A novel algorithm for the modeling of complex processes

José de Jesús Rubio[1], Edwin Lughofer[2], Angelov Plamen[3]
Juan Francisco Novoa[4], Jesus A. Meda Campaña[4]
1 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
(phone:(+52)55-57296000-64497)
(email: jrubioa@ipn.mx; rubio.josedejesus@gmail.com)
2 Department of Knowledge-Based Mathematical Systems,
Johannes Kepler University Linz, Austria,
(email: edwin.lughofer@jku.at)
3 School of Computing and Comunications
Lancaster University
Lancaster LA1 4WA, U.K.
(email: p.angelov@lancaster.ac.uk)
4 Laboratorio de Vibraciones y Rotodinámica, ESIME Zacatenco
Instituto Politécnico Nacional,
(email: jesus.meda@gmail.com)

Abstract

In this investigation, a novel algorithm is developed in a neural network for the modeling of complex processes. It presents a fuzzy transition between the recursive least square and extended Kalman filter algorithms with the objective to obtain a bounded gain such that a satisfactory modeling could be maintained. The suggested algorithm has the advantage compared with the mentioned methods that it avoids the

excessive increasing or decreasing of its gain. The gain of the proposed algorithm is uniformly stable and its convergence is found. The introduced algorithm is utilized for the modeling of two synthetic examples.

Keywords: Recursive least square, Kalman filter, modeling, complex processes.

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], [17]. The Kalman filter strategy is an algorithm that uses 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], [15].

There is some research about recursive least square algorithms. In [21], the least square and backpropagation are combined. The least square method is addressed in [8]. In [16], fuzzy least squares are suggested. The recursive fuzzily weighted least square is used for updating consequent parameters in evolving fuzzy systems [18] and [27], which is extended to a generalized form in [23]. 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.

There is some research about extended Kalman filter algorithms. In [1], [2], [3], and [28], several Kalman filter algorithms of neural networks are designed. An extended Kalman filter of a wavelet neural network is utilized in [12]. In [7] and [30], the programming with Kalman filters is described. An observer-type of Kalman filtering algorithm is discussed in [11]. In [10], the Kalman filter of nonlinear processes is designed. Single-pass active modeling filters are employed in [19], which is used in [20] for the purpose of 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 novel algorithm is employed for the updating of a neural network. Compared with the mentioned methods, the suggested algorithm is a combination between the recursive least square and extended Kalman filter such as it presents a fuzzy transition between both algorithms with the objective to obtain a bounded gain, maintaining a

satisfactory modeling.

Furthermore, the Lyapunov technique is employed to guarantee the uniform stability and convergence of the gain in the proposed algorithm. Stability is a method to analyze whether the inputs, outputs, and parameters remain bounded through the time [6], [9], [24], [29], [31]. The uniform stability is stronger than the common stability because the first is satisfied for any initial time, while the second is satisfied only for a zero initial time.

Finally, the proposed 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 considered as a well established research area [13], [14], [22].

The paper is organized as follows. The neural network, recursive least square, extended Kalman filter, and proposed algorithms are detailed in Section 2. The proposed technique is summarized in Section 3. The suggested method is applied for the modeling of two synthetic examples in Section 4. Conclusions and future research are detailed in Section 5.

2 Updating algorithms of a neural network

In this part of the article: 1) the neural network will be explained, 2) the recursive least square, extended Kalman filter, and proposed algorithms will be detailed for the updating of a neural network, and 3) the stability and convergence of the gain in the proposed algorithm will be analyzed.

2.1 The neural network

Take in account the following unknown complex process:

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

with

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

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

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 focused in a compact neural network.

