithm is uniformly stable. Now taking into account all cases. a) when $0 \leq \|G_k\| \leq \frac{1}{4} g_1$, summarize (12) from 1 to T:

$$\sum_{k=1}^{T} b^T \frac{1}{r(k)} G_k a(k) a^T(k) G_k b = L_1 - L_T + T r_1 \tag{21}$$

multiplying by $\frac{1}{T}$ and getting the limit:

$$\limsup_{T \to \infty} \frac{1}{T} \sum_{k=1}^{T} b^T \frac{1}{r(k)} G_k a(k) a^T(k) G_k b = r_1 \tag{22}$$

b) when $\frac{1}{4} g_1 < \|G_k\| \leq \frac{1}{2} g_1$, summarize (14) from 1 to T:

$$\sum_{k=1}^{T} b^T \frac{1}{r(k)} G_k a(k) a^T(k) G_k b = L_1 - L_T + T \frac{3}{4} r_1 \tag{23}$$

multiplying by $\frac{1}{T}$ and getting the limit:

$$\limsup_{T \to \infty} \frac{1}{T} \sum_{k=1}^{T} b^T \frac{1}{r(k)} G_k a(k) a^T(k) G_k b = \frac{3}{4} r_1 \tag{24}$$

c) when $\frac{1}{2} g_1 < \|G_k\| \leq \frac{3}{4} g_1$, summarize (14) from 1 to T:

$$\sum_{k=1}^{T} b^T \frac{1}{r(k)} G_k a(k) a^T(k) G_k b = L_1 - L_T + T \frac{1}{2} r_1 \tag{25}$$

multiplying by $\frac{1}{T}$ and getting the limit:

$$\limsup_{T \to \infty} \frac{1}{T} \sum_{k=1}^{T} b^T \frac{1}{r(k)} G_k a(k) a^T(k) G_k b = \frac{1}{2} r_1 \tag{26}$$

d) when $\frac{3}{4} g_1 < \|G_k\| \leq g_1$, summarize (14) from 1 to T:

$$\sum_{k=1}^{T} b^T \frac{1}{r(k)} G_k a(k) a^T(k) G_k b = L_1 - L_T + T \frac{1}{4} r_1 \tag{27}$$

multiplying by $\frac{1}{T}$ and getting the limit:

$$\limsup_{T\to\infty}\frac{1}{T}\sum_{k=1}^{T} b^T\frac{1}{r(k)}G_k a(k)a^T(k)G_k b=\frac{1}{4}r_1 \tag{28}$$

e) when $g_1<\|G_k\|$, summarize (14) from 1 to T:

$$\sum_{k=1}^{T} b^T\frac{1}{r(k)}G_k a(k)a^T(k)G_k b=L_1-L_T \tag{29}$$

multiplying by $\frac{1}{T}$ and getting the limit:

$$\limsup_{T\to\infty}\frac{1}{T}\sum_{k=1}^{T} b^T\frac{1}{r(k)}G_k a(k)a^T(k)G_k b=0. \tag{30}$$

Since the limit (22) is the weakest of all, taking into account all cases, the gain of the convergence (22) is (8), it holds the result. □

Remark 2.2. Even if the neural network of this research is equal with the neural network of [28], there are three differences between the proposed research and the investigation of [28]: first, in the investigation of [28] the least square algorithm is employed as the updating law of the neural network, while in this research a new algorithm with a fuzzy transition between the recursive least square and extended Kalman filter methods is proposed as the updating law of the neural network; second, in the investigation of [28] the stability and convergence of the modeling error is ensured, while in this research the stability and convergence of the gain in the new algorithm is ensured; and third, the algorithm of [28] is applied in the modeling of two crude oil blending processes, while the algorithm of this research is applied in the modeling of two synthetic examples.

Remark 2.3. There is some research about fuzzy transition systems. A fuzzy transition based model is recommended in [7] and [18]. In [19, 34, 35, 36], nondeterministic fuzzy transition systems are explored. Contrary to the nondeterministic fuzzy transition applied in the aforementioned texts, the deterministic fuzzy transition is employed in this investigation.

