Neurocomputing 74 (2011) 2062–2071

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Unsupervised kernel least mean square algorithm for solving ordinary differential equations

Hadi Sadoghi Yazdi [a,*], Morteza Pakdaman [b], Hamed Modaghegh [a]

[a] Engineering Department, Ferdowsi University of Mashhad, Mashhad, Iran
[b] Sama Technical and Vocational Training College, Islamic Azad University, Mashhad Branch, Mashhad, Iran

ARTICLE INFO

Article history:
Received 17 February 2010
Received in revised form
12 December 2010
Accepted 17 December 2010
Communicated by N.T. Nguyen
Available online 7 April 2011

Keywords:
Unsupervised learning
Kernel least mean square
Ordinary differential equation
Neuro-fuzzy approach

ABSTRACT

In this paper a novel method is introduced based on the use of an unsupervised version of kernel least mean square (KLMS) algorithm for solving ordinary differential equations (ODEs). The algorithm is unsupervised because here no desired signal needs to be determined by user and the output of the model is generated by iterating the algorithm progressively. However, there are several new approaches in literature to solve ODEs but the new approach has more advantages such as simple implementation, fast convergence and also little error. Furthermore, it is also a KLMS with obvious characteristics. In this paper the ability of KLMS is used to estimate the answer of ODE. First a trial solution of ODE is written as a sum of two parts, the first part satisfies the initial condition and the second part is trained using the KLMS algorithm so as the trial solution solves the ODE. The accuracy of the method is illustrated by solving several problems. Also the sensitivity of the convergence is analyzed by changing the step size parameters and kernel functions. Finally, the proposed method is compared with neuro-fuzzy [21] approach.

1. Introduction

Differential equations are basic structure in representation of engineering problems. Many problems in science and engineering can be reduced to a set of differential equations (DEs) through a process of mathematical modeling. In most cases it is not easy to obtain the exact solutions of DEs, so numerical methods must be applied. There are a lot of mathematical methods to solve DEs, such as the finite difference method (FDM) and the finite element method (FEM). These methods require the definition of a mesh (domain discretization) where the functions are approximated locally. The construction of a mesh in two or more dimensions is not a simple problem. Usually, in practice, only low-order approximations are employed resulting in a continuous approximation of the function across the mesh but not its partial derivatives. The discontinuity of the approximation of the derivative can adversely affect the stability of the solution. While higher-order schemes are necessary for more accurate approximations of the spatial derivatives, they usually involve additional computational cost. To increase the accuracy of the low-order schemes, it is required that the computational mesh refined with a higher density of elements in the regions near the contours. This, however, is also achieved at the expense of increased computational cost, thus although these methods have been highly successful, they require time-consuming procedures to build numerous volumetric elements and solve large-size linear systems of equations (see [19]).

In recent years, researches tried to find new methods. The main new proposed method was based on the use of neural network models. Neural network methods in comparison with other numerical methods have more advantages. Most other techniques offer a discrete solution (for example predictor–corrector, or Runge–Kutta methods) or a solution of limited differentiability (for example finite elements) while the solution of DE in neural network methods are differentiable and continuous. The methods which are based on the use of neural network can be realized in hardware and hence offers the opportunity to tackle in real time difficult differential equation problems arising in many engineering applications (e. g. see [10,17,22]).

The least mean squares (LMS) algorithm which is used in adaptive supervised learning immensely (from system identification to channel equalization) was introduced by Widrow and Hoff in 1959 [1]. The LMS algostochastic gradient instead of the deterministic gradient used in the method of steepest descent. On the other hand in recent years, kernel methods are applied successfully in classification, regression problems and generally machine learning (support vector machines (SVM) [2], regularization networks [3], kernel principal component analysis (K-PCA) [4], kernel independent component analysis (K-ICA) [5]). Kernel methods have been used to

* Corresponding author.
E-mail addresses: h-sadoghi@um.ac.ir (H. Sadoghi Yazdi), pakdaman.m@gmail.com (M. Pakdaman).

0925-2312/$ - see front matter Crown Copyright © 2011 Published by Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2010.12.026

extend linear adaptive filters expressed in inner products to nonlinear algorithms [6,2]. Pokharel et al. [6] and Liu et al. [7] have applied this "kernel trick" [2] to the least mean square (LMS) algorithm [8,9] to attain a nonlinear adaptive filter in reproducing kernel Hilbert spaces (RKHS), which they have coined kernel least mean square (KLMS). Modaghegh et al. [29] introduced normalized kernel least mean square (NKLMS) algorithm which has applications in system modeling and pattern recognition. Their method improved the convergence speed of KLMS and system tracking. Furthermore, they applied the proposed algorithm to channel modeling. Ghafarian and Sadoghi Yazdi [30] used kernel least mean square features for HMM-based signal recognition. Kivinen et al. [32] considered online learning in a reproducing kernel Hilbert space. By considering classical stochastic gradient descent within a feature space and the use of some straightforward tricks, they developed simple and computationally efficient algorithms for a wide range of problems such as classification, regression, and novelty detection. There are some other applications of KLMS (e.g. [33,34]). Yang and Cui [31] introduced a new algorithm for giving the approximate solution of a class of nonlinear integro-differential equations in the reproducing kernel space. Considering KLMS algorithm as a learning approach guides us to use it similar to neural networks and neuro-fuzzy systems however, with kernel capacities. In the previous work [21] we used an unsupervised version of adaptive neuro-fuzzy inference system (ANFIS) for solving differential equations. In this paper we propose a new unsupervised algorithm to solve DEs by the use of KLMS algorithm.

1.1. Related works in neural networks

In recent years, several approaches are proposed to solve ordinary differential equations (ODEs) as well as partial differential equations (PDEs). We consider a general differential equation which is given by (1)

$$G\left(x,y(x),\nabla y(x),\nabla^2 y(x),\ldots\right)=0,\quad x\in\overline{D}\subseteq R^n, \tag{1}$$

where $y(x)$ denotes the solution, G is the function that defines the structure of the differential equation, ∇ is some differential operator, and $\overline{D}$ is the problem domain. The basic idea, called collocation method, is to discretize the domain $\overline{D}$ over a finite set of points D. Thus (1) becomes a system of equations. An approximation of the solution $y(x)$ is given by the trial solution $y_t(x)$. As a measure for the degree of fulfillment of the original differential equation (1) an error function similar to the mean squared error is defined:

$$E=\frac{1}{|D|}\sum_{x_i\in D}\left[G(x_i,y_t,\nabla y_t,\nabla^2 y_t,\ldots)\right]^2 \tag{2}$$

Therefore, finding an approximation of the solution of (1) is equal to finding a function which minimizes the error E. Since multilayer feed forward neural networks are universal approximators [11] the trial solution $y_t(x)$ can be represented by such an artificial neural network as (3) (see [11])

$$y_t(x)=A(x)+F(x,N(x,p)) \tag{3}$$

where $A(x)$ contains no adjustable parameters and satisfies the boundary conditions and $F(x,N(x,p))$ is a single output feed forward neural network with input vector x and adjustable parameters p. Thus the problem is reduced to finding a configuration of weights, corresponding to the neural network architecture, that minimizes (2). As E is differentiable with respect to the weights for most differential equations, efficient, gradient-based learning algorithms for artificial neural networks can be employed to minimize (2). In recent years, neural network based approaches are presented to solve differential equation as follows.

