LS-SVM APPROXIMATE SOLUTION FOR AFFINE NONLINEAR SYSTEMS WITH PARTIALLY UNKNOWN FUNCTIONS

GUOSHAN ZHANG, SHIWEI WANG AND YIMING WANG

Tianjin Key Laboratory of Process Measurement and Control
School of Electrical Engineering and Automation
Tianjin University, Tianjin, 300072, China

WANQUAN LIU

Department of Computing
Curtin University, WA, 6102, Australia

(Communicated by Song Wang)

Abstract. By using the Least Squares Support Vector Machines (LS-SVMs), we develop a numerical approach to find an approximate solution for affine nonlinear systems with partially unknown functions. This approach can obtain continuous and differential approximate solutions of the nonlinear differential equations, and can also identify the unknown nonlinear part through a set of measured data points. Technically, we first map the known part of the affine nonlinear systems into high dimensional feature spaces and derive the form of approximate solution. Then the original problem is formulated as an approximation problem via kernel trick with LS-SVMs. Furthermore, the approximation of the known part can be expressed via some linear equations with coefficient matrices as coupling square matrices, and the unknown part can be identified by its relationship to the known part and the approximate solution of affine nonlinear systems. Finally, several examples for different systems are presented to illustrate the validity of the proposed approach.

1. Introduction. Affine nonlinear system is an important type of nonlinear systems and it has numerous applications in mechanics, physics, biology, aerospace and various engineering fields. In recent decades, the investigation of the affine nonlinear systems has obtained substantial progress [5, 11]. Typically, an affine nonlinear system is generally described by

\[ \dot{x} = f(x) + g(x)u(t) \]  (1)

where \( f(x) \) and \( g(x) \) are nonlinear functions and \( u(t) \) is the control input. System (1) has much significance in analyzing and designing the nonlinear systems and thus finding its solutions becomes very important. If \( f(x), g(x)u(t) \) are known, we generally regard the affine nonlinear systems as general nonlinear Ordinary Differential Equations (ODEs), and one can obtain their numerical solutions for low-index problem through some typical numerical methods, such as Euler method, Runge-Kutta method [4], Adomian decomposition method [18]. A direct computational method by Chebyshev series [1] is developed to obtain solutions for high-index problem.
Some other techniques have developed to produce continuous approximation solutions. Linearization method [12] has been developed to yield piecewise analytic solutions. Many researchers also proposed some methods based on the neural network models; see [7–9, 13, 16]. In detail, [8] introduced Hopfield neural network models to solve the first order differential equations. [7] presented a method based on function approximation with feed forward neural networks. Although the neural networks have nice properties such as universal approximation, they are limited for broad applications due to some inherent drawbacks, such as the multiple local optimal solutions, the difficult choice for the number of hidden units and the danger of over fitting. For this reason, a new method based on the Least squares support vector machines (LS-SVMs) has been proposed to solve ODEs in [10]. In fact, SVMs are proposed in [17] as a newly developed technique based on statistical learning theory. It adopts the structure risk minimization principle, which avoids local minimum and can effectively solve the over-fitting problem. It guarantees good generalization capability as well as providing better predict accuracy. The LS-SVMs are an extension of SVMs and have good performance in function identification, estimation and robust regression [3, 14, 15]. The LS-SVMs can achieve the global solution by solving a set of linear equations instead of a quadratic programming problem for general SVMs, which allows the LS-SVMs to be faster than SVMs and this reduces the computational costs significantly. Further, the LS-SVMs regression requires model parameters far less than methods based on neural networks.

It should be noted that all methods mentioned above only solve the initial value and boundary value problems for differential equations. These methods could not be directly applicable to the nonlinear system with an unknown part. From the point view of control theory, it is necessary to consider the approximate solutions of nonlinear systems with unknown control part by utilizing some known sampled observations. The motivation for assuming that the control part is unknown in this paper is based on the following observations in practical systems: one is that the control sensors/actuators usually have some noises which can be modeled by an unknown part and the other reason is that the affine model is a popular model in many control systems.

In this paper, by considering better generalization capability of the LS-SVMs, we propose a new method based on LS-SVMs for finding approximate solutions in closed form (i.e., explicit, continuous and differential) to the affine nonlinear system (1) with known function \( f(x) \) and unknown function \( g(x)u(t) \) when a set of measured data points of states are known. And also, we can derive the approximate solutions for the unknown part by its relationship to the known part and the approximate solution of affine nonlinear systems.

In technique, we first divide the nonlinear part into known part and unknown part and then map the known part into a high dimensional feature space. The known part can then be expressed as some linear form whose coefficient matrices are coupling square matrixes. Finally, we utilize these feature maps to transform \( x(t), f(x) \) into the same feature space and then \( g(x)u(t) \) is expressed by its relationship to the approximation of \( f(x) \) and the derivative of \( x(t) \). Finally we solve them as a linear regression problem.

This paper is organized as follows. Section 2 presents a brief review of the least squares support vector regression problem. Some notations and preliminaries are given in section 3. In Section 4, we present our main results and discuss the selection
of tuning parameters. In section 5, some examples are given to illustrate the validity of the proposed method. Section 6 gives the conclusions.

