Neural Comput & Applic (2012) 21 (Suppl 1):S79–S85
DOI 10.1007/s00521-011-0621-7

ORIGINAL ARTICLE

Ordinary differential equations solution in kernel space

Hadi Sadoghi Yazdi · Hamed Modaghegh · Morteza Pakdaman

Received: 4 December 2010 / Accepted: 3 May 2011 / Published online: 27 May 2011

Abstract This paper presents a new method based on the use of an optimization approach along with kernel least mean square (KLMS) algorithm for solving ordinary differential equations (ODEs). The new approach in comparison with the other existing methods (such as numerical methods and the methods that are based on neural networks) has more advantages such as simple implementation, fast convergence, and also little error. In this paper, we use the ability of KLMS in prediction by applying an optimization method to predict the solution of ODE. The basic idea is that first a trial solution of the ODE is written by using the KLMS structure, and then by defining an error function and minimizing it via an optimization algorithm (in this paper, we used the quasi-Newton BFGS method), the parameters of KLMS are adjusted such that the trial solution satisfies the DE. After the optimization step, the achieved optimal parameters of the KLMS model are replaced in the trial solution. The accuracy of the method is illustrated by solving several problems.

Keywords Least mean square · Kernel least mean square · Ordinary differential equation · BFGS algorithm

H. Sadoghi Yazdi · H. Modaghegh
Engineering Department, Ferdowsi University of Mashhad, Mashhad, Iran
e-mail: h-sadoghi@um.ac.ir

H. Modaghegh
e-mail: hamed.modaghegh@stu.um.ac.ir

M. Pakdaman (✉)
Sama Technical and Vocational Training College, Islamic Azad University, Mashhad Branch, Mashhad, Iran
e-mail: pakdaman@mshdiau.ac.ir; pakdaman.m@gmail.com

1 Introduction

Kernel methods are applied successfully in classification, regression problems, and generally machine learning (support vector machines, SVM [1], regularization networks [2], kernel principal component analysis, K-PCA [3], kernel independent component analysis, K-ICA [4]). Kernel methods have been used to extend linear adaptive filters expressed in inner products to nonlinear algorithms [1, 3, 4]. Pokharel et al. [5, 6] have applied this "kernel trick" to the least mean square (LMS) algorithm to attain a nonlinear adaptive filter in reproducing kernel Hilbert spaces (RKHS), which they have coined kernel least mean square (KLMS). Another application of KLMS can be found in [7].

Now before introducing the new method based on KLMS algorithm, we briefly mention the basic idea for solving differential equations by neural networks, which was presented by Lagaris et al. [8]. Consider the differential equation in the following form:

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

Here, $\psi(x)$ denotes the solution, G defines the structure of differential equation, ∇ is some differential operator, and $\bar{D}$ is the problem domain. The basic idea, called collocation method, is to discretize the domain $\bar{D}$ over a finite set of points D. Thus, (1) becomes a system of equations. An approximation of the solution $\psi(x)$ is given by the trial solution $\psi_t(x)$. 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:

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

Therefore, minimizing the error function E in (2) is equal to finding an approximated solution of (1). Since

multilayer feed forward neural networks are universal approximators (Hornik 1989), the trial solution $\psi_t(x)$ can be represented by such an artificial neural network as (3) [9].

$$\psi_t(x) = A(x) + F(x, N(x, p)) \quad (3)$$

where $A(x)$ contains no adjustable parameter and satisfies the boundary conditions and $F(x, N(x, p))$ is a single output feed forward neural network with adjustable parameters p and the input vector x. Consequently, the problem is to find adjusted weights of neural network, which minimize (2). Since E is differentiable with respect to the weights for most differential equations, efficient, gradient-based learning algorithms for artificial neural networks can be employed for minimizing (2). The history of solving differential equation using neural networks is reviewed in the rest of this section.

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 [10]. In [11], Brause used differential equations for the modeling of biochemical pathways, and these equations were solved using neural networks. In [12], 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.

Also, Manevitz et al. [13] 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 [14] presented fuzzy linguistic model in neural network to solve differential equations and applied it as universal approximators for any nonlinear continuous function.

