Approximate Solutions to Ordinary Differential Equations Using Least Squares Support Vector Machines

Siamak Mehrkanoon, Tillmann Falck, and Johan A. K. Suykens, *Senior Member, IEEE*

***Abstract*— In this paper, a new approach based on least squares support vector machines (LS-SVMs) is proposed for solving linear and nonlinear ordinary differential equations (ODEs). The approximate solution is presented in closed form by means of LS-SVMs, whose parameters are adjusted to minimize an appropriate error function. For the linear and nonlinear cases, these parameters are obtained by solving a system of linear and nonlinear equations, respectively. The method is well suited to solving mildly stiff, nonstiff, and singular ODEs with initial and boundary conditions. Numerical results demonstrate the efficiency of the proposed method over existing methods.**

***Index Terms*— Closed-form approximate solution, collocation method, least squares support vector machines (LS-SVMs), ordinary differential equations (ODEs).**

I. Introduction

DIFFERENTIAL equations can be found in the mathematical formulation of physical phenomena in a wide variety of applications, especially in science and engineering [1], [2]. Depending upon the form of the boundary conditions to be satisfied by the solution, problems involving ordinary differential equations (ODEs) can be divided into two main categories, namely initial value problems (IVPs) and boundary value problems (BVPs). Analytic solutions for these problems are not generally available and hence numerical methods must be applied.

Manuscript received July 7, 2011; revised May 14, 2012; accepted May 15, 2012. Date of publication June 22, 2012; date of current version August 1, 2012. This work was supported in part by the Research Council KUL GOA/11/05 Ambiorics, GOA/10/09 MaNet, CoE EF/05/006 Optimization in Engineering (OPTEC), IOF-SCORES4CHEM, several Ph.D and Post-Doctoral and Fellowship Grants, the Flemish Government (FWO: Ph.D. and Post-Doctoral Grants), under Project G0226.06 (cooperative systems and optimization), Project G0321.06 (Tensors), Project G.0302.07 (SVM/Kernel), Project G.0320.08 (convex MPC), Project G.0558.08 (Robust MHE), Project G.0557.08 (Glycemia2), Project G.0588.09 (Brain-Machine), Project G.0377.12 (structured models) research communities (WOG: ICCoS, ANMMM, MLDM), Project G.0377.09 (Mechatronics MPC) IWT: Ph.D. Grants, Eureka-Flite+, SBO LeCoPro, SBO Climaqs, SBO POM, O&O-Dsquare, the Belgian Federal Science Policy Office: IUAP P6/04 (DYSCO, Dynamical systems, control and optimization, 2007–2011), EU (ERNSI), FP7-HD-MPC (INFSO-ICT-223854), COST intelliCIS, FP7-EMBOCON (ICT-248940), Contract Research (AMINAL), Helmholtz (viCERP), ACCM, Bauknecht, Hoerbiger, and the ERC under Advanced Grant A-DATADRIVE-B.

The authors are with the Department of Electrical Engineering ESAT-SCD-SISTA, Katholieke Universiteit Leuven, Leuven B-3001, Belgium (e-mail: siamak.mehrkanoon@esat.kuleuven.be; tillmann.falck@esat.kuleuven.be; johan.suykens@esat.kuleuven.be).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2012.2202126

Many methods have been developed for solving IVPs of ODEs, such as Runge–Kutta, finite difference, predictor-corrector, and collocation methods [3]–[6]. Generally speaking, numerical methods for approximating the solution of the BVPs fall into two classes: the difference methods (e.g., shooting method) and weighted residual or series methods. In the shooting method, one tries to reduce the problem to IVPs by providing a sufficiently good approximation of the derivative values at the initial point.

Concerning higher order ODEs, the most common approach is the reduction of the problem to a system of first-order differential equations and then solve the system by employing one of the available methods, which notably has been studied in the literature, see [4], [7], and [8]. However, as some authors have remarked, this approach wastes a lot of computer time and human effort [9], [10].

Most of the traditional numerical methods provide the solution, in the form of an array, at specific preassigned mesh points in the domain (discrete solution) and they need an additional interpolation procedure to yield the solution for the whole domain. In order to have an accurate solution, one either has to increase the order of the method or decrease the step size. This, however, increases the computational cost.

To overcome these drawbacks, attempts have been made to develop new approaches to not only solve the higher order ODEs directly without reducing it to a system of first-order differential equations, but also to provide the approximate solution in closed form (i.e., continuous and differentiable) thereby avoiding an extra interpolation procedure. One of these classes of methods is based on the use of neural network models, see [11]–[17]. Lee and Kang [12] used Hopfield neural networks models to solve first-order differential equations. The authors in [18] introduced a method based on feedforward neural networks to solve ordinary and partial differential equations. In that model, the approximate solution was chosen such that it, by construction, satisfied the supplementary conditions. Therefore, the model function was expressed as a sum of two terms. The first term, which contains no adjustable parameters, satisfied the initial/boundary conditions and the second term involved a feedforward neural network to be trained. An unsupervised kernel least mean square algorithm was developed for solving ODEs in [19].

Despite the fact that the classical neural networks have nice properties, such as universal approximation, they still suffer from having two persistent drawbacks. The first problem is the

2162–237X/$31.00 © 2012 IEEE

existence of many local minima solutions. The second problem is how to choose the number of hidden units.

Support vector machines (SVMs) are a powerful method for solving pattern recognition and function estimation problems [20], [21]. In this method, one maps data into a high-dimensional feature space and there solves a linear regression problem. It leads to solving quadratic programming problems. least squares (LS)-SVMs for function estimation, classification, problems in unsupervised learning and others has been investigated in [22]–[26]. In this case, the problem formulation involves equality instead of inequality constraints. The training for regression and classification problems is then done by solving a set of linear equations. It is the purpose of this paper to introduce a new approach based on LS-SVMs for solving ODEs.

This paper uses the following notation. Vector-valued variables are denoted in lowercase boldface, whereas variables that are neither boldfaced nor capitalized are scalar valued. Matrices are denoted in capital. Euler Script (euscript) font is used for operators.

This paper is organized as follows. In Section II, the problem statement is given. In Section III, we formulate our LS-SVMs method for the solution of linear differential equations. Section IV is devoted to the formulation of the method for nonlinear first-order ODEs. Model selection and the practical implementation of the proposed method are discussed in Section V. Section VI describes the numerical experiments, discussion, and comparison with other known methods.

II. Problem Statement

This section describes the problem statement. In Section II-A, a short introduction to LS-SVMs for regression is given to highlight the difference to the problem considered in this paper. Finally, some operators that will be used in the following sections are defined.

Consider the general m-th order linear ODE with time varying coefficients of the form

$$\mathcal{L}[y] \equiv \sum_{\ell=0}^{m} f_\ell(t) y^{(\ell)}(t) = r(t) \qquad t \in [a, c] \tag{1}$$

where $\mathcal{L}$ represents an m-th order linear differential operator, $[a, c]$ is the problem domain, and $r(t)$ is the input signal. $f_\ell(t)$ are known functions and $y^{(\ell)}(t)$ denotes the ℓ-th derivative of y with respect to t. The $m-1$ necessary initial or boundary conditions for solving the above differential equations are

IVP

$$\mathcal{IC}_\mu[y(t)] = p_\mu, \quad \mu = 0, \ldots, m-1$$

BVP

$$\mathcal{BC}_\mu[y(t)] = q_\mu, \quad \mu = 0, \ldots, m-1$$

where $\mathcal{IC}_\mu$ are the initial conditions (all constraints are applied at the same value of the independent variable i.e., $t = a$) and $\mathcal{BC}_\mu$ are the boundary conditions (the constraints are applied at multiple values of the independent variable t, typically at the ends of the interval $[a, c]$ in which the solution is sought). p_μ and q_μ are given scalars.

A differential equation (1) is said to be stiff when its exact solution consists of a steady state term that does not grow significantly with time, together with a transient term that decays exponentially to zero. Problems involving rapidly decaying transient solutions occur naturally in a wide variety of applications, including the study of damped mass spring system and the analysis of control systems (see [4] for more details).

If the coefficient functions $f_\ell(t)$ of (1) fail to be analytic at point $x = a$, then (1) is called singular ODE.

The approaches given in [18] and [19], define a trial solution to be a sum of two terms i.e., $y(t) = H(t) + F(t, N(t, P))$. The first term $H(t)$, which has to be defined by the user and in some cases is not straightforward, satisfies the initial/boundary conditions, and the second term $F(t, N(t, P))$ is a single-output feedforward neural network with input t and parameters P. In contrast with the approaches given in [18] and [19], we build the model by incorporating the initial/boundary conditions as constraints of an optimization problem. This significantly reduces the burden placed on the user as a potentially difficult problem is handled automatically by the proposed technique.

A. LS-SVM Regression

Let us consider a given training set $\{x_i, y_i\}_{i=1}^N$ with input data $x_i \in \mathbb{R}$ and output data $y_i \in \mathbb{R}$. For the purpose of this paper, we only use an 1-D input space. The goal in a regression problem is to estimate a model of the form $y(x) = \boldsymbol{w}^T \boldsymbol{\varphi}(x) + b$.

The primal LS-SVM model for regression can be written as follows [23]:

$$\begin{aligned} &\underset{\boldsymbol{w},b,\boldsymbol{e}}{\text{minimize}} \quad \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \frac{\gamma}{2}\boldsymbol{e}^T\boldsymbol{e} \\ &\text{s.t.} \quad y_i = \boldsymbol{w}^T\boldsymbol{\varphi}(x_i) + b + e_i, \quad i = 1, \ldots, N \end{aligned} \tag{2}$$

where $\gamma \in \mathbb{R}^+, b \in \mathbb{R}$, $\boldsymbol{w} \in \mathbb{R}^h$. $\boldsymbol{\varphi}(\cdot) : \mathbb{R} \to \mathbb{R}^h$ is the feature map and h is the dimension of the feature space. The dual solution is then given by

$$\left[\begin{array}{c|c} \Omega + I_N/\gamma & \mathbf{1_N} \\ \hline \mathbf{1_N}^T & 0 \end{array}\right] \left[\begin{array}{c} \boldsymbol{\alpha} \\ \hline b \end{array}\right] = \left[\begin{array}{c} \boldsymbol{y} \\ \hline 0 \end{array}\right]$$

where $\Omega_{ij} = K(x_i, x_j) = \boldsymbol{\varphi}(x_i)^T\boldsymbol{\varphi}(x_j)$ is the ij-th entry of the positive definite kernel matrix. $\mathbf{1_N} = [1, \ldots, 1]^T \in \mathbb{R}^N$, $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_N]^T$, $\boldsymbol{y} = [y_1, \ldots, y_N]^T$ and I_N is the identity matrix. The model in the dual form becomes: $y(x) = \sum_{i=1}^N \alpha_i K(x, x_i) + b$. It should be noted that if $b = 0$, for an explicitly known and finite dimensional feature map $\boldsymbol{\varphi}$ the problem could be solved in primal (ridge regression) by eliminating $\boldsymbol{e}$ and then $\boldsymbol{w}$ would be the only unknown. But in the LS-SVM approach, the feature map $\boldsymbol{\varphi}$ is not explicitly known in general and can be infinite dimensional. Therefore, the kernel trick is used and the problem is solved in dual [22]. When we deal with differential equations, the target values y_i are not directly available, so the regression approach does not directly apply. Nevertheless, we can incorporate the

underlying differential equation in the learning process to find an approximation for the solution.