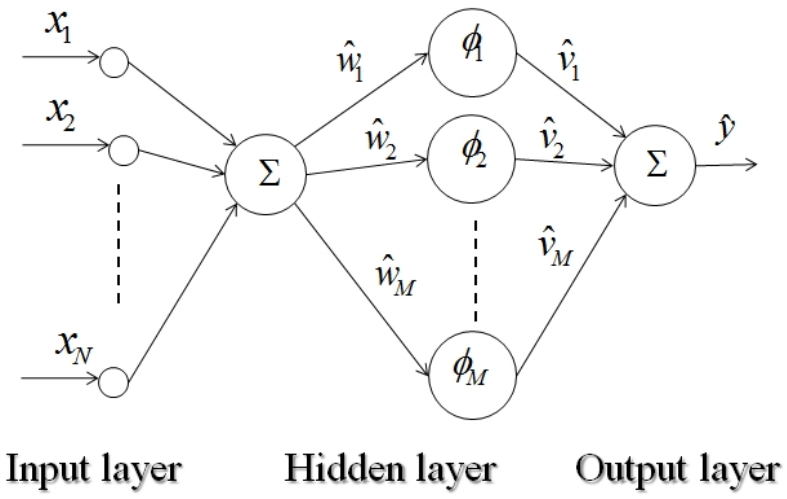

Figure 1: The neural network structure

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

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

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

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

The modeling error $e(k) \in \Re$ is described as follows:

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

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

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

2.2 The recursive least square algorithm

The recursive least square algorithm is described 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 recursive least square algorithm utilized as the adapting law of the neural network (2) for the modeling of the complex process (1) is represented as follows [25]:

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

with

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

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

$0 < r_2 \in \Re$ is a forgetting factor, $e(k)$ is the modeling error of (3), $\phi_j(k)$ and $\sigma_j(k)$ are described in (2). $G_{k+1} \in \Re^{2M \times 2M}$ 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 assure 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 described 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 extended Kalman filter algorithm utilized as the adapting law of the neural network (2) for the modeling of the complex process (1) is represented as follows [26]:

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

with

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

$$r(k) = q(k) + a^T(k)G_k a(k)$$
$$q(k) = r_2 + a^T(k)G_k a(k) \in \Re$$

$0 < r_2 \in \Re$ is a forgetting factor, $e(k)$ is the modeling error of (3), $\phi_j(k)$ and $\sigma_j(k)$ are described in (2). $G_{k+1} \in \Re^{2M \times 2M}$ 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 assure an acceptable convergence, and $I \in \Re^{2M \times 2M}$ is the identity matrix. $R_1 = r_1 I$, $0 < r_1 \in \Re$.

2.4 The proposed algorithm

The proposed algorithm is described in this subsection for the updating of a neural network. It presents a fuzzy transition between the recursive least square and extended Kalman filter algorithms with the objective to obtain a bounded gain G_k, maintaining a satisfactory modeling. Figure 2 shows the proposed algorithm.

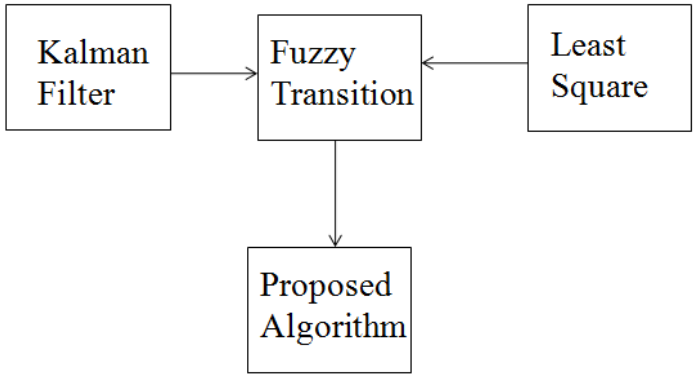

Figure 2: The proposed algorithm

The proposed algorithm utilized as the adapting law of the neural network (2) for the modeling of the complex process (1) is represented as follows:

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

with

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

$$r(k) = q(k) + a^T(k) G_k a(k)$$
$$q(k) = r_2 + a^T(k) G_k a(k) \in \Re$$

$0 < r_2 \in \Re$ is a forgetting factor, $e(k)$ is the modeling error of (3), $\phi_j(k)$ and $\sigma_j(k)$ are described in (2). $G_{k+1} \in \Re^{2M \times 2M}$ 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 assure an acceptable convergence, and $I \in \Re^{2M \times 2M}$ is the identity matrix. 5 rules are used to obtain R_1 as a fuzzy transition between the recursive least square $R_1 = 0I$ and the extended Kalman filter $R_1 = r_1 I$ as follows:

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

with $0 < r_1 \in \Re$.

The stability and convergence of the gain in the introduced algorithm are analyzed by the following Theorem.

Theorem 1 *The gain of the proposed algorithm (6) (7) for the updating of the neural network (2) (3) is uniformly stable and the following 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 proposed algorithm described in (6), $a(k)$ and $r(k)$ are described in (6), and r_1 is described in (7).

