East Asian Journal on Applied Mathematics
doi: 10.4208/eajam.310121.120821

Vol. **12**, No. 1, pp. 145-162
February 2022

A Support Vector Machine Method for Two Time-Scale Variable-Order Time-Fractional Diffusion Equations

Zhiwei Yang[1], Huan Liu[1], Xu Guo[2,*] and Hong Wang[3]

[1]*School of Mathematical Sciences, Fudan University, Shanghai 200433, China.*
[2]*Geotechnical and Structural Research Center, Shandong University, Jinan, Shandong 250061, China.*
[3]*Department of Mathematics, University of South Carolina, Columbia, South Carolina 29208, USA.*

Received 31 January 2021; Accepted (in revised version) 12 August 2021.

Abstract. An efficient least-square support vector machine (LS-SVM) method for a two time-scale variable-order time-fractional diffusion equation is developed. The method is particularly suitable for problems defined on complex physical domains or in high spatial dimensions. The problem is discretised by the L1 scheme and the Euler method. The temporal semi-discrete problem obtained is reformulated as a minimisation problem. The Karush-Kuhn-Tucker optimality condition is used to determine the minimiser of the optimisation problem and, hence, the solution sought. Numerical experiments show the efficiency and high accuracy of the method.

AMS subject classifications: 65M22, 65M70

Key words: Least-square support vector machine, variable-order, time-fractional diffusion equation, irregular domain.

1. Introduction

The classical Fickian diffusion partial differential equation (PDE) was derived under the assumptions that the underlying particle jumps have a mean waiting time and a finite variance [30]. The latter is characterised by solutions with Gaussian type symmetric and exponentially decaying tails and have been observed for the diffusive transport of solute in homogeneous porous media [2] under certain assumptions. However, the transport of solute in heterogeneous porous media exhibits power-law decaying tails, which probably do not accurately modelled by the Fickian diffusion PDE [30]. The time-fractional diffusion equation (tFDE) is derived via a continuous time random walk under the assumption that

*Corresponding author. *Email address:* guoxu@sdu.edu.cn (X. Guo)

 ©2022 Global-Science Press

the corresponding waiting time probability density function has a power-law decaying tail — cf. [27,30],

$$\partial_t^\alpha u - \Delta u = f(\boldsymbol{x},t), \quad 0<\alpha<1, \tag{1.1}$$

where ∂_t^α is the Caputo fractional differential operator defined by

$${}_0I_t^\alpha g(t) := \frac{1}{\Gamma(\alpha)}\int_0^t \frac{g(s)}{(t-s)^{1-\alpha}}ds, \quad \partial_t^\alpha g(t) := {}_0I_t^{1-\alpha} g'(t)$$

and $\Gamma(\alpha)$ refers to the Gamma function [31]. It can accurately describe the power-law decaying behavior of the subdiffusive transport of solute in heterogeneous media. Furthermore, in such applications as bioclogging [1], nonconventional hydrocarbon or gas recovery [11], design of shape memory polymers [20], manufacturing of viscoelastic materials [31] and biomaterials in orthopedic implants [42], the structure of porous materials may evolve in time. Since the order α of the tFDE (1.1) is related to the fractal dimension of the porous material via the Hurst index [27], these problems lead to variable-order tFDEs [7,24,34,38,44,47].

As is recently shown [36,41,46], the two time-scale variable-order tFDEs

$$\begin{aligned} &\partial_t u + k(t)\partial_t^{\alpha(t)} u - \Delta u = f(\boldsymbol{x},t), && (\boldsymbol{x},t)\in\Omega\times(0,T],\\ &u(\boldsymbol{x},0) = u_0(\boldsymbol{x}), && \boldsymbol{x}\in\Omega,\\ &u(\boldsymbol{x},t) = g(\boldsymbol{x},t), && (\boldsymbol{x},t)\in\partial\Omega\times[0,T] \end{aligned} \tag{1.2}$$