Let us assume an explicit model $\hat{y}(t) = \boldsymbol{w}^T\boldsymbol{\varphi}(t) + b$ as an approximation for the solution of the differential equation. Since there are no data available in order to learn from the differential equation, we have to substitute our model into the given differential equation. Therefore, we need to define the derivative of the kernel function. Making use of Mercer's theorem [21], derivatives of the feature map can be written in terms of derivatives of the kernel function [27]. Let us define the following differential operator, which will be used in subsequent sections:

$$\nabla_n^m \equiv \frac{\partial^{n+m}}{\partial u^n \partial v^m}. \tag{3}$$

If $\boldsymbol{\varphi}(u)^T\boldsymbol{\varphi}(v) = K(u,v)$, then one can show that

$$\begin{aligned}[\boldsymbol{\varphi}^{(n)}(u)]^T\boldsymbol{\varphi}^{(m)}(v) &= \nabla_n^m[\boldsymbol{\varphi}(u)^T\boldsymbol{\varphi}(v)] \\ &= \nabla_n^m[K(u,v)] = \frac{\partial^{n+m}K(u,v)}{\partial u^n \partial v^m}. \end{aligned}\tag{4}$$

Using (4), it is possible to express all derivatives of the feature map in terms of the kernel function itself (provided that the kernel function is sufficiently differentiable). For instance, the following relations hold:

$$\begin{aligned}\nabla_1^0[K(u,v)] &= \frac{\partial(\boldsymbol{\varphi}(u)^T\boldsymbol{\varphi}(v))}{\partial u} = \boldsymbol{\varphi}^{(1)}(u)^T\boldsymbol{\varphi}(v) \\ \nabla_0^1[K(u,v)] &= \frac{\partial(\boldsymbol{\varphi}(u)^T\boldsymbol{\varphi}(v))}{\partial v} = \boldsymbol{\varphi}(u)^T\boldsymbol{\varphi}^{(1)}(v) \\ \nabla_2^0[K(u,v)] &= \frac{\partial^2(\boldsymbol{\varphi}(u)^T\boldsymbol{\varphi}(v))}{\partial u^2} = \boldsymbol{\varphi}^{(2)}(u)^T\boldsymbol{\varphi}(v).\end{aligned}$$

III. Formulation of the Method for the Linear ODE Case

Let us assume that a general approximate solution to (1) is of the form $\hat{y}(t) = \boldsymbol{w}^T\boldsymbol{\varphi}(t) + b$, where $\boldsymbol{w}$ and b are unknowns of the model that have to be determined. To obtain the optimal value of these parameters, collocation methods can be used [28], which assume a discretization of the interval $[a, c]$ into a set of collocation points $\Upsilon = \{a = t_1 < t_2 < \cdots < t_N = c\}$. Therefore, the $\boldsymbol{w}$ and b are to be found by solving the following optimization problem.

For the IVP Case:

$$\begin{aligned}\underset{\hat{y}}{\text{minimize}} \quad & \frac{1}{2}\sum_{i=1}^{N}\Big[(\mathcal{L}[\hat{y}] - r)(t_i)\Big]^2 \\ \text{s.t.} \quad & \mathcal{IC}_\mu[\hat{y}(t)] = p_\mu, \quad \mu = 0, \ldots, m-1.\end{aligned}\tag{5}$$

For the BVP Case:

$$\begin{aligned}\underset{\hat{y}}{\text{minimize}} \quad & \frac{1}{2}\sum_{i=1}^{N}\Big[(\mathcal{L}[\hat{y}] - r)(t_i)\Big]^2 \\ \text{s.t.} \quad & \mathcal{BC}_\mu[\hat{y}(t)] = q_\mu, \quad \mu = 0, \ldots, m-1\end{aligned}\tag{6}$$

where N is the number of collocation points (which is equal to the number of training points) used to undertake the learning process. In what follows we formulate the optimization problem in the LS-SVM framework for solving linear ODEs. For notational convenience, let us list the following notations, which are used in the following sections:

$$\begin{aligned}[\nabla_n^m K](t,s) &= [\nabla_n^m K(u,v)]\Big|_{u=t,v=s} \\ [\Omega_n^m]_{i,j} &= \nabla_n^m[K(u,v)]\Big|_{u=t_i,v=t_j} = \frac{\partial^{n+m}K(u,v)}{\partial u^n \partial v^m}\Big|_{u=t_i,v=t_j} \\ [\Omega_0^0]_{i,j} &= \nabla_0^0[K(u,v)]\Big|_{u=t_i,v=t_j} = K(t_i,t_j)\end{aligned}$$

where $[\Omega_n^m]_{i,j}$ denotes the (i,j)-th entry of matrix Ω_n^m. The notation $M_{k:l,m:n}$ is used for selecting a submatrix of matrix M consisting of rows k to l and columns m to n. $M_{i,:}$ denotes the i-th row of matrix M. $M_{:,j}$ denotes the j-th column of matrix M.

A. First-Order IVP

As a first example, consider the following first-order IVP:

$$y'(t) - f_1(t)y(t) = r(t), \quad y(a) = p_1, \quad a \le t \le c. \tag{7}$$

In the LS-SVM framework, the approximate solution can be obtained by solving the following optimization problem:

$$\begin{aligned}\underset{\boldsymbol{w},b,\boldsymbol{e}}{\text{minimize}} \quad & \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \frac{\gamma}{2}\boldsymbol{e}^T\boldsymbol{e} \\ \text{s.t.} \quad & \boldsymbol{w}^T\boldsymbol{\varphi}'(t_i) = f_1(t_i)\Big[\boldsymbol{w}^T\boldsymbol{\varphi}(t_i) + b\Big] \\ & r(t_i) + e_i, \quad i = 2, \ldots, N \\ & \boldsymbol{w}^T\boldsymbol{\varphi}(t_1) + b = p_1.\end{aligned}\tag{8}$$

This problem is obtained by combining the LS-SVM cost function with constraints constructed by imposing the approximate solution $\hat{y}(t) = \boldsymbol{w}^T\boldsymbol{\varphi}(t) + b$, given by the LS-SVM model, to satisfy the given differential equation with corresponding initial condition at collocation points $\{t_i\}_{i=1}^N$. Problem (8) is a quadratic minimization under linear equality constraints, which enables an efficient solution.

Lemma 3.1: Given a positive definite kernel function $K : \mathbb{R}\times\mathbb{R} \to \mathbb{R}$ with $K(t,s) = \boldsymbol{\varphi}(t)^T\boldsymbol{\varphi}(s)$ and a regularization constant $\gamma \in \mathbb{R}^+$, the solution to (8) is obtained by solving the following dual problem:

$$\left[\begin{array}{c|c|c} \mathcal{K} + I_{N-1}/\gamma & \boldsymbol{h_{p_1}} & -\boldsymbol{f_1} \\ \hline \boldsymbol{h_{p_1}}^T & 1 & 1 \\ \hline -\boldsymbol{f_1}^T & 1 & 0 \end{array}\right]\left[\begin{array}{c} \boldsymbol{\alpha} \\ \hline \beta \\ \hline b \end{array}\right] = \left[\begin{array}{c} \boldsymbol{r} \\ \hline p_1 \\ \hline 0 \end{array}\right] \tag{9}$$

with

$$\begin{aligned}\boldsymbol{\alpha} &= [\alpha_2, \ldots, \alpha_N]^T,\ \boldsymbol{f_1} = [f_1(t_2), \ldots, f_1(t_N)]^T \in \mathbb{R}^{N-1} \\ \boldsymbol{r} &= [r(t_2), \ldots, r(t_N)]^T \in \mathbb{R}^{N-1} \\ \mathcal{K} &= \tilde{\Omega}_1^1 - D_1\tilde{\Omega}_1^0 - \tilde{\Omega}_0^1 D_1 + D_1\tilde{\Omega}_0^0 D_1 \\ \boldsymbol{h_{p_1}} &= [\Omega_0^1]_{1,2:N}^T - D_1[\Omega_0^0]_{1,2:N}^T.\end{aligned}$$

D_1 is a diagonal matrix with the elements of $\boldsymbol{f_1}$ on the main diagonal. $[\Omega_n^m]_{1,2:N} = [[\Omega_n^m]_{1,2}, \ldots, [\Omega_n^m]_{1,N}]$ and $\tilde{\Omega}_n^m = [\Omega_n^m]_{2:N,2:N}$ for $n, m = 0, 1$. Also note that $\mathcal{K} \in \mathbb{R}^{(N-1)\times(N-1)}$ and $\boldsymbol{h_{p_1}} \in \mathbb{R}^{N-1}$.

Proof: The Lagrangian of the constrained optimization problem (8) becomes

$$\mathcal{L}(\boldsymbol{w}, b, e_i, \alpha_i, \beta) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \frac{\gamma}{2}\boldsymbol{e}^T\boldsymbol{e} - \sum_{i=2}^{N}\alpha_i\Big[\boldsymbol{w}^T\Big(\boldsymbol{\varphi}'(t_i) - f_1(t_i)\boldsymbol{\varphi}(t_i)\Big) - f_1(t_i)b - r_i - e_i\Big] - \beta\Big(\boldsymbol{w}^T\boldsymbol{\varphi}(t_1) + b - p_1\Big)$$

where $\{\alpha_i\}_{i=2}^N$ and β are Lagrange multipliers and $r_i = r(t_i)$ for $i = 2, \ldots, N$. Then the Karush–Kuhn–Tucker (KKT) optimality conditions are as follows:

$$\begin{aligned}
&\frac{\partial\mathcal{L}}{\partial\boldsymbol{w}} = 0 \rightarrow \boldsymbol{w} = \sum_{i=2}^{N}\alpha_i\Big(\boldsymbol{\varphi}'(t_i) - f_1(t_i)\boldsymbol{\varphi}(t_i)\Big) + \beta\boldsymbol{\varphi}(t_1)\\
&\frac{\partial\mathcal{L}}{\partial b} = 0 \rightarrow \sum_{i=2}^{N}\alpha_i f_1(t_i) - \beta = 0\\
&\frac{\partial\mathcal{L}}{\partial e_i} = 0 \rightarrow e_i = -\frac{\alpha_i}{\gamma} \qquad i = 2, \ldots, N\\
&\frac{\partial\mathcal{L}}{\partial\alpha_i} = 0 \rightarrow \boldsymbol{w}^T\Big(\boldsymbol{\varphi}'(t_i) - f_1(t_i)\boldsymbol{\varphi}(t_i)\Big) - f_1(t_i)b\\
&-e_i = r_i, \quad i = 2, \ldots, N, \frac{\partial\mathcal{L}}{\partial\beta} = 0 \rightarrow \boldsymbol{w}^T\boldsymbol{\varphi}(t_1) + b = p_1.
\end{aligned}$$

After elimination of the primal variables $\boldsymbol{w}$ and $\{e_i\}_{i=2}^N$ and making use of Mercer's theorem, the solution is given in the dual by

$$\begin{cases}
r_i = \displaystyle\sum_{j=2}^{N}\alpha_j\Big[[\Omega_1^1]_{j,i} - f_1(t_i)\Big([\Omega_1^0]_{j,i} - f_1(t_j)[\Omega_0^0]_{j,i}\Big) - f_1(t_j)[\Omega_0^1]_{j,i}\Big] + \beta\Big([\Omega_0^1]_{1,i} - f_1(t_i)[\Omega_0^0]_{1,i}\Big)\\
\quad + \frac{\alpha_i}{\gamma} - f_1(t_i)b, \quad i = 2, \ldots, N\\
p_1 = \displaystyle\sum_{j=2}^{N}\alpha_j\Big([\Omega_1^0]_{j,1} - f_1(t_j)[\Omega_0^0]_{j,1}\Big) + \beta\,[\Omega_0^0]_{1,1} + b\\
0 = \displaystyle\sum_{j=2}^{N}\alpha_j f_1(t_j) - \beta
\end{cases}$$

and writing these equations in matrix form gives the linear system in (9). ∎

The model in the dual form becomes

$$\hat{y}(t) = \sum_{i=2}^{N}\alpha_i\Big([\nabla_1^0 K](t_i, t) - f_1(t_i)[\nabla_0^0 K](t_i, t)\Big) + \beta\,[\nabla_0^0 K](t_1, t) + b \tag{10}$$

where K is the kernel function.