Proof. Select the following Lyapunov function:

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

obtaining ΔL_k as follows:

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

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

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

The following result is obtained:

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

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 proposed algorithm is uniformly stable. b) when $\frac{1}{4} g_1 < \|G_k\| \leq \frac{1}{2} g_1$, considering (6) and that $b^T b = 1$ gives:

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

The following result is obtained:

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

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 proposed algorithm is uniformly stable. c) when $\frac{1}{2} g_1 < \|G_k\| \leq \frac{3}{4} g_1$, considering (6) and that $b^T b = 1$ gives:

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

The following result is obtained:

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

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 proposed algorithm is uniformly stable. d) when $\frac{3}{4} g_1 < \|G_k\| \leq g_1$, considering (6) and that $b^T b = 1$ gives:

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

The following result is obtained:

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

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 proposed algorithm is uniformly stable. e) when $g_1 < \|G_k\|$, considering (6) and that $b^T b = 1$ gives:

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

The following result is obtained:

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

since $b^T \frac{1}{r(k)} G_k a(k) a^T(k) G_k b \geq 0$, the gain of the proposed algorithm is asymptotically stable. Since the uniform stability is weaker than the asymptotic stability [6], [9], [24], [29], [31], considering all cases, the gain of the proposed algorithm is uniformly stable. Now considering 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 obtaining the limit gives:

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

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 obtaining the limit gives:

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

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 obtaining the limit gives:

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

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 obtaining the limit gives:

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

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 obtaining the limit gives:

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

Since the limit (22) is the weakest of all, considering all cases, the gain of the convergence (22) is (8), it establishes the result. ■

Remark 1 *Even if the neural network of this research is equal with the neural network of [25], there are three differences between the proposed research and the investigation of [25]: 1) in the investigation of [25] 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; 2) in the investigation of [25] the stability and convergence of the modeling error is assured, while in this research the stability and convergence of the gain in the proposed algorithm is assured; and 3) the algorithm of [25] 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.*

3 Detailed steps of the proposed algorithm

The steps of the proposed algorithm are detailed as follows:

1) The complex process output $y(k)$ is obtained 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) Consider the elements as follows; chose weights $\widehat{v}_j(1)$ and $\widehat{w}_j(1)$ 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 obtained with equation (2), the modeling error $e(k)$ is obtained with equation (3), $\widehat{\psi}(k)$ is obtained with weights $\widehat{v}_j(k)$ and $\widehat{w}_j(k)$ utilizing (6), (7), $a^T(k)$ is obtained 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 $\widehat{v}_j(k+1)$ and $\widehat{w}_j(k+1)$ with $\widehat{\psi}(k+1)$ are updated utilizing (6), (7).

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

4 Examples

In this part of the article, the proposed algorithm is applied for the modeling of two synthetic examples. In all cases, the proposed algorithm called Proposed will be compared with the least square algorithm of [25] called Least Square, and with the extended Kalman filter algorithm of [26] called Kalman Filter. The differences between three algorithms were described in past sections. The root mean square error denoted as RMSE is employed for comparisons and is expressed as follows:

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

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

4.1 Example 1

The complex process of the example 1 is detailed as follows [27]:

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

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 used for the training and the data of the least 200 iterations is used for the testing.

The Least Square of [25] is detailed 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 $\widehat{w}_j(1)$ employ random numbers between 0 and 0.5.

The Kalman Filter [26] is detailed 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 $\widehat{w}_j(1)$ employ random numbers between 0 and 0.5.

The Proposed algorithm is detailed 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 $\widehat{w}_j(1)$ 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 Least Square, Kalman Filter, and Proposed. The training and testing RMSE comparisons of (31) are shown in Table 1.

Table 1: Comparisons for the example 1

Strategies	**Training RMSE**	**Testing RMSE**
Least Square	0.0693	0.0153
Kalman Filter	0.0547	0.0367
Proposed	0.0518	0.0153

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 Proposed improves both the Least Square and Kalman Filter because the signal of the first follows better the signal of the plant than the signal of the other. From Table 1, it is observed that the Proposed achieves better accuracy when compared with both the Least Square and Kalman Filter because the RMSE is smaller for the first. Thus, the Proposed is the best option for the modeling in the example 1.