retain the long-term subdiffusive behavior of the typical tFDE (1.1). Moreover they are able to eliminate the nonphysical initial weak singularity behavior of the classical constant-order tFDE (1.1) [8,12,22,23,32,33,37], which can properly address the impact of the deformed porous media. Here, $\Omega\subset\mathbb{R}^d$ is a bounded domain with the boundary $\partial\Omega$, $\boldsymbol{x} := (x_1,\ldots,x_d)$, f, u_0, and g are prescribed source term, initial data, and boundary data, respectively. According to [24,34,38], the variable-order fractional differential operator $\partial_t^{\alpha(t)}u$ is defined by

$$\partial_t^{\alpha(t)} u(\boldsymbol{x},t) := \frac{1}{\Gamma(1-\alpha(t))}\int_0^t \frac{\partial_s u(\boldsymbol{x},s)}{(t-s)^{\alpha(t)}}ds, \quad 0\le\alpha(t)\le\alpha_*<1. \tag{1.3}$$

In this paper we develop a least-square support vector machine (LS-SVM) method for the two time-scale variable-order tFDE (1.2) with a fast solution technique by exploring the feature of the governing equation. Our goal is the simulation of problems on complex physical domains or in high spatial dimensions. The rest of the paper is organised as follows. In Section 2 we reformulate the variable-order tFDE (1.2) as a minimisation problem and obtain an LS-SVM spatial discretisation by enforcing the variable-order tFDE (1.2) at a series of collocation points. In Section 3, we discretise the considered problem in time using the $L1$ scheme and the Euler method, derive the Karush-Kuhn-Tucker (KKT) optimality condition and thus the numerical solution of the variable-order tFDE (1.2). Section 4 presents a recently developed technique to improve the efficiency of the LS-SVM method. In Section 5 we carry out numerical experiments to investigate the performance of the developed LS-SVM scheme. Section 6 contains concluding remarks.

2. An LS-SVM Spatial Discretisation

We carry out the spatial discretisation of the variable-order tFDE (1.2) by the LS-SVM framework.

2.1. Brief review of LS-SVM framework

We begin with the linear regression model: Given $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ with the input data $\boldsymbol{x}_i \in \mathbb{R}^d$ and output data $y_i \in \mathbb{R}$, let $\hat{y}_i \in \mathbb{R}$ be predictive data. Then the linear regression model

$$\hat{y}_i = \boldsymbol{w}^T \boldsymbol{x}_i + b, \quad \boldsymbol{w} \in \mathbb{R}^d, \quad b \in \mathbb{R}, \quad i = 1, 2, \ldots, N \tag{2.1}$$

can be reformulated as the following optimisation problem:

$$\begin{aligned} &\min_{\boldsymbol{w},b,\boldsymbol{\epsilon}} \ \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w} + \frac{\lambda}{2}\boldsymbol{\epsilon}^\top \boldsymbol{\epsilon}, \\ &\quad \text{s.t. } y_i = \boldsymbol{w}^\top \boldsymbol{x}_i + b + \epsilon_i, \quad i = 1, \ldots, N, \end{aligned} \tag{2.2}$$

where $\lambda > 0$ is the penalty parameter and $\boldsymbol{\epsilon} = (\epsilon_1, \epsilon_2, \ldots, \epsilon_d) \in \mathbb{R}^d$ the residual vector. According to the LS-SVM framework — cf. [4, 13, 43], the linear model (2.1) can be generalised by introducing a feature map $\boldsymbol{\phi} = [\phi_1, \phi_2, \ldots, \phi_{d'}]^\top : \mathbb{R}^d \to \mathbb{R}^{d'}$, considering the corresponding model

$$\hat{y}_i = \mathbf{w}^\top \boldsymbol{\phi}(\boldsymbol{x}_i) + b, \quad \mathbf{w} \in \mathbb{R}^{d'}, \quad i = 1, 2, \ldots, N \tag{2.3}$$