In [15], Malek and Beidokhti presented a hybrid method based on optimization techniques and neural networks methods for solving high-order ordinary differential equations. They proposed a new solution method for the approximated solution of high-order ordinary differential equations using innovative mathematical tools and neural-like systems. Hybrid method could result in improved numerical methods for solving initial/boundary value problems without using pre-assigned discretization points.

In another work, Mai-Duy and Tran-Cong [16] presented mesh-free procedures for solving linear differential equations, ordinary differential equations, and elliptic partial differential equations based on multi-quadric radial basis function networks. In [21], Sadoghi Yazdi and Pourreza (our previous work) solved ODE using new version of adaptive neuro-fuzzy inference system. Also, Jianyu et al. in [17] described a neural network for solving partial differential equations, which activation functions of the hidden nodes were the radial basis functions (RBF) whose parameters were learnt by a two-stage gradient descent strategy. Also, solving differential equation using neural network was applied to real problems such as a non-steady fixed bed non-catalytic solid/gas reactor [18].

In recent work, Sadoghi Yazdi et al. [22] applied the unsupervised KLMS algorithm to solve ordinary differential equations which is an unsupervised version of KLMS. The main difference between the method that will be introduced in this paper and the UKLMS in [22] is that the UKLMS is unsupervised and no desired signal needs to be determined by user and the output of the model is generated by iterating the algorithm progressively. But in the new following method, the parameters of KLMS are adjusted via an optimization algorithm. The use of neural networks was not limited just to ODEs. Effati and Pakdaman [20] used the artificial neural networks to solve fuzzy differential equations.

In the aforementioned works, some notes are observed as follows:

- Solving differential equation (ordinary or partial) or high order by using neural networks.
- Some researchers performed logistic function using neural networks for solving differential equations.
- Applying the solution of differential equations via neural networks in real-world problems.

Now, based on KLMS algorithm, we present a new method to solve ordinary differential equations. The paper is organized as follows. The basic requirements of KLMS are explained in Sect. 2 (Preliminaries). Section 3 is devoted to method formulation. Experimental results are discussed in Sect. 4, and finally Sect. 5 contains concluding remarks.

2 Preliminaries

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—$y(n)$—is to be identified and the LMS algorithm attempts to adapt the filter $\hat{y}(n)$ to make it as close as possible to $y(n)$. The algorithm uses $u(n)$ 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. Now, the weight vector is updated based on the following rule:

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

where $w(n)$ is weight vector and μ is step size and $u(n)$ is input vector. The filter output y is calculated by

$$y(n) = \bar{w} \times u(n). \tag{5}$$

Successive corrections of the weight vector eventually leads to the minimum value of the mean squared error. Further information about LMS can be found in [6]. 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.

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.

As presented in [6], 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.

$$\Omega(n+1) = \Omega(n) + 2\mu \times e(n) \times \Phi(u(n)) \tag{6}$$

where $\Omega(n)$ is weight vector in the HDS. The estimated output $y(n)$ will be calculated by

$$y(n) = \langle \Omega(n), \Phi(u(n)) \rangle. \tag{7}$$

The input vector $u(n)$ will be transformed to the infinite feature vector $\Phi(u(n))$, whose components are then linearly combined by the infinite dimensional weight vector. Non-recursive type of equation (6) can be written as

$$\Omega(n) = \Omega(0) + 2\mu \sum_{i=0}^{n-1} e(i)\Phi(u(i)). \tag{8}$$

By choosing $\Omega(0) = 0$,

$$\Omega(n) = 2\mu \sum_{i=0}^{n-1} e(i)\Phi(u(i)). \tag{9}$$

Based on (7) and (9),

$$\begin{aligned} y(n) &= \langle \Omega(n) \times \Phi(u(n)) \rangle \\ &= \left\langle 2\mu \sum_{i=0}^{n-1} e(i)\Phi(u(i)), \Phi(u(n)) \right\rangle \\ &= 2\mu \sum_{i=0}^{n-1} e(i) \langle \Phi(u(i)), \Phi(u(n)) \rangle. \end{aligned} \tag{10}$$

We can use kernel trick here to calculate $y(n)$:

$$y(n) = \mu \sum_{i=0}^{n-1} e(i)k(u(i), u(n)). \tag{11}$$

Equation (11) is named 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:

$$y(n) = \mu \sum_{i=0}^{\zeta} e(i)k(u(i), u(n)). \tag{12}$$

This change decreases the complexity of algorithm. Thus, we can train the system with fewer data and also use it to predict new data.