2. LS-SVM function estimation. As presented in [14] for the LS-SVM as a function estimation, the equation used to represent the identified model is described by

$$y(x) = \omega^T \phi(x) + b$$

(2)

where $$\omega \in \mathbb{R}^{n_h}$$, $$b \in \mathbb{R}$$, $$\phi(\cdot) : \mathbb{R}^n \rightarrow \mathbb{R}^{n_h}$$ is a nonlinear function which maps the input space into a high dimensional feature space. For given a training set \( \{(x_k, y_k)\}_{k=1}^N \) with $$x_k \in \mathbb{R}^n$$, $$y_k \in \mathbb{R}$$ being the input data and output data respectively, we aim to find the parameters $$\omega$$ and $$b$$ for better approximation of the training data set with the model. The optimization problem of LS-SVM can be formulated as follows

$$\min_{\omega, b, e} J(\omega, e) = \frac{1}{2} \omega^T \omega + \frac{\gamma}{2} e^T e$$

s.t. $$y_k = \omega^T \phi(x_k) + b + e_k$$

(3)

where $$\gamma \in \mathbb{R}_+$$, $$b \in \mathbb{R}$$, and $$\phi(\cdot) : \mathbb{R}^n \rightarrow \mathbb{R}^{n_h}$$ is a feature space mapping and $$n_h$$ is the dimension of feature space.

In order to solve the optimization problem in equation (3), we define the following Lagrangian function:

$$L(\omega, b, e, \alpha) = J(\omega, e) - \sum_{k=1}^N \alpha_k [\omega^T \phi(x_k) + b + e_k - y_k]$$

(4)

where $$\alpha_k$$ is Lagrange multipliers as support vectors. With the Karush-Kuhn-Tucker conditions [17] one can calculate the partial derivatives of the Lagrangian:

$$\begin{cases}
\frac{\partial L}{\partial e_k} = 0 & \rightarrow e_k = \frac{\alpha_k}{\gamma}, \\
\frac{\partial L}{\partial \omega} = 0 & \rightarrow \omega = \sum_{i=1}^N \alpha_i \phi(x_i), \\
\frac{\partial L}{\partial b} = 0 & \rightarrow \sum_{i=1}^N \alpha_i = 0, \\
\frac{\partial L}{\partial \alpha_k} = 0 & \rightarrow \omega^T \phi(x_k) + b + e_k - y_k = 0.
\end{cases}$$

After eliminating $$\omega$$ and $$e_k$$, the solution can be given as in a linear form:

$$\begin{bmatrix}
\Omega + I_N / \gamma & 1_N \\
1_N^T & 0
\end{bmatrix}
\begin{bmatrix}
\alpha \\
b
\end{bmatrix} =
\begin{bmatrix}
y \\
0
\end{bmatrix}$$

(5)

where $$1_N = [1, ..., 1]^T \in \mathbb{R}^N$$, $$\alpha = [\alpha_1, \alpha_2, \cdots, \alpha_N]^T$$ and $$y = [y_1, y_2, \cdots, y_N]^T$$.

From application of the Mercer condition [17], one can obtain $$\Omega = K(x_i, x_j) = \phi^T(x_i) \phi(x_j), i, j = 1, \cdots, N$$.

Finally the identified model approximate solution can be represented as

$$\hat{y} = \sum_{k=1}^N \alpha_k K(x_k, x) + b$$

(6)

where $$\alpha_k$$ and $$b$$ are the solution to equation (5), $$K(x, x_k)$$ is kernel function which will be selected in advance as explained in next section.
3. Notations and preliminaries. The following notations and definitions will be used throughout this paper. By utilizing the Mercer’s condition [17], we adopt the defined method of differential operator in [10]. For example, if we choose the Gaussian kernel function as \( K(x, y) = \exp(-\frac{(x-y)^2}{2\sigma^2}) \), then the following relations hold,
\[
\nabla_0^0[K(x, y)] = K(x, y),
\nabla_0^1[K(x, y)] = \frac{\partial (\phi(x)^T \phi(y))}{\partial x} = \phi'(x)^T \phi(y) = -\frac{(x-y)}{\sigma^2} K(x, y), \tag{7}
\nabla_0^1[K(x, y)] = \frac{\partial (\phi(x)^T \phi(y))}{\partial y} = \phi(x)^T \phi'(y) = \frac{(x-y)}{\sigma^2} K(x, y).
\]

Remark 1. In order to keep the notations as precise as possible, we utilize the same feature map \( \phi(t) \) for all the states, i.e., \( \hat{x}(t) = \omega^T \phi(t) + b_1 \) and use the same feature map \( \varphi(x_i), i = 1, \cdots, m \), to transform \( f(x) \) to a linear combination form, and then \( g(x)u(t) \), which can be expressed by \( gu(x, t) \), can be written as an expression (9) in next section.

Nevertheless it is possible to use different mapping functions, as long as the corresponding kernel function satisfying the Mercer’s condition [17], and in this case one may have more model parameters.