B. Second-Order IVP and BVP

1) IVP Case: Let us consider a second-order IVP of the form

$$y''(t) = f_1(t)y'(t) + f_2(t)y(t) + r(t) \quad t \in [a, c]$$
$$y(a) = p_1, \quad y'(a) = p_2.$$

The approximate solution, $\hat{y}(t) = \boldsymbol{w}^T\boldsymbol{\varphi}(t) + b$, is then obtained by solving the following optimization problem:

$$\begin{aligned}
\underset{\boldsymbol{w}, b, \boldsymbol{e}}{\text{minimize}} \quad & \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \frac{\gamma}{2}\boldsymbol{e}^T\boldsymbol{e}\\
\text{s.t.} \quad & \boldsymbol{w}^T\boldsymbol{\varphi}''(t_i) = f_1(t_i)\boldsymbol{w}^T\boldsymbol{\varphi}'(t_i) + f_2(t_i)[\boldsymbol{w}^T\boldsymbol{\varphi}(t_i) + b] + r(t_i) + e_i, \quad i = 2, \ldots, N\\
& \boldsymbol{w}^T\boldsymbol{\varphi}(t_1) + b = p_1\\
& \boldsymbol{w}^T\boldsymbol{\varphi}'(t_1) = p_2.
\end{aligned} \tag{11}$$

Lemma 3.2: Given a positive definite kernel function $K : \mathbb{R}\times\mathbb{R} \rightarrow \mathbb{R}$ with $K(t, s) = \boldsymbol{\varphi}(t)^T\boldsymbol{\varphi}(s)$ and a regularization constant $\gamma \in \mathbb{R}^+$, the solution to (11) is obtained by solving the following dual problem:

$$\left[\begin{array}{c|c|c|c} \mathcal{K} + I_{N-1}/\gamma & \boldsymbol{h_{p_1}} & \boldsymbol{h_{p_2}} & -\boldsymbol{f_2}\\ \hline \boldsymbol{h_{p_1}}^T & 1 & 0 & 1\\ \hline \boldsymbol{h_{p_2}}^T & 0 & [\Omega_1^1]_{1,1} & 0\\ \hline -\boldsymbol{f_2}^T & 1 & 0 & 0 \end{array}\right]\left[\begin{array}{c} \boldsymbol{\alpha}\\ \hline \beta_1\\ \hline \beta_2\\ \hline b \end{array}\right] = \left[\begin{array}{c} \boldsymbol{r}\\ \hline p_1\\ \hline p_2\\ \hline 0 \end{array}\right] \tag{12}$$

where

$$\begin{aligned}
\boldsymbol{\alpha} &= [\alpha_2, \ldots, \alpha_N]^T, \ \boldsymbol{f_1} = [f_1(t_2), \ldots, f_1(t_N)]^T \in \mathbb{R}^{N-1}\\
\boldsymbol{f_2} &= [f_2(t_2), \ldots, f_2(t_N)]^T \in \mathbb{R}^{N-1}\\
\boldsymbol{r} &= [r(t_2), \ldots, r(t_N)]^T \in \mathbb{R}^{N-1}\\
\mathcal{K} &= \tilde{\Omega}_2^2 - D_1\tilde{\Omega}_2^1 - D_2\tilde{\Omega}_2^0 - \tilde{\Omega}_1^2 D_1 - \tilde{\Omega}_0^2 D_2 + D_1\tilde{\Omega}_1^1 D_1 + D_1\tilde{\Omega}_0^1 D_2 + D_2\tilde{\Omega}_1^0 D_1 + D_2\tilde{\Omega}_0^0 D_2\\
\boldsymbol{h}_{p_1} &= [\Omega_0^2]_{1,2:N}^T - D_1[\Omega_0^1]_{1,2:N}^T - D_2[\Omega_0^0]_{1,2:N}^T\\
\boldsymbol{h}_{p_2} &= [\Omega_1^2]_{1,2:N}^T - D_1[\Omega_1^1]_{1,2:N}^T - D_2[\Omega_1^0]_{1,2:N}^T.
\end{aligned}$$

D_1 and D_2 are diagonal matrices with the elements of $\boldsymbol{f_1}$ and $\boldsymbol{f_2}$ on the main diagonal, respectively. Note that $\mathcal{K} \in \mathbb{R}^{(N-1)\times(N-1)}$ and $\boldsymbol{h}_{p_1}, \boldsymbol{h}_{p_2} \in \mathbb{R}^{N-1}$. $[\Omega_n^m]_{1,2:N} = [[\Omega_n^m]_{1,2}, \ldots, [\Omega_n^m]_{1,N}]$ for $n = 0, 1$ and $m = 0, 1, 2$. $\tilde{\Omega}_n^m = [\Omega_n^m]_{2:N,2:N}$ for $m, n = 0, 1, 2$.

Proof: Consider the Lagrangian of (11)

$$\begin{aligned}
&\mathcal{L}(\boldsymbol{w}, b, e_i, \alpha_i, \beta_1, \beta_2)\\
&= \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \frac{\gamma}{2}\boldsymbol{e}^T\boldsymbol{e} - \sum_{i=2}^{N}\alpha_i\Big[\boldsymbol{w}^T\Big(\boldsymbol{\varphi}''(t_i) - f_1(t_i)\boldsymbol{\varphi}'(t_i) - f_2(t_i)\boldsymbol{\varphi}(t_i)\Big) - f_2(t_i)b - r_i - e_i\Big] - \beta_1\Big(\boldsymbol{w}^T\boldsymbol{\varphi}(t_1) + b - p_1\Big)\\
&\quad - \beta_2\Big(\boldsymbol{w}^T\boldsymbol{\varphi}'(t_1) - p_2\Big)
\end{aligned} \tag{13}$$

where $\{\alpha_i\}_{i=2}^N, \beta_1$ and β_2 are Lagrange multipliers. The KKT optimality conditions are as follows:

$$\begin{aligned}
&\frac{\partial\mathcal{L}}{\partial\boldsymbol{w}} = 0 \rightarrow \boldsymbol{w} = \sum_{i=2}^{N}\alpha_i\Big(\boldsymbol{\varphi}_i'' - f_1(t_i)\boldsymbol{\varphi}_i' - f_2(t_i)\boldsymbol{\varphi}_i\Big) + \beta_1\boldsymbol{\varphi}_1 + \beta_2\boldsymbol{\varphi}_1' \qquad \frac{\partial\mathcal{L}}{\partial b} = 0 \rightarrow \sum_{i=2}^{N}\alpha_i f_2(t_i) - \beta_1 = 0\\
&\frac{\partial\mathcal{L}}{\partial e_i} = 0 \rightarrow e_i = -\frac{\alpha_i}{\gamma}, \quad i = 2, \ldots,
\end{aligned}$$

$$N\frac{\partial \mathcal{L}}{\partial \alpha_i} = 0 \rightarrow \boldsymbol{w}^T\left(\boldsymbol{\varphi}_i'' - f_1(t_i)\boldsymbol{\varphi}_i' - f_2(t_i)\boldsymbol{\varphi}_i\right) - f_2(t_i)b - e_i = r_i,$$
$$i = 2, \ldots, N \quad \frac{\partial \mathcal{L}}{\partial \beta_1} = 0 \rightarrow \boldsymbol{w}^T\boldsymbol{\varphi}_1 + b = p_1$$
$$\frac{\partial \mathcal{L}}{\partial \beta_2} = 0 \rightarrow \boldsymbol{w}^T\boldsymbol{\varphi}_1' = p_2$$

where $\boldsymbol{\varphi}_i = \boldsymbol{\varphi}(t_i)$, $\boldsymbol{\varphi}_i' = \boldsymbol{\varphi}'(t_i)$ and $\boldsymbol{\varphi}_i'' = \boldsymbol{\varphi}''(t_i)$ for $i = 1, \ldots, N$.

Applying the kernel trick and eliminating $\boldsymbol{w}$ and $\{e_i\}_{i=2}^N$ leads to

$$\begin{cases} r_i = \sum_{j=2}^N \alpha_j \Big[[\Omega_2^2]_{j,i} \\ \quad -f_1(t_i)\Big([\Omega_2^1]_{j,i} - f_1(t_j)[\Omega_1^1]_{j,i} - f_2(t_j)[\Omega_0^1]_{j,i}\Big) \\ \quad -f_2(t_i)\Big([\Omega_2^0]_{j,i} - f_1(t_j)[\Omega_1^0]_{j,i} - f_2(t_j)[\Omega_0^0]_{j,i}\Big) \\ \quad -f_1(t_j)[\Omega_1^2]_{j,i} - f_2(t_j)[\Omega_0^2]_{j,i}\Big] \\ \quad +\beta_1\Big([\Omega_0^2]_{1,i} - f_1(t_i)[\Omega_0^1]_{1,i} - f_2(t_i)[\Omega_0^0]_{1,i}\Big) \\ \quad +\beta_2\Big([\Omega_1^2]_{1,i} - f_1(t_i)[\Omega_1^1]_{1,i} - f_2(t_i)[\Omega_1^0]_{1,i}\Big) \\ \quad +\frac{\alpha_i}{\gamma} - f_2(t_i)b, \quad i = 2, \ldots, N \end{cases}$$

$$\begin{cases} p_1 = \sum_{j=2}^N \alpha_j\Big([\Omega_2^0]_{j,1} - f_1(t_j)[\Omega_1^0]_{j,1} - f_2(t_j)[\Omega_0^0]_{j,1}\Big) \\ \quad +\beta_1[\Omega_0^0]_{1,1} + \beta_2[\Omega_1^0]_{1,1} + b \\ p_2 = \sum_{j=2}^N \alpha_j\Big([\Omega_2^1]_{j,1} - f_1(t_j)[\Omega_1^1]_{j,1} - f_2(t_j)[\Omega_0^1]_{j,1}\Big) \\ \quad +\beta_1[\Omega_0^1]_{1,1} + \beta_2[\Omega_1^1]_{1,1} \\ 0 = \sum_{j=2}^N \alpha_j f_2(t_j) - \beta_1. \end{cases}$$