Good prediction ability in non-linear channels is one of the algorithm advantages, but results are sensitive to step size and signal amplitude stability.

3 New method

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

$$\begin{aligned} &a_n \frac{\mathrm{d}^n y(x)}{\mathrm{d}x^n} + a_{n-1} \frac{\mathrm{d}^{n-1} y(x)}{\mathrm{d}x^{n-1}} + \cdots + a_0 y(x) = v_o(x), \\ &x \in [a, b] \end{aligned} \tag{13}$$

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

$$\begin{aligned} &y(0) = y_0^0, \quad y^{(1)}(0) = y_0^{(1)}, \ldots, y^{n-1}(0) = y_0^{n-1} \\ &\text{or} \\ &y(x_0) = y_{x_0}, \quad y(x_1) = y_{x_1}, \ldots, y(x_n) = y_{x_n} \end{aligned} \tag{14}$$

Following [15], we propose a trial solution for the above differential equation is like (15).

$$\begin{aligned} y_T(x, E) &= A(x, y_0^{(0)}, y_0^{(1)}, \ldots, y_0^{(n-1)}) + F(x, K(x, E)) \\ &= A(x, C) + F(x, K(x, E)) \end{aligned} \tag{15}$$

In (15), $y_T(x, E)$ is composed of two parts. The first part, $A(x, C)$, is a function for satisfaction of initial/boundary conditions (C denotes the initial/boundary conditions) and contains no adjustable parameters. In the second part, $F(x, K(x, E))$ contains a KLMS part which the errors of it must be adjusted so as the trial solution satisfy the DE. Also, $F(x, K(x, E))$ must be zero at initial points. $E = (e_1, e_2, \ldots, e_n)$ is the error vector as in (12). By this construction, $y_T(x, E)$ satisfies the DE as well as the initial/boundary conditions and the error vector must be adjusted. To train the KLMS part, we divide the interval $[a, b]$ into n equal parts $(x_0 = a, x_1, \ldots, x_{n-1} = b)$. Then, we put these points into the KLMS function. If we rewrite (12), we have:

$$K(x, E) = \mu \sum_{i=0}^{n-1} e(i)k(x_i, x). \tag{16}$$

Put (16) in (15),

$$y_T(x,E) = A(x,C) + F\left(x, \mu \sum_{i=0}^{n-1} e(i)k(x_i,x)\right). \tag{17}$$

Substituting (17) into the (13), we have

$$a_n \frac{\mathrm{d}^n y_T(x,E)}{\mathrm{d}x^n} + a_{n-1} \frac{\mathrm{d}^{n-1} y_T(x,E)}{\mathrm{d}x^{n-1}} + \cdots + a_0 y_T(x,E) - v_o(t) = 0. \tag{18}$$

To solve (18), we define the following unconstraint optimization problem:

$$\min_E \{G(x,E)\}^2 \tag{19}$$

where

$$G(x,E) = a_n \frac{\mathrm{d}^n y_T(x,E)}{\mathrm{d}x^n} + a_{n-1} \frac{\mathrm{d}^{n-1} y_T(x,E)}{\mathrm{d}x^{n-1}} + \cdots + a_0 y_T(x,E) - v_o(x)$$

To demonstrate the proposed procedure, we mention some examples.

Consider the following first-order DE:

$$\frac{\mathrm{d}y(x)}{\mathrm{d}x} = v_o(y,x), \quad x \in [0,1],\ y(0) = A. \tag{20a}$$

Hence,

$$A(x,C) = A \Rightarrow y_T(x,E) = A + xK(x,E). \tag{20b}$$

Following (19) and by replacing (20b) in (20a), we can introduce the following optimization problem:

$$\min_E \left[K(x,E) + x\frac{\partial K(x,E)}{\partial x} - v_0(y_T,x)\right]^2. \tag{21}$$

The optimization problem (21) can be solved by any optimization technique.