Lagaris et al. [10] used neural networks to solve ODEs and PDEs. They used multilayer perceptron in their network architecture. Shekari Beidokhti and Malek [22] proposed a hybrid method based on artificial neural networks, minimization techniques and collocation methods to solve a general system of time dependent partial differential equations. They also applied the method to solve high order differential equations [17]. Another hybrid method which creates trial solutions in neural network form using a scheme based on grammatical evolution was proposed by Tsoulos [23]. Multilayer perceptron and radial basis function (RBF) neural networks with a new unsupervised training method proposed in [24] for numerical solution of partial differential equations. Smaoui and Al-Enezi analyzed dynamics of two nonlinear partial differential equations known as the Kuramoto–Sivashinsky (K–S) equation and the two-dimensional Navier–Stokes (N–S) equations using Karhunen–Loeve (K–L) decomposition and artificial neural networks [12]. In [13], Brause used differential equations for modeling of biochemical pathways and these equations were solved using neural networks. In [14], Hea et al. used feed forward neural network with the extended back propagation algorithm to solve a class of first-order partial differential equations for input-to-state linearizable or approximate linearizable systems. The use of neural networks not only limited to solve ODEs but also extended to solve fuzzy differential equations. The authors of the current paper, applied a mulilayer perceptron to solve fuzzy differential equations in [25]. Also Manevitz et al. [15] presented basic learning algorithms and the neural network model to the problem of mesh adaptation for the finite-element method to solve time-dependent partial differential equations. Time-series prediction via the neural network methodology was used to predict the areas of "interest" in order to obtain an effective mesh refinement at the appropriate times. Leephakpreeda [16] presented fuzzy linguistic model in neural network to solve differential equations and applied it as universal approximators for any nonlinear continuous functions. In another work, Mai-Duy and Tran-Cong [18] presented mesh-free procedures to solve linear differential equations, ordinary differential equations and elliptic partial differential equations based on multi-quadric radial basis function networks. Also Jianyu et al. [19] described a neural network to solve partial differential equations which the activation functions of the hidden nodes were the radial basis functions (RBF) whose parameters were learnt by a two-stage gradient descent strategy (see [28]). Also solving differential equation using neural network was applied to real problems such as a non-steady fixed bed non-catalytic solid/gas reactor [20].

The new proposed method against the past that are based on neural networks and also our previous work [21] which was based on neuro-fuzzy algorithm, uses unsupervised KLMS (UKLMS) to solve differential equations. In the end of this paper, we compare the results of this paper with the results in [21].

1.2. Motivation

KLMS is a new algorithm, which was proposed in 2007 with high ability in learning area, but for solving DE we need an unsupervised version of KLMS which we present it in this work. Also KLMS can be described as an ideal ANFIS (which it is not possible to implement practically), furthermore, the use of unsupervised KLMS (UKLMS) is a new concept in solving DEs such that, it can be implemented easily with a fast convergence. Some of the main advantages of the new proposed method are as follows:

- Simple implementation.
- As it can be observed in Section 3 the method has a fat convergence.

- The algorithm is unsupervised because no desired signals need to be determined by user.
- In comparison with mathematical methods, the new method has the advantages that the solution of DE is differentiable and continuous such that we can calculate the answer at every point in the training interval.
- In comparison with the neural network based methods, which need an optimization step to improve the corresponding weights of the network, the new method does not need an optimization step and the parameters of the KLMS algorithm are improved via a recursive scheme.

The paper is organized as follows. The basic requirements of KLMS and the formulation of the new method are presented in Section 2. Experimental results and error analysis are discussed in Section 3 and finally Section 4 contains concluding remarks.

2. Unsupervised kernel least mean square (UKLMS) in DE solving

In this section, first we introduce the LMS and KLMS algorithms, and then describe the new proposed UKLMS algorithm.

2.1. The KLMS algorithm

In 1959, the LMS algorithm was introduced as a simple way of training a linear adaptive system with mean square error minimization. An unknown system – $K(n)$ – is to be identified and the LMS algorithm attempts to adapt the filter $\hat{K}(n)$ to make it as close as possible to $K(n)$. The algorithm uses $x(n)$ as the input, $d(n)$ as desired output and $e(n)$ as calculated error.

LMS uses steepest-descent algorithm to update the weight vector so that the weight vector converges to optimum Wiener solution. Updating weight vector is applied based on the following rule:

$$w(n+1)=w(n)-\mu' \times \nabla_w e^2(n) \tag{4a}$$

where $w(n)$ is the weight vector, μ' is the step size, $x(n)$ is the input vector and $e(n)=d(n)-K(n)$, thus considering the filter output K by $K(n)=w\times x(n)$ then (4a) can be rewritten as

$$w(n+1)=w(n)+2\mu' \times e(n) \times x(n) \tag{4b}$$

Successive corrections of the weight vector eventually leads to the minimum value of the mean squared error. You can find further information about LMS in [1]. On the other hand, Kernel methods are applied to map the input data into a high dimensional space (HDS). In HDS a variety of methods can be used to find linear relations in the data. Mapping procedure is handled by Φ functions (Fig. 1).

Kernel functions help the algorithm to handle the converted input data in the HDS ever without knowing the coordinates of data in that space; simply by computing the kernel of input data instead of calculating the inner products between images of all pairs of data in HDS. This method is called the kernel trick.

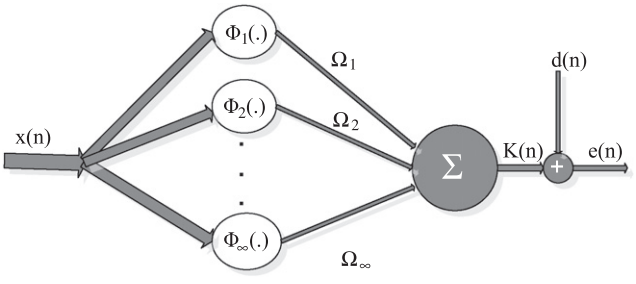

Fig. 1. Kernel estimation system.

As presented in [7] estimation and prediction of some time-series could be optimized with a new approach, named KLMS. The basic idea is to perform the linear LMS algorithm in the kernel space

$$\Omega(n+1)=\Omega(n)+\mu \times e(n) \times \varphi(x(n)) \tag{5}$$

where $\Omega(n)$ is weight vector in the HDS and for simplicity $\mu=2\mu'$. The estimated output $K(n)$ will be calculated by

$$K(n)=\langle \Omega(n),\varphi(x(n))\rangle \tag{6}$$

Fig. 1 shows the input vector $x(n)$ being transformed to the infinite feature vector $\Phi(x(n))$, whose components are then linearly combined by the infinite dimensional weight vector. Non-recursive type of Eq. (5) can be written as

$$\Omega(n)=\Omega(0)+\mu\sum_{i=0}^{n-1} e(i)\Phi(x(i)) \tag{7a}$$

By choosing $\Omega(0)=0$

$$\Omega(n)=\mu\sum_{i=0}^{n-1} e(i)\Phi(x(i)) \tag{7b}$$

Based on Eqs. (6) and (7b):

$$\begin{aligned} K(n)&=\langle \Omega(n)\times\Phi(x(n))\rangle \\ &=\langle \mu\sum_{i=0}^{n-1} e(i)\Phi(x(i)),\Phi(x(n))\rangle \\ &=\mu\sum_{i=0}^{n-1} e(i)\langle \Phi(x(i)),\Phi(x(n))\rangle \end{aligned} \tag{8}$$

We can use kernel trick here to calculate $K(n)$. Kernel functions help the algorithm to handle the converted input data in the HDS ever without knowing the coordinates of data in that space; simply by computing the kernel of input data instead of calculating the inner products between images of all pairs of data in HDS. This method is called the kernel trick

$$K(n)=\mu\sum_{i=0}^{n-1} e(i)\ker(x(i),x(n)) \tag{9}$$

and (9) is the main equation in Kernel LMS algorithm. As error of system reduces by time, we can ignore the $e(n)$ after ξ sample and predict new data with previous error:

$$K(n)=\mu\sum_{i=0}^{\xi} e(i)\ker(x(i),x(n)) \tag{10}$$

This change decreases the complexity of algorithm. After that, we can train the model of system with fewer numbers of data and use it to predict new data. Now if we rewrite (10) with finite training data then we will have

$$K(x,P)=K(x,(E_{1\times\zeta},X_{1\times\zeta}))=\mu\sum_{i=0}^{\xi} e(i)\ker(x(i),x(n)) \tag{10a}$$

where P is the parameter of KLMS model that includes $(E_{1\times\zeta},X_{1\times\zeta})$ and x is the new input data. E is error vector which can be obtained after training the model with ξ input data, and X is the vector of input learning. By using (10a) we can calculate the output of KLMS for each input data x.