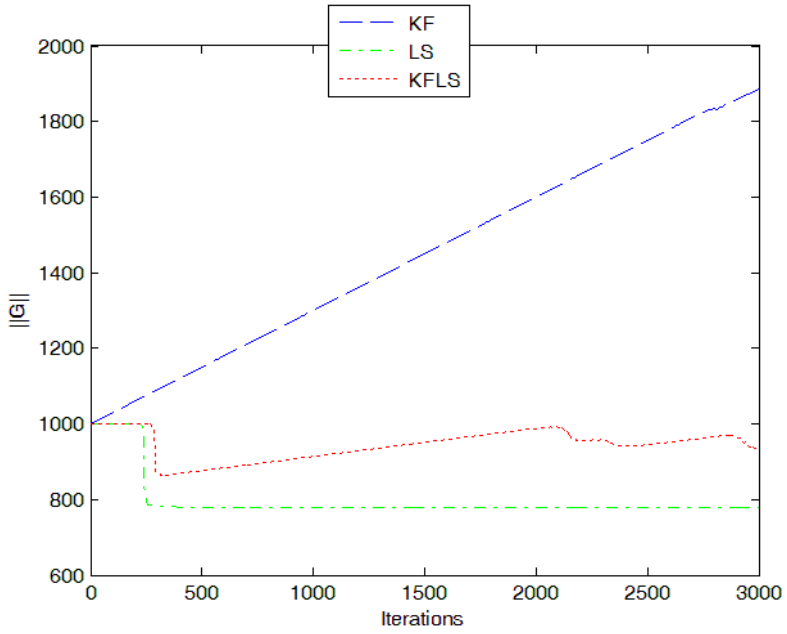

Figure 3: Norm of the gain for the example 1

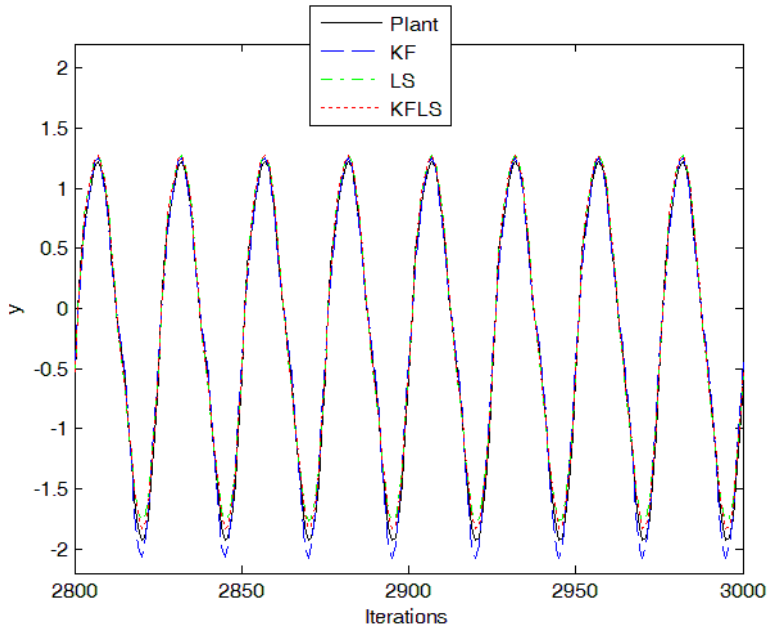

Figure 4: Training for the example 1

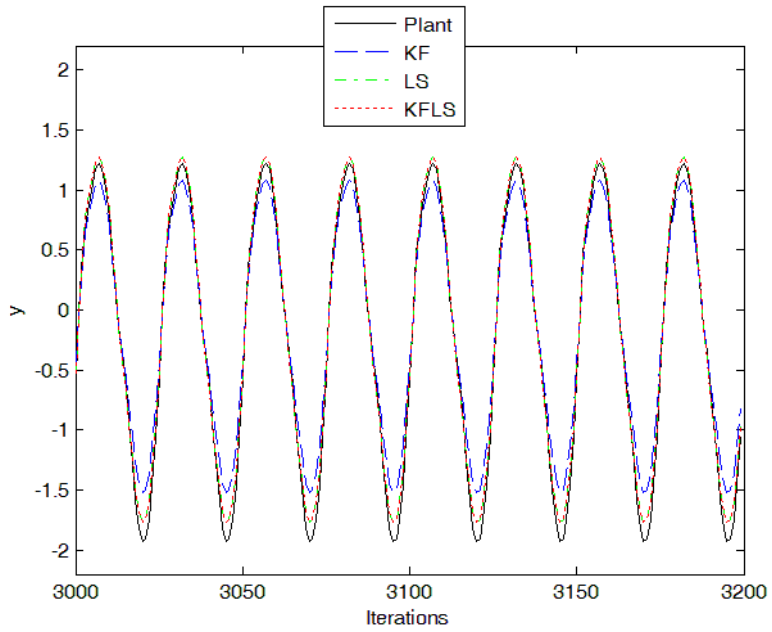

Figure 5: Testing for the example 1

4.2 Example 2

The complex process of the example 2 is detailed as follows [27]:

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

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 used for the training and the data of the least 200 iterations is used for the testing.

The Least Square of [25] is detailed 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 $\widehat{w}_j(1)$ employ random numbers between 0 and 1.

The Kalman Filter of [26] is detailed 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 $\widehat{w}_j(1)$ employ random numbers between 0 and 1.