Now consider the following second-order DE.

$$\frac{\mathrm{d}^2 y(x)}{\mathrm{d}x^2} = v_o\left(\frac{\mathrm{d}y}{\mathrm{d}x}, y, x\right), \quad x \in [0,1]. \tag{22}$$

The trial solution of this DE is considered in two cases. In the first case, these initial conditions are considered as $y(0) = A$ and $\frac{\mathrm{d}y}{\mathrm{d}x}(0) = A'$ thus

$$A(x,C) = A + A'x \Rightarrow y_T(x,E) = A + A'x + x^2 K(x,E). \tag{23}$$

Again by replacing (23) in (22), we can define the following optimization problem:

$$\min_E \left[2K(x,E) + 4x\frac{\partial K(x,E)}{\partial x} + x^2\frac{\partial^2 K(x,E)}{\partial x^2} - v_0\left(\frac{dy_T}{dx}, y_T, x\right)\right]^2. \tag{24}$$

In the second case, the boundary conditions are considered as $y(x_0) = y_0$, $y(x_1) = y_1$. Therefore,

$$A(x,C) = \frac{x_1 y_0 + x_0 y_1}{x_1 - x_0} + \frac{y_1 - y_0}{x_1 - x_0}x \Rightarrow$$
$$y_T(x,E) = \frac{x_1 y_0 + x_0 y_1}{x_1 - x_0} + \frac{y_1 - y_0}{x_1 - x_0}x + (x - x_0)(x - x_1)K(x,E). \tag{25}$$

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

The optimization problem (19) (or 21 and 24) is an unconstrained optimization problem, and thus, we can use any optimization techniques such as steepest descent method and conjugate gradient or quasi Newton methods. Newton method is one of the important algorithms in nonlinear optimization. In this paper, we use the Quasi-Newton BFGS method. Quasi-Newton methods were originally proposed by Davidon in 1959 and were later developed by Fletcher and Powell (1963). The most fundamental idea in quasi Newton methods is that an approximation of the Hessian matrix is calculated. Here, the quasi Newton BFGS (Broyden–Fletcher–Goldfarb–Shanno) method is used. This method is quadratically convergent (for more details see [19]). After the optimization step, optimal value of the vector E is obtained, thus, by replacing the optimal parameter E in (15), the trial solution (15) will be the approximated solution of (13).

3.1 Method advantages

The proposed method has several advantages: first that the implementation of the method is simple and also has a quick convergence, second that we can apply the method to solve higher-order ODEs as well as PDEs. We can increase the number of iterations of optimization step to attain more accurate results. We can also apply it to solve a system of ODEs.

4 Numerical simulations

To illustrate the method advantages, in this section, five problems are solved. All problems are solved over the interval [0,1]. The solved problem has transient response in this interval. For all problems, the interval [0,1] is discretized to 20 equal sections. By this selection, E will have 20 elements. To minimize the objective function in (19), we used MATLAB7 optimization toolbox.

Example 1 Consider the following first-order differential equation with constant excitation.

$$\frac{\mathrm{d}}{\mathrm{d}x}y(x) + 5y(x) = 4, \quad y(0) = 1. \tag{26}$$

The related trial function (similar to (17)) can be considered in the following form:

$$y_T(x) = 1 + (x - 0)K(x, E). \tag{27}$$

After substituting (27) in (26), we can define the following optimization problem:

$$\min_E \left[1 + (1 + 5x)K(x, E) + x\frac{\partial K(x, E)}{\partial x}\right]^2. \tag{28}$$

By running the BFGS algorithm in MATLAB7 optimization toolbox, we obtain the optimal value of error vector as reported in Table 1.

By replacing the error vector from Table 1 in (16), the KLMS model will be as (29).

$$\begin{aligned} K(x, E) = {} & 0.0845 \times e^{x^2} + 0.1366 \times e^{(x-0.05)^2} \\ & + 0.4486 \times e^{(x-0.1)^2} + \cdots - 0.6953 \\ & \times e^{(x-0.90)^2} - 0.5779 \times e^{(x-0.95)^2}. \end{aligned} \tag{29}$$

Analytical solution and the KLMS solution are compared in Fig. 1. As it can be seen, the KLMS solution approximates the analytical solution.

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

$$\frac{\mathrm{d}}{\mathrm{d}x}y(x) + 10y(x) = -\sin(10x), \quad y(0) = 1. \tag{30}$$

Similar to Example 1, the related trial solution can be constructed as

$$y_T(x) = 1 + (x - 0)K(x, E). \tag{31}$$

After substituting (31) in (30), we can define the following optimization problem:

$$\min_E \left[10 + (1 + 10x)K(x, E) + x\frac{\partial K(x, E)}{\partial x} + \sin(10x)\right]^2. \tag{32}$$

Analytical solution and obtained solution via KLMS algorithm are compared in Fig 2.