2.2. Proposed UKLMS algorithm for solving DEs

A linear differential equation (DE) with constant coefficients can be expressed in the following form:

$$a_n \frac{d^n y(x)}{dx^n} + a_{n-1} \frac{d^{n-1} y(x)}{dx^{n-1}} + \cdots + a_0 y(x) = v_o(x), \quad x \in [a,b] \tag{11}$$

where $a_n,\ldots,a_0$ are constant coefficients and $[a,b]$ is the problem domain. The $n-1$ necessary initial or boundary conditions for solving above DE are

$$y(0) = y_0^0, \quad y^{(1)}(0) = y_0^{(1)},\ldots,y^{(n-1)}(0) = y_0^{(n-1)}$$

or

$$y(x_0) = y_{x_0}, \quad y(x_1) = y_{x_1},\ldots,y(x_n) = y_{x_n} \tag{12}$$

As have been pointed in (3), a trial solution (3) is like (13)

$$\begin{aligned} y_p(x) &= f(x,y_0^{(0)},y_0^{(1)},\ldots,y_0^{(n-1)}) + g(h,K(x,P)) \\ &= f(x,C) + g(h,K(x,P)) \end{aligned} \tag{13}$$

where $f(x,C)$ is a function for satisfaction of initial/boundary conditions, C describes the initial conditions and $g(h,K(x,P))$ is a function which is zero in initial points and is $K(x,P)$ in other points. $K(x,P)$ is the out put of KLMS algorithm and plays a high important role, principally $K(x,P)$ is the main answer without contemplate of initial points. h is used to suppress the $g(h,K(x,P))$ term in initial/boundary points. Hence an easy and suitable form of $g(h,K(x,P))$ is $h \times K(x,P)$. $f(x,C)$ and h take different forms depending on the initial/boundary conditions and the order of differential equation and there is no clear procedure to choose the most appropriate ones. The selection of $f(x,C)$ and h for several types of DEs is explained in [17]; however, we repeat some ordinary differential equations here to make it easy for the readers to follow the procedure of solving DE using KLMS algorithm.

Consider the first order DE in (14)

$$\frac{dy(x)}{dx} = v_o(y,x), \quad x \in [0,1], \quad y(0) = A \tag{14}$$

hence

$$f(x,C) = A, \quad h = x \Rightarrow y_p(x) = A + xK(x,P) \tag{15}$$

It is easy to check that $y_p(x)$, which is the trial solution of (14), satisfies the initial condition.

Now consider the following second order DE:

$$\frac{d^2 y(x)}{dx^2} = v_o\left(\frac{dy}{dx},y,x\right), \quad x \in [0,1] \tag{16}$$

The trial solution of this DE is written in two cases. In the first case these initial conditions are considered as: $y(0)=A$ and $(dy/dx)(0)=A'$. Thus

$$f(x,C) = A + A'x, \quad h = x^2 \Rightarrow y_p(x) = A + A'x + x^2 K(x,P) \tag{17}$$

In the second case these boundary conditions are considered as: $y(0)=A$ and $y(1)=B$. Therefore

$$f(x,C) = A(1-x) + Bx, h = x(1-x) \Rightarrow y_p(x) = A(1-x) + Bx + x(1-x)K(x,P) \tag{18}$$

The same procedure can be performed to find the trial solution of higher order ordinary differential equations.

Pursuing the procedure of solving DE, (13) is substituted in (11) and we can write:

$$a_n \frac{d^n y_p(x)}{dx^n} + a_{n-1} \frac{d^{n-1} y_p(x)}{dx^{n-1}} + \cdots + a_0 y_p(x) = v_o(x) \tag{19}$$

Then putting $y_p(x)=f(x,C)+h \times K(x,P)$ (19) becomes

$$\hat{f}(x,C) + b_n \frac{d^n K(x,P)}{dx^n} + b_{n-1} \frac{d^{n-1} K(x,P)}{dx^{n-1}} + \cdots + b_0 K(x,P) = v_o(x) \tag{20}$$

where $\hat{f}(x,C) = a_n(d^n f(x,C)/dx^n) + a_{n-1}(d^{n-1} f(x,C)/dx^{n-1}) + \cdots + a_0 f(x,C)$, b_i are coefficients which are functions of x and are related to the effect of h. Also P is the parameter of KLMS Model that includes error vector **E** and learning data x_i. Finally, we can obtain desired output of $K(x,P)$ from (20) for learning of KLMS as follows:

$$K(x,P) = -\frac{1}{b_0}\left(v_o(x) - \left(\hat{f}(x,C) + b_n \frac{d^n K(x,P)}{dx^n} + b_{n-1} \frac{d^{n-1} K(x,P)}{dx^{n-1}} + \cdots\right)\right) \tag{21}$$

Already, we have acquired an equation that can be used to calculate the desired outputs for random inputs. These input–output pairs are finally used in KLMS. This process is explained in details below.

Given a differential equation, $f(x,C)$ and h should be determined as explained above as well as $K(x,P)$ using (21). Then we must acquire some learning samples to train UKLMS, therefore an input vector (X) is generated so that it covers the problem domain uniformly with step length (dx). Since the implemented UKLMS is unsupervised, the output vector (K) to the random input must be calculated automatically. We have utilized an iterative algorithm to calculate the referred outputs. To do so, K_0 is initialized randomly and is updated using (22)

$$K_{i+1} = -\frac{1}{b_0}\left(v_o(x) - \left(\hat{f}(x,C) + b_n \frac{d^n K_i}{dx^n} + b_{n-1} \frac{d^{n-1} K_i}{dx^{n-1}} + \cdots\right)\right). \tag{22}$$

The algorithm is unsupervised because; here no desired signal is required to be determined by user and thus the output of the model is generated by iterating the algorithm progressively. The derivatives of K in (22) are calculated numerically; the first derivative is the first order difference of K divided by dx, the second derivative is the second order difference of K divided by dx^2 and so on.

The iteration is stopped when the stopping criteria is met. The criteria includes a small difference between K_{i+1} and K_i or a large number of iteration. Hereby some input–output pairs are generated and in the next step, UKLMS is generated and trained according to the available learning samples. The final result is achieved by combining UKLMS (K), $f(x,C)$ and h according to (13).

Now, we can complete the proposed unsupervised KLMS for solving differential equations by the following algorithm:

Comment: Unsupervised KLMS Algorithm to solve differential equations

Initialize:

ξ: Resolution

x: Input vector includes ξ points in $[a,b]$.

$\boldsymbol{K_1}$: initial random parameters of KLMS Model.

$K_{\text{Old}}=K_1$;

While Iteration < Max iteration **or** Criteria < threshold

1: $[b, f\_\text{hat}\,]$=**Call** *DF_Equ_Parameter*; **Comment:** calculation of b_i, $\hat{f}$ with regard to ODE.

2: [dK]=**Call** *Ndifference*; **Comment:** calculation of numerical derivative of K_i $(K_i^{(n)},\ldots,K_i^{(1)})$

3: Desired_Output=*Calculate of* (22);

4: $[P_{i+1}]$=**Call** learning the KLMS with (x, *Desired_Output*)

5: K_{i+1}=**Call** KLMS (x,P); **Comment:** calculate the output of new KLMS Model

6: Criteria=$Abs(K_{i+1}-K_{\text{old}})$;

7: $K_{\text{old}}=K_{i+1}$;

End While

8:**Display** (13)

In the above algorithm, firstly, step initialization is performed which includes determination of number of points for analysis (ξ).

For the first time we cannot calculate difference of K in line 2 algorithm so before **While** we have generated an initial K. For this purpose, we can use an arbitrary random data.

A loop is necessary to receive the convergence condition which includes the maximum number of iterations or small change to K in two consecutive iterations. Hands down we see if K have no considerable change, and then it satisfies (21) thus ($K(x,P)$) can be used as the answer of ODE in (13). After that we can easily calculate the answer of ODE at any arbitrary point $x \in [a,b]$.

Remainder of algorithm includes line 1 to obtain b_i and $\hat{f}$, line 2 to compute $K^{(n)},\ldots,K^{(1)}$ for substitution in (22), line 3 to calculate (22), line 4 to learn KLMS model, line 5 to calculate the output of model and line 6 is the measurement of stop criteria.