for the feature space and replacing (2.2) by the primal LS-SVM optimisation model for the regression by using the nonlinear map (2.3), i.e. by

$$\begin{aligned} &\min_{\mathbf{w},b,\boldsymbol{e}} \ \frac{1}{2}\mathbf{w}^\top \mathbf{w} + \frac{\lambda}{2}\boldsymbol{e}^\top \boldsymbol{e}, \\ &\quad \text{s.t. } y_i = \mathbf{w}^\top \boldsymbol{\phi}(\boldsymbol{x}_i) + b + e_i, \quad i = 1, \ldots, N. \end{aligned} \tag{2.4}$$

The construction of the feature space is motivated by the fact that any symmetric square integrable kernel $K(\boldsymbol{x}, \boldsymbol{z})$ can be expanded in the series of eigenfunctions of the corresponding integral operator — i.e.

$$K(\boldsymbol{x}, \boldsymbol{z}) = \sum_{i=1}^{\infty} \sigma_i \phi_i(\boldsymbol{x}) \phi_i(\boldsymbol{z}).$$

It is worth noting that although the LS-SVM method relies on the functions ϕ_i, the resulting scheme in the kernel method uses only information of the kernel $K(\boldsymbol{x}, \boldsymbol{z})$ but not explicit representation of functions ϕ_i.

2.2. LS-SVM spatial discretisation of variable-order tFDEs

We use (2.3) to define the approximate solution to the variable-order tFDE (1.2) at each time $t \in [0,T]$ as

$$\hat{u}(\boldsymbol{x},t) := \mathbf{w}(t)^\top \boldsymbol{\phi}(\boldsymbol{x}) + r(t) \tag{2.5}$$

with undetermined parameters $\mathbf{w}(t)^\top$ and $r(t)$. Consider the set $N = N_I + N_B$ of collocation points, where N_I and N_B denote the sets of interior points $\mathscr{Z}_I := \{\boldsymbol{x}_i^I\}_{i=1}^{N_I} \subset \Omega$ and boundary points $\mathscr{Z}_B := \{\boldsymbol{x}_i^B\}_{i=1}^{N_B} \subset \partial\Omega$, respectively. It follows from (2.5) that

$$\hat{u}(\boldsymbol{x}_i^B,t) = g(\boldsymbol{x}_i^B,t), \quad i = 1,2,\ldots,N_B, \quad t \in [0,T].$$

For any $t \in [0,T]$, the vector $\boldsymbol{e}(t) = [e_1(t),\ldots,e_{N_I}(t)]^\top$ can be assembled by collecting the residuals at each interior points $\boldsymbol{x}_i^I$, i.e.

$$e_i(t) = \partial_t \hat{u}(\boldsymbol{x}_i^I,t) - k(t)\partial_t^{\alpha(t)}\hat{u}(\boldsymbol{x}_i^I,t) - \Delta\hat{u}(\boldsymbol{x}_i^I,t) - f(\boldsymbol{x}_i^I,t), \quad i = 1,2,\ldots,N_I, \quad t \in (0,T].$$

The LS-SVM minimisation model (2.4) now takes the form

$$\begin{aligned} &\min_{\boldsymbol{\theta}(t)} \frac{1}{2}\mathbf{w}(t)^\top \mathbf{w}(t) + \frac{\lambda}{2}\boldsymbol{e}(t)^\top \boldsymbol{e}(t), \\ &\text{s.t. } \partial_t \hat{u}(\boldsymbol{x}_i^I,t) - k(t)\partial_t^{\alpha(t)}\hat{u}(\boldsymbol{x}_i^I,t) - \Delta\hat{u}(\boldsymbol{x}_i^I,t) - f(\boldsymbol{x}_i^I,t) = e_i(t), \quad i = 1,\ldots,N_I, \\ &\qquad \hat{u}(\boldsymbol{x}_i^B,t) = g(\boldsymbol{x}_i^B,t), \quad i = 1,\ldots,N_B, \end{aligned} \tag{2.6}$$

where $\boldsymbol{\theta}(t) = \{\mathbf{w}(t), r(t), \boldsymbol{e}(t)\}$ [40].