Table 1 Numerical value of error vector for KLMS model in example 1

Error	Value	Error	Value
e_0	0.0845	e_{10}	−0.2569
e_1	0.1366	e_{11}	0.5066
e_2	0.4486	e_{12}	0.2224
e_3	−0.4329	e_{13}	0.7038
e_4	−0.3387	e_{14}	0.2044
e_5	−0.2746	e_{15}	0.0833
e_6	−0.2542	e_{16}	0.3872
e_7	0.1881	e_{17}	−0.1148
e_8	−0.0932	e_{18}	−0.6953
e_9	−0.1176	e_{19}	−0.5779

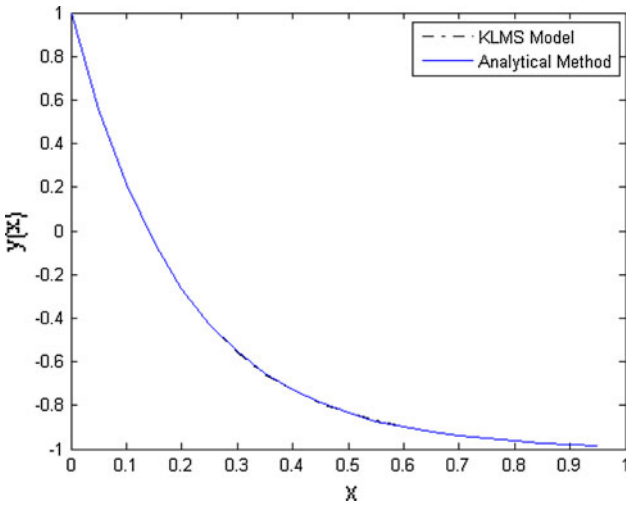

Fig. 1 Comparing analytical solution and the solution via KLMS for example 1

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

$$\frac{\mathrm{d}}{\mathrm{d}x}y(x) + y(x) = (4x)^3 \sin(5x), \quad y(0) = 1. \tag{33}$$

The related trial solution can be constructed as

$$y_T(x) = 1 + (x - 0)K(x, E). \tag{34}$$

After substituting (34) in (33), we can define the following optimization problem

$$\min_E \left[1 + (1 + x)K(x, E) + x\frac{\partial K(x, E)}{\partial x} - (4x)^3 \sin(5x)\right]^2. \tag{35}$$

Analytical solution and obtained solution via KLMS algorithm are compared in Fig 3.