Remark 2.4. The description about how the new algorithm improves both the least square and Kalman filter is mathematically explained. From the gain $G_{k+1}=G_k-\frac{1}{r(k)}G_k a(k)a^T(k)G_k$ of equation (4), it can be seen that through the time G_k of the least square algorithm could converge to zero due to $r(k)$ is a positive term. From the gain $G_{k+1}=G_k-\frac{1}{r(k)}G_k a(k)a^T(k)G_k+R_1$ of equation (5), it can be seen that through the time G_k of the Kalman filter algorithm could converge to infinity due to $r(k)$ is a positive term and $R_1=r_1 I$ is a constant positive term with value r_1. From the gain $G_{k+1}=G_k-\frac{1}{r(k)}G_k a(k)a^T(k)G_k+R_1$ of equation (6), it can be seen that through the time G_k of the new algorithm eludes the convergences to zero or infinity due to $r(k)$ is a positive term and R_1 is a positive semidefinite term which changes from 0 to r_1 depending of the value of $\|G_k\|$ as can be seen in equation (7).

Remark 2.5. $g_1 > 0$ is a scalar constant that must be big enough to ensure an acceptable convergence, it is due to if g_1 is very small in the least square its gain G_k could converge to zero, or if g_1 is very big in the Kalman filter algorithm its gain G_k could converge to infinity. Consequently, g_1 must be chosen inside of an interval which will be commented in the next section.

3. STEPS OF THE NEW ALGORITHM

The next steps explain the new algorithm:

1. The complex process output $y(k)$ is gotten with equation (1). The complex process should has the form represented by the equation (1); the element N is chosen in concordance with this complex process.

2. Take into account the next elements; chose weights $\widehat{\upsilon}_j(1)$ and $\widehat{\omega}_j(1)$ for (2) as random numbers between 0 and 1; chose the number of hidden layer neurons M for (2) with an integer value, chose the initial algorithm gain g_1 with a positive value, and elements r_1 and r_2 for (6), (7) with positive values.

3. For each iteration k, the neural network output $\widehat{y}(k)$ is gotten with equation (2), the modeling error $e(k)$ is gotten with equation (3), $\widehat{\psi}(k)$ is gotten with weights $\widehat{\upsilon}_j(k)$ and $\widehat{\omega}_j(k)$ utilizing (6), (7), $a^T(k)$ is gotten with $\sigma_j(k)$ and $\phi_j(k)$ utilizing (2), (6), (7), the element $\widehat{\psi}(k+1)$ is updated with equations (6), (7), weights of the neural network $\widehat{\upsilon}_j(k+1)$ and $\widehat{\omega}_j(k+1)$ with $\widehat{\psi}(k+1)$ are updated utilizing (6), (7).

4. The behavior of the algorithm could be modified by choosing different values in the elements $M \in [N, 5N]$, $g_1 \in \left[1 \times 10^2, 1 \times 10^4\right]$, $r_1 \in \left[5 \times 10^{-5}, 5 \times 10^1\right]$, or $r_2 \in \left[8 \times 10^{-2}, 5 \times 10^{-1}\right]$.

4. EXAMPLES

In this part of the article, the advised algorithm is applied for the modeling of two synthetic examples. The two chosen synthetic examples have the two main characteristics: first, they are nonlinear with the structure of equation (1), and second, they let to show the advantage in the recommended algorithm of maintaining its gain bounded. In all cases, the new algorithm called KFLS will be compared with the least square algorithm of [28] called LS, and with the extended Kalman filter algorithm of [29] called KF. The differences between three algorithms were explained in past sections. The next root mean square error denoted as MSE is employed for comparisons:

$$MSE = \left(\frac{1}{N} \sum_{k=1}^{N} e^2(k) \right)^{\frac{1}{2}} \tag{31}$$

with $e(k)$ as the modeling error of (3).