Finally, writing these equations in matrix form will result in the linear system (12). ∎

The LS-SVM model for the solution and its derivative in the dual form becomes

$$\hat{y}(t) = \sum_{i=2}^N \alpha_i\Big([\nabla_2^0 K](t_i,t) - f_1(t_i)[\nabla_1^0 K](t_i,t) - f_2(t_i)[\nabla_0^0 K](t_i,t)\Big) + \beta_1[\nabla_0^0 K](t_1,t) + \beta_2[\nabla_1^0 K](t_1,t) + b$$

$$\frac{d\hat{y}(t)}{dt} = \sum_{i=2}^N \alpha_i\Big([\nabla_2^1 K](t_i,t) - f_1(t_i)[\nabla_1^1 K](t_i,t) - f_2(t_i)[\nabla_0^1 K](t_i,t)\Big) + \beta_1[\nabla_0^1 K](t_1,t) + \beta_2[\nabla_1^1 K](t_1,t).$$

2) BVP Case: Consider the second-order BVP of ODEs of the form

$$y''(t) = f_1(t)y'(t) + f_2(t)y(t) + r(t) \quad t \in [a,c]$$
$$y(a) = p_1, \quad y(c) = q_1.$$

Then the parameters of the closed-form approximation of the solution can be obtained by solving the following optimization problem:

$$\begin{aligned} \underset{\boldsymbol{w},b,\boldsymbol{e}}{\text{minimize}} \ & \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \frac{\gamma}{2}\boldsymbol{e}^T\boldsymbol{e} \\ \text{s.t.} \quad & \boldsymbol{w}^T\boldsymbol{\varphi}''(t_i) = f_1(t_i)\boldsymbol{w}^T\boldsymbol{\varphi}'(t_i) + f_2(t_i)[\boldsymbol{w}^T\boldsymbol{\varphi}(t_i) + b] + r(t_i) + e_i \\ & \qquad i = 2, \ldots, N-1 \\ & \boldsymbol{w}^T\boldsymbol{\varphi}(t_1) + b = p_1 \\ & \boldsymbol{w}^T\boldsymbol{\varphi}(t_N) + b = q_1. \end{aligned} \tag{14}$$

The same procedure can be applied to derive the Lagrangian and afterward the KKT optimality conditions. Then, one can show that the solution to (14) is obtained by solving the following linear system:

$$\left[\begin{array}{c|c|c|c} \mathcal{K} + I_{N-2}/\gamma & \boldsymbol{h_{p_1}} & \boldsymbol{h_{q_1}} & -\boldsymbol{f_2} \\ \hline \boldsymbol{h_{p_1}}^T & 1 & [\Omega_0^0]_{N,1} & 1 \\ \hline \boldsymbol{h_{q_1}}^T & [\Omega_0^0]_{1,N} & 1 & 1 \\ \hline -\boldsymbol{f_2}^T & 1 & 1 & 0 \end{array}\right]\left[\begin{array}{c} \boldsymbol{\alpha} \\ \hline \beta_1 \\ \hline \beta_2 \\ \hline b \end{array}\right] = \left[\begin{array}{c} \boldsymbol{r} \\ \hline p_1 \\ \hline q_1 \\ \hline 0 \end{array}\right]$$

where

$$\begin{aligned} \boldsymbol{\alpha} &= [\alpha_2, \ldots, \alpha_{N-1}]^T \\ \boldsymbol{f_1} &= [f_1(t_2), \ldots, f_1(t_{N-1})]^T \in \mathbb{R}^{N-2} \\ \boldsymbol{f_2} &= [f_2(t_2), \ldots, f_2(t_{N-1})]^T \in \mathbb{R}^{N-2} \\ \boldsymbol{r} &= [r(t_2), \ldots, r(t_{N-1})]^T \in \mathbb{R}^{N-2} \\ \mathcal{K} &= \tilde{\Omega}_2^2 - D_1\tilde{\Omega}_2^1 - D_2\tilde{\Omega}_2^0 - \tilde{\Omega}_1^2 D_1 - \tilde{\Omega}_0^2 D_2 \\ &\quad + D_1\tilde{\Omega}_1^1 D_1 + D_1\tilde{\Omega}_0^1 D_2 + D_2\tilde{\Omega}_1^0 D_1 + D_2\tilde{\Omega}_0^0 D_2 \\ \boldsymbol{h_{p_1}} &= [\Omega_0^2]_{1,2:N-1}^T - D_1[\Omega_0^1]_{1,2:N-1}^T - D_2[\Omega_0^0]_{1,2:N-1}^T \\ \boldsymbol{h_{q_1}} &= [\Omega_0^2]_{N,2:N-1}^T - D_1[\Omega_0^1]_{N,2:N-1}^T - D_2[\Omega_0^0]_{N,2:N-1}^T. \end{aligned}$$

D_1 and D_2 are diagonal matrices with the elements of $\boldsymbol{f_1}$ and $\boldsymbol{f_2}$ on the main diagonal, respectively. Note that $\mathcal{K} \in \mathbb{R}^{(N-2)\times(N-2)}$ and $\boldsymbol{h_{p_1}}, \boldsymbol{h_{q_1}} \in \mathbb{R}^{N-2}$. $[\Omega_n^m]_{1,2:N-1} = [[\Omega_n^m]_{1,2}, \ldots, [\Omega_n^m]_{1,N-1}]$ and $[\Omega_n^m]_{N,2:N-1} = [[\Omega_n^m]_{N,2}, \ldots, [\Omega_n^m]_{N,N-1}]$ for $n = 0, 1$ and $m = 0, 1, 2$. $\tilde{\Omega}_n^m = [\Omega_n^m]_{2:N-1,2:N-1}$ for $m, n = 0, 1, 2$.

The LS-SVM model for the solution and its derivative are expressed in dual form as

$$\hat{y}(t) = \sum_{i=2}^{N-1} \alpha_i\Big([\nabla_2^0 K](t_i,t) - f_1(t_i)[\nabla_1^0 K](t_i,t) - f_2(t_i)[\nabla_0^0 K](t_i,t)\Big) + \beta_1[\nabla_0^0 K](t_1,t) + \beta_2[\nabla_0^0 K](t_N,t) + b$$

$$\frac{d\hat{y}(t)}{dt} = \sum_{i=2}^{N-1} \alpha_i \Big([\nabla_2^1 K](t_i,t) - f_1(t_i)[\nabla_1^1 K](t_i,t) - f_2(t_i)[\nabla_0^1 K](t_i,t) \Big) + \beta_1 [\nabla_0^1 K](t_1,t) + \beta_2 [\nabla_0^1 K](t_N,t).$$

C. m-th Order Linear ODE

Let us now consider the general m-th order IVP of the following form:

$$y^{(m)}(t) - \sum_{i=1}^{m} f_i(t) y^{(m-i)}(t) = r(t) \quad t \in [a,c]$$
$$\begin{cases} y(a) = p_1 \\ y^{(i-1)}(a) = p_i, \quad i = 2,\ldots,m. \end{cases} \tag{15}$$

The approximate solution can be obtained by solving the following optimization problem:

$$\begin{aligned} \underset{\boldsymbol{w},b,\boldsymbol{e}}{\text{minimize}} \ & \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \frac{\gamma}{2}\boldsymbol{e}^T\boldsymbol{e} \\ \text{s.t.} \quad & \boldsymbol{w}^T\boldsymbol{\varphi}^{(m)}(t_i) = \boldsymbol{w}^T\Big[\sum_{k=1}^{m} f_k(t_i)\boldsymbol{\varphi}_i^{(m-k)}\Big] + f_m(t_i)b + r(t_i) + e_i \\ & \qquad i = 2,\ldots,N \\ & \boldsymbol{w}^T\boldsymbol{\varphi}(t_1) + b = p_1 \\ & \boldsymbol{w}^T\boldsymbol{\varphi}^{(i-1)}(t_1) = p_i, \quad i = 2,\ldots,m. \end{aligned} \tag{16}$$

Lemma 3.3: Given a positive definite kernel function $K : \mathbb{R}\times\mathbb{R}\to\mathbb{R}$ with $K(t,s) = \boldsymbol{\varphi}(t)^T\boldsymbol{\varphi}(s)$ and a regularization constant $\gamma \in \mathbb{R}^+$, the solution to (16) is obtained by solving the following dual problem:

$$\left[\begin{array}{c|c|c} \mathcal{K} + I_{N-1}/\gamma & \mathcal{K}_{\text{in}} & -\boldsymbol{f_m} \\ \hline \mathcal{K}_{\text{in}}^T & \Delta & C \\ \hline -\boldsymbol{f_m}^T & C^T & 0 \end{array}\right] \left[\begin{array}{c} \boldsymbol{\alpha} \\ \hline \boldsymbol{\beta} \\ \hline b \end{array}\right] = \left[\begin{array}{c} \boldsymbol{r} \\ \hline \boldsymbol{p} \\ \hline 0 \end{array}\right] \tag{17}$$

with

$$\begin{aligned} \boldsymbol{f_k} &= [f_k(t_2),\ldots,f_k(t_N)]^T \in \mathbb{R}^{N-1}, \quad k = 1,\ldots,m \\ \boldsymbol{h_{p_\ell}} &= [\Omega_{\ell-1}^m]_{1,2:N}^T - \sum_{k=1}^{m} D_k [\Omega_{\ell-1}^{m-k}]_{1,2:N}^T, \ \ell = 1,\ldots,m \\ \mathcal{K}_{\text{in}} &= [\boldsymbol{h_{p_1}},\ldots,\boldsymbol{h_{p_m}}] \in \mathbb{R}^{(N-1)\times m}, \quad \boldsymbol{h_{p_\ell}} \in \mathbb{R}^{(N-1)} \\ \boldsymbol{p} &= [p_1,\ldots,p_m]^T \in \mathbb{R}^m, \quad C = [1,0,\ldots,0]^T \in \mathbb{R}^m \\ \boldsymbol{r} &= [r(t_2),\ldots,r(t_N)]^T \in \mathbb{R}^{N-1}, \ \boldsymbol{\alpha} = [\alpha_2,\ldots,\alpha_N]^T \\ \boldsymbol{\beta} &= [\beta_1,\ldots,\beta_m]^T, \ \bar{\Omega}_1 = [\tilde{\Omega}_m^0,\ldots,\tilde{\Omega}_m^{m-1}]^T \\ \bar{D} &= [D_m,\ldots,D_1], \ \ \bar{\Omega} = \begin{bmatrix} \tilde{\Omega}_0^0 & \cdots & \tilde{\Omega}_{m-1}^0 \\ \vdots & \ddots & \vdots \\ \tilde{\Omega}_0^{m-1} & \cdots & \tilde{\Omega}_{m-1}^{m-1} \end{bmatrix} \\ \mathcal{K} &= \tilde{\Omega}_m^m - \bar{D}\,\bar{\Omega}_1 - \bar{\Omega}_1^T\bar{D}^T + \bar{D}\,\bar{\Omega}\,\bar{D}^T \\ \Delta &= \begin{bmatrix} [\Omega_0^0]_{1,1} & \cdots & [\Omega_{m-1}^0]_{1,1} \\ \vdots & \ddots & \vdots \\ \left[\Omega_0^{m-1}\right]_{1,1} & \cdots & [\Omega_{m-1}^{m-1}]_{1,1} \end{bmatrix}_{m\times m}. \end{aligned}$$

$\{D_i\}_{i=1}^m$ are diagonal matrices with the elements of $\{\boldsymbol{f_i}\}_{i=1}^m$ on the main diagonal, respectively. Also $\bar{\Omega}_1$, $\bar{\Omega}$, and $\bar{D}$ are block matrices. $\tilde{\Omega}_n^m = [\Omega_n^m]_{2:N,2:N}$. Note that $\mathcal{K} \in \mathbb{R}^{(N-1)\times(N-1)}$.