To show that how the method works (in details) we describe it by solving six problems in Section 3.

3. Numerical simulations

In this section, to illustrate the method applicability, six problems are solved. For each example there is a discussion about the amount of step size parameters and the kernel function used.

Example 1. Consider the following ordinary differential equation:

$$\frac{d}{dx}y(x)+2y(x)=1, \quad y(0)=1 \tag{23}$$

We can write the corresponding trial solution in the following form:

$$y_p(x)=1+(x-0)K(x,P) \tag{24}$$

It is easy to check that (24) satisfies the initial condition in (23). If we substitute (24) in (23) desired answer will be obtained:

$$K(x,P)=\frac{-1-x(d/dx)K(x,P)}{1+2x} \tag{25}$$

Now we can apply the KLMS algorithm. Analytical solution and the solution which is obtained with KLMS algorithm ($y_p(x)$) are shown in Fig. 2 (for $x \in [0,10]$). Note that in this problem, the interval [0,10] discretized into 50 equal parts.

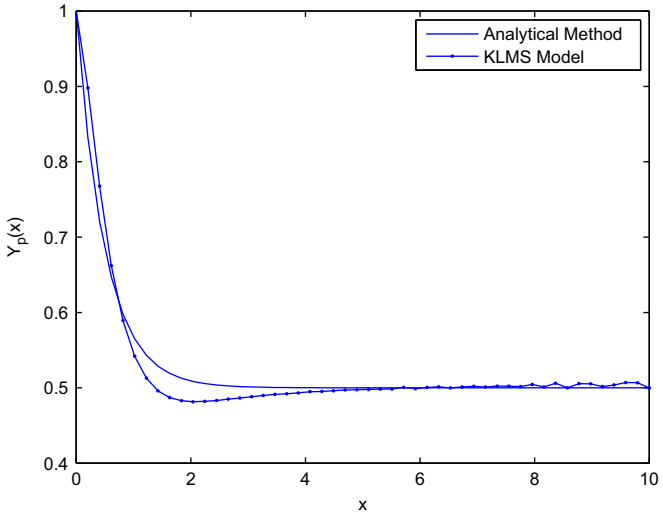

Fig. 2. Solution of Example 1.

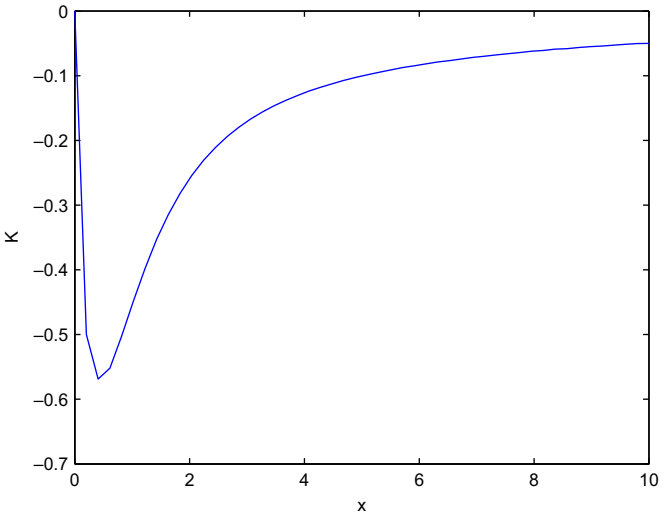

Fig. 3. The output of KLMS after convergence for Example 1.

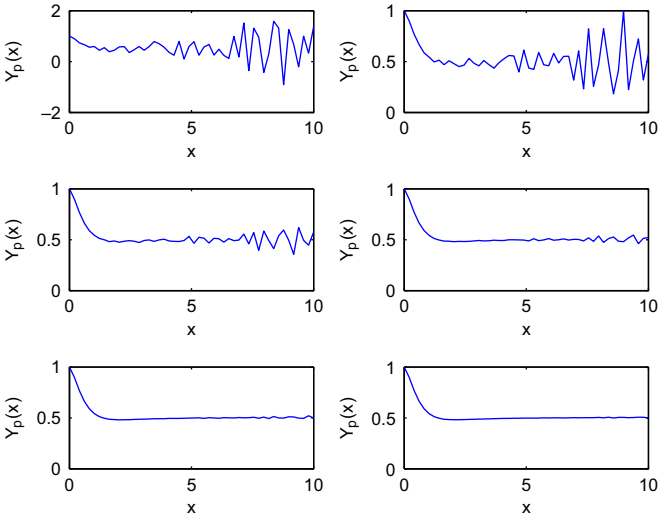

Fig. 4. Convergence of answer for Example 1 in selected six numbers of iterations.

The output of KLMS algorithm (K), which can be calculated by (21), is shown in Fig. 3.

To show the convergent behavior of the method we have selected six numbers of iterations which can be observed in Fig. 4.

Finally, the error (difference between output of algorithm and analytic answer of ODE) is plotted in Fig. 5.

Example 2. Consider the following first order differential equation with sinusoidal excitation:

$$\frac{d}{dx}y(x)+2y(x)=\sin(x), \quad y(0)=1 \tag{26}$$

Desired response of KLMS algorithm (K_p) can be found as follows:

$$K(x,P)=\frac{\sin(x)-x(d/dx)K(x,P)-2}{1+2x} \tag{27}$$

Analytical solution and obtained solution via KLMS algorithm are compared in Fig. 6. The output of KLMS algorithm (K) is shown in Fig. 7. To show the convergent behavior of the method we have

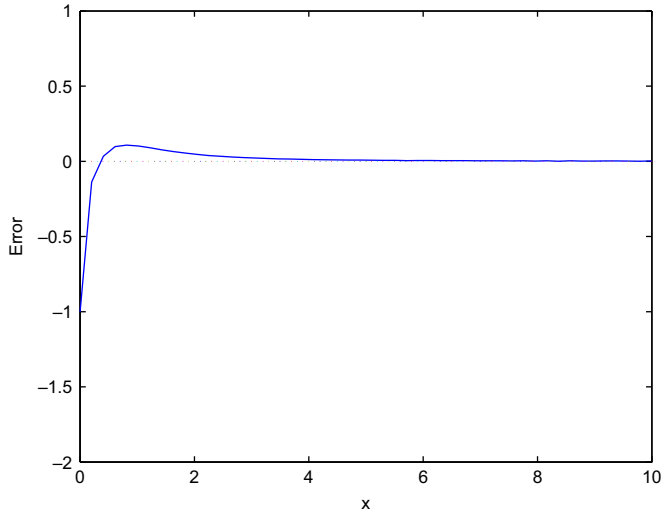

Fig. 5. Error function (difference between output of algorithm and analytic answer of ODE) for Example 1.

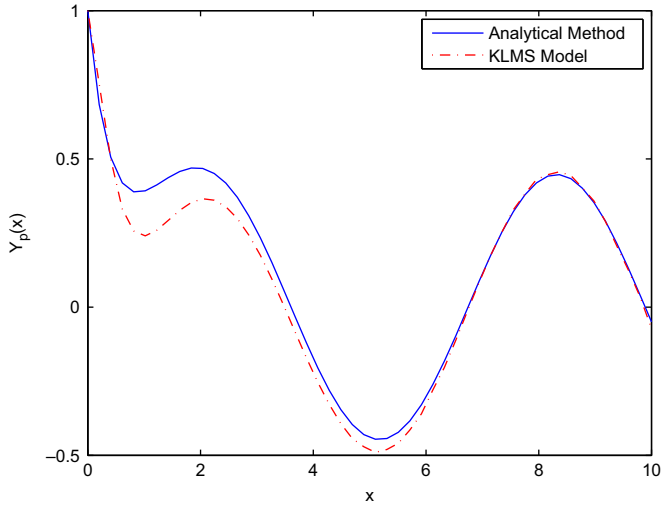

Fig. 6. Solution of Example 2.