4.1. Example 1

The next complex process of the example 1 is explained [30]:

$$y(k) = \frac{y(k-1)y(k-2)\left[y(k-1)-0.5\right]}{1+y(k-1)^2+y(k-2)^2} + u(k-1) \tag{32}$$

with

$$u(k-1) = \sin\left(\frac{2\pi(k-1)}{25}\right).$$

The complex process of equations (1), (32) is utilized where inputs are $x_1(k) = y(k-1)$, $x_2(k) = y(k-2)$, $x_3(k) = u(k-1)$ and the output is $y(k) = y(k)$. The data of 3000 iterations is employed for the training and the data of the least 200 iterations is employed for the testing.

The LS of [28] is explained as (2), (3), (4) with elements $N = 3$, $M = 5$, $g_1 = 1 \times 10^3$, $r_2 = 0.2$, $\widehat{v}_j(1)$ and $\widehat{\omega}_j(1)$ employ random numbers between 0 and 0.5.

The KF [29] is explained as (2), (3), (5) with elements $N = 3$, $M = 5$, $g_1 = 1 \times 10^3$, $r_1 = 3 \times 10^{-1}$, $r_2 = 0.2$, $\widehat{v}_j(1)$ and $\widehat{\omega}_j(1)$ employ random numbers between 0 and 0.5.

The KFLS is explained as (2), (3), (6), (7) with elements $N = 3$, $M = 5$, $g_1 = 1 \times 10^3$, $r_1 = 3 \times 10^{-1}$, $r_2 = 0.2$, $\widehat{v}_j(1)$ and $\widehat{\omega}_j(1)$ employ random numbers between 0 and 0.5.

Figures 3, 4, and 5 show the comparisons for the norm of the gain ($\|G_k\|$), the training, and testing of the LS, KF, and KFLS. The training and testing MSE comparisons of (31) are shown in Table 1.

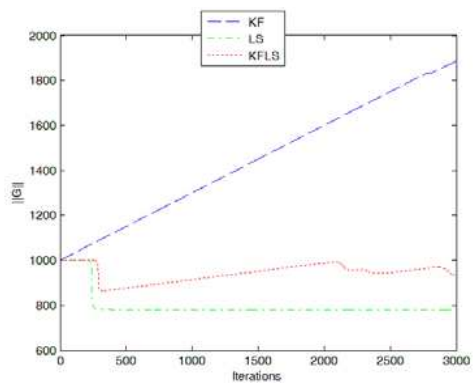

Fig. 3. Norm of the gain for the example 1.

Strategies	**Training MSE**	**Testing MSE**
LS	0.0693	0.0153
KF	0.0547	0.0367
KFLS	0.0518	0.0153

Tab. 1. Comparisons for the example 1.

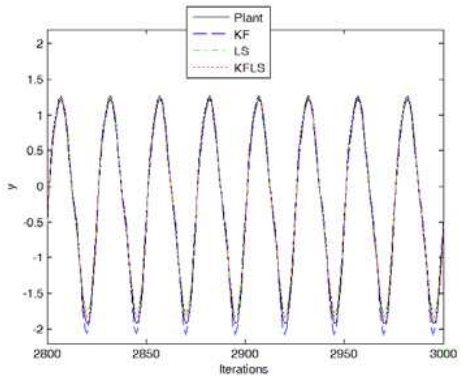

Fig. 4. Training for the example 1.

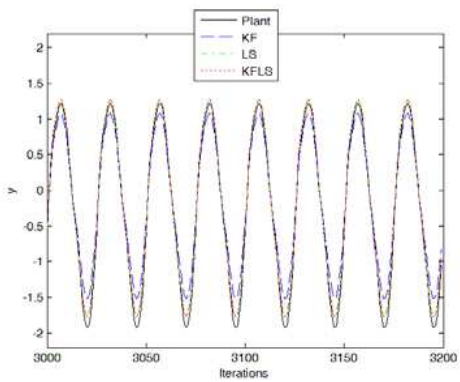

Fig. 5. Testing for the example 1

From the Figure 3, it is observed that all algorithms show bounded norms of gains. From Figures 4 and 5, it is observed that the KFLS improves both the LS and KF due to the signal of the first reaches better the signal of the plant than the signal of the other. From Table 1, it is observed that the KFLS achieves better accuracy when compared with both the LS and KF due to the MSE is smaller for the first. Then, the KFLS is the best option for the modeling in the Example 1.