Proof: The Lagrangian for (16) is given by

$$\begin{aligned} &\mathcal{L}(\boldsymbol{w},b,e_i,\alpha_i,\beta_i) \\ &= \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \frac{\gamma}{2}\boldsymbol{e}^T\boldsymbol{e} - \sum_{i=2}^{N}\alpha_i\Big[\boldsymbol{w}^T\Big(\boldsymbol{\varphi}^m(t_i) - \sum_{k=1}^{m} f_k(t_i)\boldsymbol{\varphi}^{(m-k)}(t_i)\Big) - f_m(t_i)b - r_i - e_i\Big] - \beta_1\Big(\boldsymbol{w}^T\boldsymbol{\varphi}(t_1) + b - p_1\Big) \\ &\quad - \beta_2\Big(\boldsymbol{w}^T\boldsymbol{\varphi}'(t_1) - p_2\Big) - \cdots - \beta_m\Big(\boldsymbol{w}^T\boldsymbol{\varphi}^{m-1}(t_1) - p_m\Big). \end{aligned}$$

Eliminating $\boldsymbol{w}$ and $\{e_i\}_{i=2}^N$ from the corresponding KKT optimality conditions yields the following set of equations:

$$\begin{cases} r_i = \sum_{j=2}^{N}\alpha_j\Big[[\Omega_m^m]_{j,i} - \sum_{\ell=1}^{m} f_\ell(t_i)\Big([\Omega_m^{m-\ell}]_{j,i} - \sum_{k=1}^{m} f_k(t_j)[\Omega_{m-k}^{m-\ell}]_{j,i}\Big) - \sum_{k=1}^{m} f_k(t_j)[\Omega_{m-k}^m]_{j,i} + \sum_{\ell=1}^{m}\beta_\ell\Big([\Omega_{\ell-1}^m]_{1,i} - \sum_{k=1}^{m} f_k(t_i)[\Omega_{\ell-1}^{m-k}]_{1,i}\Big) + \frac{\alpha_i}{\gamma} - f_m(t_i)b, \quad i = 2,\ldots,N \\ p_1 = \sum_{j=2}^{N}\alpha_j\Big([\Omega_m^0]_{j,1} - \sum_{k=1}^{m} f_k(t_j)[\Omega_{m-k}^0]_{j,1}\Big) + \sum_{k=1}^{m}\beta_k[\Omega_{k-1}^0]_{1,1} + b \\ \vdots \\ p_m = \sum_{j=2}^{N}\alpha_j\Big([\Omega_m^{m-1}]_{j,1} - \sum_{k=1}^{m} f_k(t_j)[\Omega_{m-k}^{m-1}]_{j,1}\Big) + \sum_{k=1}^{m}\beta_k[\Omega_{k-1}^{m-1}]_{1,1} \\ 0 = \sum_{j=2}^{N}\alpha_j f_m(t_j) - \beta_1. \end{cases}$$

Rewriting the above system in matrix form will result in (17). ∎

The LS-SVM model for the solution and its derivatives can be expressed in dual form as follows:

$$\begin{aligned} \hat{y}(t) &= \sum_{i=2}^{N}\alpha_i\Big([\nabla_m^0 K](t_i,t) - \sum_{k=1}^{m} f_k(t_i)[\nabla_{m-k}^0 K](t_i,t)\Big) + \sum_{k=1}^{m}\beta_k[\nabla_{k-1}^0 K](t_1,t) + b \\ \frac{d^p\,\hat{y}(t)}{dt^p} &= \sum_{i=2}^{N}\alpha_i\Big([\nabla_m^p K](t_i,t) - \sum_{k=1}^{m} f_k(t_i)[\nabla_{m-k}^p K](t_i,t)\Big) + \sum_{k=1}^{m}\beta_k[\nabla_{k-1}^p K](t_1,t) \quad p = 1,\ldots,m-1. \end{aligned}$$

Lemma 3.4: The matrix $\mathcal{K}$ is positive semi-definite.

Proof: Let $\tilde{D} = [\bar{D}, -I_{N-1}]$ and

$$\tilde{\Omega} = \begin{bmatrix} \bar{\Omega} & \bar{\Omega}_1 \\ \bar{\Omega}_1^T & \Omega_m^m \end{bmatrix}.$$

Then the matrix $\mathcal{K}$ can be written as $\tilde{D}\tilde{\Omega}\tilde{D}^T$. To show that $x^T\mathcal{K}x \geq 0$ for any x, it is sufficient to show that $\tilde{x}^T\tilde{\Omega}\tilde{x} \geq 0$ for any $\tilde{x}$ because $x = \tilde{D}\tilde{x}$ is included as a special case. Now consider matrices of evaluations of the feature map and its derivatives $\Phi^{(i)} = [\boldsymbol{\varphi}^{(i)}(t_2), \ldots, \boldsymbol{\varphi}^{(i)}(t_N)]$ for $i = 0, \ldots, m-1$ and denote their concatenation as $\tilde{\Phi} = [\Phi^{(0)}, \ldots, \Phi^{(m-1)}]$. Then $\|\tilde{\Phi}\tilde{x}\|_2^2 = \tilde{x}^T\tilde{\Phi}^T\tilde{\Phi}\tilde{x} = \tilde{x}^T\tilde{\Omega}\tilde{x}$ holds, where the last equality follows from an application of the kernel trick. With the property that the norm of any vector is a nonnegative real number, $\|\tilde{\Phi}\tilde{x}\|_2$ is greater than or equal to zero. Therefore, also its squared form $\tilde{x}^T\tilde{\Omega}\tilde{x}$ is nonnegative, which concludes the proof. ∎

IV. Formulation of the Method for the Nonlinear ODE Case

In this section, we formulate an optimization problem based on LS-SVMs for solving nonlinear first-order ODEs of the following form:

$$y' = f(t, y), \qquad y(a) = p_1, \quad a \leq t \leq c. \tag{18}$$

One starts with assuming the approximate solution to be of the form $\hat{y}(t) = \boldsymbol{w}^T\boldsymbol{\varphi}(t) + b$. Additional unknowns y_i are introduced to keep the constraints linear in $\boldsymbol{w}$. This yields the following nonlinear optimization problem:

$$\begin{aligned} \underset{\boldsymbol{w},b,\boldsymbol{e},\boldsymbol{\xi},y_i}{\text{minimize}} \ & \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \frac{\gamma}{2}\boldsymbol{e}^T\boldsymbol{e} + \frac{\gamma}{2}\boldsymbol{\xi}^T\boldsymbol{\xi} \\ \text{s.t.} \quad & \boldsymbol{w}^T\boldsymbol{\varphi}'(t_i) = f(t_i, y_i) + e_i, \quad i = 2, \ldots, N \\ & \boldsymbol{w}^T\boldsymbol{\varphi}(t_1) + b = p_1 \\ & y_i = \boldsymbol{w}^T\boldsymbol{\varphi}(t_i) + b + \xi_i, \quad i = 2, \ldots, N. \end{aligned} \tag{19}$$

The Lagrangian of the constrained optimization problem (19) becomes

$$\begin{aligned} \mathcal{L}(\boldsymbol{w}, b, e_i, \xi_i, y_i, \alpha_i, \eta_i, \beta) &= \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \frac{\gamma}{2}\boldsymbol{e}^T\boldsymbol{e} + \frac{\gamma}{2}\boldsymbol{\xi}^T\boldsymbol{\xi} \\ &- \sum_{i=2}^{N}\alpha_i\Big(\boldsymbol{w}^T\boldsymbol{\varphi}'(t_i) - f(t_i, y_i) - e_i\Big) \\ &- \beta\Big(\boldsymbol{w}^T\boldsymbol{\varphi}(t_1) + b - p_1\Big) \\ &- \sum_{i=2}^{N}\eta_i\Big(y_i - \boldsymbol{w}^T\boldsymbol{\varphi}(t_i) - b - \xi_i\Big). \end{aligned}$$

After obtaining KKT optimality conditions, and elimination of the primal variables $\boldsymbol{w}$, $\{e_i\}_{i=2}^N$ and $\{\xi_i\}_{i=2}^N$ and making use of Mercer's theorem, the solution is obtained in the dual by solving the following nonlinear system of equations:

$$\left[\begin{array}{c|c|c|c|c} \widehat{\Omega_1^1} & \tilde{\Omega}_0^1 & \boldsymbol{h_1}^T & \boldsymbol{0}_{N-1} & 0_{(N-1)\times(N-1)} \\ \hline (\tilde{\Omega}_0^1)^T & \widehat{\Omega}_0^0 & \boldsymbol{h_0}^T & \boldsymbol{1}_{N-1} & -I_{N-1} \\ \hline \boldsymbol{h_1} & \boldsymbol{h_0} & [\Omega_0^0]_{1,1} & 1 & \boldsymbol{0}_{N-1}^T \\ \hline \boldsymbol{0}_{N-1}^T & \boldsymbol{1}_{N-1}^T & 1 & 0 & \boldsymbol{0}_{N-1}^T \\ \hline D(\boldsymbol{y}) & I_{N-1} & \boldsymbol{0}_{N-1} & \boldsymbol{0}_{N-1} & 0_{(N-1)\times(N-1)} \end{array}\right]$$

$$\left[\begin{array}{c} \boldsymbol{\alpha} \\ \hline \boldsymbol{\eta} \\ \hline \beta \\ \hline b \\ \hline \boldsymbol{y} \end{array}\right] = \left[\begin{array}{c} \boldsymbol{f(y)} \\ \hline \boldsymbol{0}_{N-1} \\ \hline p_1 \\ \hline 0 \\ \hline \boldsymbol{0}_{N-1} \end{array}\right] \tag{20}$$

where

$$\begin{aligned} \widehat{\Omega}_1^1 &= \tilde{\Omega}_1^1 + I_{N-1}/\gamma, \quad \widehat{\Omega}_0^0 = \tilde{\Omega}_0^0 + I_{N-1}/\gamma \\ D(\boldsymbol{y}) &= \operatorname{diag}(\boldsymbol{f'(y)}) \\ \boldsymbol{f(y)} &= [f(t_2, y_2), \ldots, f(t_N, y_N)]^T \\ \boldsymbol{f'(y)} &= \left[\left.\frac{\partial f(t,y)}{\partial y}\right|_{t=t_2,y=y_2}, \ldots, \left.\frac{\partial f(t,y)}{\partial y}\right|_{t=t_N,y=y_N}\right] \\ \boldsymbol{\alpha} &= [\alpha_2, \ldots, \alpha_N]^T, \ \boldsymbol{\eta} = [\eta_2, \ldots, \eta_N]^T \\ \boldsymbol{y} &= [y_2, \ldots, y_N]^T, \ \tilde{\Omega}_0^0 = [\Omega_0^0]_{2:N,2:N} \\ \tilde{\Omega}_1^1 &= [\Omega_1^1]_{2:N,2:N}, \ \tilde{\Omega}_0^1 = [\Omega_0^1]_{2:N,2:N} \\ \boldsymbol{h_0} &= \left[[\Omega_0^0]_{1,2}, \ldots, [\Omega_0^0]_{1,N}\right] \\ \boldsymbol{h_1} &= \left[[\Omega_0^1]_{1,2}, \ldots, [\Omega_0^1]_{1,N}\right] \\ \boldsymbol{0}_{N-1} &= [0, \ldots, 0]^T \in \mathbb{R}^{N-1}. \end{aligned}$$

