Neurocomputing 159 (2015) 105–116

Contents lists available at ScienceDirect

Neurocomputing

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

Learning solutions to partial differential equations using LS-SVM

Siamak Mehrkanoon*, Johan A.K. Suykens

KU Leuven, ESAT-STADIUS, Kasteelpark Arenberg 10, B-3001 Leuven (Heverlee), Belgium

ARTICLE INFO

Article history:
Received 28 August 2014
Received in revised form
7 February 2015
Accepted 7 February 2015
Communicated by Wei Chiang Hong
Available online 19 February 2015

Keywords:
Least squares support vector machines
Partial differential equations
Closed form approximate solution
Collocation method

ABSTRACT

This paper proposes an approach based on Least Squares Support Vector Machines (LS-SVMs) for solving second order partial differential equations (PDEs) with variable coefficients. Contrary to most existing techniques, the proposed method provides a closed form approximate solution. The optimal representation of the solution is obtained in the primal–dual setting. The model is built by incorporating the initial/boundary conditions as constraints of an optimization problem. The developed method is well suited for problems involving singular, variable and constant coefficients as well as problems with irregular geometrical domains. Numerical results for linear and nonlinear PDEs demonstrate the efficiency of the proposed method over existing methods.

© 2015 Elsevier B.V. All rights reserved.

1. Introduction

Partial differential equations (PDEs) are widely used in the mathematical modelling of scientific and engineering problems. In most applications the analytic solutions of the underlying PDEs are not available and therefore numerical methods must be applied. For that reason, a number of numerical methods such as Finite Difference methods (FDM) [1], Finite Element methods (FEM) [2], Splines [3,4], and methods based on feedforward neural network [5–9] and genetic programming approaches [10–12] have been developed. Some other related works for the numerical solution of PDEs are for instance Spectral based methods [13], Boundary Integral Equation methods (BIE) [14] and Boundary Element methods [15]. A combined finite volume method and spectral element technique for solving an unsteady magnetohydrodynamic equation is also introduced in [16]. A method based on Radial Basis Functions for solving PDEs on arbitrary surfaces is discussed in [17].

The finite difference methods provide the solution at specific preassigned mesh points only (discrete solution) and they need an additional interpolation procedure to yield the solution for the whole domain. Furthermore conditional stability as well as the lower accuracy on irregular domains limits the applicability of these methods.

The finite-element method (FEM) is the most popular discretization method in engineering applications. An important feature of the FEM is that it requires a discretization of the domain via meshing, and therefore belongs to the class of mesh-based methods. For problems involving complex geometries or higher dimensional problems, generating a mesh might be challenging and is indeed the most time consuming part of the solution process. Therefore in case of three or higher dimensional problems, FEM requires high memory. In addition this approach approximates the functions locally, therefore it provides the solution at mesh points only and additional interpolation is required in order to find the solution at an arbitrary point in the domain.

In contrast with mesh-based approaches such as finite difference and finite element methods, the solution obtained from neural network approaches (see [18–20]) is in closed form (continuous and differentiable) and it does not require a mesh topology. In addition it can achieve the desired accuracy even if the domain of interest is presented by scattered discrete points (therefore it can be referred to as mesh-less approach). It has been shown in [8] that neural approach requires less number of parameter to achieve the same accuracy as with FEM on the grid points. Moreover for FEM approach it has been observed that the accuracy at arbitrary points on the domain is order of magnitude lower that of training points.

Despite the fact that the classical neural networks have nice properties such as universal approximation, they still suffer from having two persistent weak points. The first problem is the existence of many local minima solutions. The second problem is how to choose the number of hidden units.

* Corresponding author.
E-mail addresses: siamak.mehrkanoon@esat.kuleuven.be (S. Mehrkanoon), johan.suykens@esat.kuleuven.be (J.A.K. Suykens).

http://dx.doi.org/10.1016/j.neucom.2015.02.013
0925-2312/© 2015 Elsevier B.V. All rights reserved.

Support Vector Machines (SVMs) [21,22] have been successful for solving pattern recognition and function estimation problems. SVM solutions for function estimation are characterized by convex quadratic programming problems in the dual. Therefore the weak points of the classical neural network approaches have been overcome. Least squares SVMs (LS-SVMs) [23], in classification and regression, have been proposed as a way to replace the quadratic programming problems by solving a linear system. This is achieved by applying a least square loss function in the objective function and changing the inequality constraints to equality constraints [24].

LS-SVM approaches have been primarily successfully applied for solving initial and boundary value problems for ordinary differential equations (ODEs) and differential algebraic equations (DAEs) [25,26] as well as parameter estimation of dynamical systems [27,28]. It is the purpose of this paper to extend the method developed in [25] to approximate the solution of one dimensional second order time dependent PDEs.

We propose a kernel based method in the LS-SVM framework. It should be noted that one can derive a kernel based model in two ways: one is using a primal–dual setting and the other one is by using function estimation in a reproducing kernel Hilbert space and the corresponding representer theorem. The primal–dual approach has the advantage that it is usually straightforward to incorporate additional structure or knowledge into the estimation problem. For instance in the context of learning the solution of PDEs, one may know in advance that the underlying solution has to satisfy an additional constraint (like non-local conservation condition [29]). Then one can incorporate it to the estimation problem by adding a suitable constraint. Furthermore, contrary to the classical mesh-free approaches, the primal and dual formulation of the method allows to obtain the optimal representation of the solution. That means in the primal one starts with a simple representation of the solution and by incorporating the initial/boundary conditions and the given PDE, one may obtain the optimal representation of the solution in the dual. That is in contrast with most existing approaches that produce a closed form solution. More precisely, unlike the approach described in [8] that the user has to define a form of a trial solution, which in some cases is not straightforward, in the proposed approach the optimal model is derived by incorporating the initial/boundary conditions as constraints of an optimization problem.

This paper is organized as follows. In Section 2, a brief review about least squares support vector regression problems together with preliminary definitions is given. In Section 3, we formulate our least squares support vector machines method to the solution of second order linear time dependent hyperbolic PDEs in one space dimension. The formulation of the method for nonlinear PDEs is discussed in Section 4. Section 5 describes the results of numerical experiments, discussion and comparison with other known methods.

2. LS-SVM regression and preliminary definitions

Let us consider a given training set $\{z_i, y_i\}_{i=1}^N$ with input data $z_i \in \mathbb{R}^m$ and output data $y_i \in \mathbb{R}$. The goal in regression is to estimate a model of the form $\hat{y} = w^T\varphi(z)+d$.

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

$$\begin{aligned} &\underset{w,d,e}{\text{minimize}} \quad \frac{1}{2}w^Tw+\frac{\gamma}{2}e^Te \\ &\text{subject to} \quad y_i = w^T\varphi(z_i)+d+e_i, \quad i=1,\ldots,N \end{aligned} \tag{1}$$

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

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

where $\Omega_{ij} = K(z_i, z_j) = \varphi(z_i)^T\varphi(z_j)$ is the (i,j)-th entry of the positive definite kernel matrix for a positive definite kernel function $K(z,r) = \varphi(z)^T\varphi(r)$ and $1_N = [1;\ldots;1] \in \mathbb{R}^N$. The vector of Lagrange multipliers is denoted by $\alpha = [\alpha_1;\ldots;\alpha_N]$, $y = [y_1;\ldots;y_N]$ and I_N is the identity matrix. When we deal with differential equations, the target values y_i are no longer directly available, so the regression approach is not directly applicable. Nevertheless we can incorporate the underlying differential equation in the learning process to find an approximation for the solution.

First 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. Without loss of generality let us assume that $d=2$, i.e. the training points $z_i \in \mathbb{R}^2$. Suppose that $z_1 = (x_1, t_1)^T$ and $z_2 = (x_2, t_2)^T$ are two arbitrary points in $\mathbb{R}^2$ (x, t-coordinates). Then let us define the following differential operator which will be used in subsequent sections

$$\nabla_{s^{(n)},p^{(m)}} \equiv \frac{\partial^{n+m}}{\partial s^n \partial p^m}. \tag{2}$$

If $\varphi(z_1)^T\varphi(z_2) = K(z_1, z_2)$, then one can show that

$$\begin{aligned} \left[\varphi_{x^{(n)}}(z_1)\right]^T\varphi_{x^{(m)}}(z_2) &= \nabla_{x_1^{(n)},x_2^{(m)}}[K(z_1,z_2)] = \frac{\partial^{n+m}K(z_1,z_2)}{\partial x_1^n \partial x_2^m}, \\ \left[\varphi_{x^{(n)}}(z_1)\right]^T\varphi_{t^{(m)}}(z_2) &= \nabla_{x_1^{(n)},t_2^{(m)}}[K(z_1,z_2)] = \frac{\partial^{n+m}K(z_1,z_2)}{\partial x_1^n \partial t_2^m}, \\ \left[\varphi_{t^{(n)}}(z_1)\right]^T\varphi_{t^{(m)}}(z_2) &= \nabla_{t_1^{(n)},t_2^{(m)}}[K(z_1,z_2)] = \frac{\partial^{n+m}K(z_1,z_2)}{\partial t_1^n \partial t_2^m}, \\ \left[\varphi_{t^{(n)}}(z_1)\right]^T\varphi_{x^{(m)}}(z_2) &= \nabla_{t_1^{(n)},x_2^{(m)}}[K(z_1,z_2)] = \frac{\partial^{n+m}K(z_1,z_2)}{\partial t_1^n \partial x_2^m}. \end{aligned} \tag{3}$$

Here $\varphi_{x^{(n)}}$ and $\varphi_{t^{(n)}}$ are the n-th derivative of the feature map φ with respect to variable x and t respectively. Note that if either m or n is zero, we do not take the derivative of the term w.r.t. to the corresponding variable. More precisely suppose $m=0$ then we use the following notations:

$$\begin{aligned} \left[\varphi_{x^{(n)}}(z_1)\right]^T\varphi_{x^{(0)}}(z_2) &= \nabla_{x_1^{(n)},x_2^{(0)}}[K(z_1,z_2)] \\ &\triangleq \nabla_{x_1^{(n)},0}[K(z_1,z_2)] = \frac{\partial^n K(z_1,z_2)}{\partial x_1^n}. \end{aligned}$$

For instance if K is chosen to be the RBF kernel

$$K(z_1,z_2) = \exp\left(-\frac{\|z_1-z_2\|^2}{\sigma^2}\right)$$

then the following relations hold:

$$\begin{aligned} \left[\varphi(z_1)\right]^T\varphi_x(z_2) &= \nabla_{0,x_2}[K(z_1,z_2)] = \frac{2(x_1-x_2)}{\sigma^2}K(z_1,z_2), \\ \left[\varphi(z_1)\right]^T\varphi_t(z_2) &= \nabla_{0,t_2}[K(z_1,z_2)] = \frac{2(t_1-t_2)}{\sigma^2}K(z_1,z_2). \end{aligned}$$

3. Formulation of the method

The general form of a linear second-order PDE with two independent variables x and t is

$$a\frac{\partial^2 u}{\partial x^2}+b\frac{\partial^2 u}{\partial x \partial t}+c\frac{\partial^2 u}{\partial t^2}+d\frac{\partial u}{\partial x}+e\frac{\partial u}{\partial t}+l_1 u = l_2. \tag{4}$$