4.2. Example 2

The next complex process of the example 2 is explained [30]:

$$y(k) = 0.3y(k-1) + 0.6y(k-2) + f(u(k-1)) \tag{33}$$

with

$$f(u(k-1)) = 0.6\sin(\pi u(k-1)) + 0.3\sin(3\pi u(k-1)) + 0.1\sin(5\pi u(k-1))$$

$$u(k-1) = \sin\left(\frac{2\pi(k-1)}{200}\right)$$

The complex process of equations (1), (33) where inputs are $x_1(k) = y(k-1)$, $x_2(k) = y(k-2)$, $x_3(k) = u(k-1)$ and the output is $y(k) = y(k)$. The data of 3000 iterations is employed for the training and the data of the least 200 iterations is employed for the testing.

The LS of [28] is explained as (2), (3), (4) with elements $N = 3$, $M = 5$, $g_1 = 1 \times 10^3$, $r_2 = 0.2$, $\widehat{v}_j(1)$ and $\widehat{\omega}_j(1)$ employ random numbers between 0 and 1.

The KF of [29] is explained as (2), (3), (5) with elements $N = 3$, $M = 5$, $g_1 = 1 \times 10^3$, $r_1 = 1 \times 10^{-1}$, $r_2 = 0.2$, $\widehat{v}_j(1)$ and $\widehat{\omega}_j(1)$ employ random numbers between 0 and 1.

The KFLS is explained as (2), (3), (6), (7) with elements $N = 3$, $M = 5$, $g_1 = 1 \times 10^3$, $r_1 = 1 \times 10^{-1}$, $r_2 = 0.2$, $\widehat{v}_j(1)$ and $\widehat{\omega}_j(1)$ employ random numbers between 0 and 1.

Figures 6, 7, and 8 show the comparisons for the norm of the gain ($\|G_k\|$), the training, and testing of the LS, KF, and KFLS. The training and testing MSE comparisons of (31) are shown in Table 2.

Strategies	**Training MSE**	**Testing MSE**
LS	0.1892	0.0320
KF	0.1125	0.0273
KFLS	0.1087	0.0263

Tab. 2. Comparisons for the example 2.

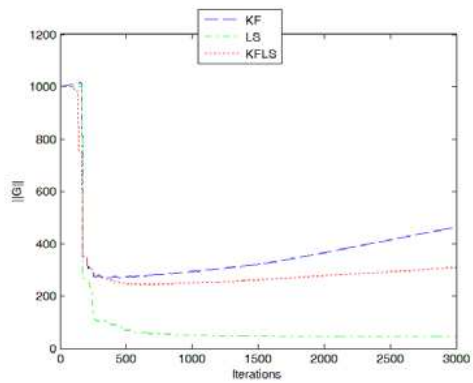

Fig. 6. Norm of the gain for the example 2.

From the Figure 6, it is observed that all algorithms show bounded norms of gains. From Figures 7 and 8, it is observed that the KFLS algorithm improves both the LS and KF due to the signal of the first reaches better the signal of the plant than the signal of the other. From Table 2, it is observed that the KFLS achieves better accuracy when compared with both the LS and KF due to the MSE is smaller for the first. Then,the KFLS is the best option for the modeling in the Example 2.

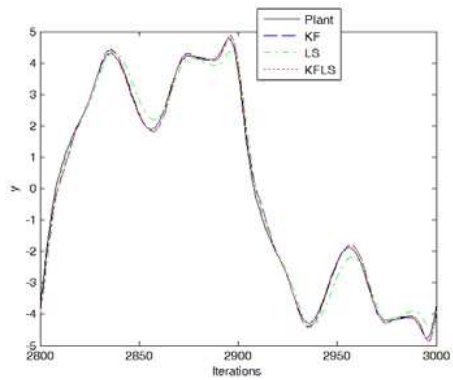

Fig. 7. Training for the example 2.