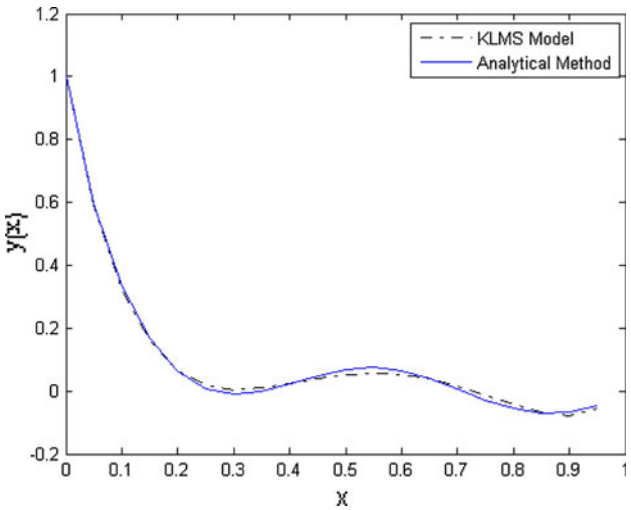

Fig. 2 Solution of example 2

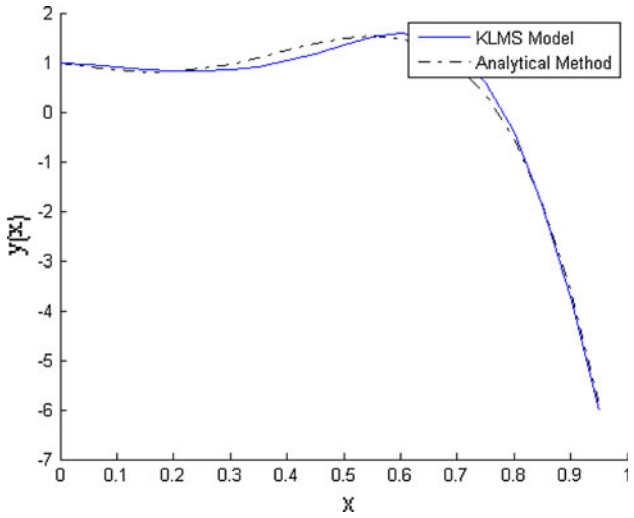

Fig. 3 Solution of example 3

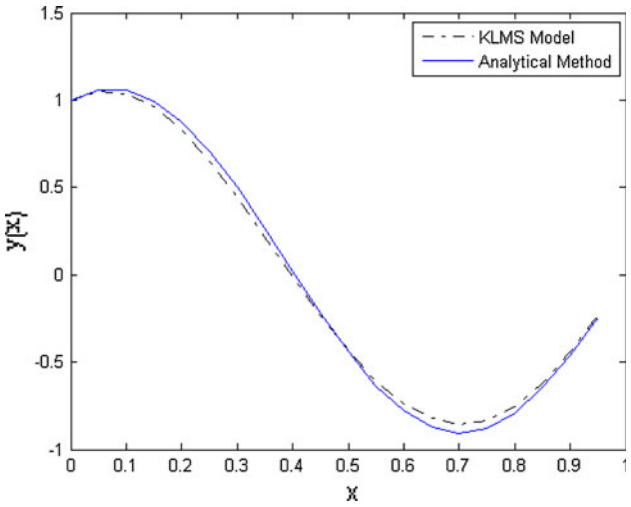

Fig. 4 Solution of example 4

As it can be seen in Examples 4 and 5 despite sinusoidal excitation, the KLMS algorithm could follow the analytical solution. This examples show that this algorithm can get transient and steady-state responses of differential equation.

Example 4 Consider the following second-order differential equation with constant excitation:

$$\frac{\mathrm{d}^2}{\mathrm{d}x^2}y(x) + 25y(x) = 2, \quad y(0) = 1,\ y(1) = 0. \tag{36}$$

For this type of DE, we use (23). The related trial function would be in the following form if $y(0) = A$, $y(1) = B$.

$$\begin{aligned} &A(x, C) = A(1 - x) + Bx \Rightarrow \\ &y_T(x, E) = A(1 - x) + Bx + x(1 - x)K(x, E). \end{aligned} \tag{37}$$

After substitution in (36), we have

$$y_T(x) = (1 - x) + x(1 - x)K(x, E). \tag{38}$$

The trial solution (38) satisfies the boundary conditions. Similar to previous examples, we can define the following optimization problem:

$$\begin{aligned} \min_E \Big[&23 - 25x + (-2 + 25x - 25x^2)K(x, E) \\ &+ (2 - 4x)\frac{\partial K(x, E)}{\partial x} + (x - x^2)\frac{\partial^2 K(x, E)}{\partial x^2}\Big]^2. \end{aligned} \tag{39}$$

Analytical solution and the obtained solution via KLMS algorithm are compared in Fig 4.

Example 5 Consider the second-order differential equation with time-varying input signal. This example shows the case of variable–time input signal.

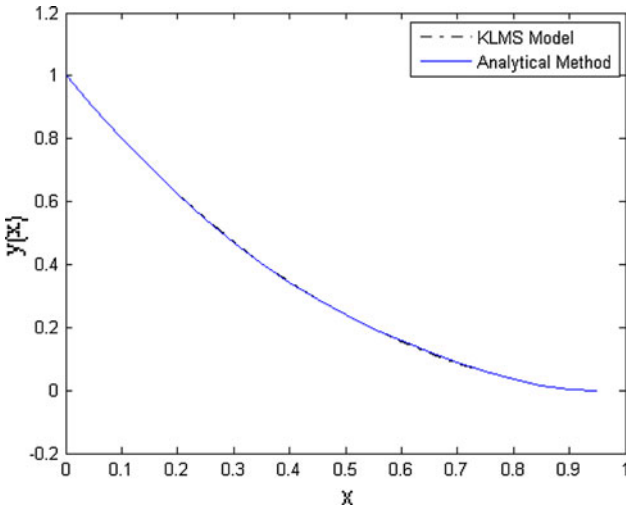

Fig. 5 Solution of example 5

$$\begin{aligned} &\frac{\mathrm{d}^2}{\mathrm{d}x^2}y(x) + y(x) = 2 + 2\sin(4x)\cos(3x) \\ &y(0) = 1, \quad y(1) = 0, \end{aligned} \tag{40}$$

Analytical solution and obtained solution via KLMS algorithm are compared in Fig 5.