The first three terms containing the second derivatives are called the principal part of the PDE. The coefficients of the principal part can be used to classify the PDE into elliptic, parabolic and

hyperbolic. In the case that the coefficients a, b and c are variable (i.e. functions of x or y, or both), then the categorization of the equation could vary throughout the solution region. Consider the one space dimensional linear second order equation with variable coefficients of the following form:

$$\mathcal{L}u(z) = f(z), \quad z \in \Sigma \in \mathbb{R}^2 \tag{5}$$

subject to the boundary conditions of the form

$$\mathcal{B}u(z) = g(z), \quad z \in \partial\Sigma$$

where $u(z) = u(x,t)$, t and x are time and space variables respectively and $z = (x,t)^T$. Σ is a bounded domain, which can be either rectangular or irregular, and $\partial\Sigma$ represent its boundary. $\mathcal{B}$ and $\mathcal{L}$ are differential operators. In this study we consider the case where

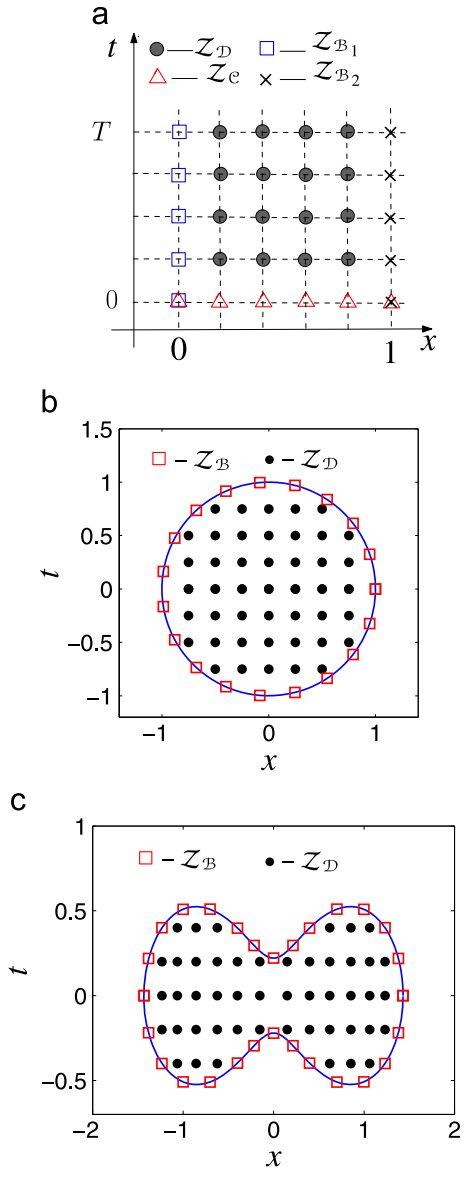


Fig. 1. $\mathcal{Z}_\mathcal{D}$ and $\mathcal{Z}_\mathcal{B}$ are the set of grid points which are located inside and on the boundary of the domain respectively. (a) Grid points used in the learning process for the rectangular domain. (b) Grid points used in the learning process for the circular domain. (c) Grid points used in the learning process for the irregular domain.

$\mathcal{L}$ is defined as follows:

$$\mathcal{L} \equiv \frac{\partial^2 u}{\partial t^2} + a(x,t)\frac{\partial u}{\partial t} + b(x,t)u - c(x,t)\frac{\partial^2 u}{\partial x^2}. \tag{6}$$

Remark 3.1. It should be noted that the presented approach can be applied for the general linear second order PDE (4) but for the sake of notational simplicity, the method is given for the differential operator $\mathcal{L}$ in (6).

Let us assume that a general approximate solution to (5) is of the form of $\hat{u}(z) = w^T\varphi(z) + d$, where w and d are parameters of the model that have to be determined. To obtain the optimal value of these parameters, collocation methods can be used which assume a discretization of the domain Σ into a set of collocation points defined as follows:

$$\mathcal{Z} = \left\{ z^k \middle| z^k = (x_k, t_k),\ k = 1, \ldots, k_{end} \right\},$$

where k_{end} is a user defined number. Let us decompose $\mathcal{Z}$ into two disjoint non-empty sets $\mathcal{Z}_\mathcal{D}$ and $\mathcal{Z}_\mathcal{B}$, i.e. $\mathcal{Z} = \mathcal{Z}_\mathcal{D} \cup \mathcal{Z}_\mathcal{B}$, where $\mathcal{Z}_\mathcal{D} = \{z_\mathcal{D}^i\}_{i=1}^{|\mathcal{Z}_\mathcal{D}|}$ and $\mathcal{Z}_\mathcal{B} = \{z_\mathcal{B}^i\}_{i=1}^{|\mathcal{Z}_\mathcal{B}|}$. $\mathcal{Z}_\mathcal{D}$ denotes the set of collocation points located inside the domain and $\mathcal{Z}_\mathcal{B}$ represents the collocation points situated on the boundary. Therefore the adjustable parameters w and d are to be found by solving the following optimization problem:

$$\begin{aligned} &\underset{\hat{u}}{\text{minimize}} \quad \frac{1}{2}\sum_{i=1}^{|\mathcal{Z}_\mathcal{D}|} \left[(\mathcal{L}[\hat{u}] - f)(z_\mathcal{D}^i) \right]^2 \\ &\text{subject to} \quad \mathcal{B}[\hat{u}(z_\mathcal{B}^j)] = g(z_\mathcal{B}^j),\ j = 1, \ldots, |\mathcal{Z}_\mathcal{B}|. \end{aligned} \tag{7}$$

Here $|\mathcal{Z}_\mathcal{D}|$ and $|\mathcal{Z}_\mathcal{B}|$ are the cardinality of sets $\mathcal{Z}_\mathcal{D}$ and $\mathcal{Z}_\mathcal{B}$ respectively. Furthermore $|\mathcal{Z}_\mathcal{D}| + |\mathcal{Z}_\mathcal{B}|$ is equal to the number of training points used in the learning process. In what follows we formulate the optimization problem in the LS-SVM framework for solving linear second order time varying partial differential equation given in (5) and (6). Suppose that $z_i \in \mathcal{S}$ and $z_j \in \mathcal{T}$ are two arbitrary points and $\mathcal{S}, \mathcal{T} \subseteq \mathbb{R}^2$. Now for notational convenience let us list the following notations which are used in the following sections:

$$\begin{aligned} &\left[\Omega_{s^{(n)},p^{(m)}}\right]_{ij}^{\mathcal{S},\mathcal{T}} = \left[\nabla_{s^{(n)},p^{(m)}} K\right](z_i, z_j), \\ &\left[\Omega\right]_{ij}^{\mathcal{S},\mathcal{T}} = \left[\nabla_{s^{(0)},p^{(0)}} K\right](z_i, z_j) = \left[\nabla_{0,0} K\right](z_i, z_j) = K(z_i, z_j), \end{aligned}$$

where $\left[\Omega_{s^{(n)},p^{(m)}}\right]_{ij}^{\mathcal{S},\mathcal{T}}$ denotes the (i,j)-th entry of matrix $\left[\Omega_{s^{(n)},p^{(m)}}\right]^{\mathcal{S},\mathcal{T}}$. In the case that $\mathcal{S} = \mathcal{T}$, we denote the matrix by $\left[\Omega_{s^{(n)},p^{(m)}}\right]^{\mathcal{S}}$. Here s and p can take values for any t_1, t_2, x_1 and x_2 combinations see (3).

3.1. PDEs on rectangular domains

Consider the PDE (5), with the operator $\mathcal{L}$ in (6), defined on a rectangular domain subject to the initial conditions of the form

$$u(x,0) + \frac{\partial u(x,0)}{\partial t} = h(x), \quad 0 \le x \le 1 \tag{8}$$

and boundary conditions at $x=0$ and $x=1$ of the form

$$u(0,t) = g_0(t), \quad u(1,t) = g_1(x), \quad 0 \le t \le T. \tag{9}$$

Therefore now the set $\mathcal{Z}_\mathcal{B}$, defined previously, can be written as $\mathcal{Z}_\mathcal{B} = \mathcal{Z}_\mathcal{C} \cup \mathcal{Z}_{\mathcal{B}_1} \cup \mathcal{Z}_{\mathcal{B}_2}$, (see Fig. 1), where

$$\begin{aligned} &\mathcal{Z}_\mathcal{C} = \{(x,0) | \forall x \in [0,1]\}, \\ &\mathcal{Z}_{\mathcal{B}_1} = \{(0,t) | \forall t \in [0,T]\}, \\ &\mathcal{Z}_{\mathcal{B}_2} = \{(1,t) | \forall t \in [0,T]\}. \end{aligned}$$

Furthermore let us assume that $N = |\mathcal{Z}_\mathcal{D}|$ and $M = M_1 + M_2 + M_3 = |\mathcal{Z}_\mathcal{C}| + |\mathcal{Z}_{\mathcal{B}_1}| + |\mathcal{Z}_{\mathcal{B}_2}|$.

In the LS-SVM framework the approximate solution, $\hat{u}(z) = w^T\varphi(z) + d$, can be obtained by solving the following

optimization problem:

$$\begin{aligned}
&\underset{w,d,e}{\text{minimize}} \quad \frac{1}{2}w^Tw+\frac{\gamma}{2}e^Te\\
&\text{subject to} \quad w^T\Big[\varphi_{tt}(z_{\mathcal{D}}^i)+a(z_{\mathcal{D}}^i)\varphi_t(z_{\mathcal{D}}^i)+b(z_{\mathcal{D}}^i)\varphi(z_{\mathcal{D}}^i)\\
&\qquad -c(z_{\mathcal{D}}^i)\varphi_{xx}(z_{\mathcal{D}}^i)\Big]+b(z_{\mathcal{D}}^i)d\\
&\qquad =f(z_{\mathcal{D}}^i)+e_i,\ \ i=1,\ldots,|\mathcal{Z}_{\mathcal{D}}|,\\
&w^T\Big[\varphi(z_{\mathcal{C}}^i)+\varphi_t(z_{\mathcal{C}}^i)\Big]+d=h(x_i),\ \ i=1,\ldots,|\mathcal{Z}_{\mathcal{C}}|,\\
&w^T\varphi(z_{\mathcal{B}_1}^i)+d=g_0(t_i),\ \ i=1,\ldots,|\mathcal{Z}_{\mathcal{B}_1}|,\\
&w^T\varphi(z_{\mathcal{B}_2}^i)+d=g_1(t_i),\ \ i=1,\ldots,|\mathcal{Z}_{\mathcal{B}_2}|,
\end{aligned}\tag{10}$$

where

$$\varphi_t=\frac{\partial\varphi}{\partial t},\quad \varphi_{tt}=\frac{\partial^2\varphi}{\partial t^2},\quad \varphi_{xx}=\frac{\partial^2\varphi}{\partial x^2},\quad \varphi_x=\frac{\partial\varphi}{\partial x}.$$

Problem (10) is obtained by combining the LS-SVM cost function with constraints constructed by imposing the approximate solution $\hat{u}(z)=w^T\varphi(z)+d$, given by the LS-SVM model, to satisfy the given differential equation as well as the initial and boundary conditions at the collocation points. We note here that (10) is a quadratic minimization under linear equality constraints, which enables an efficient solution.

The proof of the following lemma is reported in the appendix.