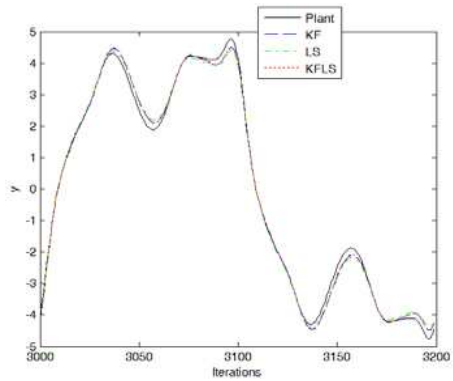

Fig. 8. Testing for the example 2.

5. CONCLUSION

In this research, a new algorithm was introduced for the updating of a neural network with one hidden layer. The stability and convergence of the gain in the advised algorithm were ensured by the Lyapunov technique. From examples, it was seen that the recommended algorithm achieves better accuracy when compared with both the recursive least square algorithm and extended Kalman filter for the modeling of two complex processes. The algorithm could be employed to as the updating of the fuzzy or evolving intelligent systems. In the future research, the new algorithm will be applied to real world complex processes or it will be employed for the control, pattern recognition, prediction, or classification.

ACKNOWLEDGEMENT

Authors are grateful to the editors and the reviewers for their valuable comments. Authors thank the Instituto Politécnico Nacional, Secretaría de Investigación y Posgrado, the Comisión de Operación y Fomento de Actividades Académicas, and Consejo Nacional de Ciencia y Tecnología for their help in this research. The second author acknowledges the support of the Austrian COMET-K2 programme of the Linz Center of Mechatronics (LCM), funded by the Austrian federal government and the federal state of Upper Austria. This publication reflects only the authors views.

(Received March 4, 2017)

REFERENCES

[1] A. Y. Alanis, L. J. Ricalde, C. Simetti, and F. Odone: Neural model with particle swarm optimization Kalman learning for forecasting in smart grids. Math. Problems Engrg. (2013), 9 pages. DOI:10.1155/2013/197690

[2] A. Y. Alanis, E. N. Sanchez, and A. G. Loukianov: A wind speed neural model with particle swarm optimization Kalman learning. In: International Joint Conference on Neural Networks 2006, pp. 1993–2000.

[3] A. Y. Alanis, C. Simetti, L. J. Ricalde, and F. Odone: A wind speed neural model with particle swarm optimization Kalman learning. In: World Automation Congress 2012, pp. 1-5.

[4] K. J. Astrom and B. Wittenmark: Adaptive Control. Second edition. Addison-Wesley Longman Publishing Co., Inc., Boston (1994).

[5] A. Bifet and R. Gavalda: Kalman filters and adaptive windows for learning in data streams. In: Discovery Science (L. Todorovski, N. Lavrac, K. P. Jantke, eds.), Lecture Notes in Computer Science *4265* (2006), pp. 29-40, Springer, Berlin, Heidelberg. DOI:10.1007/11893318

[6] S. Čelikovský: Topological equivalence and topological linearization of controlled dynamical systems. Kybernetika *31* (1995), 141–150.

[7] M. Cerrada, C. Li, R. V. Sanchez, F. Pacheco, D. Cabrera, and J. Valente: A fuzzy transition based approach for fault severity prediction in helical gearboxes. Fuzzy Sets and Systems *337* (2018), 52–73. DOI:10.1016/j.fss.2016.12.017

[8] G. Chen, Q. Xie, and L. S. Shieh: Fuzzy Kalman filtering. J. Inform. Sci. *109* (1998), 197–209. DOI:10.1016/s0020-0255(98)10002-6