The nonlinear system (20), which consists of $3N-1$ equations with $3N-1$ unknowns $(\boldsymbol{\alpha}, \boldsymbol{\eta}, \beta, b, \boldsymbol{y})$, is solved by Newton's method. The model in the dual form becomes

$$\begin{aligned} \hat{y}(t) = &\sum_{i=2}^{N}\alpha_i[\nabla_1^0 K](t_i, t) + \sum_{i=2}^{N}\eta_i[\nabla_0^0 K](t_i, t) \\ &+\beta\,[\nabla_0^0 K](t_1, t) + b. \end{aligned} \tag{21}$$

V. Practical Implementation and Model Selection

A. Solution on a Long Time Interval

Consider now the situation where a given differential equation has to be solved for a large time interval $[a, c]$. It should be noted that in order to improve the accuracy (or maintain the same order of accuracy on the whole domain), we then need to increase the number of collocation points. This approach, however, leads to a larger system of equations.

In order to implement the proposed method for solving problems involving large time intervals efficiently, the domain decomposition technique is applied [29]. At first, the domain $\Xi = [a, c]$ is decomposed into S segments as $\Xi = \bigcup_{k=1}^{S}\Xi_k$. We assume that the approximate solution on the k-th segment has the form $\hat{y}_k(t) = \boldsymbol{w}_k^T\boldsymbol{\varphi}(t) + b_k$. Then the problem is solved in each sub-domain Ξ_k using the described method in previous sections. The computed approximate solution at the final point in the sub-domain Ξ_k is used as starting point (initial value) for the consecutive sub-domain Ξ_{k+1}.

Utilizing this approach will result in solving S smaller systems of equations, which is computationally more efficient than solving a very large system of equations obtained by considering the whole domain Ξ (with the same total number of collocation points). The procedure is outlined in Algorithm 1.

Algorithm 1 Approximating the Solution on a Large Interval

1: Decompose the domain $\Xi = [a, c]$ into S sub-domains.
2: set $\Gamma = (c-a)/S$, $t_{\text{in}} := a$, $y_{\text{in}} := p_1$, $t_f := t_{\text{in}} + \Gamma$.
3: **for** $k = 1$ **to** S **do**
4: Obtain a LS-SVM model for the k-th segment $[t_{\text{in}}, t_f]$ i.e., $\hat{y}_k(t) = \boldsymbol{w}_k^T \boldsymbol{\varphi}_k(t) + b_k$.
5: set $t_{\text{in}} := t_f$, $y_{\text{in}} := \hat{y}(t_f)$, $t_f := t_{\text{in}} + \Gamma$
6: **end for**
7: For a given test point t:
1) check to which segment it belongs;
2) use the corresponding model to compute the approximate solution at the given point.

B. Parameter Tuning

The performance of the LS-SVM model depends on the choice of the tuning parameters. In this paper, for all experiments, the Gaussian RBF kernel is used. Therefore, a model is determined by the regularization parameter γ and the kernel bandwidth σ.

It should be noted that unlike the regression case, we do not have target values, and consequently we do not have noise. Therefore, a quite large value should be taken for the regularization constant γ so that the error $\boldsymbol{e}$ is sharply minimized or equivalently the constraints are well satisfied.

In all the experiments, the chosen value for γ was 10^{10}, except for the problem with large interval for which γ is set to 10^5 in order to avoid ill conditioning.

Therefore, the only parameter left that has to be tuned is the kernel bandwidth. In this paper, the optimal values of σ are obtained by evaluating the performance of the model on a validation set using a meaningful range of possible (σ). The validation set is defined to be the set of midpoints $V \equiv \{v_i = (t_i + t_{i+1})/2, i = 1, \ldots, N-1\}$ where $\{t_i\}_{i=1}^N$ are training points. The values that minimize the mean squared error (MSE) on this validation set are then selected.

Remark 5.1: In some cases, an extremely large value for γ, normally greater than 10^7, can make the matrix in (9) close to singular.

VI. Numerical Results

In this section, we have tested the performance of the proposed method on seven problems, four first-order and three second-order ODEs. For the first three problems and Problem 5, a comparison is made between the solutions obtained in [19] and our computed solutions. The numerical results of the Problems 4 and 6 are compared with those given in [18]. Problem 7, which has no analytic solution and is a singular problem, is solved and the computed solution is compared with that reported in [30]. In order to show the approximation and generalization capabilities of the proposed method, we compare the exact solution with the computed solution inside and outside of the domain of consideration. Furthermore, the proposed method is successfully applied to solve Problem 1 for a very large time interval. For all experiments, the RBF kernel is used, $K(u, v) = \exp(-(u-v)^2/\sigma^2)$, so the following relations hold:

$$\nabla_1^0[K(u,v)] = -\frac{2(u-v)}{\sigma^2}K(u,v)$$
$$\nabla_0^1[K(u,v)] = \frac{2(u-v)}{\sigma^2}K(u,v)$$
$$\nabla_2^0[K(u,v)] = \left[\frac{4(u-v)^2}{\sigma^4} - \frac{2}{\sigma^2}\right]K(u,v).$$

MATLAB 2010b is used to implement the code and all computations were carried out on a windows 7 system with Intel-core i7 CPU and 4.00 GB RAM.

TABLE I
Numerical Results of the Proposed Method for Solving Problems 2–4

Problem	Domain	$\|y - \hat{y}\|_\infty$	MSE	STD
2	**Inside**	4.56×10^{-3}	1.47×10^{-6}	1.16×10^{-3}
	Outside	4.62×10^{-1}	3.85×10^{-2}	1.56×10^{-1}
3	**Inside**	5.43×10^{-3}	8.94×10^{-6}	1.60×10^{-3}
	Outside	8.46×10^{-2}	1.49×10^{-3}	2.27×10^{-2}
4	**Inside**	1.46×10^{-4}	8.15×10^{-9}	3.90×10^{-5}
	Outside	6.76×10^{-2}	5.53×10^{-4}	2.20×10^{-2}

Note: STD is the standard deviation.

A. First-Order ODEs

Problem 1: Consider the following first-order ODE [19, eq. 2]:

$$\frac{d}{dt}y(t) + 2y(t) = \sin(t) \quad y(0) = 1, \quad t \in [0, 10].$$

The approximate solution obtained by the proposed method is compared with the true solution and results are depicted in Fig. 1. From the obtained results, it is apparent that our method outperforms the method in [19] in terms of accuracy (see [19, Fig. 6]), although training was performed using fewer points (one fourth). In addition, we also considered points outside the training interval, and Fig. 1(d) and (e) shows that the extrapolation error remains low for the points near the domain of equation. As expected, by increasing the number of mesh points (training points), the error decreases both inside and outside of the training interval. Fig. 1(c) and (f) indicates the performance of the method when nonuniform partitioning is used for creating training points.

Problem 2: First-order differential equation with nonlinear sinusoidal excitation [19, eq. 3]

$$\frac{d}{dt}y(t) + 2y(t) = t^3 \sin\left(\frac{t}{2}\right) \quad y(0) = 1, \quad t \in [0, 10].$$

The interval $[0, 10]$ is discretized into $N = 20$ points $t_1 = 0, \ldots, t_{20} = 10$ using the grid $t_i = (i-1)h$, $i = 1, \ldots, N$, where $h = (10/N - 1)$. In Fig. 2(a), we compare the exact solution with the computed solution at grid points (circles) as well as for other points inside and outside the domain of equation. The obtained absolute errors for points inside and outside the domain $[0, 10]$ are tabulated in Table I. It is clear that the solution is of higher accuracy compared to the solution obtained in [19], despite the fact that fewer training points are used. (Note that in [19], 100 equidistant points are used for

TABLE II
NUMERICAL RESULTS OF THE PROPOSED METHOD FOR SOLVING PROBLEMS 5–7

Problem	Domain	Variable	$\|y-\hat{y}\|_\infty$	MSE	STD
5	**Inside**	y	1.14×10^{-6}	4.16×10^{-13}	6.43×10^{-7}
		y'	4.81×10^{-5}	6.78×10^{-11}	8.21×10^{-6}
	Outside	y	4.20×10^{-2}	2.64×10^{-4}	1.26×10^{-2}
		y'	1.00×10^{-1}	3.27×10^{-3}	3.87×10^{-2}
6	**Inside**	y	5.88×10^{-6}	1.49×10^{-11}	1.63×10^{-6}
		y'	7.34×10^{-6}	2.18×10^{-11}	3.28×10^{-6}
	Outside	y	3.96×10^{-4}	2.39×10^{-8}	1.19×10^{-4}
		y'	5.15×10^{-4}	7.11×10^{-8}	1.74×10^{-4}
7	**Inside**	y	6.64×10^{-9}	2.01×10^{-16}	4.07×10^{-9}
		y'	8.41×10^{-8}	1.30×10^{-15}	3.59×10^{-8}
	Outside	y	6.51×10^{-2}	3.90×10^{-4}	1.65×10^{-2}
		y'	7.80×10^{-2}	7.31×10^{-4}	2.15×10^{-2}

Note: STD is the standard deviation.