Lemma 3.1. *Given a positive definite kernel function $K:\mathbb{R}^2\times\mathbb{R}^2\to\mathbb{R}$ with $K(z_1,z_2)=\varphi(z_1)^T\varphi(z_2)$ and a regularization constant $\gamma\in\mathbb{R}^+$, the solution to (10) is given by the following dual problem:*

$$\begin{bmatrix}\mathcal{K}+\gamma^{-1}I_N & S & b\\ S^T & \Delta & 1_M\\ b^T & 1_M^T & 0\end{bmatrix}\begin{bmatrix}\alpha\\ \beta\\ d\end{bmatrix}=\begin{bmatrix}f\\ v\\ 0\end{bmatrix}.\tag{11}$$

The dual model representation of the solution is as follows:

$$\begin{aligned}
\hat{u}(z)=d+&\sum_{i=1}^{|\mathcal{Z}_{\mathcal{D}}|}\alpha_i\Big(\Big[\nabla_{t_1^{(2)},0}K\Big](z_{\mathcal{D}}^i,z)+a(z_{\mathcal{D}}^i)[\nabla_{t_1,0}K](z_{\mathcal{D}}^i,z)\\
&+b(z_{\mathcal{D}}^i)[\nabla_{0,0}K](z_{\mathcal{D}}^i,z)-c(z_{\mathcal{D}}^i)\Big[\nabla_{x_1^{(2)},0}K\Big](z_{\mathcal{D}}^i,z)\Big)\\
&+\sum_{i=1}^{|\mathcal{Z}_{\mathcal{C}}|}\beta_i^1[\nabla_{0,0}K+\nabla_{t_1,0}K](z_{\mathcal{C}}^i,z)\\
&+\sum_{i=1}^{|\mathcal{Z}_{\mathcal{B}_1}|}\beta_i^2[\nabla_{0,0}K](z_{\mathcal{B}_1}^i,z)+\sum_{i=1}^{|\mathcal{Z}_{\mathcal{B}_2}|}\beta_i^3[\nabla_{0,0}K](z_{\mathcal{B}_2}^i,z).
\end{aligned}$$

The elements of (11) are given by

$$\begin{aligned}
&\alpha=[\alpha_1,\ldots,\alpha_N]^T,\quad \beta=[\beta^1,\beta^2,\beta^3]^T\in\mathbb{R}^M,\\
&\beta^1=[\beta_1^1,\ldots,\beta_{M_1}^1],\quad \beta^2=[\beta_1^2,\ldots,\beta_{M_2}^2],\quad \beta^3=[\beta_1^3,\ldots,\beta_{M_3}^3],\\
&v=\mathrm{vec}[H,G_0,G_1]\in\mathbb{R}^M,\quad H=[h(z_{\mathcal{C}}^1),\ldots,h(z_{\mathcal{C}}^{M_1})]^T\in\mathbb{R}^{M_1},\\
&G_0=[g_0(z_{\mathcal{B}_1}^1),\ldots,g_0(z_{\mathcal{B}_1}^{M_2})]^T\in\mathbb{R}^{M_2},\\
&G_1=[g_1(z_{\mathcal{B}_2}^1),\ldots,g_1(z_{\mathcal{B}_2}^{M_3})]^T\in\mathbb{R}^{M_3},\\
&\Delta=\begin{bmatrix}\Delta_{11} & \Delta_{12} & \Delta_{13}\\ \Delta_{12}^T & \Delta_{22} & \Delta_{23}\\ \Delta_{13}^T & \Delta_{23}^T & \Delta_{33}\end{bmatrix}\in\mathbb{R}^{M\times M},\\
&\Delta_{11}=[\Omega]^{\mathcal{X}_{\mathcal{C}}}+[\Omega_{t_1,0}]^{\mathcal{Z}_{\mathcal{C}}}+[\Omega_{0,t_2}]^{\mathcal{Z}_{\mathcal{C}}}+[\Omega_{t_1,t_2}]^{\mathcal{Z}_{\mathcal{C}}},\\
&\Delta_{12}=[\Omega]^{\mathcal{Z}_{\mathcal{B}_1},\mathcal{Z}_{\mathcal{C}}}+[\Omega_{0,t_2}]^{\mathcal{Z}_{\mathcal{B}_1},\mathcal{Z}_{\mathcal{C}}},\\
&\Delta_{13}=[\Omega]^{\mathcal{Z}_{\mathcal{B}_2},\mathcal{Z}_{\mathcal{C}}}+[\Omega_{0,t_2}]^{\mathcal{Z}_{\mathcal{B}_2},\mathcal{Z}_{\mathcal{C}}},\\
&\Delta_{22}=[\Omega]^{\mathcal{Z}_{\mathcal{B}_1}},\quad \Delta_{23}=[\Omega]^{\mathcal{Z}_{\mathcal{B}_2},\mathcal{Z}_{\mathcal{B}_1}},\quad \Delta_{33}=[\Omega]^{\mathcal{Z}_{\mathcal{B}_2}},\\
&S=[S_{\mathcal{C}},S_{\mathcal{B}_1},S_{\mathcal{B}_2}],\quad \in\mathbb{R}^{N\times M},
\end{aligned}$$

$$\begin{aligned}
S_{\mathcal{C}}=&\Big[\Omega_{0,t_2^{(2)}}\Big]^{\mathcal{Z}_{\mathcal{C}},\mathcal{Z}_{\mathcal{D}}}+D_a[\Omega_{0,t_2}]^{\mathcal{Z}_{\mathcal{C}},\mathcal{Z}_{\mathcal{D}}}+D_b[\Omega]^{\mathcal{Z}_{\mathcal{C}},\mathcal{Z}_{\mathcal{D}}}\\
&-D_c\Big[\Omega_{0,x_2^{(2)}}\Big]^{\mathcal{Z}_{\mathcal{C}},\mathcal{Z}_{\mathcal{D}}}+\Big[\Omega_{t,t_2^{(2)}}\Big]^{\mathcal{Z}_{\mathcal{C}},\mathcal{Z}_{\mathcal{D}}}+D_a[\Omega_{t_1,t_2}]^{\mathcal{Z}_{\mathcal{C}},\mathcal{Z}_{\mathcal{D}}}\\
&+D_b[\Omega_{t_1,0}]^{\mathcal{Z}_{\mathcal{C}},\mathcal{Z}_{\mathcal{D}}}-D_c\Big[\Omega_{t_1,x_2^{(2)}}\Big]^{\mathcal{Z}_{\mathcal{C}},\mathcal{Z}_{\mathcal{D}}},\\
S_{\mathcal{B}_1}=&\Big[\Omega_{0,t_2^{(2)}}\Big]^{\mathcal{Z}_{\mathcal{B}_1},\mathcal{Z}_{\mathcal{D}}}+D_a[\Omega_{0,t_2}]^{\mathcal{Z}_{\mathcal{B}_1},\mathcal{Z}_{\mathcal{D}}}\\
&+D_b[\Omega]^{\mathcal{Z}_{\mathcal{B}_1},\mathcal{Z}_{\mathcal{D}}}-D_c\Big[\Omega_{0,x_2^{(2)}}\Big]^{\mathcal{Z}_{\mathcal{B}_1},\mathcal{Z}_{\mathcal{D}}},\\
S_{\mathcal{B}_2}=&\Big[\Omega_{0,t_2^{(2)}}\Big]^{\mathcal{Z}_{\mathcal{B}_2},\mathcal{Z}_{\mathcal{D}}}+D_a[\Omega_{0,t_2}]^{\mathcal{Z}_{\mathcal{B}_2},\mathcal{Z}_{\mathcal{D}}}\\
&+D_b[\Omega]^{\mathcal{Z}_{\mathcal{B}_2},\mathcal{Z}_{\mathcal{D}}}-D_c\Big[\Omega_{0,x_2^{(2)}}\Big]^{\mathcal{Z}_{\mathcal{B}_2},\mathcal{Z}_{\mathcal{D}}},\\
D_a=&\mathrm{diag}\big(a(z_{\mathcal{D}}^1),\ldots,a(z_{\mathcal{D}}^N)\big),\\
D_b=&\mathrm{diag}\big(b(z_{\mathcal{D}}^1),\ldots,b(z_{\mathcal{D}}^N)\big),\quad f=[f(z_{\mathcal{D}}^1),\ldots,f(z_{\mathcal{D}}^N)]^T\\
D_c=&\mathrm{diag}\big(c(z_{\mathcal{D}}^1),\ldots,c(z_{\mathcal{D}}^N)\big),\quad b=[b(z_{\mathcal{D}}^1),\ldots,b(z_{\mathcal{D}}^N)]^T,\\
\mathcal{K}=&\Big[\Omega_{t_1^{(2)},t_2^{(2)}}\Big]^{\mathcal{Z}_{\mathcal{D}}}+D_a[\Omega_{t_1,t_2}]^{\mathcal{Z}_{\mathcal{D}}}D_a+D_b[\Omega]^{\mathcal{Z}_{\mathcal{D}}}D_b\\
&+D_c\Big[\Omega_{x_1^{(2)},x_2^{(2)}}\Big]^{\mathcal{Z}_{\mathcal{D}}}D_c+\Big(D_a\Big[\Omega_{t_1^{(2)},t_2}\Big]^{\mathcal{Z}_{\mathcal{D}}}+\Big[\Omega_{t_1,t_2^{(2)}}\Big]^{\mathcal{Z}_{\mathcal{D}}}D_a\Big)\\
&+\Big(D_b\Big[\Omega_{t_1^{(2)},0}\Big]^{\mathcal{Z}_{\mathcal{D}}}+\Big[\Omega_{0,t_2^{(2)}}\Big]^{\mathcal{Z}_{\mathcal{D}}}D_b\Big)\\
&-\Big(D_c\Big[\Omega_{t_1^{(2)},x_2^{(2)}}\Big]^{\mathcal{Z}_{\mathcal{D}}}+\Big[\Omega_{x_1^{(2)},t_2^{(2)}}\Big]^{\mathcal{Z}_{\mathcal{D}}}D_c\Big)\\
&+\Big(D_b[\Omega_{t_1,0}]^{\mathcal{Z}_{\mathcal{D}}}D_a+D_a[\Omega_{0,t_2}]^{\mathcal{Z}_{\mathcal{D}}}D_b\Big)\\
&-\Big(D_c\Big[\Omega_{t_1,x_2^{(2)}}\Big]^{\mathcal{Z}_{\mathcal{D}}}D_a+D_a\Big[\Omega_{x_1^{(2)},t_2}\Big]^{\mathcal{Z}_{\mathcal{D}}}D_c\Big)\\
&-\Big(D_c\Big[\Omega_{0,x_2^{(2)}}\Big]^{\mathcal{Z}_{\mathcal{D}}}D_b+D_b\Big[\Omega_{x_1^{(2)},0}\Big]^{\mathcal{Z}_{\mathcal{D}}}D_c\Big)\in\mathbb{R}^{N\times N},
\end{aligned}$$

where $\mathrm{vec}(\cdot)$ *denotes the vectorization of a matrix. Also note that* $\mathcal{K}=\mathcal{K}^T$.