[9] J. K. Coelho, M.Ḋ. Pena, and O. J. Romero: Pore-scale modeling of oil mobilization trapped in a square cavity. IEEE Latin Amer. Trans. *14* (2016), 4, 1800–1807. DOI:10.1109/tla.2016.7483518

[10] Z. Deng, X. Wang, and Y. Hong: Distributed optimisation design with triggers for disturbed continuous-time multi-agent systems. IET Control Theory Appl. *11* (2017), 2, 282–290. DOI:10.1049/iet-cta.2016.0795

[11] K. Dolinský and S. Čelikovský: Kalman filter under nonlinear system transformations. In: American Control Conference 2012, pp. 4789-4794. DOI:10.1109/acc.2012.6315366

[12] S. M. Guo, L. S. Shieh, G. Chen, and N. P. Coleman: Observer-type Kalman innovation filter for uncertain linear systems. IEEE Trans. Aerospace Eelectron. Systems *37* (2001), 4, 1406–1418. DOI:10.1109/7.976975

[13] J. E.Guillermo, L. J. Ricalde Castellanos, E. N. Sanchez, and A. Y. Alanis: Detection of heart murmurs based on radial wavelet neural network with Kalman learning. Neurocomputing *164* (2015), 307–317. DOI:10.1016/j.neucom.2014.12.059

[14] E. A. Hernandez-Vargas, P. Colaneri, and R. H. Middleton: Switching strategies to mitigate HIV mutation. IEEE Trans. Control Systems Technol. *22* (2014), 4, 1623–1628. DOI:10.1109/tcst.2013.2280920

[15] E. A. Hernandez-Vargas, P. Colaneri, and R. H. Middleton: Optimal therapy scheduling for a simplified HIV infection model. Automatica *49* (2013), 2874–2880. DOI:10.1016/j.automatica.2013.06.001

[16] R. E. Kalman: A New approach to linear filtering and prediction problems. Trans. ASME, J. Basic Engrg. *82* (1960), 35–45. DOI:10.1115/1.3662552

[17] R. Khemchandani, A. Pal, and S. Chandra: Fuzzy least squares twin support vector clustering. Neural Computing Appl. *29* (2018), 553–563. DOI:10.1007/s00521-016-2468-4

[18] I. Lizasoain and M. Gomez: Products of lattice-valued fuzzy transition systems and induced fuzzy transformation semigroups. Fuzzy Sets and Systems *317* (2017), 133–150. DOI:10.1016/j.fss.2017.01.006

[19] F. Liu, R. Zhao, T. Tan, and Q. Zhang: Existence and verification for decentralized nondeterministic discrete-event systems under bisimulation equivalence. Asian J. Control *18* (2016), 5, 1679–1687. DOI:10.1002/asjc.1253

[20] L. Ljung: System Identification: Theory for the User. Prentice Hall PTR, Prentic Hall Inc., Upper Saddle River, New Jersey 1999.

[21] E. Lughofer: Evolving Fuzzy Systems – Methodologies, Advanced Concepts and Applications. Springer, Berlin, Heidelberg 2011.

[22] E. Lughofer: Single-pass active learning with conflict and ignorance. Evolving Systems *3* (2012), 4, 251–271. DOI:10.1007/s12530-012-9060-7

[23] E. Lughofer, E. Weigl, W. Heidl, C. Eitzinger, and T. Radauer: Recognizing input space and target concept drifts in data streams with scarcely labeled and unlabelled instances. Inform. Sci. *355-356* (2016), 127–151. DOI:10.1016/j.ins.2016.03.034

[24] I. Mansouri, A. Gholampour, O. Kisi, and T. Ozbakkaloglu: Evaluation of peak and residual conditions of actively confined concrete using neuro-fuzzy and neural computing techniques. Neural Computing Appl. *29* (2018), 873–888. DOI:10.1007/s00521-016-2492-4

[25] V. K. Nguyen, F. Klawonn, R. Mikolajczyk, and E. A. Hernandez-Vargas: Analysis of practical identifiability of a viral infection model. Plos One (2016), 1–16. DOI:10.1371/journal.pone.0167568