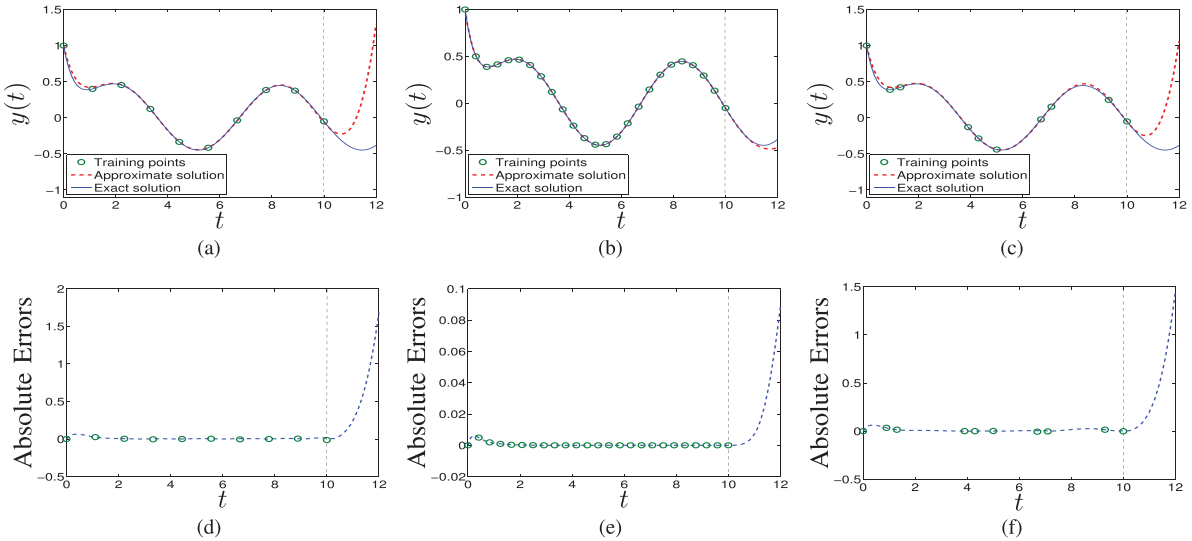


Fig. 1. Numerical results for Problem 1. (a) Ten equidistant points in [0, 10] are used for training. (b) 25 equidistant points in [0, 10] are used for training. (c) Nonuniform partitions of [0, 10] using ten points, which are used for training. (d) Obtained absolute errors on the interval [0, 12] when [0, 10] is discretized into nine equal parts. (e) Obtained absolute errors on the interval [0, 12] when [0, 10] is discretized into 24 equal parts. (f) Obtained absolute errors on the interval [0, 12] when [0, 10] is discretized into nine nonuniform parts.

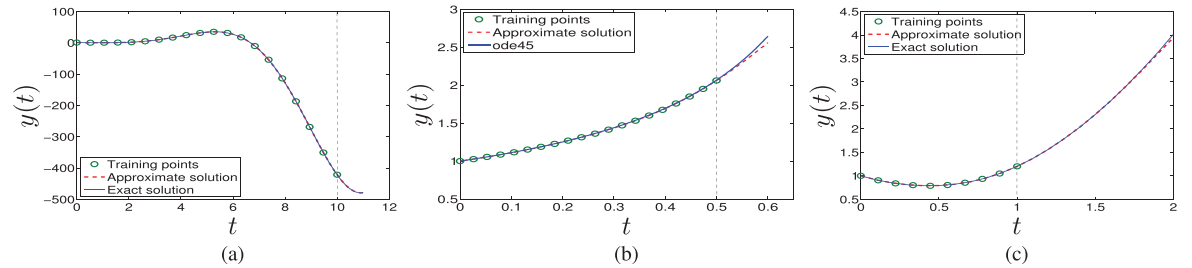


Fig. 2. (a) Numerical results for Problem 2. Twenty equidistant points in [0, 10] are used for training. (b) Numerical results for Problem 3. Twenty equidistant points in [0, 0.5] are used for training. (c) Numerical results for Problem 4. Ten equidistant points in [0, 1] are used for training.

training and the maximum absolute error shown in [19, Fig. 13] is approximately 25×10^{-2}.)

Problem 3: Consider the following nonlinear first-order ODE, which has no analytic solution [19, eq. 6]:

$$\frac{d}{dt}y(t) = y(t)^2 + t^2, \quad y(0) = 1, \quad t \in [0, 0.5].$$

Twenty equidistant points in the given interval are used for the training process. The obtained approximate solution by the proposed method and the solution obtained by MATLAB built-in solver ODE45 are displayed in Fig. 2(b). The obtained absolute errors for points inside and outside the domain [0, 0.5] are tabulated in Table I. The proposed method shows a better

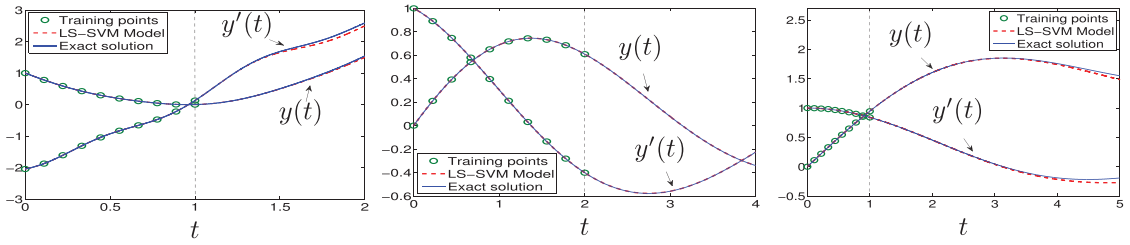


Fig. 3. (a) Numerical results for Problem 5. Ten equidistant points in [0, 1] are used for training. (b) Numerical results for Problem 6. Ten equidistant points in [0, 2] are used for training. (c) Numerical results for Problem 7. Ten equidistant points in [0, 1] are used for training.

performance in comparison with the described method in [19] in terms of accuracy, despite the fact that much less number of training points are used. (Note that in [19], the problem is solved over domain $[0, 0.2]$ by using 100 equidistant training points and the maximum absolute error shown in [19, Fig. 22] is approximately 4×10^{-2}.)

Problem 4: Consider the following first-order ODE with time varying coefficient [18, eq. 1]:

$$\frac{d}{dt}y(t) + \left(t + \frac{1+3t^2}{1+t+t^3}\right)y(t) = t^3 + 2t + t^2\frac{1+3t^2}{1+t+t^3}$$
$$y(0) = 1, \quad t \in [0, 1].$$

In order to have a fair comparison with the results reported in [18], ten equidistant points in the given interval are used for the training process. The analytic solution and obtained solution via our proposed method are displayed in Fig. 2(c). The obtained absolute errors for points inside and outside the domain [0, 1] are recorded in Table I, which shows the superiority of the proposed method over the described method in [18]. (Note that in [18, Fig. 2], the maximum absolute error outside the domain [0, 1] is approximately 12×10^{-2}.)

B. Second-Order ODEs

Problem 5: Consider the following second-order BVP with time-varying input signal [19, eq. 4]:

$$\frac{d^2}{dt^2}y(t) + y(t) = 2 + 2\sin(4t)\cos(3t)$$
$$y(0) = 1, \quad y(1) = 0.$$

Ten equidistant points in the given interval are used for the training process. The analytic solution and the obtained solution via our proposed method are displayed in Fig. 3(a). The obtained absolute errors for points inside and outside the domain [0, 1] are recorded in Table II, which shows the superiority of the proposed method over the described method in [19]. (Note that in [19], 100 equidistant points are used for training and the maximum absolute error shown in [19, Fig. 17] is approximately 5×10^{-1}.)

Problem 6: Consider the following second-order ODE with time-varying input signal [18, Problem 3]:

$$\frac{d^2}{dt^2}y(t) + \frac{1}{5}\frac{d}{dt}y(t) + y(t) = -\frac{1}{5}e^{(\frac{-t}{5})}\cos(t)$$
$$y(0) = 1, \quad y'(0) = 1.$$

TABLE III
NUMERICAL RESULT OF THE PROPOSED METHOD FOR SOLVING PROBLEM 1 WITH TIME INTERVAL $[0, 10^5]$. N IS THE NUMBER OF LOCAL COLLOCATION POINTS, AND S IS THE NUMBER SUB-DOMAINS

N	S	CPU time	MSE	
			Training	Test
20	1000	5.5	2.4×10^{-2}	7.2×10^{-2}
	2000	10.6	1.3×10^{-3}	3.3×10^{-3}
	5000	29.5	8.4×10^{-8}	2.3×10^{-7}
30	1000	6.6	2.2×10^{-2}	5.9×10^{-2}
	2000	13.4	4.1×10^{-6}	1.3×10^{-5}
	5000	37.1	8.2×10^{-9}	2.7×10^{-8}
40	1000	9.6	5.8×10^{-4}	1.4×10^{-3}
	2000	20.1	1.7×10^{-7}	5.8×10^{-7}
	5000	54.2	2.3×10^{-9}	8.1×10^{-9}

Note: The execution time is in seconds.

Ten equidistant points in the interval [0, 2] are used for the training process. The analytic solution and the obtained solution by the proposed method are shown in Fig. 3(b). The obtained absolute errors for points inside and outside the domain [0, 2] are tabulated in Table II, which again shows the improvement of the proposed method over the described method in [18]. (Note that in [18, Fig. 4], the maximum absolute error outside the domain $[0, 2]$ is 8×10^{-4}.)

Problem 7: Consider the following singular second-order ODE, which has no analytical closed-form solution [30, eq. 1]:

$$\frac{d^2}{dt^2}y(t) + \frac{1}{t}\frac{d}{dt}y(t) - \frac{1}{t}\cos(t) = 0, \quad y(0) = 0, \quad y'(0) = 1.$$
$$\text{Exact solution} \quad y(t) = \int_0^t \frac{\sin(x)}{x}\,dx.$$

Ten equidistant points in the interval [0, 1] are used as training points, and the obtained results are shown in Fig. 3(c) and recorded in Table II. The obtained maximum absolute error outside the domain [0, 1] is 6.51×10^{-2}, which is smaller than 14×10^{-1} shown in [30, Fig. 6].

C. Sensitivity of the Solution w.r.t the Parameter

In order to illustrate the sensitivity of the result with respect to the parameter of the model (σ), for two examples, we have plotted the MSE, on the validation set, versus the kernel

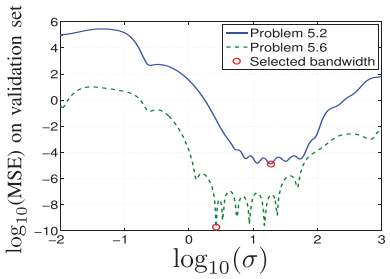


Fig. 4. Sensitivity of the obtained result with respect to model parameter σ. $\log_{10}$ (MSE) versus $\log_{10}\sigma$ is plotted for Problems 2 and 6.

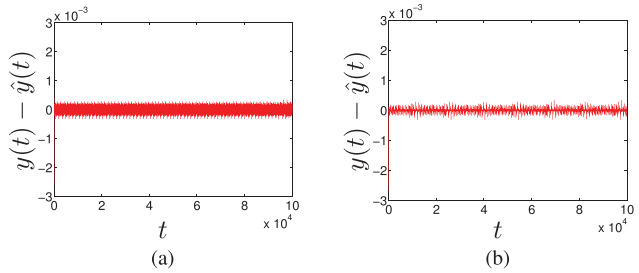


Fig. 5. (a) Residual $y(t) - \hat{y}(t)$ when Problem 1 is solved on the interval $[0, 10^5]$, by using 5000 sub-intervals and 50 local collocation points. (b) Obtained residual $y(t) - \hat{y}(t)$ for the same problem by using 500 sub-intervals and 500 local collocation points.

bandwidth on logarithmic scales in Fig. 4. From this figure, it is apparent that there exists a range of σ for which the MSE on the validation set is quite small.