The Proposed algorithm is detailed 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 $\widehat{w}_j(1)$ 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 Least Square, Kalman Filter, and Proposed. The training and testing RMSE comparisons of (31) are shown in Table 2.

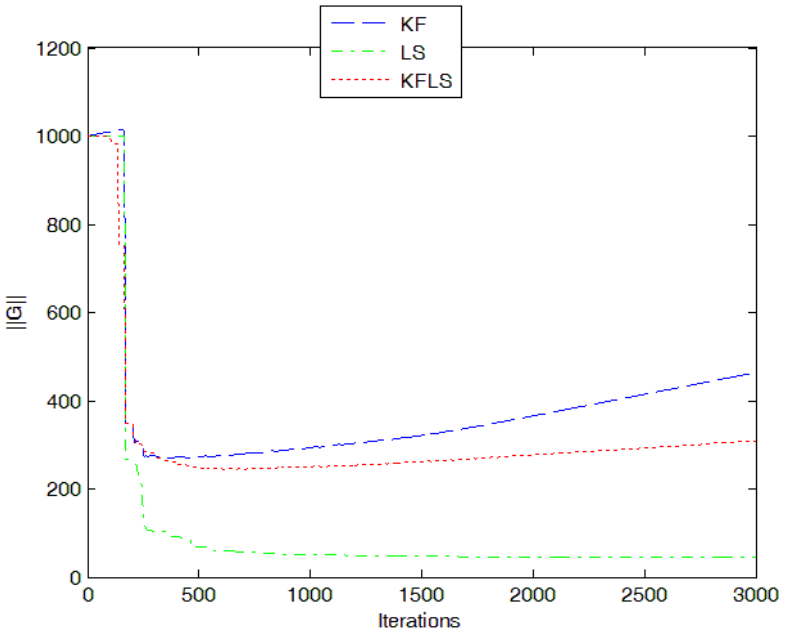

Figure 6: Norm of the gain for the example 2

Table 2: Comparisons for the example 2		
Strategies	**Training RMSE**	**Testing RMSE**
Least Square	0.1892	0.0320
Kalman Filter	0.1125	0.0273
Proposed	0.1087	0.0263

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 Proposed algorithm improves both the Least Square and Kalman Filter because the signal of the first follows better the signal of the plant than the signal of the other. From Table 2, it is observed that the Proposed achieves better accuracy when compared with both the Least Square and Kalman Filter because the RMSE