3. A Fully-Discrete LS-SVM Scheme

Let us note that the accuracy of the kernel-based methods relies on the solution smoothness. Therefore, specific kernel functions have to be chosen according to the solution properties. As is recently shown, the solution to the variable-order tFDE (1.2) may exhibit a non-physical singularity at the initial time $t = 0$, which differs from its behavior in the spatial domain [41]. Consequently, the kernel function for the variable-order tFDE (1.2) has to be considered separately in space and in time. However, the fractional derivative of the kernel function is expensive to compute. Moreover, the kernel function considered here, depends on the spatial variables only. This is very different from the approach [28, 29]. In what follows, we use the well-known $L1$ scheme to discretise the variable-order time-fractional derivative and develop a fully-discrete LS-SVM scheme. In the next section we show that the efficiency of the $L1$ temporal discretisation scheme can be significantly improved. Thus this efficient fully-discrete LS-SVM scheme is especially suitable for solving the tFDE (1.2) in high spatial dimensional or irregular domains.

3.1. A fully-discrete numerical scheme

Defining the uniform temporal partition on $[0,T]$ by $t_m := m\tau$, $m = 0,1,\ldots,M$ with the time step size $\tau := T/M$, we respectively discretise $\partial_t \hat{u}$ and $\partial_t^{\alpha(t_m)}\hat{u}$ at $t = t_m$ by the implicit Euler method and the $L1$ discretisation scheme [21,39], i.e.

$$\begin{aligned}
&\partial_t \hat{u}(\boldsymbol{x}, t_m) \approx \delta_\tau \hat{u}_m(\boldsymbol{x}) := \frac{\hat{u}_m(\boldsymbol{x}) - \hat{u}_{m-1}(\boldsymbol{x})}{\tau},\\
&\partial_t^{\alpha(t_m)} \hat{u}(\boldsymbol{x}, t_m) \approx \delta_\tau^{\alpha(t_m)} \hat{u}_m(\boldsymbol{x}) := \frac{1}{\Gamma(1-\alpha(t_m))} \sum_{k=1}^{m} \int_{t_{k-1}}^{t_k} \frac{\delta_\tau \hat{u}_k(\boldsymbol{x}) dt}{(t_m - t)^{\alpha(t_m)}}\\
&= \frac{1}{\Gamma(2-\alpha(t_m))} \sum_{k=1}^{m} \Big[(t_m - t_{k-1})^{1-\alpha(t_m)} - (t_m - t_k)^{1-\alpha(t_m)} \Big] \delta_\tau \hat{u}_k(\boldsymbol{x})\\
&= \frac{1}{\Gamma(2-\alpha(t_m))} \sum_{k=1}^{m} b_{m,k} \big(\hat{u}_k(\boldsymbol{x}) - \hat{u}_{k-1}(\boldsymbol{x}) \big), \quad 1 \le k \le m, \quad 1 \le m \le M.
\end{aligned}$$

Here, $\hat{u}_m(\boldsymbol{x})$ is the approximation to the solution of (1.2), $f_m(\boldsymbol{x}) := f(\boldsymbol{x}, t_m)$ and

$$b_{m,k} := \big[(m-k+1)^{1-\alpha(t_m)} - (m-k)^{1-\alpha(t_m)} \big] / \tau^{\alpha(t_m)}.$$

The fully-discrete numerical scheme for the problem (1.2) is stated as follows: Find $\hat{u}_m(\boldsymbol{x}_i^I)$, $i = 1,2,\ldots,N_I$ such that the equations

$$\begin{aligned}
&\delta_\tau \hat{u}_m(\boldsymbol{x}_i^I) + k(t_m) \delta_\tau^{\alpha(t_m)} \hat{u}_m(\boldsymbol{x}_i^I) - \Delta \hat{u}_m(\boldsymbol{x}_i^I) = f_m(\boldsymbol{x}_i^I),\\
&\hat{u}_m(\boldsymbol{x}_i^B) = g_m(\boldsymbol{x}_i^B), \quad i = 1,2,\ldots,N_B
\end{aligned}$$

hold for all $m = 1,2,\ldots,M$.