4. Problem formulation and main results. Now we consider the following affine nonlinear systems:
\[
\dot{x} = f(x) + g(x)u(t), \quad t \in [t_i, t_f] \tag{8}
\]
where the state vector \( x = [x_1, \cdots, x_m]^T \in \mathbb{R}^m \) is assumed completely measurable in given data points, i.e., \( x(t_i) = [x_1(t_i), \cdots, x_m(t_i)]^T, \quad i = 1, \cdots, N. \) \( u(t) \in \mathbb{R}^p \) is the control input and \( f(x) \) is assumed to be known and \( g(x)u(t) \) is assumed to be unknown, and they are in the following form,
\[
f(x) = \begin{pmatrix}
f_1(x) \\
f_2(x) \\
\vdots \\
f_m(x)
\end{pmatrix}, \quad g(x) = \begin{pmatrix}
g_{11}(x) & \cdots & g_{1p}(x) \\
\vdots & \ddots & \vdots \\
g_{m1}(x) & \cdots & g_{mp}(x)
\end{pmatrix}
\]

Also we assume to have some observed samples for the state \( x(t) \) in some interval as stated below. The aim of this paper is to find an approximate solution \( \hat{x}(t) \) of (8) and identify the unknown part \( g(x)u(t) \) for system (8). In order to achieve these goals, we assume that a general approximate solution of (8) is of the form of \( \hat{x}(t) = \omega^T \phi(t) + b_1 \). Further, we also map \( f(x) \) into a high dimensional feature space and express it as linear combination forms. That is to say, \( f(x) \) and \( g(x)u(t) \) can be expressed as below.
\[
\hat{f}(x) = \begin{pmatrix}
\tilde{\omega}_{11}^T & \cdots & \tilde{\omega}_{1m}^T \\
\vdots & \ddots & \vdots \\
\tilde{\omega}_{m1}^T & \cdots & \tilde{\omega}_{mm}^T
\end{pmatrix} \begin{pmatrix}
\varphi(x_1) \\
\varphi(x_2) \\
\vdots \\
\varphi(x_m)
\end{pmatrix} + \begin{pmatrix}
\tilde{b}_1 \\
\tilde{b}_2 \\
\vdots \\
\tilde{b}_m
\end{pmatrix} \tag{9}
\]
\[
\hat{g}(x)u(t) = \hat{g}u(x, t) = \dot{\hat{x}} - \hat{f}(\hat{x})
\]
where $[\bar{\omega}l]_{m \times m}$ are the coupled matrices and $\omega_t, \bar{\omega}l_k, b_l$ are parameters of the model that have to be determined.

In order to obtain the optimal value of these parameters, the collocation method [6] is used here in which we assume that $x(t)$ is measured on a discretization of the interval $[t_i, t_f]$ in a set of collocation points $\gamma = \{t_m = t_1 < t_2 < \cdots < t_N = t_f\}$. Therefore, the adjustable parameters $\omega_t, \bar{\omega}l_k, b_l$; $k = 1, \cdots, m$, can be found by solving the following optimization problem:

$$\begin{align*}
\min_{\hat{x}, \eta} & \frac{1}{2} \sum_{i=1}^{N} [[\hat{x}' - \hat{f}(X) - g(X)u](t_i)]^2 \\
\text{s.t.} & \hat{x}(t_i) = X(t_i) + \xi(t_i) 
\end{align*}$$

(10)

where $N$ is the number of collocation points (which is equal to the number of training points) used to undertake the learning process. With LS-SVMs, we utilize the nonlinear feature map $\phi(x)$ to map the training data into a feature space and transform the nonlinear problem into a linear problem. Therefore, in order to solve the problem (10), we make use of two nonlinear feature map functions $\phi(t), \varphi(x)$ to map $X(t), f(x)$ into their feature spaces and transform the original problem into a linear problem which can be solved in the framework of LS-SVMs. For notational convenience, let us list the following notations which are used in the following sections. $K_1, K_2$ represent two different types of kernel functions denoted as follows:

$$\begin{align*}
\nabla^n_m [K_1(u, v)] &= \nabla^n_m [K_1(t, s)]|_{t=u, s=v} \\
\Omega^n_m (t, v) &= \nabla^n_m [K_1(t, s)]|_{t=t, s=v} = \frac{\partial^n+m}{\partial t^n \partial s^m} |_{t=t, s=s} K_1(t, s) \\
\Omega^n_0 (t, v) &= \nabla^n_0 [K_1(t, s)]|_{t=t, s=v} = K_1(t, s) |_{t=t, s=v} \\
\Omega^n_0 (x_k^l, x_k^l) &= \nabla^n_0 [K_2(t, s)]|_{t=x_k^l, s=x_k^l} = K_2(t, s) |_{t=x_k^l, s=x_k^l} \varphi(t)^T \varphi(s) |_{t=x_k^l, s=x_k^l}
\end{align*}$$

where $\Omega^n_m (t, v)$ denotes the $(i, v)-$th entry of matrix $\Omega^n_m$.

**Remark 2.** The LS-SVM is used for dynamic data in this paper and it is usually used for static data in machine learning [2]. One can find that the objective function (10) includes the derivative item of state, which is different from the case for static data. This difference is that the derivative of the kernel function is introduced in the solution.

Now we consider optimization problem (10). Let $\hat{x}_l(t) = \omega^T l \phi(t) + b_l$, and now we introduce additional unknowns $gu(x^l, t_i)$, $l = 1, \cdots, m; i = 1, \cdots, N$; into the optimization goal. Then let $gu(x^l, t_i)$ optimally approximate the exact value in discrete points by optimizing the goal. Thus, $gu(x^l, t_i)$ together with the parameters of framework of LS-SVMs constructs the following optimization problem.

$$\begin{align*}
\min_{\hat{x}, \eta} & \frac{1}{2} \sum_{l=1}^{m} \omega^T_l \omega_l + \frac{C}{2} \sum_{l=1}^{m} \epsilon^T_l \epsilon_l + \frac{1}{2} \sum_{l=1}^{m} gu^T gu_l \\
& + \frac{1}{2} \sum_{l=1}^{m} \sum_{k=1}^{m} \bar{\omega}l^T_k \bar{\omega}l_k + \frac{\bar{\epsilon}}{2} \sum_{l=1}^{m} \xi^T_l \xi_l + \frac{\bar{\epsilon}}{2} \sum_{l=1}^{m} \bar{\epsilon}^T_l \bar{\epsilon}_l
\end{align*}$$