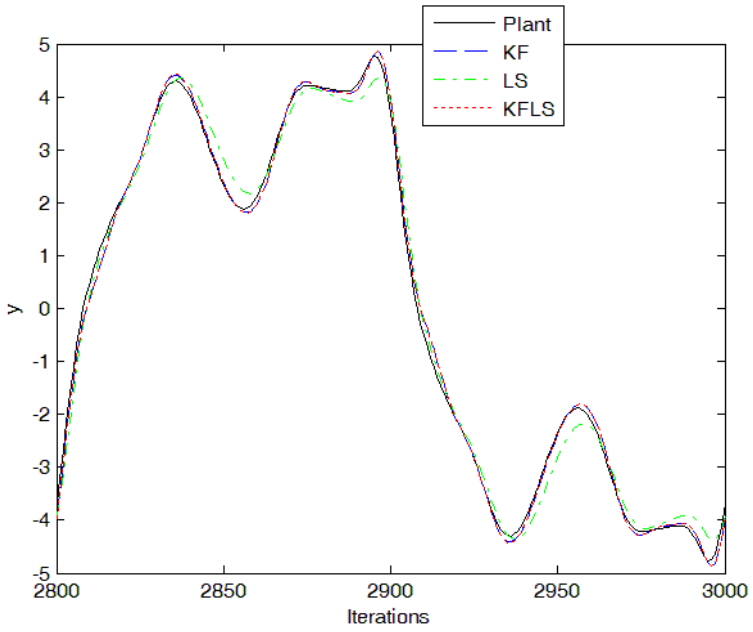

Figure 7: Training for the example 2

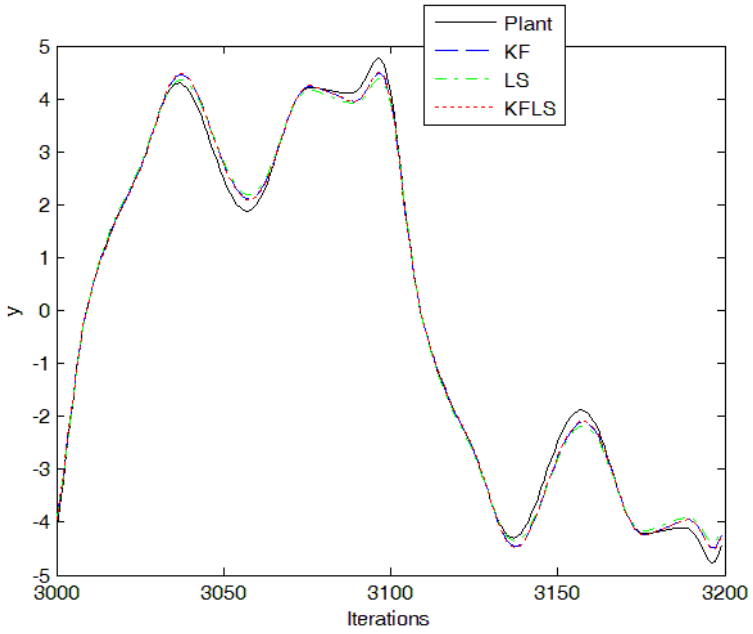

Figure 8: Testing for the example 2

is smaller for the first. Thus,the Proposed is the best option for the modeling in the example 2.

5 Conclusion

In this research, a novel algorithm was introduced for the updating of a neural network. The stability and convergence of the gain in the proposed algorithm were assured by the Lyapunov technique. From examples, it was seen that the proposed 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 proposed algorithm could be used to train a fuzzy system, or it could be used as the updating of an evolving intelligent system. In the future research, the proposed method will be used for the control, pattern recognition, prediction, or classification.

6 Acknowledgements

Authors are grateful to the editors and the reviewers for their valuable comments. Authors thank the Secretaría de Investigación y Posgrado, the Comisión de Operación y Fomento de Actividades Académicas del Instituto Politécnico Nacional, 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.

References

[1] A. Y. Alanis, L. J. Ricalde, C. Simetti, F. Odone, Neural model with particle swarm optimization Kalman learning for forecasting in smart grids, Mathematical Problems in Engineering, 2013 (2013) 1-9.

[2] A. Y. Alanis, E. N. Sanchez, A. G. Loukianov, Discrete-time recurrent neural induction motor control using Kalman learning, International Joint Conference on Neural Networks, (2006) 1993-2000.

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

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

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

[6] S. Celikovsky, Topological equivalence and topological linearization of controlled dynamical systems, Kybernetika, 31 (1995) 141-150.

[7] G. Chen, Q. Xie, L. S. Shieh, Fuzzy Kalman filtering, Journal of Information Sciences, 109 (1998) 197-209.

[8] J. K. Coelho, M. D. Pena, O. J. Romero, Pore-scale modeling of oil mobilization trapped in a square cavity, IEEE Latin America Transactions, 14 (4) (2016) 1800-1807.

[9] Z. Deng, X. Wang, Y. Hong, Distributed optimisation design with triggers for disturbed continuous-time multi-agent systems, IET Control Theory and Applications, 11 (2) (2017) 282-290.