We can see in Examples 4 and 5, that KLMS method can be applied to solve higher-order differential equations. Also, comapring the results in Fig. 5 with the obtained results via UKLMS in [22] shows that the optimization-based KLMS has more accurate results.

5 Concluding remarks

In this paper, we proposed a new approach for solving ordinary differential equations by the use of kernel least

mean square (KLMS) algorithm and an optimization approach. 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 the accuracy and fast convergence of the new approach. Furthermore, implementing the KLMS algorithm is very simple. In optimization step, we can use other optimization techniques. Indeed, the results can be compared with the other methods, which are based on neural networks. In future, we are going to apply and extend the method to solve PDEs as well as integral differential equations.

References

1. Vapnik V (1995) The nature of statistical learning theory. Springer, New York
2. Girosi F, Jones M, Poggio T (1995) Regularization theory and neural networks architectures. Neural Comput 7(2):219–269
3. Scholkopf B, Smola A, Muller KR (1998) Nonlinear component analysis as a kernel eigenvalue problem. Neural Comput 10:1299–1319
4. Bach FR, Jordan MI (2002) Kernel independent component analysis. J Mach Learn Res 3:1–48
5. Pokharel P, Liu W, Principe JC (2007) Kernel lms. In: Proceedings of international conference on acoustics, speech and signal processing
6. Liu W, Pokharel P, Principe JC (2008) The kernel least mean square algorithm. IEEE Trans Signal Process 56:543–554
7. Gunduz A, Kwon J-P, Sanchez JC, Principe JC (2009) 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
8. Lagaris IE, Likas A, Fotiadis DI (1998) Artificial neural network for solving ordinary and partial differential equations. IEEE Trans Neural Netw 9:987–1000
9. Hornik K (1989) Multilayer feedforward networks are universal approximators. Neural Netw 2:359–366
10. Smaoui N, Al-Enezi S (2004) Modeling the dynamics of nonlinear partial differential equations using neural networks. J Comput Appl Math 170:27–58
11. Brause R (2003) Adaptive modeling of biochemical pathways. In: Proceedings of the 15th IEEE international conference on tools with artificial intelligence (ICTAI'03)
12. Hea S, Reif K, Unbehauen R (2000) Multilayer neural networks for solving a class of partial differential equations. Neural Netw 13:385–396
13. Manevitz L, Bitar A, Givoli D (2005) Neural network time series forecasting of finite-element mesh adaptation. Neurocomputing 63:447–463
14. Leephakpreeda T (2002) Novel determination of differential-equation solutions: universal approximation method. J Comput Appl Math 146:443–457
15. Malek A, Shekari Beidokhti R (2006) Numerical solution for high order differential equations using a hybrid neural network—optimization method. Appl Math Comput 183:260–271
16. Mai-Duy N, Tran-Cong T (2001) Numerical solution of differential equations using multi quadric radial basis function networks. Neural Netw 14:185–199
17. Jianyu L, Siwei L, Yingjian Q, Yaping H (2003) Numerical solution of elliptic partial differential equation using radial basis function neural networks. Neural Netw 16:729–734
18. Parisi DR, Mariani MC, Laborde MA (2003) Solving differential equations with unsupervised neural networks. Chem Eng Process 42:715–721
19. Luenberger DG (1984) Linear and nonlinear programming, 2nd edn. Addison-wesley, Boston
20. Effati S, Pakdaman M (2010) Artificial neural network approach for solving fuzzy differential equations. Inf Sci 180:1434–1457
21. Sadoghi Yazdi H, Pourreza R (2010) Unsupervised adaptive neural-fuzzy inference system for solving differential equations. Appl Soft Comput 10:267–275
22. Sadoghi Yazdi H, Pakdaman M, Modaghegh H (2011) Unsupervised kernel least mean square algorithm for solving ordinary differential equations. Neuromputing 74:2062–2071