At each time step t_m, the residual vector $\boldsymbol{e}_m = [e_{m,1},\ldots,e_{m,N_I}]^\top$ is given by

$$e_{m,i} := \delta_\tau \hat{u}_m(\boldsymbol{x}_i^I) + k(t_m) \delta_\tau^{\alpha(t_m)} \hat{u}_m(\boldsymbol{x}_i^I) - \Delta \hat{u}_m(\boldsymbol{x}_i^I) - f_m(\boldsymbol{x}_i^I), \quad 1 \le i \le N_I.$$

The fully-discrete analogue of (2.6) is formulated in a time-marching fashion as follows: For $m = 1,2,\ldots,M$,

$$\begin{aligned}
\min_{\boldsymbol{\theta}_m} \ & \frac{1}{2} \mathbf{w}_m^\top \mathbf{w}_m + \frac{\lambda}{2} \boldsymbol{e}_m^\top \boldsymbol{e}_m,\\
\text{s.t. } & \mathbf{w}_m^\top \boldsymbol{\phi}(\boldsymbol{x}_i^I) + r_m = \hat{u}_{m-1}(\boldsymbol{x}_i^I) - \tau k(t_m) \delta_\tau^{\alpha(t_m)} \hat{u}_m(\boldsymbol{x}_i^I) + \tau \Delta \hat{u}_m(\boldsymbol{x}_i^I)\\
& \qquad\qquad + \tau f_m(\boldsymbol{x}_i^I) + \tau e_{m,i}, \quad i = 1,\ldots,N_I,\\
& \mathbf{w}_m^\top \boldsymbol{\phi}(\boldsymbol{x}_i^B) + r_m = g_m(\boldsymbol{x}_i^B), \qquad\qquad i = 1,\ldots,N_B,
\end{aligned} \tag{3.1}$$

where $\boldsymbol{\theta}_m = \{\mathbf{w}_m, r_m, \boldsymbol{e}_m\}$.

3.2. The Karush-Kuhn-Tucker optimality condition

In order to find the solution of tFDE (1.2) [13], we employ the Karush-Kuhn-Tucker (KKT) optimality condition and reformulate the minimisation problem (3.1) as a linear system.

Theorem 3.1. *Let $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ be a symmetric and positive-definite kernel function such that $K(\boldsymbol{x}_1, \boldsymbol{x}_2) := \boldsymbol{\phi}(\boldsymbol{x}_1)^\top \boldsymbol{\phi}(\boldsymbol{x}_2)$ and $\lambda > 0$ be a regularisation parameter. Then the solution of the optimisation problem* (3.1) *is characterised by the linear system*

$$\boldsymbol{A}_m \boldsymbol{v}_m := \begin{pmatrix} \boldsymbol{T}_m & \boldsymbol{P}_m & (1+\mu_m)\mathbf{1}_{N_I} \\ \boldsymbol{P}_m^\top & \boldsymbol{S} & \mathbf{1}_{N_B} \\ (1+\mu_m)\mathbf{1}_{N_I}^\top & \mathbf{1}_{N_B}^\top & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{\xi}_m \\ \boldsymbol{\eta}_m \\ r_m \end{pmatrix} = \begin{pmatrix} \boldsymbol{h}_m \\ \boldsymbol{g}_m \\ 0 \end{pmatrix} =: \boldsymbol{F}_m, \tag{3.2}$$