(11)
This constraint optimization problem (11) can be solved by introducing the Lagrange multipliers $\alpha_l^i, \beta_l^i, \gamma_l^i \geq 0$ associated with the following Lagrangian function:

$$
L (\omega_l, \omega_{lk}, c_l^i, d_l^i, e_l^i, f_l^i, \alpha_l^i, \beta_l^i, \gamma_l^i, gu_l) = \frac{1}{2} \sum_{l=1}^{m} \omega_l^T \omega_l + \frac{C}{2} \sum_{l=1}^{m} c_l^T c_l \\
+ \frac{1}{2} \sum_{l=1}^{m} g_{ul}^T g_{ul} + \frac{1}{2} \sum_{k=1}^{m} \sum_{l=1}^{m} \omega_{lk}^T \omega_{lk} - \sum_{l=1}^{m} \sum_{i=1}^{N} \beta_l^i (\omega_l^T \phi (t_l) + b_l - x_l^i - \xi_l^i) \\
+ \frac{\epsilon}{2} \sum_{l=1}^{m} \sum_{k=1}^{N} \gamma_l^i \left( f_l (x^i) - \bar{b}_l - \bar{\xi}_l^i - \sum_{k=1}^{m} \omega_{lk} (x_k^i) \right) \\
- \sum_{l=1}^{m} \sum_{i=1}^{N} \alpha_l^i \left( \omega_l^T \phi' (t_l) - (\bar{b}_l + \bar{c}_l^i) - \sum_{k=1}^{m} \omega_{lk} \varphi (x_k^i) - gu_l (x^i, t_l) \right)
$$
where \( i = 1, \ldots, N, l = 1, \ldots, m \). Then the Karush-Kuhn-Tucker (KKT) optimality conditions can be derived as follows:

\[
\frac{\partial L}{\partial e_i^l} = 0 \rightarrow e_i^l = -\frac{\alpha_i^l}{C}, l = 1, \ldots, m; i = 1, \ldots, N;
\]

\[
\frac{\partial L}{\partial \xi_i^l} = 0 \rightarrow \xi_i^l = -\frac{\beta_i^l}{\bar{e}}, l = 1, \ldots, m; i = 1, \ldots, N;
\]

\[
\frac{\partial L}{\partial b_l} = 0 \rightarrow \sum_{i=1}^N \alpha_i^l + \sum_{i=1}^N \gamma_i^l = 0, l = 1, \ldots, m; i = 1, \ldots, N;
\]

\[
\frac{\partial L}{\partial \omega_i} = 0 \rightarrow \omega_i = \sum_{i=1}^N (\alpha_i^l \varphi' (t_i) + \beta_i^l \phi (t_i)), l = 1, \ldots, m;
\]

\[
\frac{\partial L}{\partial \gamma_i^l} = 0 \rightarrow \bar{b} + \bar{e} + \varepsilon = 0, l = 1, \ldots, m; i = 1, \ldots, N;
\]

\[
\frac{\partial L}{\partial \varphi (x^i)} = 0 \rightarrow f_i (x^i) = \sum_{k=1}^m \tilde{\omega}_{ik} \varphi (x^i_k) + \tilde{b}_i + \tilde{e}_i, l = 1, \ldots, m; i = 1, \ldots, N;
\]

\[
\frac{\partial L}{\partial g_u} = 0 \rightarrow 0 = \alpha_i^l + gu_l (x^i, t_i), l = 1, \ldots, m; i = 1, \ldots, N;
\]

\[
\frac{\partial L}{\partial \varphi (x^i)} = 0 \rightarrow \omega_i \varphi (t_i) + b_i - x_i - \xi_i^l = 0, l = 1, \ldots, m; i = 1, \ldots, N;
\]

\[
\frac{\partial L}{\partial \alpha_i^l} = 0 \rightarrow \omega_i \varphi' (t_i) - \sum_{k=1}^m \tilde{\omega}_{ik} \varphi (x^i_k) - gu_l (x^i, t_i) - \bar{b}_i - \varepsilon_i^l = 0,
\]

\[
l = 1, \ldots, m; i = 1, \ldots, N.
\]

After eliminating \( \omega_i, \tilde{\omega}_{ik}, \{e_i^l\}_{i=1}^N, \{\varepsilon_i^l\}_{i=1}^N, \{\xi_i^l\}_{i=1}^N \) for \( l = 1, \ldots, m \) from the corresponding KKT [17] optimality conditions, we can yield the following set of equations.