The LS-SVM model for the solution derivative, with respect to space (x) and time (t), in the dual form becomes

$$\begin{aligned}
\frac{\partial\hat{u}(z)}{\partial x}=&\sum_{i=1}^{|\mathcal{Z}_{\mathcal{D}}|}\alpha_i\Big(\Big[\nabla_{t_1^{(2)},x_2}K\Big](z_{\mathcal{D}}^i,z)\\
&+a(z_{\mathcal{D}}^i)[\nabla_{t_1,x_2}K](z_{\mathcal{D}}^i,z)+b(z_{\mathcal{D}}^i)[\nabla_{0,x_2}K](z_{\mathcal{D}}^i,z)\\
&-c(z_{\mathcal{D}}^i)\Big[\nabla_{x_1^{(2)},x_2}K\Big](z_{\mathcal{D}}^i,z)\Big)+\sum_{i=1}^{|\mathcal{Z}_{\mathcal{C}}|}\beta_i^1[\nabla_{0,x_2}K\\
&+\nabla_{t_1,x_2}K](z_{\mathcal{C}}^i,z)+\sum_{i=1}^{|\mathcal{X}_{\mathcal{B}_1}|}\beta_i^2[\nabla_{0,x_2}K](z_{\mathcal{B}_1}^i,z)\\
&+\sum_{i=1}^{|\mathcal{Z}_{\mathcal{B}_2}|}\beta_i^3[\nabla_{0,x_2}K](z_{\mathcal{B}_2}^i,z),
\end{aligned}$$

$$\begin{aligned}
\frac{\partial\hat{u}(z)}{\partial t}=&\sum_{i=1}^{|\mathcal{X}_{\mathcal{D}}|}\alpha_i\Big(\Big[\nabla_{t_1^{(2)},t_2}K\Big](z_{\mathcal{D}}^i,z)\\
&+a(z_{\mathcal{D}}^i)[\nabla_{t_1,t_2}K](z_{\mathcal{D}}^i,z)+b(z_{\mathcal{D}}^i)[\nabla_{0,t_2}K](z_{\mathcal{D}}^i,z)\\
&-c(z_{\mathcal{D}}^i)\Big[\nabla_{x_1^{(2)},t_2}K\Big](z_{\mathcal{D}}^i,z)\Big)+\sum_{i=1}^{|\mathcal{Z}_{\mathcal{C}}|}\beta_i^1[\nabla_{0,t_2}K\\
&+\nabla_{t_1,t_2}K](z_{\mathcal{C}}^i,z)+\sum_{i=1}^{|\mathcal{X}_{\mathcal{B}_1}|}\beta_i^2[\nabla_{0,t_2}K](\boldsymbol{x}_{\mathcal{B}_1}^i,z)
\end{aligned}$$

$$+\sum_{i=1}^{|\mathcal{Z}_{\mathcal{B}_2}|}\beta_i^3\left[\nabla_{0,t_2}K\right](z_{\mathcal{B}_2}^i,z),$$

where K is the kernel function.

3.2. PDEs on irregular domains

Consider the PDE (5), with operator $\mathcal{L}$ in (6), defined on an irregular domain subject to a Dirichlet boundary condition, i.e.

$u(z)=g(z)$ for all $z\in\partial\Sigma$.

The approximate solution, $\hat{u}(z)=w^T\varphi(z)+d$, can then be obtained by solving the following optimization problem:

$$\begin{aligned}
&\underset{w,d,e}{\text{minimize}} \quad \frac{1}{2}w^Tw+\frac{\gamma}{2}e^Te\\
&\text{subject to} \quad w^T\Big[\varphi_{tt}(z_{\mathcal{D}}^i)+a(z_{\mathcal{D}}^i)\varphi_t(z_{\mathcal{D}}^i)+b(z_{\mathcal{D}}^i)\varphi(z_{\mathcal{D}}^i)\\
&\qquad -c(z_{\mathcal{D}}^i)\varphi_{xx}(z_{\mathcal{D}}^i)\Big]+b(z_{\mathcal{D}}^i)d\\
&\qquad =f(z_{\mathcal{D}}^i)+e_i,\ \ i=1,\ldots,|\mathcal{Z}_{\mathcal{D}}|,\\
&w^T\varphi(z_{\mathcal{B}}^i)+d=g(t_i),\ \ i=1,\ldots,|\mathcal{Z}_{\mathcal{B}}|.
\end{aligned}\tag{12}$$

Here $\mathcal{Z}_{\mathcal{D}}$ and $\mathcal{Z}_{\mathcal{B}}$ are defined as previously.

Lemma 3.2. *Given a positive definite kernel function $K:\mathbb{R}^2\times\mathbb{R}^2\to\mathbb{R}$ with $K(z_1,z_2)=\varphi(z_1)^T\varphi(z_2)$ and a regularization constant $\gamma\in\mathbb{R}^+$, the solution to (12) is given by the following dual problem:*

$$\begin{bmatrix}\mathcal{K}+\gamma^{-1}I_N & S_{\mathcal{B}} & b\\ S_{\mathcal{B}}^T & \Delta_{\mathcal{B}} & 1_M\\ b^T & 1_M^T & 0\end{bmatrix}\begin{bmatrix}\alpha\\ \beta\\ d\end{bmatrix}=\begin{bmatrix}f\\ g\\ 0\end{bmatrix}\tag{13}$$

with

$$\begin{aligned}
&N=|\mathcal{Z}_{\mathcal{D}}|,\quad M=|\mathcal{Z}_{\mathcal{B}}|,\quad \beta=[\beta_1,\ldots,\beta_M]^T\in\mathbb{R}^M,\\
&g=[g(z_{\mathcal{B}}^1),\ldots,g(z_{\mathcal{B}}^M)]^T\in\mathbb{R}^M,\quad \Delta_{\mathcal{B}}=\left[\Omega\right]^{\mathcal{Z}_{\mathcal{B}},\mathcal{Z}_{\mathcal{B}}}\in\mathbb{R}^{M\times M},\\
&S_{\mathcal{B}}=\left[\Omega_{0,t_2^{(2)}}\right]^{\mathcal{Z}_{\mathcal{B}},\mathcal{Z}_{\mathcal{D}}}+D_a\left[\Omega_{0,t_2}\right]^{\mathcal{Z}_{\mathcal{B}},\mathcal{Z}_{\mathcal{D}}}\\
&D_b\left[\Omega\right]^{\mathcal{Z}_{\mathcal{B}},\mathcal{Z}_{\mathcal{D}}}-D_c\left[\Omega_{0,x_2^{(2)}}\right]^{\mathcal{Z}_{\mathcal{B}},\mathcal{Z}_{\mathcal{D}}}
\end{aligned}$$

where $\mathcal{K}$, b, f, D_a, D_b, α and D_c are defined as previously.

The dual model representation of the solution is as follows:

$$\begin{aligned}
\hat{u}(z)=&\sum_{i=1}^{|\mathcal{Z}_{\mathcal{D}}|}\alpha_i\Big(\left[\nabla_{t_1^{(2)},0}K\right](z_{\mathcal{D}}^i,z)+a(z_{\mathcal{D}}^i)\left[\nabla_{t_1,0}K\right](z_{\mathcal{D}}^i,z)\\
&+b(z_{\mathcal{D}}^i)\left[\nabla_{0,0}K\right](z_{\mathcal{D}}^i,z)-c(z_{\mathcal{D}}^i)\left[\nabla_{x_1^{(2)},0}K\right](z_{\mathcal{D}}^i,z)\Big)\\
&+\sum_{i=1}^{|\mathcal{Z}_{\mathcal{B}}|}\beta_i\left[\nabla_{0,0}K\right](z_{\mathcal{B}}^i,z)+d.
\end{aligned}$$

Proof. It follows from constructing the Lagrangian of the constrained optimization (12) as in Lemma 2.1, then obtaining the Karush–Kuhn–Tucker optimality conditions and eliminating the primal variables w and e. □

The LS-SVM model for the solution derivative, with respect to space (x) and time (t), in the dual form becomes

$$\begin{aligned}
\frac{\partial\hat{u}(z)}{\partial x}=&\sum_{i=1}^{|\mathcal{Z}_{\mathcal{D}}|}\alpha_i\Big(\left[\nabla_{t_1^{(2)},x_2}K\right](z_{\mathcal{D}}^i,z)+a(z_{\mathcal{D}}^i)\left[\nabla_{t_1,x_2}K\right](z_{\mathcal{D}}^i,z)\\
&+b(z_{\mathcal{D}}^i)\left[\nabla_{0,x_2}K\right](z_{\mathcal{D}}^i,z)-c(z_{\mathcal{D}}^i)\left[\nabla_{x_1^{(2)},x_2}K\right](z_{\mathcal{D}}^i,z)\Big)\\
&+\sum_{i=1}^{|\mathcal{Z}_{\mathcal{B}}|}\beta_i\left[\nabla_{0,x_2}K\right](z_{\mathcal{B}}^i,z),
\end{aligned}$$

$$\begin{aligned}
\frac{\partial\hat{u}(z)}{\partial t}=&\sum_{i=1}^{|\mathcal{Z}_{\mathcal{D}}|}\alpha_i\Big(\left[\nabla_{t_1^{(2)},t_2}K\right](z_{\mathcal{D}}^i,z)+a(z_{\mathcal{D}}^i)\left[\nabla_{t_1,t_2}K\right](z_{\mathcal{D}}^i,z)\\
&+b(z_{\mathcal{D}}^i)\left[\nabla_{0,t_2}K\right](z_{\mathcal{D}}^i,z)-c(z_{\mathcal{D}}^i)\left[\nabla_{x_1^{(2)},t_2}K\right](z_{\mathcal{D}}^i,z)\Big)\\
&+\sum_{i=1}^{|\mathcal{Z}_{\mathcal{B}}|}\beta_i\left[\nabla_{0,t_2}K\right](z_{\mathcal{B}}^i,z),
\end{aligned}$$

where K is the kernel function.

Remark 3.2. Although in Section 3.2, the formulation of the method is presented for a Dirichlet boundary condition, it can be easily adapted, by adopting suitable constraints, for the Neumann or Robbins (a linear combination of the Dirichlet and Neumann) type boundary conditions.

4. Formulation of the method for nonlinear PDE

Inspired by the approach described in [25] for nonlinear ODEs, we formulate an optimization problem based on least squares support vector machines for solving nonlinear partial differential equations. For the sake of notational simplicity let us assume that the nonlinear PDE has the following form:

$$\frac{\partial^2u}{\partial t^2}+\frac{\partial^2u}{\partial x^2}+f(u)=g(z),\quad z\in\Sigma\in\mathbb{R}^2\tag{14}$$

subject to the boundary conditions of the form

$$u(z)=h(z),\quad z\in\partial\Sigma\tag{15}$$

where f is a nonlinear function. The approximate solution $\hat{u}(z)=w^T\varphi(z)+d$ for the given nonlinear PDE can be obtained by solving the following optimization problem:

$$\begin{aligned}
&\underset{w,d,e,\xi,u}{\text{minimize}} \quad \frac{1}{2}w^Tw+\frac{\gamma}{2}(e^Te+\xi^T\xi)\\
&\text{subject to} \quad w^T\left[\varphi_{tt}(z_{\mathcal{D}}^i)+\varphi_{xx}(z_{\mathcal{D}}^i)\right]+f(u(z_{\mathcal{D}}^i))\\
&\qquad =g(z_{\mathcal{D}}^i)+e_i,\ \ i=1,\ldots,|\mathcal{Z}_{\mathcal{D}}|,\\
&w^T\varphi(z_{\mathcal{D}}^i)+d=u(z_{\mathcal{D}}^i)+\xi_i,\ \ i=1,\ldots,|\mathcal{Z}_{\mathcal{D}}|,\\
&w^T\varphi(z_{\mathcal{B}}^i)+d=h(z_{\mathcal{B}}^i),\ \ i=1,\ldots,|\mathcal{Z}_{\mathcal{B}}|.
\end{aligned}\tag{16}$$