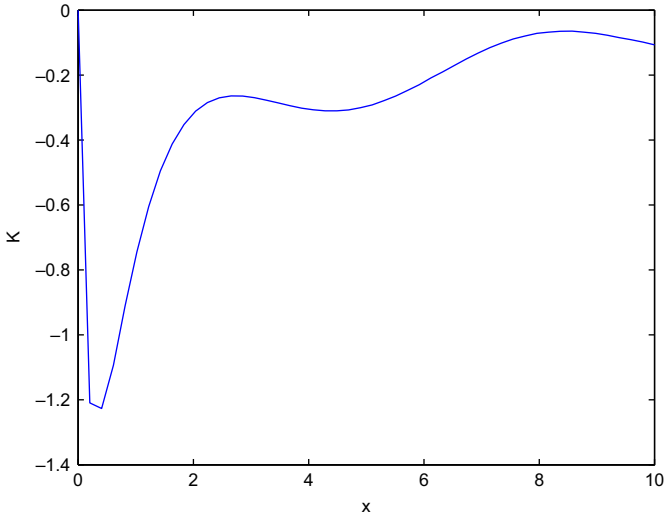

Fig. 7. Convergence of the out put of KLMS for Example 2.

selected six numbers of iterations which can be observed in Fig. 8. Finally, the error is plotted in Fig. 9. Note that in this problem, the interval [0,10] discretized into 50 equal parts.

Example 3. Consider the following first order differential equation with nonlinear sinusoidal excitation:

$$\frac{d}{dx}y(x)+2y(x)=x^3\sin\left(\frac{1}{2}x\right),\quad y(0)=1 \tag{28}$$

Analytical solution and obtained solution via KLMS algorithm are compared in Fig. 10. The output of KLMS algorithm (K) is shown in Fig. 11. To show the convergent behavior of the method we have selected six numbers of iterations which can be observed in Fig. 12. Finally, the error is plotted in Fig. 13. Note that in this problem, the interval [0,10] discretized in to 100 equal parts.

Example 4. Consider the second order differential equation with time-varying input signal. This example shows the case of variable-time input signal.

$$\frac{d^2}{dx^2}y(x)+y(x)=2+2\sin(4x)\cos(3x)$$
$$y(0)=1,\quad y(1)=0 \tag{29}$$

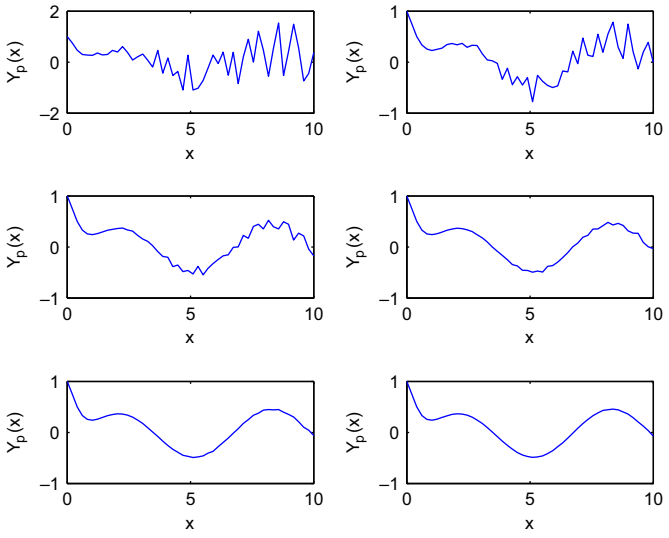

Fig. 8. Convergence of answer for Example 2 in selected six numbers of iterations.

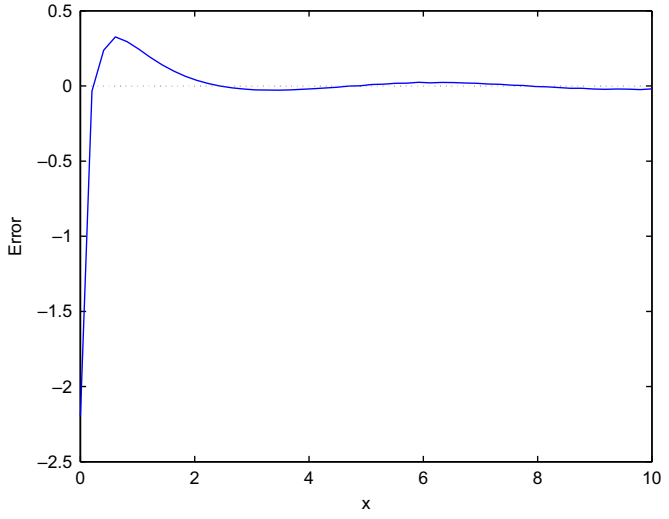

Fig. 9. Errors function for Example 2.

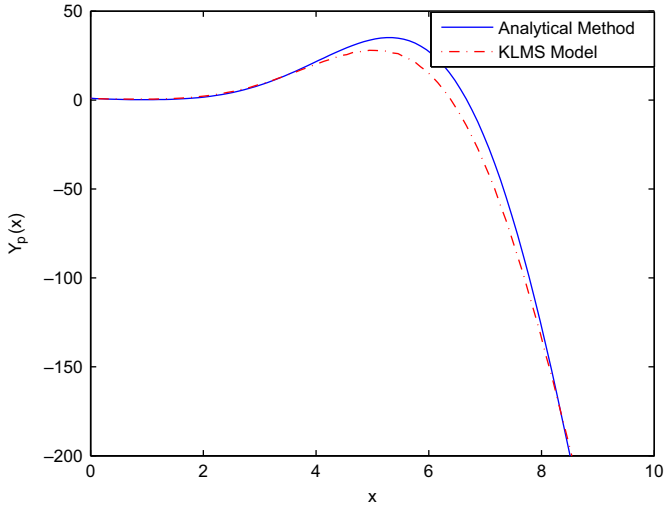

Fig. 10. Solution of Example 3.

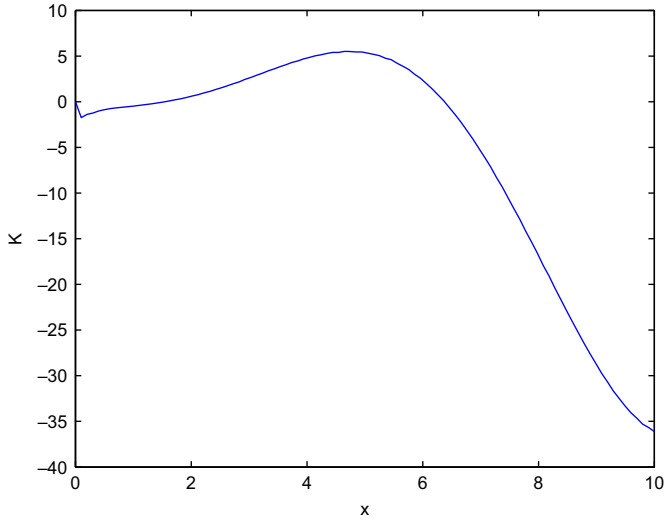

Fig. 11. Convergence of KLMS in different iterations for Example 3.

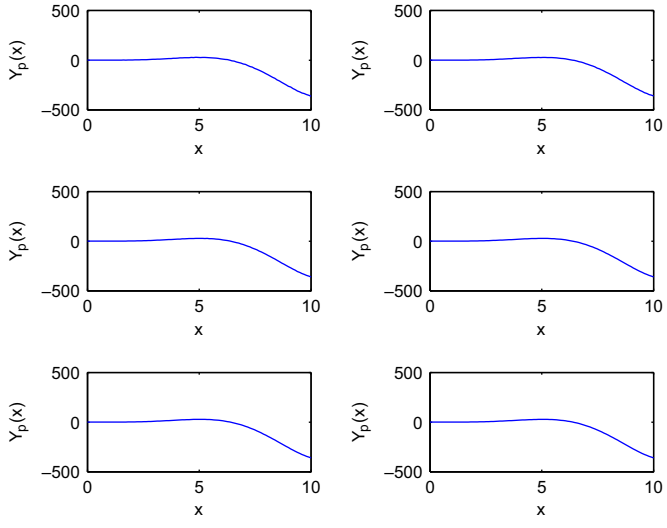

Fig. 12. Convergence of answer for Example 3 in selected six numbers of iterations.