\[
\sum_{i=1}^N \alpha_i^l \Omega_1^l (t_i, t_j) + \sum_{k=1}^m \sum_{i=1}^N \alpha_i^l \varphi (x_k^i) + \sum_{i=1}^N \beta_i^l \Omega_0^l (t_i, t_j)
\]

\[
- gu_l (x^i, t_i) - \bar{b}_i + \frac{\alpha_i^l}{C} = 0, l = 1, \ldots, m; j = 1, \ldots, N;
\]

\[
\sum_{i=1}^N \alpha_i^l + \sum_{i=1}^N \gamma_i^l = 0, l = 1, \ldots, m;
\]

\[
- \sum_{i=1}^N \beta_i^l = 0, l = 1, \ldots, m; j = 1, \ldots, N;
\]
\[
\sum_{i=1}^{N} \alpha_i^0 \Omega_0^0(t_i, t_j) + \sum_{i=1}^{N} \beta_i^0 \Omega_0^0(t_i, t_j) + b_j + \frac{\beta_j^0}{\varepsilon} = x_i^l, l = 1, \ldots, m; j = 1, \ldots, N;
\]

\[
- \sum_{k=1}^{m} \sum_{v=1}^{N} (\alpha_i^v + \gamma_i^v) \bar{\Omega}_0^0(x_k^v, x_k^1) + \bar{b} - \frac{\gamma_i^v}{\tau} = f_i(x^v), l = 1, \ldots, m; i = 1, \ldots, N;
\]

\[0 = \alpha_i^v + gu(x^v, t_k), l = 1, \ldots, m; i = 1, \ldots, N.\]

Then we can write these equations in a compact matrix form as follows.

\[
\begin{bmatrix}
K & H & S & O & -I & -E \\
-H^T & -T & Z & O & I & Z \\
S^T & Z & Q & I & O & Z \\
O^T & O^T & -I^T & 0_{m \times m} & 0_{m \times m} & O^T \\
I^T & I^T & O^T & 0_{m \times m} & 0_{m \times m} & O^T \\
E & Z & Z & O & O & E
\end{bmatrix}
\begin{bmatrix}
\alpha \\
\gamma \\
\beta \\
b \\
\bar{b} \\
gu
\end{bmatrix}
= 
\begin{bmatrix}
0 \\
f \\
x \\
b \\
0 \\
0
\end{bmatrix}
\]

with

\[O = [0, \ldots, 0]_{mN \times m}, I = [1, \ldots, 1]_{mN \times m},\]

\[Z = \begin{pmatrix}
0 & \ldots & 0 \\
\vdots & \ddots & \vdots \\
0 & \ldots & 0
\end{pmatrix}_{mN \times mN}, E = \begin{pmatrix}
1 & \ldots & 0 \\
\vdots & \ddots & \vdots \\
0 & \ldots & 1
\end{pmatrix}_{mN \times mN},\]

\[b = [b_1, \ldots, b_m]^T, \bar{b} = [\bar{b}_1, \ldots, \bar{b}_m]^T,\]

\[\gamma = \begin{bmatrix}
\gamma_1 \\
\gamma_2 \\
\vdots \\
\gamma_m
\end{bmatrix} = [\gamma_1^1; \ldots; \gamma_1^N; \ldots; \gamma_m^1; \ldots; \gamma_m^N] \in \mathbb{R}^{mN},\]

\[\alpha = \begin{bmatrix}
\alpha_1 \\
\alpha_2 \\
\vdots \\
\alpha_m
\end{bmatrix} = [\alpha_1^1; \ldots; \alpha_1^N; \ldots; \alpha_m^1; \ldots; \alpha_m^N] \in \mathbb{R}^{mN},\]

\[\beta = \begin{bmatrix}
\beta_1 \\
\beta_2 \\
\vdots \\
\beta_m
\end{bmatrix} = [\beta_1^1; \ldots; \beta_1^N; \ldots; \beta_m^1; \ldots; \beta_m^N] \in \mathbb{R}^{mN},\]

\[gu = \begin{bmatrix}
gu_1 \\
gu_2 \\
\vdots \\
gu_m
\end{bmatrix} = [gu_1(x^1, t_1); \ldots; gu_1(x^N, t_N); \ldots; gu_m(x^1, t_1); \ldots; gu_m(x^N, t_N)] \in \mathbb{R}^{mN}.\]
In order to write the final solution in an elegant form, let $\nabla_{ij} = \Omega_0^1(t_i, t_j)$, we have

$$S_{11} = \begin{bmatrix} \nabla_{11} & \cdots & \nabla_{1N} \\ \vdots & \ddots & \vdots \\ \nabla_{N1} & \cdots & \nabla_{NN} \end{bmatrix}, \quad S = \begin{bmatrix} S_{11} & 0 & \cdots & 0 \\ 0 & S_{11} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & S_{11} \end{bmatrix}_{mN \times mN}$$

with $\Delta_{ij} = \Omega_1^1(t_i, t_j) + \sum_{k=1}^{m} \Omega_0^0(x_k^i, x_k^j)$, then

$$K_{11} = \begin{bmatrix} \Delta_{11} + \frac{1}{\varepsilon} & \cdots & \Delta_{1N} \\ \vdots & \ddots & \vdots \\ \Delta_{N1} & \cdots & \Delta_{NN} + \frac{1}{\varepsilon} \end{bmatrix}_{N \times N}, \quad K = \begin{bmatrix} K_{11} & 0 \\ 0 & K_{11} \end{bmatrix}_{mN \times mN}$$