Note that the second set of additional constraints is introduced to keep the optimization problem linear in w. As before, we assume that $N=|\mathcal{Z}_{\mathcal{D}}|$, $M=|\mathcal{Z}_{\mathcal{B}}|$. After deriving the Lagrangian, taking the KKT conditions and eliminating the primal variables w, e, ξ one obtains the following nonlinear system of equations:

$$\begin{cases}
\mathcal{K}\alpha+S_1\eta^1+S_2\eta^2-f(u)=0\\
S_1^T\alpha+\Delta_{11}\eta^1+\Delta_{12}\eta^2+1_Nd-I_Nu=0\\
S_2^T\alpha+\Delta_{12}^T\eta^1+\Delta_{22}\eta^2+1_Md=0\\
1_N^T\eta^1+1_M^T\eta^2=0\\
\mathrm{diag}(f_u)\alpha-\eta^1=0
\end{cases}\tag{17}$$

where η_1, η_2 and α are Lagrange multipliers and $u=[u(z_{\mathcal{D}}^1),\ldots,u(z_{\mathcal{D}}^N)]^T$, $f_u=[(d/du)f(u(z_{\mathcal{D}}^1)),\ldots,(d/du)f(u(z_{\mathcal{D}}^N))]$ and $\mathrm{diag}(f_u)$ is a diagonal matrix with elements of f_u on the diagonal

$$\begin{aligned}
\mathcal{K}=&\left[\Omega_{t_1^{(2)},t_2^{(2)}}\right]^{\mathcal{Z}_{\mathcal{D}}}+\left[\Omega_{x_1^{(2)},x_2^{(2)}}\right]^{\mathcal{Z}_{\mathcal{D}}}\\
&+\left[\Omega_{t_1^{(2)},x_2^{(2)}}\right]^{\mathcal{Z}_{\mathcal{D}}}+\left[\Omega_{x_1^{(2)},t_2^{(2)}}\right]^{\mathcal{Z}_{\mathcal{D}}}+\gamma^{-1}I_N\in\mathbb{R}^{N\times N}\\
S_1=&\left[\Omega_{0,t_2^{(2)}}\right]^{\mathcal{Z}_{\mathcal{D}}}+\left[\Omega_{0,x_2^{(2)}}\right]^{\mathcal{Z}_{\mathcal{D}}}
\end{aligned}$$

$$S_2 = \left[\Omega_{0,t_2^{(2)}}\right]^{\mathcal{Z}_\mathcal{B},\mathcal{Z}_\mathcal{D}} + \left[\Omega_{0,x_2^{(2)}}\right]^{\mathcal{Z}_\mathcal{B},\mathcal{Z}_\mathcal{D}}$$

$$\Delta_{11} = [\Omega]^{\mathcal{Z}_\mathcal{D}} + \gamma^{-1} I_N, \quad \Delta_{12} = [\Omega]^{\mathcal{Z}_\mathcal{B},\mathcal{Z}_\mathcal{D}}, \quad \Delta_{22} = [\Omega]^{\mathcal{Z}_\mathcal{B}}.$$

The nonlinear system (17) is solved for $(\alpha, \eta^1, \eta^2, d, u)$ using Newton's method. The Jacobian of (17) can be explicitly represented as follows:

$$J = \begin{bmatrix} \mathcal{K} & S_1 & S_2 & 0_N & -\mathrm{diag}(f_u) \\ S_1^T & \Delta_{11} & \Delta_{12} & 1_N & -I_N \\ S_2^T & \Delta_{12}^T & \Delta_{22} & 1_M & 0_{M\times N} \\ 0_N^T & 1_N^T & 1_M^T & 0 & 0_N^T \\ \mathrm{diag}(f_u) & -I_N & 0_{N\times M} & 0_N & \mathrm{diag}(f_{uu} \odot \alpha) \end{bmatrix}$$

where $f_{uu} = [(d^2/du^2)f(u(z_\mathcal{D}^1)), \ldots, (d^2/du^2)f(u(z_\mathcal{D}^N))]$ and $\odot$ denotes the element-wise multiplication. The dual model representation of the solution is as follows:

$$\hat{u}(z) = \sum_{i=1}^{|\mathcal{Z}_\mathcal{D}|} \alpha_i \left(\left[\nabla_{t_1^{(2)},0} K\right](z_\mathcal{D}^i, z) + \left[\nabla_{x_1^{(2)},0} K\right](z_\mathcal{D}^i, z) \right)$$
$$+ \sum_{i=1}^{|\mathcal{Z}_\mathcal{D}|} \eta_i^1 [\nabla_{0,0} K](z_\mathcal{D}^i, z) + \sum_{i=1}^{|\mathcal{Z}_\mathcal{B}|} \eta_i^2 [\nabla_{0,0} K](z_\mathcal{B}^i, z) + d.$$

Remark 4.1. One may also discretize the given PDE in space and reduce it to system of nonlinear ordinary differential equations. Then the available ode solver can be used. But in that case one also should consider the stability of the difference scheme that is being used.

5. Numerical results

In this section, six experiments are performed to demonstrate the capability of the proposed method for solving second order partial differential equations. The accuracy of an approximate solution is measured by means of different error norms which are defined as follows:

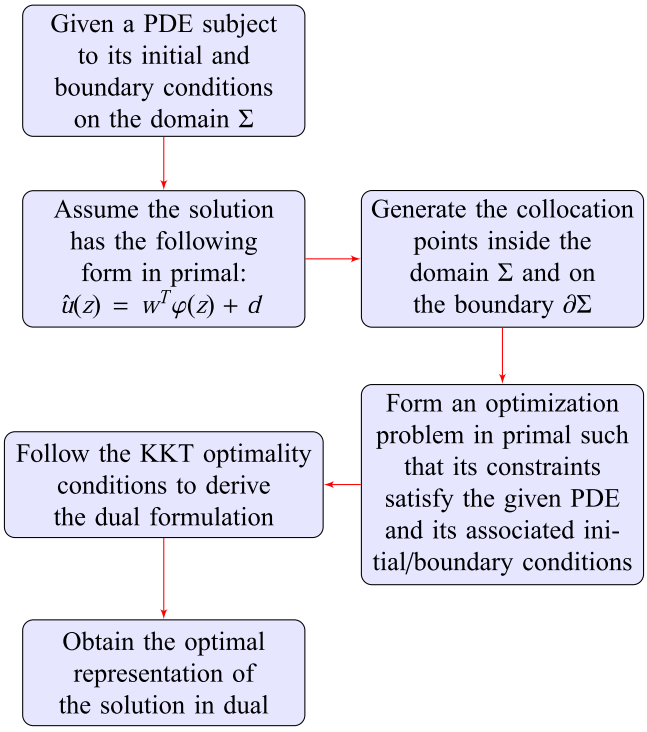
$$\mathrm{RMSE}_{test} = \sqrt{\frac{\sum_{i=1}^{N_{test}} e_i^2}{N_{test}}}, \quad L_\infty = \|e\|_\infty,$$

where $e_i = u(z_i) - \hat{u}(z_i)$. The test set consists of a grid of N_{test} points inside the given domain.

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. Note that any positive definite kernel function can be used as well. In the case of Gaussian RBF kernel the model is determined by the regularization parameter γ and the kernel bandwidth σ. It should be noted that as opposed to the regression case, we do not have the target values and we do not have noise. Therefore a quite large value can be taken for the regularization constant γ, so that the optimization (10) is sharply minimized, compared to the noisy case in regression problems.

In this work, the optimal values for γ and σ are obtained by evaluating the performance of the model on a validation set, which consists of new mesh points that do not belong to the training set, using a meaningful grid of possible (γ, σ) combinations.

The general stages (methodology) of the procedure are described by the following flow-chart:



5.1. Rectangular domains

Problem 5.1. Consider the linear second order hyperbolic equation with variable coefficients defined on a rectangular domain [1, Example 1]

$$u_{tt} + 2e^{x+t} u_t + (\sin^2(x+t))u = (1+x^2)u_{xx} + e^{-2t}(x^2 + 4e^{t+x} - \sin^2(t+x) - 3)\sinh(x), \quad 0 < x < 1,\ 0 < t < T,$$

subject to the initial and boundary conditions (8) and (9) with exact solution $u(x,t) = e^{-2t}\sinh(x)$. The approximate solution obtained by the proposed method is compared with the exact solution in Fig. 2. The step length used to generate the mesh points for the training set is $\frac{1}{10}$ and thus the number of collocation points (training points) inside and on the boundary of the domain is as follows: for $T=1$,

$$|\mathcal{Z}_\mathcal{D}| = 81, \quad |\mathcal{Z}_\mathcal{C}| = |\mathcal{Z}_{\mathcal{B}_1}| = |\mathcal{Z}_{\mathcal{B}_2}| = 10,$$

and for $T=2$,

$$|\mathcal{Z}_\mathcal{D}| = 171, \quad |\mathcal{Z}_\mathcal{C}| = 10, \quad |\mathcal{Z}_{\mathcal{B}_1}| = |\mathcal{X}_{\mathcal{B}_2}| = 20.$$

In order to make a fair comparison with the results reported in [1], the step length $\frac{1}{64}$ is used for generating mesh points for the test set. The obtained results are tabulated in Table 1.

From Table 1, in terms of accuracy the proposed method outperforms the method described in [1] despite the fact that the method presented in this paper utilizes a much less number of mesh points. Also unlike the method in [1] that provides the solution at grid points only, here a closed form solution is presented.

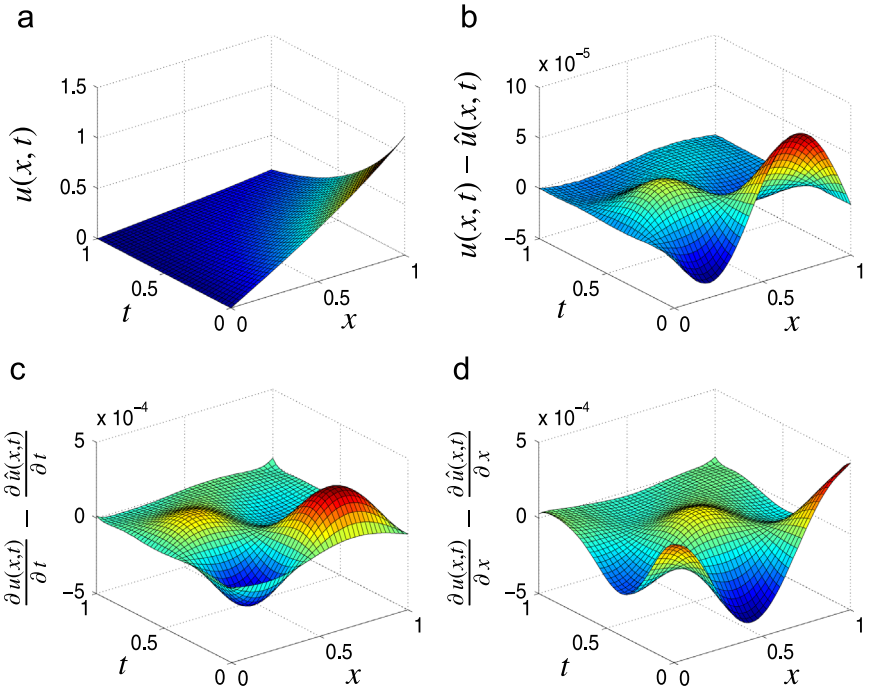


Fig. 2. Obtained model errors for Problem 5.1, when a grid consists of 81 mesh points inside the domain $[0,1]\times[0,1]$ are used for training.

Table 1
Numerical result of the proposed method for solving Problem 5.1 with time interval $[0,T]$.

Method	T	RMSE		L_∞	
		Training	Test	Training	Test
LSSVM	1	1.75×10^{-5}	1.94×10^{-5}	5.31×10^{-5}	6.71×10^{-5}
FDM [1]		–	0.74×10^{-4}	–	–
LSSVM	2	3.18×10^{-5}	3.49×10^{-5}	1.30×10^{-4}	1.51×10^{-4}
FDM [1]		–	0.43×10^{-4}	–	–

Table 2
The effect of number of training points on the approximate solution of Problem 5.1 with time interval $[0,1]$.