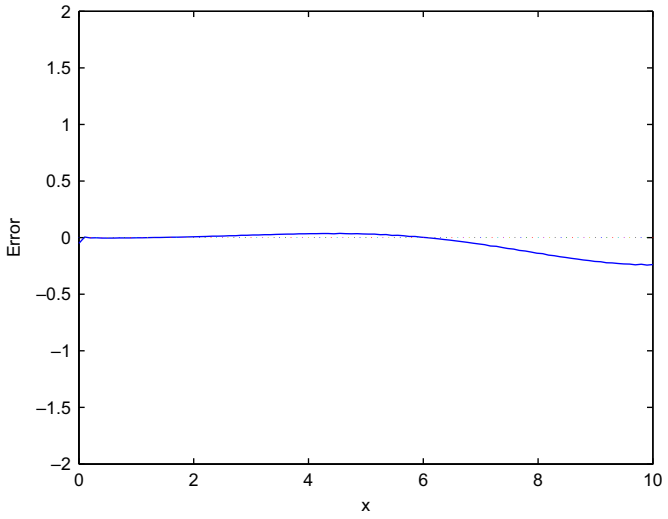

Fig. 13. Errors function for Example 3.

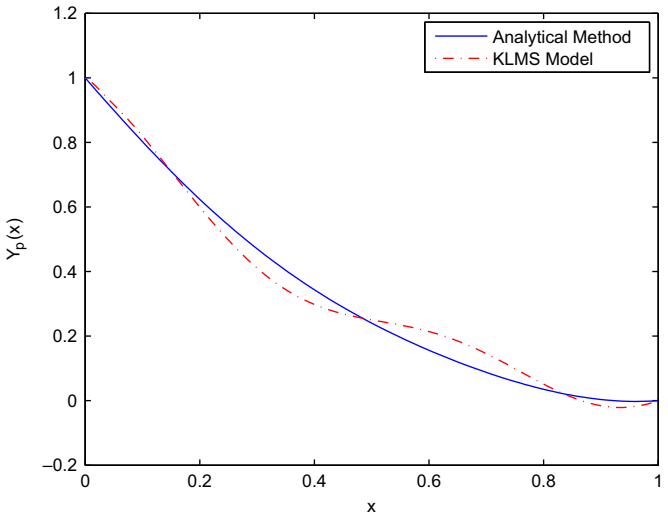

Fig. 14. Solution of Example 4.

Desired response of KLMS algorithm (K) easily can be found. Here we can write the trial solution of (29) in the following form:

$$y_p(x) = x(x-1)K(x,P)+1-x \tag{30}$$

Analytical solution and obtained solution via KLMS algorithm are compared in Fig. 14. The output of KLMS algorithm (K) is shown in Fig. 15. To show the convergent behavior of the method we have selected six numbers of iterations which can be observed in Fig. 16. Finally, the error is plotted in Fig. 17. Note that in this problem, the interval [0,1] discretized in to 100 equal parts.

Example 5. Consider the following second order differential equation with constant excitation:

$$\frac{d^2}{dx^2}y(x)+y(x)=2, \quad y(0)=1, \quad y(1)=0 \tag{31}$$

The related trial function would be in the following form if $y(x_0)=y_0$, $y(x_1)=y_1$

$$y_p(t) = \frac{x_1y_0+x_0y_1}{x_1-x_0}+\frac{y_1-y_0}{x_1-x_0}x+(x-x_0)(x-x_1)K(x,P) \tag{32}$$

This solution satisfies the boundary condition. Analytical solution and obtained solution via KLMS algorithm are compared in

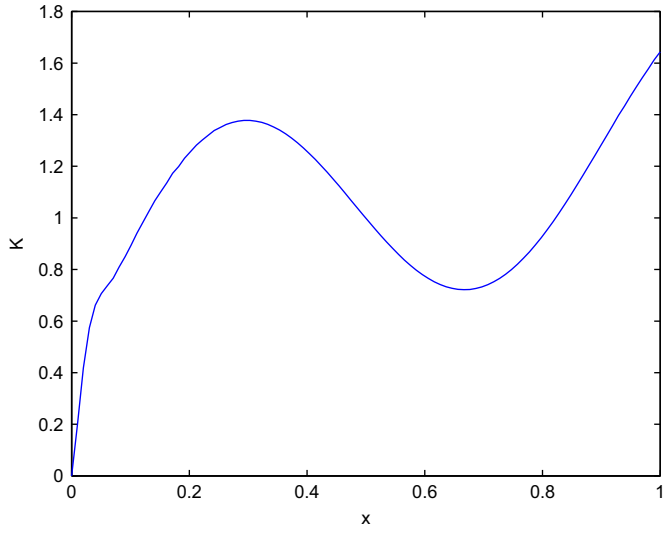

Fig. 15. KLMS output after convergence for Example 4.

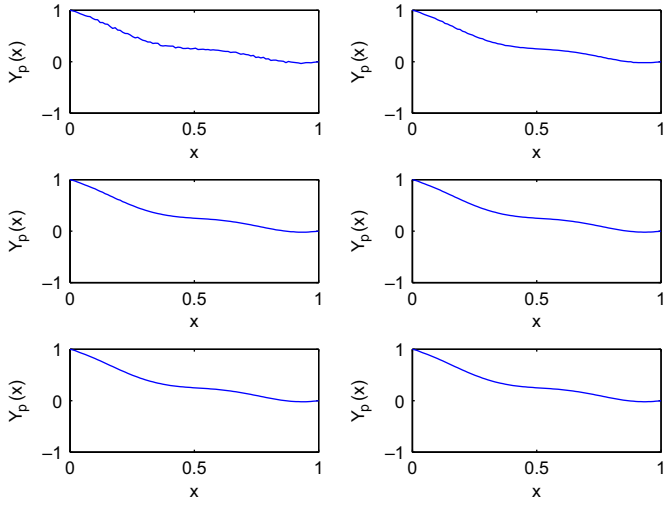

Fig. 16. Convergence of answer for Example 4 in selected six numbers of iterations.

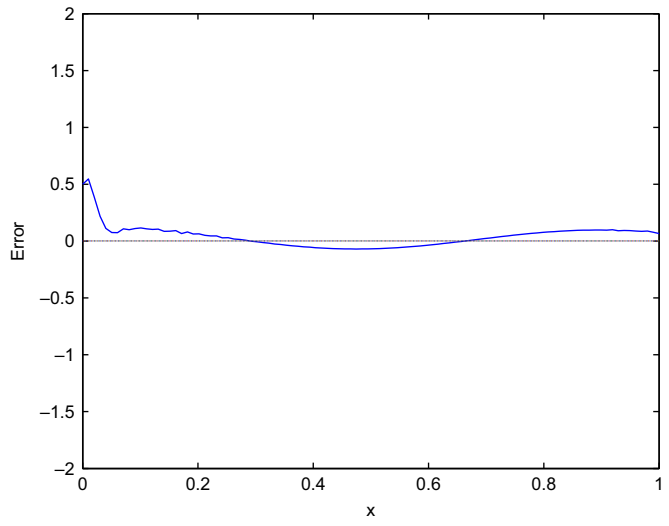

Fig. 17. Errors function for Example 4.

Fig. 18. The output of KLMS algorithm (K) is shown in Fig. 19. To show the convergent behavior of the method we have selected six numbers of iterations which can be observed in Fig. 20. Finally, the error is plotted in Fig. 21. Note that in this problem, the interval [0,1] discretized into 50 equal parts.

3.1. Comparison by UANFIS (see [21])

To show that the new current method is more accurate, we compare the obtained results in the above examples by the results in [21]. Table 1 shows the comparison results.

It is straightforward to see that the new results by KLMS algorithm are more accurate.

Example 6. Consider the following differential equation which has no analytical solution:

$$\frac{dy(x)}{dx} = y^2(x) + x^2, \quad y(0) = 1 \tag{33}$$

We solved this problem for $t \in [0,0.2]$. Fig. 22 shows the comparison between the numerical method and the KLMS output.