[26] M. Pratama, J. Lu, S. Anavatti, E. Lughofer, and C. P. Lim: An incremental meta-cognitive-based scaffolding fuzzy neural network. Neurocomputing *171* (2016), 89–105. DOI:10.1016/j.neucom.2015.06.022

[27] B. Rehák and S. Čelikovský: Numerical method for the solution of the regulator equation with application to nonlinear tracking. Automatica *44* (2008), 1358–1365. DOI:10.1016/j.automatica.2007.10.015

[28] J. J. Rubio: Least square neural network model of the crude oil blending process. Neural Networks *78* (2016), 88–96. DOI:10.1016/j.neunet.2016.02.006

[29] J. J. Rubio: Stable Kalman filter and neural network for the chaotic systems identification. J. Franklin Inst. *354* (2017), 7444–7462. DOI:10.1016/j.jfranklin.2017.08.038

[30] J. J. Rubio: SOFMLS: Online self-organizing fuzzy modified least square network. IEEE Trans. Fuzzy Systems *17* (2009), 6, 1296–1309. DOI:10.1109/tfuzz.2009.2029569

[31] E. N. Sanchez, A. Y. Alanis, and J. Rico: Electric load demand prediction using neural networks trained by Kalman filtering. In: IEEE International Conference on Neural Networks 2004, pp. 2111–2775. DOI:10.1109/ijcnn.2004.1381093

[32] X. M. Sun, X. F. Wang, Y. Hong, and W. Xia: Stabilization control design with parallel-triggering mechanism. IEEE Trans. Industr. Electron. *64* (2017), 3260–3267. DOI:10.1109/tie.2016.2637888

[33] Z. Weng, G. Chen, L. S. Shieh, and J. Larsson: Evolutionary programming Kalman filter. Inform. Sci. *129* (2000), 197–210. DOI:10.1016/s0020-0255(00)00064-5

[34] H. Wu and Y. Deng: Logical characterizations of simulation and bisimulation for fuzzy transition systems. Fuzzy Sets and Systems *301* (2016), 19–36. DOI:10.1016/j.fss.2015.09.012

[35] H. Wu and Y. Deng: Distribution-based behavioural distance for nondeterministic fuzzy transition systems. IEEE Trans. Fuzzy Systems *99* (2017), 1–1. DOI:10.1109/tfuzz.2017.2670605

[36] H. Wu, Y. Chen, T. Bu, and Y. Deng: Algorithmic and logical characterizations of bisimulations for non-deterministic fuzzy transition systems. Fuzzy Sets and Systems *333* (2017), 106–123. DOI:10.1016/j.fss.2017.02.008

[37] D. Xu, X. Wang, Y. Hong, and Z. P. Jiang: Global robust distributed output consensus of multi-agent nonlinear systems: an internal model approach Systems Control Lett. *87* (2016), 64–69. DOI:10.1016/j.sysconle.2015.11.002

José de Jesús Rubio, Sección de Estudios de Posgrado e Investigación, ESIME Azcapotzalco, Instituto Politécnico Nacional, Av. de las Granjas no.682, Col. Santa Catarina, México D.F., 02250. México.
e-mail: jrubioa@ipn.mx; rubio.josedejesus@gmail.com

Edwin Lughofer, Department of Knowledge-Based Mathematical Systems, Johannes Kepler University Linz. Austria.
e-mail: edwin.lughofer@jku.at

Angelov Plamen, School of Computing and Communications, Lancaster University. United Kingdom.
e-mail: p.angelov@lancaster.ac.uk

Juan Francisco Novoa, Laboratorio de Vibraciones y Rotodinámica, ESIME Zacatenco, Instituto Politécnico Nacional. México.
e-mail: jnovoa@ipn.mx

Jesús A. Meda-Campaña, Laboratorio de Vibraciones y Rotodinámica, ESIME Zacatenco, Instituto Politécnico Nacional. México.
e-mail: jesus.meda@gmail.com