$\lvert\mathcal{X}_\mathcal{D}\rvert$	σ	RMSE		L_∞	
		Training	Test	Training	Test
4	225.04	1.76×10^{-3}	2.78×10^{-3}	3.50×10^{-3}	1.01×10^{-2}
25	12.61	6.26×10^{-4}	7.57×10^{-4}	1.76×10^{-3}	2.32×10^{-3}
49	5.99	2.58×10^{-4}	2.86×10^{-4}	7.31×10^{-4}	8.93×10^{-4}
81	4.13	1.75×10^{-5}	1.94×10^{-5}	5.31×10^{-5}	6.71×10^{-5}

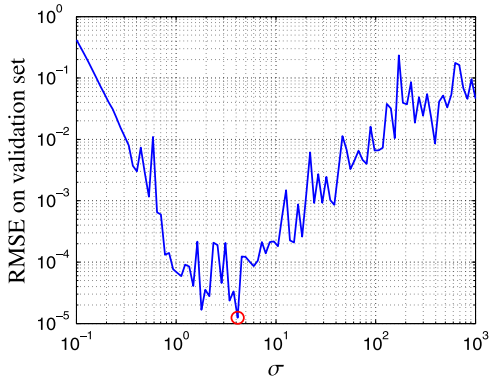


Fig. 3. Tuning the kernel bandwidth (σ) using validation set for Problem 5.1. The red circle indicates the location of selected bandwidth. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

Fig. 3 shows the RMSE obtained on the validation set versus the kernel bandwidth. The bandwidth that results in minimum RMSE on the validation set is then selected and used for evaluating the model on test set. The effect of increasing number of training points (collocation points) on the performance of the approach is demonstrated in Table 2. The training computational time is reported in Table 3.

Table 3
CPU time taken for solving the tested problems.

	Problem					
	5.1	5.2	5.3	5.4	5.5	5.6
Training time (s)	0.07	0.04	0.01	0.02	0.01	0.23

Note: The execution time is in seconds.

Problem 5.2. Consider the singular linear second order partial differential equation defined on a rectangular domain [1, Example 2]

$$u_{tt}+\frac{2}{x^2}u_t+\frac{1}{x^2}u=(1+x^2)u_{xx} -\frac{e^{-2t}(x^4-3x^2+3)\sinh(x)}{x^2},\quad 0<x<1,\ 0<t<T,$$

subject to the initial and boundary conditions (8) and (9) with exact solution $u(x,t)=e^{-2t}\sinh(x)$. The approximate solution obtained by

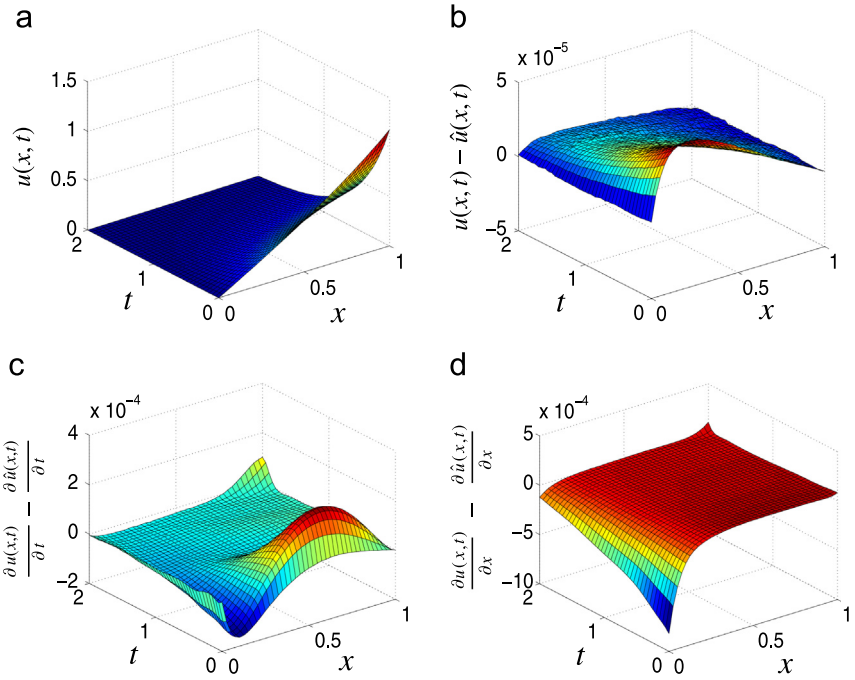


Fig. 4. Obtained model errors for Problem 5.2, when a grid consists of 171 mesh points inside the domain $[0,1]\times[0,2]$ are used for training.

Table 4
Numerical result of the proposed method for solving Problem 5.2 with time interval $[0,T]$.

Method	T	RMSE		L_∞	
		Training	Test	Training	Test
LSSVM	1	3.79×10^{-5}	3.71×10^{-5}	8.38×10^{-5}	9.52×10^{-5}
FDM [1]		–	0.22×10^{-3}	–	–
LSSVM	2	1.78×10^{-5}	1.76×10^{-5}	4.48×10^{-5}	4.89×10^{-5}
FDM [1]		–	0.62×10^{-4}	–	–

the proposed method is compared with the exact solution in Fig. 4. The same step length as in Problem 5.1 is used to generate the mesh points for the training and test sets. The training computational time is reported in Table 3. The obtained results are tabulated in Table 4. The proposed method shows a better performance in comparison with the described method in [1] in terms of accuracy despite the fact that much less number of mesh points are used.

Problem 5.3. Consider the singular linear second order equation defined on a rectangular domain [8, Example 5]

$$\nabla^2 u(x,y) = \exp(-x)(x-2+y^3+6y)$$

with $x,\ t\in[0,1]$ and the Dirichlet boundary conditions:

$$u(0,t)=y^3,\quad u(1,t)=(1+y^3)\exp(-1)$$

and

$$u(x,0)=x\exp(-x),\quad u(x,1)=x\exp(-x)(x+1).$$

The exact solution is $u(x,y)=e^{-x}(x+y^3)$. The approximate solution obtained by the proposed method is compared with the exact solution in Fig. 5. In order to make a fair comparison, the same number of grid points as in [8] is used for training and test sets. The proposed method shows slightly better performance in comparison with the described method in [8] in terms of accuracy (the maximum absolute error for training and test points shown in [8, Fig. 9 and Fig. 10] is approximately 5×10^{-7}). Furthermore as opposed to the neural networks approach here one does not need to provide a trial neural form of the solution and the solution is obtained by solving a linear system of equations. The training computational time is reported in Table 3.

5.2. Irregular domains

Problem 5.4. Consider the linear second order elliptic PDE [10, Section VB1]

$$\nabla^2 u(x,y) = 4x\cos(x)+(5-x^2-y^2)\sin(x) \tag{18}$$

defined on a circular domain, i.e.

$$\Sigma := \{(x,y)|x^2+y^2-1=0,\ -1\le x\le 1,\ -1\le y\le 1\}$$

with the Dirichlet condition $u(x,y)=0$ on $\partial\Sigma$. The exact solution is given by $u(x,y)=(x^2+y^2-1)\sin(x)$. The approximate solution obtained by the proposed method is compared with the exact solution in Fig. 6. The distribution of the collocation points used to undertake the learning process is shown in Fig. 1(b). The number of collocation points (training points) inside and on the boundary of the domain is as follows:

$$|\mathcal{Z}_\mathcal{D}|=45,\quad |\mathcal{Z}_\mathcal{B}|=19,$$

which are less than those (52 and 24 collocation points inside and on the boundary of the domain respectively) used in [10]. The training computational time is reported in Table 3. The obtained results are tabulated in Table 5. The proposed method outperforms the described method in [10] in terms of accuracy despite the fact that less number of training points are used. (Note that in [10] the maximum absolute error shown in [10, Fig. 7] is approximately 2×10^{-3} and the reported mean square error in [10, Table II], obtained by using genetic programming with boosting approach, is 2.05×10^{-4}).

Problem 5.5. Consider the second order elliptic PDE [10, Section VB2]

$$\nabla^2 u(x,y) = 2\exp(x-y) \tag{19}$$

defined on the following domain, i.e.

$$\Sigma := \{(x,y)|(x,y)=r(\theta)(\cos(\theta),\sin(\theta)),\ 0\le\theta\le 2\pi,\}$$

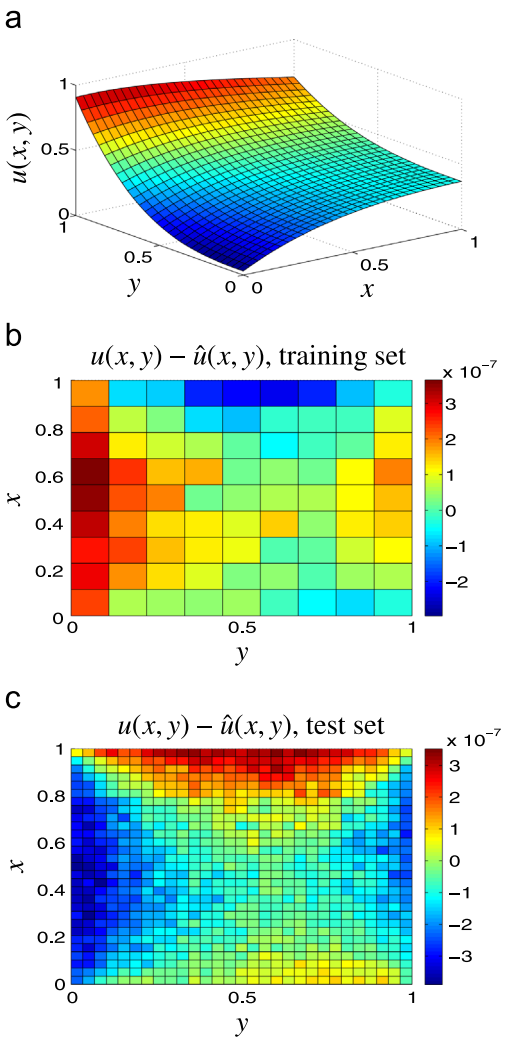


Fig. 5. The obtained approximate solution and the model errors for Problem 5.3. (a) The obtained approximate solution. (b) The model error on training set when a grid consists of 100 mesh points inside the domain $[0, 1] \times [0, 1]$ are used for training. (c) The model error on test set consists of 900 mesh points inside the domain $[0, 1] \times [0, 1]$ are used for testing.

with $r(\theta) = \sqrt{\cos(2\theta) + \sqrt{1.1 - \sin^2(2\theta)}}$ and the Dirichlet boundary condition $u(x, y) = e^{x-y} + e^x \cos(y)$ on $\partial\Sigma$. The exact solution is given by $u(x, y) = e^{x-y} + e^x \cos(y)$. The approximate solution obtained by the proposed method is compared with the exact solution in Fig. 7. The distribution of the collocation points used to undertake the learning process is shown in Fig. 1(c). The number of collocation points (training points) inside and on the boundary of the domain is as follows:

$$|\mathcal{Z}_{\mathcal{D}}| = 48, \quad |\mathcal{Z}_{\mathcal{B}}| = 28,$$

which are almost as number as those (48 and 32 collocation points inside and on the boundary of the domain respectively) used in [10]. The computed residuals are displayed in Fig. 7(b)–(e). The training computational time is reported in Table 3. The mean square errors and maximum absolute errors for the test set are also recorded in Table 5, which shows the improvement of the proposed method over the described method in [10]. (Note that in [10] the maximum absolute error shown in [10, Fig. 11] is approximately 2×10^{-2} and the reported mean square error in [10, Table II], obtained by using genetic programming with boosting approach, is 4.46×10^{-4}).