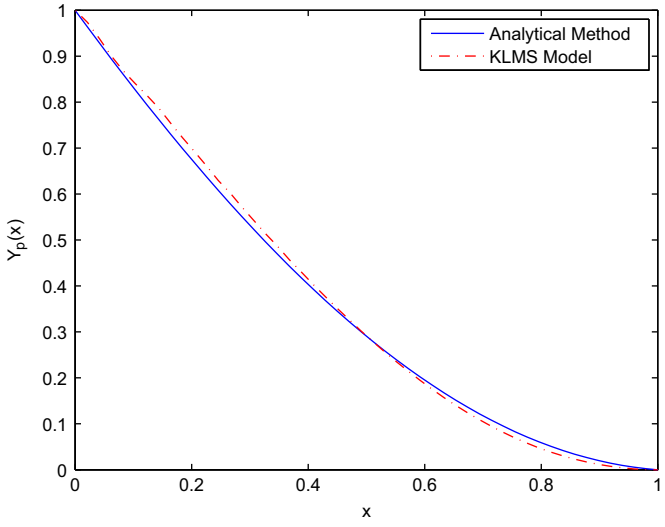

Fig. 18. Solution of Example 5.

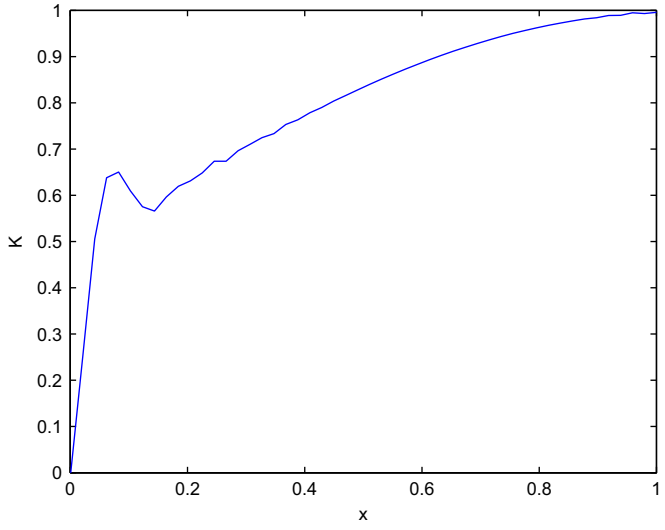

Fig. 19. KLMS output after convergence for Example 5.

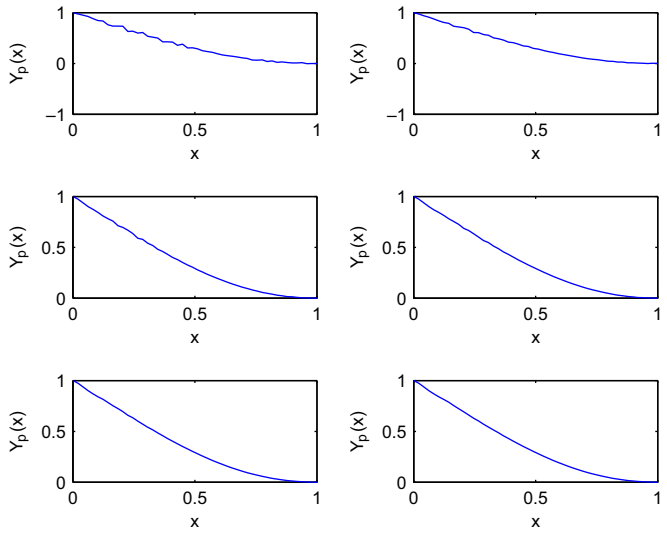

Fig. 20. Convergence of answer for Example 5 in selected six numbers of iterations.

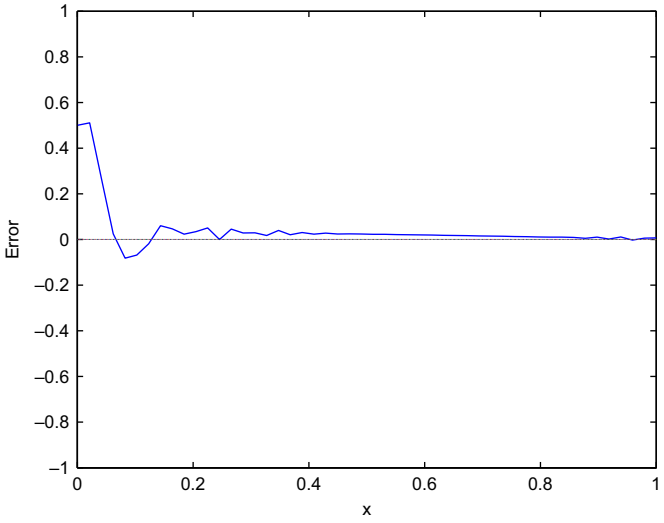

Fig. 21. Errors function for Example 5.

Table 1
Comparing error in KLMS and ANFIS.

	Example 1	Example 2	Example 3	Example 4	Example 5
UKLMS	0.0024	0.0064	0.0605	0.0098	0.0356
UANFIS	0.0155	0.0252	0.0958	0.0556	0.0967

Note that in this example, we applied the KLMS for $x \in [0,0.2]$; we can apply the method for each subintervals to obtain more accurate results. Note that in this problem, the interval [0,0.2] discretized into 100 equal parts.

3.2. Error analysis

In this subsection, to illustrate the reliability of the method, following [26,27] we calculated the amount of the variance and mean values of the errors for each example. By use of the

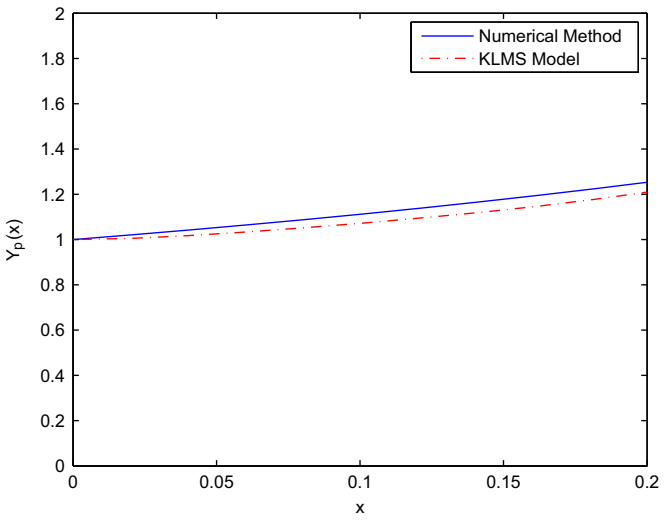

Fig. 22. Results for Example 6.

Table 2
The amount of Diebold–Mariano statistical test S_1 for each example.

	Example 1	Example 2	Example 3	Example 4	Example 5	Example 6
S_1	0.9416	1.8678	0.1702	0.6741	0.4494	1.4411

Diebold–Mariano statistical test S_1 (for more details see [26,27]) it can be inferred that the errors of solved problems has an asymptotic standard normal distribution. Table 2 shows the amount of S_1 for each solved problems.

According to the values of S_1 (which are between -1.96 and 1.96) it can be inferred that the errors of six solved problems has an asymptotic standard normal distribution.

4. Concluding remarks and future works

This paper presented a novel approach to solve ordinary differential equations by the use of an unsupervised version of kernel least mean square (KLMS) algorithm. The accuracy of the proposed method was examined by solving first-order and second-order differential equations with input excitation signal in both constant and time-varying formats. The achieved results demonstrate that the accuracy and fast convergence of the novel approach, which takes advantages of KLMS algorithm in its initial form. The results are comparable with the results of similar approaches that use neural networks. Furthermore, implementing the KLMS algorithm is very simple; and also the trial solution is in a close and differentiable form and satisfies the boundary/initial conditions. The results were compared by ANFIS method in [21]. Also we solved a DE which had no analytical solution. The main advantage of the method is that the algorithm is unsupervised, because no desired signal is required to be determined by user and the output of the model is generated by iterating the algorithm progressively.