[10] K. Dolinsky, S. Celikovsky, Kalman filter under nonlinear system transformations, American Control Conference, (2012) 4789-4794.

[11] S. M. Guo, L. S. Shieh, G. Chen, N. P. Coleman, Observer-type Kalman innovation filter for uncertain linear systems, IEEE Transactions on Aerospace and Eelectronic Systems, 37 (4) (2001) 1406-1418.

[12] J. E.Guillermo, L. J. Ricalde Castellanos, E. N. Sanchez, A. Y. Alanis, Detection of heart murmurs based on radial wavelet neural network with Kalman learning, Neurocomputing, 164 (2015) 307-317.

[13] E. A. Hernandez-Vargas, P. Colaneri, R. H. Middleton, Switching Strategies to Mitigate HIV Mutation, IEEE Transactions on Control Systems Technology, 22 (4) (2014) 1623-1628.

[14] E. A. Hernandez-Vargas, P. Colaneri, R. H. Middleton, Optimal therapy scheduling for a simplified HIV infection model, Automatica, 49 (2013) 2874-2880.

[15] R. E. Kalman, A New Approach to Linear Filtering and Prediction Problems, Transaction of the ASME, Journal of Basic Engineering, 82 (1960) 35-45.

[16] R. Khemchandani, A. Pal, S. Chandra, Fuzzy least squares twin support vector clustering, Neural Computing and Applications, (2016). DOI: 10.1007/s00521-016-2468-4

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

[18] E. Lughofer, Evolving Fuzzy Systems - Methodologies, Advanced Concepts and Applications, (2011), Springer, Berlin Heidelberg.

[19] E. Lughofer, Single-Pass Active Learning with Conflict and Ignorance, Evolving Systems, 3 (4) (2012) 251-271.

[20] E. Lughofer, E. Weigl, W. Heidl, C. Eitzinger, T. Radauer, Recognizing input space and target concept drifts in data streams with scarcely labeled and unlabelled instances, Information Sciences, 355-356 (2016) 127-151.

[21] I. Mansouri, A. Gholampour, O. Kisi, T. Ozbakkaloglu, Evaluation of peak and residual conditions of actively confined concrete using neuro-fuzzy and neural computing techniques, Neural Computing and Applications, (2016). DOI: 10.1007/s00521-016-2492-4

[22] V. K. Nguyen, F. Klawonn, R. Mikolajczyk, E. A. Hernandez-Vargas, Analysis of practical identifiability of a viral infection model, Plos one, (2016) 1-16.

[23] M. Pratama, J. Lu, S. Anavatti, E. Lughofer, C. P. Lim, An incremental meta-cognitive-based scaffolding fuzzy neural network, Neurocomputing, 171 (2016) 89-105.

[24] B. Rehaka, S. Celikovsky, Numerical method for the solution of the regulator equation with application to nonlinear tracking, Automatica, 44 (2008) 1358-1365.

[25] J. J. Rubio, Least square neural network model of the crude oil blending process, Neural Networks, 78 (2016) 88-96.

[26] J. J. Rubio, W. Yu, Nonlinear system identification with recurrent neural networks and dead-zone Kalman filter algorithm, Neurocomputing, 70 (13) (2007) 2460-2466.

[27] J. J. Rubio, SOFMLS: Online self-organizing fuzzy modified least square network, IEEE Transactions on Fuzzy Systems, 17 (6) (2009) 1296-1309.

[28] E. N. Sanchez, A. Y. Alanis, J. Rico, Electric load demand prediction using neural networks trained by Kalman filtering, IEEE International Conference on Neural Networks, (2004) 2111- 2775.

[29] X. M. Sun, X. F. Wang, Y. Hong, W. Xia, Stabilization control design with parallel-triggering mechanism, IEEE Transactions on Industrial Electronics, (2016). DOI: 10.1109/TIE.2016.2637888

[30] Z. Weng, G. Chen, L. S. Shieh, J. Larsson, Evolutionary programming Kalman filter, Information Sciences, 129 (2000) 197-210.

[31] D. Xu, X. Wang, Y. Hong, Z. P. Jiang, Global robust distributed output consensus of multi-agent nonlinear systems: an internal model approach, Systems & Control Letters, 87 (2016) 64-69.