D. Large Interval

Let us consider Problem 1 when the time interval is $[0, 10^5]$. It is known in advance that the solution of this problem is oscillating. The problem is solved by decomposing the given domain of interest into S sub-domains. Then the problem is solved on each sub-domain using N number of local collocation points. The execution time and the MSE for the training and test sets

$$\text{MSE}_{\text{train}} = \frac{\sum_{i=1}^{N\times S}(y(t_i) - \hat{y}(t_i))^2}{N \times S}$$
$$\text{MSE}_{\text{test}} = \frac{\sum_{i=1}^{M}(y(t_i) - \hat{y}(t_i))^2}{M}$$

where $N \times S$ is the total number of collocation points, and M is the total number of test points over the interval $[0, 10^5]$, are tabulated in Table III. The test set is the same for all the cases and it consists of $M = 5 \times 10^5$ points. It is apparent that when S is fixed and N increases, the accuracy is improved whereas the execution time is increased. The same pattern is observed when N is fixed and S increases. Fig. 5(a) and (b) shows the residual error $e_t = y(t) - \hat{y}(t)$ when Problem 1 is solved over the interval $[0, 10^5]$, using $N = 50$ local collocation points, $S = 5000$ sub-domains and $N = 500$, collocation points, $S = 500$ sub-domains, respectively. It should be noted that the result depicted in Fig. 5(a) is obtained much faster than that shown in Fig. 5(b).

TABLE IV
NUMERICAL RESULTS OF THE PROPOSED METHOD FOR SOLVING PROBLEM 1 WITH TIME INTERVAL [0, 4000], WHILE TOTAL NUMBER OF COLLOCATION POINTS i.e., $N \times S$ IS CONSTANT

N	S	CPU time	MSE	
			Training	Test
800	10	85.5	1.36×10^{-8}	2.06×10^{-8}
400	20	26.1	1.37×10^{-8}	2.08×10^{-8}
20	400	2.06	1.68×10^{-8}	2.52×10^{-8}

Note: The execution time is in seconds.

In Table IV, we analyze the situation where the total number of collocation points i.e., $N \times S$ in the given interval [0, 4000] is fixed. It can be seen that as the number of sub-domains increases (number of collocation points in each sub-domain increases), the computational time decreases without losing the order of accuracy. In this case, the test set consists of $M = 2 \times 10^4$ points.

VII. CONCLUSION

In this paper, a new method based on LS-SVMs was developed for solving general linear m-th order ODEs and also first-order nonlinear ODEs. On the tested problems, the method proposed in this paper is more efficient compared to methods described in [18] and [19]. Also, the proposed method is able to solve a differential equation for a large time interval, while predicting the solution with high accuracy.

In the future, the proposed method may be extended to solve a system of ODEs and partial differential equations.

REFERENCES

[1] J. Cheng, M. R. Sayeh, M. R. Zargham, and Q. Cheng, "Real-time vector quantization and clustering based on ordinary differential equations," *IEEE Trans. Neural Netw.*, vol. 22, no. 12, pp. 2143–2148, Dec. 2011.

[2] C. Y. Lai, C. Xiang, and T. H. Lee, "Data-based identification and control of nonlinear systems via piecewise affine approximation," *IEEE Trans. Neural Netw.*, vol. 22, no. 12, pp. 2189–2200, Dec. 2011.

[3] J. C. Butcher, *Numerical Methods for Ordinary Differential Equations*, 2nd ed. Chichester, U.K.: Wiley, 2008.

[4] J. D. Lambert, *Numerical Methods for Ordinary Differential Systems*. New York: Wiley, 1991.

[5] M. M. Chawla and C. P. Katti, "Finite difference methods for two-point boundary value problems involving high order differential equations," *BIT Numer. Math.*, vol. 19, no. 1, pp. 27–33, 1979.

[6] R. D. Russell and L. F. Shampine, "A collocation method for boundary value problems," *Numer. Math.*, vol. 19, no. 1, pp. 1–28, 1972.

[7] C. W. Gear, "Hybrid methods for initial value problems in ordinary differential equations," *SIAM J. Numer. Anal.*, vol. 2, no. 1, pp. 69–86, 1965.

[8] D. Sarafyan, "New algorithm for the continuous approximate solution of ordinary differential equations and the upgrading of the order of the processes," *Comput. Math. Appl.*, vol. 20, no. 1, pp. 71–100, 1990.

[9] S. N. Jator and J. Li, "A self-starting linear multistep method for a direct solution of the general second-order initial value problem," *Int. J. Comput. Math.*, vol. 86, no. 5, pp. 827–836, 2009.

[10] D. O. Awoyemi, "A new sixth-order algorithm for general second order ordinary differential equation," *Int. J. Comput. Math.*, vol. 77, no. 1, pp. 117–124, 2001.

[11] A. J. Meade and A. A. Fernadez, "The numerical solution of linear ordinary differential equations by feedforward neural networks," *Math. Comput. Model.*, vol. 19, no. 12, pp. 1–25, 1994.

[12] H. Lee and I. Kang, "Neural algorithms for solving differential equations," *J. Comput. Phys.*, vol. 91, no. 1, pp. 110–117, 1990.

[13] B. P. van Milligen, V. Tribaldos, and J. A. Jiménez, "Neural network differential equation and plasma equilibrium solver," *Phys. Rev. Lett.*, vol. 75, no. 20, pp. 3594–3597, 1995.

[14] L. P. Aarts and P. Van der Veer, *Solving Nonlinear Differential Equations by a Neural Network Method* (Lecture Notes in Computer Science), vol. 2074. New York: Springer-Verlag, 2001, pp. 181–189.

[15] P. Ramuhalli, L. Udpa, and S. S. Udpa, "Finite-element neural networks for solving differential equations," *IEEE Trans. Neural Netw.*, vol. 16, no. 6, pp. 1381–1392, Nov. 2005.

[16] 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 Trans. Neural Netw.*, vol. 20, no. 8, pp. 1221–1233, Aug. 2009.

[17] I. G. Tsoulos, D. Gavrilis, and E. Glavas, "Solving differential equations with constructed neural networks," *Neurocomputing*, vol. 72, nos. 10–12, pp. 2385–2391, 2009.

[18] I. Lagaris, A. Likas, and D. I. Fotiadis, "Artificial neural networks for solving ordinary and partial differential equations," *IEEE Trans. Neural Netw.*, vol. 9, no. 5, pp. 987–1000, Sep. 1998.

[19] H. S. Yazdi, M. Pakdaman, and H. Modaghegh, "Unsupervised kernel least mean square algorithm for solving ordinary differential equations," *Neurocomputing*, vol. 74, nos. 12–13, pp. 2062–2071, 2011.

[20] B. Schölkopf and A. Smola, *Learning with Kernels*. Cambridge, MA: MIT Press, 2002.

[21] V. Vapnik, *Statistical Learning Theory*. New York: Wiley, 1998.

[22] J. A. K. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural Process. Lett.*, vol. 9, no. 3, pp. 293–300, 1999.

[23] J. A. K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle, *Least Squares Support Vector Machines*. Singapore: World Scientific, 2002.

[24] J. A. K. Suykens, J. Vandewalle, and B. De Moor, "Optimal control by least squares support vector machines," *Neural Netw.*, vol. 14, no. 1, pp. 23–35, 2001.

[25] K. De Brabanter, J. De Brabanter, J. A. K. Suykens, and B. De Moor, "Approximate confidence and prediction intervals for least squares support vector regression," *IEEE Trans. Neural Netw.*, vol. 22, no. 1, pp. 110–120, Jan. 2011.

[26] H. Ning, X. Jing, and L. Cheng, "Online identification of nonlinear spatiotemporal systems using kernel learning approach," *IEEE Trans. Neural Netw.*, vol. 22, no. 9, pp. 1381–1394, Sep. 2011.

[27] M. Lázaro, I. Santamaria, F. Pérez-Cruz, and A. Artés-Rodriguez, "Support vector regression for the simultaneous learning of a multivariate function and its derivative," *Neurocomputing*, vol. 69, nos. 1–3, pp. 42–61, 2005.

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

[29] A. Toselli and O.B. Widlund, *Domain Decomposition Methods-Algorithms and Theory*. Berlin, Germany: Springer-Verlag, 2005.

[30] I. G. Tsoulos and I. E. Lagaris, "Solving differential equations with genetic programming," *Genetic Program. Evolvable Mach.*, vol. 7, no. 1, pp. 33–54, 2006.



Siamak Mehrkanoon received the B.S. degree in pure mathematics in 2005 and the M.S. degree in applied mathematics from the Iran University of Science and Technology, Tehran, Iran, in 2007. He is currently pursuing the Ph.D. degree with the Department of Electrical Engineering, Katholieke Universiteit Leuven, Leuven, Belgium.

His current research interests include machine learning, system identification, pattern recognition, and numerical algorithms.



Tillmann Falck received the Dipl.Ing. degree in electrical engineering from Ruhr University Bochum, Bochum, Germany, in 2007. He is pursuing the Ph.D. degree in electrical engineering from Katholieke Universiteit Leuven, Leuven, Belgium.

He has been with Robert Bosch GmbH, Wuerttemberg, Germany, since 2012. His research interests include nonlinear system identification, convex optimization, machine learning, and tracking and driver assistance systems.



Johan A. K. Suykens (SM'05) was born in Willebroek, Belgium, on May 18, 1966. He received the M.S. degree in electromechanical engineering and the Ph.D. degree in applied sciences from the Katholieke Universiteit Leuven, Leuven, Belgium, in 1989 and 1995, respectively.

He was a Visiting Post-Doctoral Researcher with the University of California, Berkeley, in 1996. He has been a Post-Doctoral Researcher with the Fund for Scientific Research FWO Flanders and is currently a Professor in Hoogleraar with K.U. Leuven. He has authored the books *Artificial Neural Networks for Modelling and Control of Non-linear Systems* (Kluwer Academic Publishers) and *Least Squares Support Vector Machines* (World Scientific), co-authored the book *Cellular Neural Networks, Multi-Scroll Chaos and Synchronization* (World Scientific), and edited the books *Nonlinear Modeling: Advanced Black-Box Techniques* (Kluwer Academic Publishers) and *Advances in Learning Theory: Methods, Models and Applications* (IOS Press). In 1998, he organized an International Workshop on Nonlinear Modeling with Time-series Prediction Competition.

Dr. Suykens has served as an Associate Editor for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS from 1997 to 1999 and from 2004 to 2007, and for the IEEE TRANSACTIONS ON NEURAL NETWORKS from 1998 to 2009. He was a recipient of the IEEE Signal Processing Society Best Paper (Senior) Award in 1999 and several Best Paper Awards at International Conferences. He was a recipient of the International Neural Networks Society INNS 2000 Young Investigator Award for significant contributions in the field of neural networks. He has served as a Director and Organizer of the NATO Advanced Study Institute on Learning Theory and Practice (Leuven, 2002), as a Program Co-Chair for the *International Joint Conference on Neural Networks* in 2004 and the International Symposium on Nonlinear Theory and its Applications 2005, as an organizer of the International Symposium on Synchronization in Complex Networks in 2007 and a co-organizer of the NIPS 2010 workshop on Tensors, Kernels and Machine Learning. He was awarded an ERC Advanced Grant in 2011.