Let $\mathcal{N}_{ij} = \Omega_0^0(t_i, t_j)$, then

$$Q_{11} = \begin{bmatrix} \mathcal{N}_{11} + \frac{1}{\varepsilon} & \cdots & \mathcal{N}_{1N} \\ \vdots & \ddots & \vdots \\ \mathcal{N}_{N1} & \cdots & \mathcal{N}_{NN} + \frac{1}{\varepsilon} \end{bmatrix}, \quad Q = \begin{bmatrix} Q_{11} & 0 & \cdots & 0 \\ 0 & Q_{11} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & Q_{11} \end{bmatrix}_{mN \times mN}$$

and let $\Lambda = \sum_{k=1}^{m} \Omega_0^0(x_k^i, x_k^j)$, then

$$H_{11} = \begin{bmatrix} \Lambda_{11} & \cdots & \Lambda_{1N} \\ \vdots & \ddots & \vdots \\ \Lambda_{N1} & \cdots & \Lambda_{NN} \end{bmatrix}_{N \times N}, \quad H = \begin{bmatrix} H_{11} & 0 \\ 0 & H_{11} \end{bmatrix}_{mN \times mN}$$

$$T_{11} = \begin{bmatrix} \Lambda_{11} + \frac{1}{\varepsilon} & \cdots & \Lambda_{1N} \\ \vdots & \ddots & \vdots \\ \Lambda_{N1} & \cdots & \Lambda_{NN} + \frac{1}{\varepsilon} \end{bmatrix}_{N \times N}, \quad T = \begin{bmatrix} T_{11} & 0 \\ 0 & T_{11} \end{bmatrix}_{mN \times mN}$$

$$f = [f_1(x^1); \cdots; f_1(x^N); \cdots; f_m(x^1); \cdots; f_m(x^N)] \in R^{mN}$$

$$X = [x_1^1; \cdots; x_1^N; \cdots; x_m^1; \cdots; x_m^N] \in R^{mN}$$

Finally, the approximate solution is

$$\hat{x}_i(t) = \sum_{i=1}^{N} \alpha_i^l \Omega_1^l(t_i, t) + \sum_{i=1}^{N} \beta_i^l \Omega_0^0(x_k^i, \hat{x}_k) + \bar{b}_l, l = 1, \cdots, m; \quad (13)$$

where $\Omega$ is a kernel function. Furthermore, we can also identify $g(x)u(t)$, and the approximation of $f(x)$ can also be obtained as follows.

$$\hat{f}_i (x) = - \sum_{k=1}^{m} \sum_{i=1}^{N} (\alpha_i^l + \gamma_i^l) \bar{\Omega}_0^0(x_k^i, \hat{x}_k) + \bar{b}_l;$$

$$\hat{g}u_i (x, t) = \sum_{i=1}^{N} \alpha_i^l \Omega_1^l(t_i, t) + \sum_{i=1}^{N} \beta_i^l \Omega_0^0(t_i, t)$$

$$+ \sum_{k=1}^{m} \sum_{i=1}^{N} (\alpha_i^l + \gamma_i^l) \bar{\Omega}_0^0(x_k^i, \hat{x}_k) - \bar{b}_l. \quad (14)$$
It should be noted that approximate solution in (13) can be obtained by solving the linear equation (12). This is the advantage of LS-SVM. Once we obtain the approximated solution, $g(x)u(t)$ can be identified easily with (14). The efficiency of the LS-SVMs model depends on the choice of the tuning parameters. All examples of this paper use the Gaussian functions as the kernel functions. Therefore a model is determined by the regularization parameter $\gamma$ and the kernel bandwidth $\sigma$. We obtain these parameters by using the fast leave-one-out cross-validation [2].

Remark 3. In order to make the tuning parameters as simple as possible, we use one common mapping function in the three groups of parameters. The results show that this simple selection will not affect the performance significantly.

5. Simulation results. In this section, four examples are used to illustrate the validity and viability of the proposed method for solving the affine nonlinear systems. In these experiments, the observed sampling data points are obtained from the analytic solution or MATLAB built-in solver ode45 whose accuracy is much reasonable. The accuracy of an approximate solution is measured by means of mean square error (MSE) defined as follows:

$$MSE_{test} = \frac{1}{M} \sum_{i=1}^{M} (x(t_i) - \hat{x}(t_i))^2$$

where $M$ is the number test points. In all the examples, $M$ is set to 100 points in the given domain. In order to illustrate the proposed method more precisely, we also calculate the real time difference $E(t) = x(t) - \hat{x}(t)$ if possible. Further, we also consider points outside the training interval in order to obtain an estimate of the exterior performance of the obtained solution. As to the known and unknown nonlinear parts, i.e., $f(x), g(x)u(t)$, we identify, approximate and compare them to the exact nonlinear functions $f(x), g(x)u(t)$.

Remark 4. Note that $f(x)$ and $g(x)u(t)$ are both the functions of variable $t$, because $x$ is function of $t$. So in simulation, $f(x), g(x)u(t)$ are regarded as the functions of $t$, i.e., $f(t), g(t)$.

Example 1. Consider the following Riccati equation, i.e., Example 3 in [1].

$$\dot{x}(t) = 2x(t) - x^2(t) + 1, 0 \leq t \leq 2;$$

Its exact solution is

$$x(t) = 1 + \sqrt{2} \tanh \left[ \sqrt{2}t + \frac{1}{2} \ln \left( \frac{\sqrt{2} - 1}{\sqrt{2} + 1} \right) \right]$$

If we assume that $f(x) = 2x$ is known, and $g(x)u(t) = -x^2 + 1$ is unknown. The approximate solution obtained by the proposed method is compared with the exact solution (see Figure 1) and the identification $g(x)u(t)$ and approximation of $f(x)$ are compared with the exact nonlinear parts for exact solutions (see Figure 2). The results with different number of training samples are recorded in Table 1, from which, it is apparent that the approximate solution converges to the exact solution as $N$ increases, where $N$ is the number of training samples.