Problem 5.6. Consider an example of nonlinear PDE

$$\nabla^2 u(x, y) + u(x, y)^2 = \sin(\pi x)\left(2 - (\pi y)^2 + y^4 \sin(\pi x)\right) \tag{20}$$

defined on a circular domain, i.e.

$$\Sigma := \{(x, y) | x^2 + y^2 - 1 = 0, \ -1 \le x \le 1, -1 \le y \le 1\}$$

with the Dirichlet condition on $\partial\Sigma$. The exact solution is given by $u(x, y) = y^2 \sin(\pi x)$. The approximate solution obtained by the proposed method is compared with the exact solution in Fig. 8(b). The training computational time is reported in Table 3. The number of collocation points (training points) inside and on the boundary of the domain is as follows:

$$|\mathcal{Z}_{\mathcal{D}}| = 24, \quad |\mathcal{Z}_{\mathcal{B}}| = 19.$$

Remark 5.1. In the presented approach, the solution to the given PDE is learned by means of a linear combination of infinite number of basis functions in the primal formulation, since the dimension of feature map can be infinite for a Gaussian kernel, as opposed to classical methods that consider a finite number of basis functions. One can use other loss functions than the least square

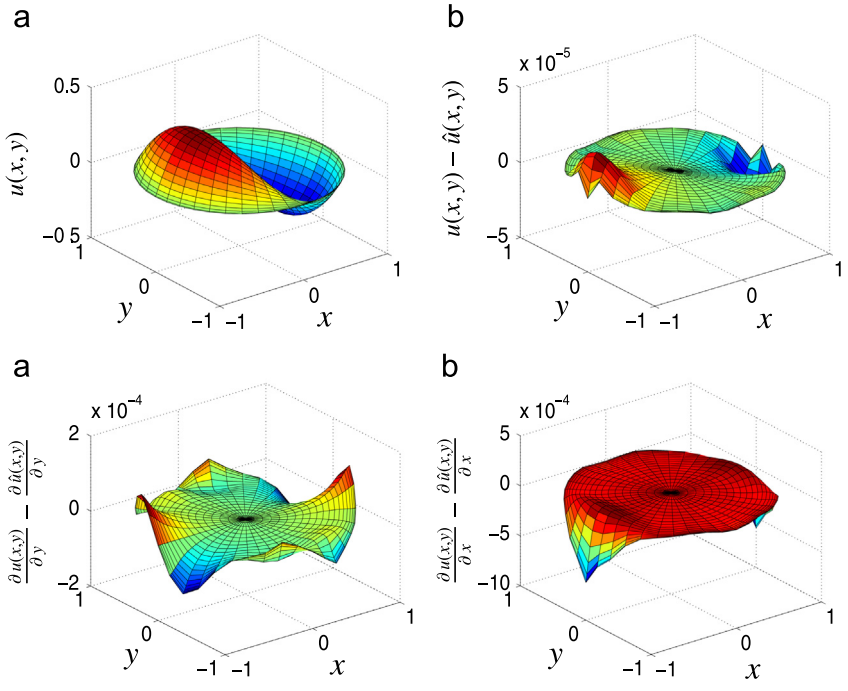


Fig. 6. Obtained model error for Problem 5.4, when a grid consists of 45 and 19 mesh points inside and on the boundary of the domain respectively are used for training.

loss function. For instance if one chooses the ϵ-insensitive loss function then the solution in the dual can be obtained by solving a quadratic programming problem and a sparse solution is then obtained.

Remark 5.2. One may obtain the explicit finite-dimensional representation of the feature map $\hat{\varphi}$ by means of Nyström approximation on a subsample of collocation points that are actively selected using quadratic Rényi entropy method. Then based on the explicit approximation $\hat{\varphi}$, the optimization problems (10) and (12) can be solved in the primal similar to Fixed-size LS-SVM approach [23,30] leading to a sparse kernel representation.

Table 5
Numerical result of the proposed method for solving Problems 5.4 and 5.5.

Problem	Method	MSE		L_∞	
		Training	Test	Training	Test
5.4	LSSVM	5.18×10^{-11}	5.94×10^{-11}	1.91×10^{-5}	2.71×10^{-5}
	GPA [10]	–	2.04×10^{-4}	–	–
5.5	LSSVM	7.93×10^{-9}	1.32×10^{-8}	3.95×10^{-4}	5.90×10^{-4}
	GPA [10]	–	4.46×10^{-4}	–	–

6. Conclusion and future work

In this paper, a new method based on least squares support vector machines is developed for solving second order linear and nonlinear partial differential equations (PDEs) in one space dimension. As opposed to ANN based approaches that need to solve a non-linear optimization problem, here in case of a linear PDE the solution is obtained by solving a system of linear equations. For a nonlinear PDE one requires applying a Newton-type iterative method. The results reveal that the method can obtain the desired accuracy with only a few number of collocation points and is able to provide a closed form approximate solution for the problem. For the future work one may consider adapting other loss functions or kernels that can deal with certain difficulties in approximating the solution of a PDE such as discontinuities, presence of shocks and low and high frequency components.

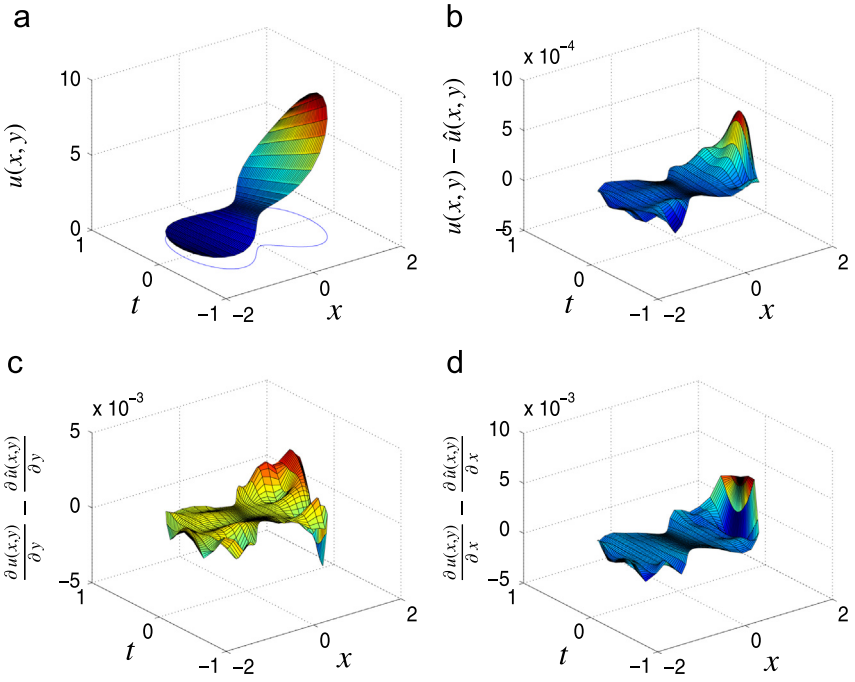


Fig. 7. Obtained model error for Problem 5.5, when a grid consists of 48 and 28 mesh points inside and on the boundary of the domain respectively are used for training.

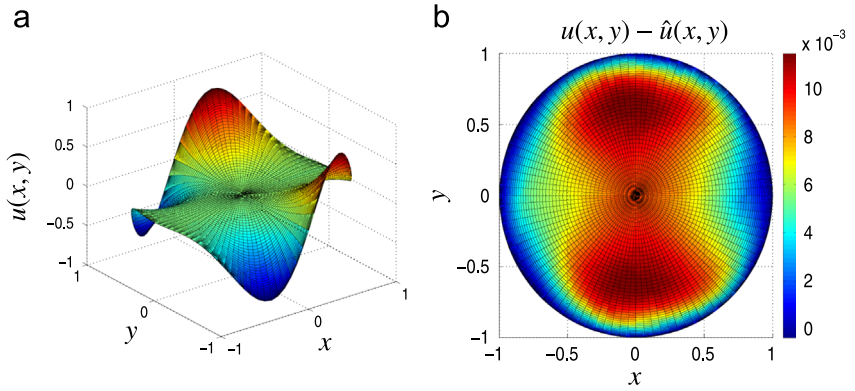


Fig. 8. Obtained model error for Problem 5.6.

Acknowledgments

• EU: The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) / ERC AdG A-DATADRIVE-B (290923). This paper reflects only the authors' views, the Union is not liable for any use that may be made of the contained information. • Research Council KUL: GOA/10/09 MaNet, CoE PFV/10/002 (OPTEC), BIL12/11T; PhD/Postdoc grants • Flemish Government: ○ FWO: projects: G.0377.12 (Structured systems), G.088114N (Tensor based data similarity); PhD/Postdoc grants ○ IWT: projects: SBO POM (100031); PhD/Postdoc grants ○ iMinds Medical Information Technologies SBO 2014 • Belgian Federal Science Policy Office: IUAP P7/19 (DYSCO, Dynamical systems, control and optimization, 2012–2017). Johan Suykens is a professor at the KU Leuven, Belgium.

Appendix: Proof of Lemma 2.1

Proof. Consider the Lagrangian of (10):

$$\begin{aligned}
&\mathcal{L}(w,d,e_i,\alpha_i,\beta_i^1,\beta_i^2,\beta_i^3)\\
&\quad=\frac{1}{2}w^Tw+\frac{\gamma}{2}e^Te-\sum_{i=1}^{|\mathcal{Z}_\mathcal{D}|}\alpha_i\Big[w^T\Big(\varphi_{tt}(z_\mathcal{D}^i)+a(z_\mathcal{D}^i)\varphi_t(z_\mathcal{D}^i)\\
&\qquad+b(z_\mathcal{D}^i)\varphi(z_\mathcal{D}^i)-c(z_\mathcal{D}^i)\varphi_{xx}(z_\mathcal{D}^i)\Big)+b(z_\mathcal{D}^i)d-f(z_\mathcal{D}^i)-e_i\Big]\\
&\qquad-\sum_{i=1}^{|\mathcal{Z}_\mathcal{C}|}\beta_i^1\Big[w^T\Big(\varphi(z_\mathcal{C}^i)+\varphi_t(z_\mathcal{C}^i)\Big)+d-h(x_i)\Big]\\
&\qquad-\sum_{i=1}^{|\mathcal{Z}_{\mathcal{B}_1}|}\beta_i^2\Big[w^T\varphi(z_{\mathcal{B}_1}^i)+d-g_0(t_i)\Big]\\
&\qquad-\sum_{i=1}^{|\mathcal{Z}_{\mathcal{B}_2}|}\beta_i^3\Big[w^T\varphi(z_{\mathcal{B}_2}^i)+d-g_1(t_i)\Big] \qquad (21)
\end{aligned}$$

where $\{\alpha_i\}_{i=1}^{|\mathcal{Z}_\mathcal{D}|}$, $\{\beta_i^1\}_{i=1}^{|\mathcal{Z}_\mathcal{C}|}$, $\{\beta_i^2\}_{i=1}^{|\mathcal{Z}_{\mathcal{B}_1}|}$ and $\{\beta_i^3\}_{i=1}^{|\mathcal{Z}_{\mathcal{B}_2}|}$ are Lagrange multipliers. The Karush–Kuhn–Tucker (KKT) optimality conditions are as follows:

$$\begin{aligned}
&\frac{\partial\mathcal{L}}{\partial w}=0\to w=\sum_{i=1}^{|\mathcal{Z}_\mathcal{D}|}\alpha_i\Big(\varphi_{tt}(z_\mathcal{D}^i)+a(z_\mathcal{D}^i)\varphi_t(z_\mathcal{D}^i)\\
&\quad+b(z_\mathcal{D}^i)\varphi(z_\mathcal{D}^i)-c(z_\mathcal{D}^i)\varphi_{xx}(z_\mathcal{D}^i)\Big)\\
&\quad+\sum_{i=1}^{|\mathcal{X}_\mathcal{C}|}\beta_i^1\Big(\varphi(z_\mathcal{C}^i)+\varphi_t(z_\mathcal{C}^i)\Big)\\
&\quad+\sum_{i=1}^{|\mathcal{Z}_{\mathcal{B}_1}|}\beta_i^2\Big(\varphi(z_{\mathcal{B}_1}^i)\Big)+\sum_{i=1}^{|\mathcal{Z}_{\mathcal{B}_2}|}\beta_i^3\Big(\varphi(z_{\mathcal{B}_2}^i)\Big)
\end{aligned}$$

$$\begin{aligned}
&\frac{\partial\mathcal{L}}{\partial d}=0\to\sum_{i=1}^{|\mathcal{Z}_\mathcal{D}|}\alpha_ib(z_\mathcal{D}^i)+\sum_{i=1}^{|\mathcal{Z}_\mathcal{C}|}\beta_i^1+\sum_{i=1}^{|\mathcal{Z}_{\mathcal{B}_1}|}\beta_i^2+\sum_{i=1}^{|\mathcal{Z}_{\mathcal{B}_2}|}\beta_i^3=0,\\
&\frac{\partial\mathcal{L}}{\partial e_i}=0\to e_i=-\frac{\alpha_i}{\gamma},\quad i=1,\ldots,|\mathcal{Z}_\mathcal{D}|,\\
&\frac{\partial\mathcal{L}}{\partial\alpha_i}=0\to w^T\Big(\varphi_{tt}(z_\mathcal{D}^i)+a(z_\mathcal{D}^i)\varphi_t(z_\mathcal{D}^i)+b(z_\mathcal{D}^i)\varphi(z_\mathcal{D}^i)\\
&\qquad-c(z_\mathcal{D}^i)\varphi_{xx}(z_\mathcal{D}^i)\Big)+b(z_\mathcal{D}^i)d-e_i\\
&\quad=f(z_\mathcal{D}^i),\quad i=1,\ldots,|\mathcal{Z}_\mathcal{D}|,\\
&\frac{\partial\mathcal{L}}{\partial\beta_i^1}=0\to w^T\Big(\varphi(z_\mathcal{C}^i)+\varphi_t(z_\mathcal{C}^i)\Big)+d\\
&\quad=h(x_i),\quad i=1,\ldots,|\mathcal{Z}_\mathcal{C}|,\\
&\frac{\partial\mathcal{L}}{\partial\beta_i^2}=0\to w^T\varphi(z_{\mathcal{B}_1}^i)+d=g_0(t_i),\quad i=1,\ldots,|\mathcal{Z}_{\mathcal{B}_1}|,\\
&\frac{\partial\mathcal{L}}{\partial\beta_i^3}=0\to w^T\varphi(z_{\mathcal{B}_2}^i)+d=g_1(t_i),\quad i=1,\ldots,|\mathcal{Z}_{\mathcal{B}_2}|.
\end{aligned}$$

Eliminating w and $\{e_i\}_{i=1}^N$, applying the kernel trick, and writing the system in matrix form results in the linear system (11). □

References

[1] R. Mohanty, An unconditionally stable finite difference formula for a linear second order one space dimensional hyperbolic equation with variable coefficients, Appl. Math. Comput. 165 (1) (2005) 229–236.
[2] V. Thomée, From finite differences to finite elements: a short history of numerical analysis of partial differential equations, J. Comput. Appl. Math. 128 (1) (2001) 1–54.
[3] A.A. Abushama, B. Bialecki, Modified nodal cubic spline collocation for Poisson's equation, SIAM J. Numer. Anal. 46 (1) (2008) 397–418.
[4] M. Kumar, Y. Gupta, Methods for solving singular boundary value problems using splines: a review, J. Appl. Math. Comput. 32 (1) (2010) 265–278.
[5] P. Ramuhalli, L. Udpa, S.S. Udpa, Finite-element neural networks for solving differential equations, IEEE Trans. Neural Netw. 16 (6) (2005) 1381–1392.
[6] K.S. McFall, J.R. Mahan, Artificial neural network method for solution of boundary value problems with exact satisfaction of arbitrary boundary conditions, IEEE Trans. Neural Netw. 20 (8) (2009) 1221–1233.
[7] I.G. Tsoulos, D. Gavrilis, E. Glavas, Solving differential equations with constructed neural networks, Neurocomputing 72 (10) (2009) 2385–2391.
[8] I.E. Lagaris, A. Likas, D.I. Fotiadis, Artificial neural networks for solving ordinary and partial differential equations, IEEE Trans. Neural Netw. 9 (5) (1998) 987–1000.
[9] R. Shekari Beidokhti, A. Malek, Solving initial-boundary value problems for systems of partial differential equations using neural networks and optimization techniques, J. Frankl. Inst. 346 (9) (2009) 898–913.
[10] A. Sóbester, P.B. Nair, A.J. Keane, Genetic programming approaches for solving elliptic partial differential equations, IEEE Trans. Evol. Comput. 12 (4) (2008) 469–478.
[11] I.G. Tsoulos, I.E. Lagaris, Solving differential equations with genetic programming, Genet. Program. Evolvable Mach. 7 (1) (2006) 33–54.
[12] N.Y. Nikolaev, H. Iba, Learning polynomial feedforward neural networks by genetic programming and backpropagation, IEEE Trans. Neural Netw. 14 (2) (2003) 337–350.
[13] A. Taleei, M. Dehghan, Time-splitting pseudo-spectral domain decomposition method for the soliton solutions of the one-and multi-dimensional nonlinear Schrödinger equations, Comput. Phys. Commun. 185 (6) (2014) 1515–1528.
[14] M. Dehghan, A. Ghesmati, Application of the dual reciprocity boundary integral equation technique to solve the nonlinear Klein–Gordon equation, Comput. Phys. Commun. 181 (8) (2010) 1410–1418.
[15] P.K. Banerjee, R. Butterfield, Boundary Element Methods in Engineering Science, vol. 17, McGraw-Hill, London, 1981.
[16] F. Shakeri, M. Dehghan, A finite volume spectral element method for solving magnetohydrodynamic (mhd) equations, Appl. Numer. Math. 61 (1) (2011) 1–23.
[17] C. Piret, The orthogonal gradients method: a radial basis functions method for solving partial differential equations on arbitrary surfaces, J. Comput. Phys. 231 (14) (2012) 4662–4675.
[18] A.J. Meade Jr., A.A. Fernandez, Solution of nonlinear ordinary differential equations by feedforward neural networks, Math. Comput. Modell. 20 (9) (1994) 19–44.
[19] B. Choi, J.-H. Lee, Comparison of generalization ability on solving differential equations using backpropagation and reformulated radial basis function networks, Neurocomputing 73 (1) (2009) 115–118.
[20] Y. Shirvany, M. Hayati, R. Moradian, Multilayer perceptron neural networks with novel unsupervised training method for numerical solution of the partial differential equations, Appl. Soft Comput. 9 (1) (2009) 20–29.
[21] V. Vapnik, Statistical Learning Theory, Wiley, New York, 1998.
[22] B. Scholkopf, A.J. Smola, Learning with kernels: support vector machines, regularization, optimization, and beyond, MIT press, 2001.
[23] J.A.K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, J. Vandewalle, Least Squares Support Vector Machines, World Scientific Pub. Co., Singapore, 2002.
[24] J.A.K. Suykens, J. Vandewalle, Least squares support vector machine classifiers, Neural Process. Lett. 9 (3) (1999) 293–300.
[25] S. Mehrkanoon, T. Falck, J.A.K. Suykens, Approximate solutions to ordinary differential equations using least squares support vector machines, IEEE Trans. Neural Netw. Learn. Syst. 23 (9) (2012) 1356–1367.
[26] S. Mehrkanoon, J.A.K. Suykens, LS-SVM approximate solution to linear time varying descriptor systems, Automatica 48 (10) (2012) 2502–2511.
[27] S. Mehrkanoon, S. Mehrkanoon, J.A. Suykens, Parameter estimation of delay differential equations: an integration-free ls-svm approach, Commun. Nonlinear Sci. Numer. Simul. 19 (4) (2014) 830–841.
[28] S. Mehrkanoon, T. Falck, J.A. Suykens, Parameter estimation for time varying dynamical systems using least squares support vector machines, in:

Proceedings of the 16th IFAC Symposium on System Identification (SYSID 2012), Brussels, Belgium, 2012, pp. 1300–1305.
[29] W.-T. Ang, A numerical method for the wave equation subject to a non-local conservation condition, Appl. Numer. Math. 56 (8) (2006) 1054–1060.
[30] K. De Brabanter, J. De Brabanter, J.A.K. Suykens, B. De Moor, Optimized fixed-size kernel models for large data sets, Comput. Stat. Data Anal. 54 (6) (2010) 1484–1504.



Siamak Mehrkanoon received BSc. in Pure Mathematics in 2005 and MSc. in Applied Mathematics from the Iran University of Science and Technology, Tehran, Iran, in 2007. He has been developing numerical methods for simulation of large-scale dynamical systems using MPI (Message Passing Interface) during 2007-2010. He is currently pursuing his Ph.D. with the Department of Electrical Engineering, ESAT-STADIUS, KU Leuven, Belgium. Siamak has been focusing on the development of kernel based approaches for clustering/classification, simulation of dynamical systems and parameter estimation. His current research interests include machine learning, pattern recognition, neural networks, numerical algorithms and optimization.



Johan A.K. Suykens (SM'05) was born in Willebroek, Belgium, on May 18, 1966. He received the M.S. degree in Electro-Mechanical Engineering and the Ph.D. degree in Applied Sciences from the Katholieke Universiteit Leuven, in 1989 and 1995, respectively. In 1996 he has been a Visiting Postdoctoral Researcher at the University of California, Berkeley. He has been a Postdoctoral Researcher with the Fund for Scientific Research FWO Flanders and is currently a Professor (Hoogleraar) with KU Leuven.

He is author of 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-author of the book Cellular Neural Networks, Multi-Scroll Chaos and Synchronization (World Scientific) and editor of the books Nonlinear Modelling: 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 Modelling with Time-series Prediction Competition. He has served as an Associate Editor for the IEEE Transactions on Circuits and Systems (1997–1999 and 2004–2007) and for the IEEE Transactions on Neural Networks (1998–2009). He received an IEEE Signal Processing Society 1999 Best Paper (Senior) Award and several Best Paper Awards at International Conferences. He is 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 2004 and the International Symposium on Nonlinear Theory and its Applications 2005, as an organizer of the International Symposium on Synchronization in Complex Networks 2007 and a co-organizer of the NIPS 2010 workshop on Tensors, Kernels and Machine Learning. He has been awarded an ERC Advanced Grant 2011 and has been elevated IEEE Fellow 2015 for developing least squares support vector machines.