In the future, we will develop our method to solve nonlinear and partial differential equations. If the capability of solving partial differential equation is added to this method, it would be easily extended to solve high-dimensional problems. Moreover, to achieve more accurate results, we can apply KLMS in subintervals distinctly.

References

[1] B. Widrow, Adaptive Filters I. Fundamentals (TR 6764-6), Technical Report. Stanford Electronics Laboratories, Stanford, CA, 1966.
[2] V. Vapnik, The Nature of Statistical Learning Theory, Springer, New York, 1995.
[3] F. Girosi, M. Jones, T. Poggio, Regularization theory and neural networks architectures, Neural Computation 7 (2) (1995) 219–269.
[4] B. Scholkopf, A. Smola, K.R. Muller, Nonlinear component analysis as a kernel eigenvalue problem, Neural Computation 10 (1998) 1299–1319.
[5] F.R. Bach, M.I. Jordan, Kernel independent component analysis, Journal of Machine Learning Research 3 (2002) 1–48.
[6] P. Pokharel, W. Liu, J.C. Principe, Kernel lms, in: Proceedings of the International Conference on Acoustics, Speech and Signal Processing, 2007.
[7] W. Liu, P. Pokharel, J.C. Principe, The kernel least mean square algorithm, IEEE Transactions on Signal Processing 56 (2008) 543–554.
[8] B. Widrow, J.R. Glover, J.M. McCool, J. Kaunitz, C.S. Williams, R.H. Hearn, J.R. Zeidler, E. Dong, R.C. Goodlin, Adaptive noise cancelling: principles and applications, Proceedings of the IEEE 63 (1975) 1692–1716.
[9] A. Gunduz, J.-P. Kwon, J.C. Sanchez, J.C. Principe, Decoding hand trajectories from ECoG recordings via kernel least-mean-square algorithm, in: Proceedings of the 4th International IEEE EMBS Conference on Neural Engineering, Antalya, Turkey, April 29–May 2, 2009.
[10] I.E. Lagaris, A. Likas, D.I. Fotiadis, Artificial neural network for solving ordinary and partial differential equations, IEEE Transactions on Neural Networks 9 (5) (1998) 987–1000.
[11] K. Hornik, Multilayer feedforward networks are universal approximators, Neural Networks 2 (1989) 359–366.
[12] N. Smaoui, S. Al-Enezi, Modeling the dynamics of nonlinear partial differential equations using neural networks, Journal of Computational and Applied Mathematics 170 (2004) 27–58.
[13] R. Brause, Adaptive modeling of biochemical pathways, in: Proceedings of the 15th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'03), 2003.
[14] S. Hea, K. Reif, R. Unbehauen, Multilayer neural networks for solving a class of partial differential equations, Neural Networks 13 (2000) 385–396.
[15] L. Manevitz, A. Bitar, D. Givoli, Neural network time series forecasting of finite-element mesh adaptation, Neurocomputing 63 (2005) 447–463.
[16] T. Leephakpreeda, Novel determination of differential-equation solutions: universal approximation method, Journal of Computational and Applied Mathematics 146 (2002) 443–457.
[17] A. Malek, R.Shekari Beidokhti, Numerical solution for high order differential equations using a hybrid neural network—optimization method, Applied Mathematics and Computation 183 (2006) 260–271.
[18] N. Mai-Duy, T. Tran-Cong, Numerical solution of differential equations using multi quadric radial basis function networks, Neural Networks 14 (2001) 185–199.
[19] L. Jianyu, L. Siwei, Q. Yingjian, H. Yaping, Numerical solution of elliptic partial differential equation using radial basis function neural networks, Neural Networks 16 (2003) 729–734.
[20] D.R. Parisi, M.C. Mariani, M.A. Laborde, Solving differential equations with unsupervised neural networks, Chemical Engineering and Processing 42 (2003) 715–721.
[21] Hadi Sadoghi Yazdi, R. Pourreza, Unsupervised adaptive neural-fuzzy inference system for solving differential equations, Applied Soft Computing 10 (2010) 267–275.
[22] R. Shekari Beidokhti, A. Malek, Solving initial-boundary value problems for systems of partial differential equations using neural networks and optimization techniques, Journal of the Franklin Institute 346 (2009) 898–913.
[23] Tsoulos Ioannis G., Dimitris Gavrilis, Euripidis Glavas, Solving differential equations with constructed neural networks, Neurocomputing 72 (2009) 2385–2391.
[24] Yazdan Shirvany, Mohsen Hayati, Rostam Moradian, Multilayer perceptron neural networks with novel unsupervised training method for numerical solution of the partial differential equations, Applied Soft Computing 9 (2009) 20–29.
[25] Sohrab Effati, Morteza Pakdaman, Artificial neural network approach for solving fuzzy differential equations, Information Sciences 180 (2010) 1434–1457.
[26] D. Harvey, S. Leybourne, P. Newbold, Testing the equality of prediction mean squared errors, International Journal of Forecasting 13 (1997) 281–291.
[27] Francis X. Diebold, Roberto S. Mariano, Comparing predictive accuracy, Journal of Business and Economic Statistics 13 (1995) 253–265.
[28] J. Li, S. Luo, Y. Qi, Y. Huang, Numerical solution of elliptic partial differential equation by growing radial basis function neural networks, in: IEEE Proceedings of the International Joint Conference on Neural Networks, vol. 1, Issue date: 20–24 July 2003, pp. 85–90.
[29] H. Modaghegh, H. Khosravi, R.S. Ahoon Manesh, H. Sadoghi Yazdi, A new modeling algorithm—normalized kernel least mean square, in: IEEE 6th International Conference on Innovations in Information Technology, Innovations'09, December 2009, pp. 120–124.
[30] S.H. Ghafarian, H. Sadoghi Yazdi, H. Baradaran Kashani, Kernel least mean square features For HMM-based signal recognition, International Journal of Computer Theory and Engineering 2 (2) (2010) 1793–8201.
[31] L. Yang, M. Cui, New algorithm for a class of nonlinear integro-differential equations in the reproducing kernel space, Applied Mathematics and Computation 174 (2006) 942–960.
[32] J. Kivinen, A. Lexander, J. Smola, R.C. Williamson, Online Learning with Kernels, IEEE Transactions on Signal Processing 52 (8) (2004) 2165–2176.
[33] H. Bao, I. Panahi,Active noise control based on kernel least-mean-square algorithm, IEEE Record of the Forty-Third Asilomar Conference on Signals, Systems and Computers, 2009, pp. 642–644.
[34] A. Gunduz, J. Kwon, J.C. Sanchez, J.C. Principe, Decoding hand trajectories from ECoG recordings via kernel least-mean-square algorithm, in: 4th international IEEE/EMBS Conference on Neural Engineering, vol. 9, 2009, pp. 267–270.

Hadi Sadoghi Yazdi received the B.S. degree in electrical engineering from Ferdowsi University of Mashhad in 1994, and then he received to the M.S. and Ph.D. degrees in electrical engineering from Tarbiat Modarres University of Tehran, Iran, in 1996 and 2005, respectively. He works in Computer Department as an associate professor at Ferdowsi University of Mashhad. His research interests include pattern recognition and optimization in signal processing. Email: h-sadoghi@um.ac.ir

Morteza Pakdaman was born in Mashhad, Iran. In 2004 he received the B.Sc. in applied mathematics from Ferdowsi University of Mashhad and the M.Sc. in optimization from Tarbiat Moallem University of Sabzevar. His current research intrests include operations research, and applications of neural networks and fuzzy systems in optimization. Email: pakdaman.m@gmail.com, pakdaman@mshdiau.ac.ir

Hamed Modaghegh was born in Mashhad, Iran. In 2005 he received the B.Sc. in Electrical Engineering from Iran University of Science and Technology and the M.Sc. in Communication from Sharif University of Technology in 2007 and he is Ph.D student in communication in Ferdowsi University of Mashhad now. His current research intrests include signal processing and data hiding. Email: hamed.modaghegh@stu-mail.um.ac.ir