Example 2. Consider the following equation which has no analytic solution, and the example is Problem 5.3 in [10]:
Table 1. Numerical results of the proposed method for solving Example 1 on time interval [0, 2], with N number of collocation points.

| \( N \) | \( \text{MSE}_{\text{test}} \) |
|---|---|
| 20 | \( 1.15 \times 10^{-6} \) |
| 40 | \( 4.14 \times 10^{-7} \) |
| 60 | \( 6.70 \times 10^{-12} \) |
| 80 | \( 4.85 \times 10^{-12} \) |

\[
\dot{y}(t) = y(t)^2 + t^2,
\]

with \( f(y) = y^2 \) and \( g(y)u(t) = t^2 \), we can obtain the training data by MATLAB built-in solver ode45 with \( y(0) = 1 \). This problem is solved for \( t \in [0, 0.5] \) and the approximate solution obtained by the proposed method is compared with the solution obtained by MATLAB built-in solver ode45 in Figure 3. The identification of \( g(x)u(t) \) and approximation of \( f(x) \) are compared with the exact nonlinear parts.
for points obtained by MATLAB built-in solver ode45 (see Figure 4). The obtained results with different number of training points are tabulated in Table 2. The proposed method shows a slightly better performance in comparison with the described method in [10], when the number of training points is twenty. Note that the subroutine DSolve of MATLAB 7.0 failed to find the analytic solution for the above equation.

Table 2. Numerical results of the proposed method for solving Example 2 on time interval [0, 0.5], with N number of collocation points.

| N   | \( \text{MSE}_{\text{test}} \) |
|-----|--------------------------------|
| 20  | 5.27e-09                       |
| 40  | 2.17e-10                       |
| 60  | 1.26e-11                       |
| 80  | 1.80e-12                       |

Figure 3. Obtained approximate solution and model errors on the interval [0, 0.8] for Example 2, when 50 equidistant mesh points on the interval [0, 0.5] are used for training.

 Figure 4. Obtained approximate \( f(t) \), \( gu(t) \) on the interval [0, 0.5] for Example 2, when 50 equidistant mesh points on the interval [0, 0.5] are used for training.

Example 3. Consider a system of index-3 defined as follows:

\[
\dot{x}(t) = f(x) + g(x)u(t)
\]

where

\[
f(x) = \begin{bmatrix} x_2 x_3 \\ 0 \\ -x_1 x_2 \end{bmatrix}, \quad g(x) = \begin{bmatrix} x_1 & 0 & 0 \\ 0 & -x_2 & 0 \\ 0 & 0 & -x_3 \end{bmatrix} \quad \text{and} \quad u(t) = \begin{bmatrix} \sin(t) & t & e^{-t} \end{bmatrix}^T.
\]

We obtain the training data by using the MATLAB built-in solver ode45 with
\[ x(0) = [1 \ 3 \ 2]^T. \] This problem is solved for \( t \in [0, 2] \) and the approximate solution obtained by the proposed method is compared with the solution obtained by MATLAB built-in solver ode45 in Figure 5. The identification of \( g(x)u(t) \) and approximation of \( f(x) \) are compared with the exact nonlinear parts of the solution obtained by MATLAB built-in ode45 (see Figure 6). The obtained results with different number of training points are tabulated in Table 3. Note that the subroutine DSolve of MATLAB7.0 failed to find the analytical solution for the above equation.

**Example 3.** Consider a system of index-2 of Problem 4 discussed in [7] and described by:

\[
\dot{x}(t) = f(x) + g(x)u(t)
\]

where

\[
f(x) = \begin{pmatrix} x_1^2 + x_2 \\ x_1x_2 \end{pmatrix} \quad \text{and} \quad g(x)u(t) = \begin{pmatrix} \cos(t) - \sin^2(t) - t^2 - 1 \\ 2t - \sin(t) - t^2 \sin(t) \end{pmatrix}
\]

with \( x(0) = [0 \ 1]^T. \) The exact solution is given by \( x_1(t) = \sin(t), \ x_2(t) = 1 + t^2. \) The

| \( N \) | \( MSE_{test} \) |
|---|---|
| \( x_1 \) | \( x_2 \) | \( x_3 \) |
| 20 | 2.23e-05 | 1.57e-06 | 2.16e-05 |
| 40 | 7.90e-09 | 1.63e-07 | 2.52e-06 |
| 60 | 1.19e-10 | 4.96e-11 | 5.22e-09 |
| 80 | 9.66e-11 | 1.32e-13 | 1.10e-10 |

**Figure 5.** Obtained approximate solution and model errors on the interval \([0, 3]\) for Example 3, when 50 equidistant mesh points on the interval \([0, 2]\) are used for training.

**Example 4.** Consider a system of index-2 of Problem 4 discussed in [7] and described by:

\[
\dot{x}(t) = f(x) + g(x)u(t)
\]

where

\[
f(x) = \begin{pmatrix} x_1^2 + x_2 \\ x_1x_2 \end{pmatrix} \quad \text{and} \quad g(x)u(t) = \begin{pmatrix} \cos(t) - \sin^2(t) - t^2 - 1 \\ 2t - \sin(t) - t^2 \sin(t) \end{pmatrix}
\]

with \( x(0) = [0 \ 1]^T. \) The exact solution is given by \( x_1(t) = \sin(t), \ x_2(t) = 1 + t^2. \) The
problem is solved on domain $t \in [0, 3]$ for different $N$ (the number of training points) values. The approximate solution obtained by the proposed method is compared with the exact solution (see Figure 7). The identification of $g(x)u(t)$ and approximation of $f(x)$ are compared with the exact nonlinear parts for exact solution (see Figure 8) and the results are recorded in Table 4, from which, it is apparent that as $N$ increases, the solution converges to the true solution.

**Table 4.** Numerical results of the proposed method for solving Example 4 on time interval $[0, 3]$, with $N$ number of collocation points.

| $N$ | $x_1$ | $x_2$ |
|-----|-------|-------|
| 20  | $1.84e-07$ | $5.91e-12$ |
| 40  | $3.90e-08$ | $3.54e-13$ |
| 60  | $2.49e-10$ | $8.67e-14$ |
| 80  | $4.77e-14$ | $4.07e-14$ |

We can see from all these examples that the approximate solutions can approach to the original solution quite well in the given interval. Also the unknown function part can be identified properly. On the other hand, the performance will become worse without training data outside the interval. This shows that the training data points are very important.

6. **Conclusions.** In this paper, a new method has been presented for solving affine nonlinear systems with partially unknown functions by using the LS-SVMs and a set of data points. It can provide accurate and differential solutions in a closed analytic form. The results of simulation show that the method can obtain the desired accuracy with only a few number of collocation points. The continuous approximate solution can be easily evaluated at any point within the interval. The method can be effectively applied to solve any index partially unknown affine nonlinear systems. As the affine nonlinear differential equations are partially unknown, this approach can offer new ideas to nonlinear system identification and can also be used to design the controller if the desired characteristics for the observed sampling data are given.

Though the idea in this paper can be used for obtaining the approximate solutions of general nonlinear systems, it will be different when one uses this idea for...
identifying some unknown parts in nonlinear systems. In the future, we will investigate control and identification problems for different types of nonlinear systems with applications.

Acknowledgments. The authors would like to thank the reviewers and Associate Editor for their helpful comments and constructive suggestions. This project is supported by NSF grant No. 61074088

REFERENCES

[1] A. Akyüzy-Dascioglu and H. Cerdik-Yaslan, The solution of high-order nonlinear ordinary differential equations by Chebyshev Series, Applied Mathematics and Computation, 217 (2011), 5658–5666.
[2] S. J. An, W. Q. Liu and S. Venkatesh, Fast Exact cross-validation of least squares support vector machines, Pattern Recognition, 40 (2007), 2154–2162.

[3] T. Falck, K. Pelckmans, J. A. K. Suykens and B. De Moor, Identification of Wiener-Hammerstein Systems using LS-SVMs, 15th IFAC Symposium on System Identification, Saint-Malo, France, 2009.

[4] Z. Guan and J. F. Lu, Basic of Numerical Analysis(Chinese), 2nd edition, Higher Education Press, Beijing, 2010.

[5] A. Isidori, Nonlinear Control Systems: An Introduction, 3rd edition, Springer-Verlag, London, 1995.

[6] D. R. Kincaid and E. W. Cheney, Numerical Analysis: Mathematics of Scientific Computing, 3rd edition, Brooks/Cole, Pacific Grove, CA, 2002

[7] I. E. Lagaris, A. Likas and D. I. Fotiadis, Artificial neural networks for solving ordinary and partial differential equations, IEEE Transactions on Neural Networks, 9 (1998), 987–1000.

[8] H. Lee and I. S. Kang, Neural algorithm for solving differential equations, Journal of Computational Physics, 91 (1990), 110–131.

[9] K. S. McFall and J. R. Mahan, Artificial neural network method for solution of boundary value problems with exact satisfaction of arbitrary boundary conditions, IEEE Transactions on Neural Networks, 20 (2009), 1221–1233.

[10] S. Mehrkanoon, T. Falck and J. A. K. Suykens, Approximate solutions to ordinary differential equations using least squares support vector machines, IEEE Trans. on Neural Networks and Learning Systems, 23 (2012), 1356–1367.

[11] M. Popescu, On minimum quadratic functional control of affine nonlinear systems, Nonlinear Analysis: Theory, Methods & Applications, 56 (2004), 1165–1173.

[12] J. I. Ramos, Linearization techniques for singular initial-value problems of ordinary differential equations, Applied Mathematics and Computation, 161 (2005), 525–542.

[13] P. Ramuhalli, L. Udpa and S. S. Udpa, Finite-element neural networks for solving differential equations, IEEE Transactions on Neural Networks, 16 (2005), 1381–1392.

[14] J. A. K. Suykens, T. V. Gestel, J. Brabanter, B. D. Moor and J. Vandewalle, Least Squares Support Vector Machines, 1st edition, World Scientific, Singapore, 2002.

[15] J. A. K. Suykens, J. Vandewalle and B. D. Moor, Optimal control by least squares support vector machines, Neural Networks, 14 (2001), 23–35.

[16] I. G. Tsoulos, D. Gavrilis and E. Glavas, Solving differential equations with constructed neural networks, Neurocomputing, 72 (2009), 2385–2391.

[17] V. Vapnik, The Nature of Statistical Learning Theory, 1st edition, Springer-Verlag, New York, 1995.

[18] A. M. Wazwaz, A new method for solving initial value problems in second-order ordinary differential equations, Applied Mathematics and Computation, 128 (2002), 45–57.

Received January 2013; 1st revision April 2013; 2nd revision June 2013.

E-mail address: zhanggs@tju.edu.cn
E-mail address: wsansree@163.com
E-mail address: 34486460@qq.com
E-mail address: w.liu@curtin.edu.au