where matrix blocks $\boldsymbol{T}_m$, $\boldsymbol{P}_m$, and $\boldsymbol{S}$ are defined by

$$\begin{aligned} &\boldsymbol{T}_m := -(1+\mu_m)^2 \boldsymbol{K}_0 + 2\tau(1+\mu_m)\boldsymbol{K}_2 - \tau^2 \boldsymbol{K}_4 - \tau^2 \boldsymbol{I}/\lambda, \\ &\boldsymbol{P}_m := -(1+\mu_m)\boldsymbol{K}_{0,B} + \tau \boldsymbol{K}_{2,B}, \qquad \boldsymbol{S} := \left[K\left(\boldsymbol{x}_i^B, \boldsymbol{x}_j^B\right)\right]_{N_B \times N_B}, \\ &\boldsymbol{K}_2 := \left[\Delta\boldsymbol{\phi}\left(\boldsymbol{x}_i^I\right)^\top \boldsymbol{\phi}\left(\boldsymbol{x}_j^I\right)\right]_{N_I \times N_I}, \qquad \boldsymbol{K}_0 := \left[K\left(\boldsymbol{x}_i^I, \boldsymbol{x}_j^I\right)\right]_{N_I \times N_I}, \\ &\boldsymbol{K}_4 := \left[\Delta\boldsymbol{\phi}\left(\boldsymbol{x}_i^I\right)^\top \Delta\boldsymbol{\phi}\left(\boldsymbol{x}_j^I\right)\right]_{N_I \times N_I}, \quad \boldsymbol{K}_{0,B} := \left[K\left(\boldsymbol{x}_i^I, \boldsymbol{x}_j^B\right)\right]_{N_I \times N_B}, \\ &\boldsymbol{K}_{2,B} := \left[\Delta\boldsymbol{\phi}\left(\boldsymbol{x}_i^I\right)^\top \boldsymbol{\phi}\left(\boldsymbol{x}_j^B\right)\right]_{N_I \times N_B}, \quad \mu_m := \frac{\tau k(t_m) b_{m,m}}{\Gamma(2-\alpha(t_m))} \end{aligned} \tag{3.3}$$

with the unknowns and the right-hand side having the form

$$\begin{aligned} &\boldsymbol{\xi}_m := (\xi_{m,1}, \ldots, \xi_{m,N_I})^\top \in \mathbb{R}^{N_I}, \qquad \mathbf{1}_{N_I} := (1, 1, \cdots, 1)^\top \in \mathbb{R}^{N_I}, \\ &\boldsymbol{\eta}_m := (\eta_{m,1}, \eta_{m,2}, \cdots, \eta_{m,N_B})^\top \in \mathbb{R}^{N_B}, \qquad \mathbf{1}_{N_B} := (1, 1, \cdots, 1)^\top \in \mathbb{R}^{N_B}, \\ &\boldsymbol{h}_m := \hat{\boldsymbol{u}}_{m-1} - \frac{\tau k(t_m)}{\Gamma(2-\alpha(t_m))}\left(\sum_{k=1}^{m-1}(b_{m,k} - b_{m,k+1})\hat{\boldsymbol{u}}_k - b_{m,1}\hat{\boldsymbol{u}}_0\right) + \tau \boldsymbol{f}_m, \\ &\hat{\boldsymbol{u}}_m := \left(\hat{u}_m\left(\boldsymbol{x}_1^I\right), \cdots, \hat{u}_m\left(\boldsymbol{x}_{N_I}^I\right)\right)^\top, \qquad \boldsymbol{f}_m := \left(f_m\left(\boldsymbol{x}_1^I\right), \cdots, f_m\left(\boldsymbol{x}_{N_I}^I\right)\right)^\top \in \mathbb{R}^{N_I}, \\ &\boldsymbol{g}_m := \left(g_m\left(\boldsymbol{x}_1^B\right), \cdots, g_m\left(\boldsymbol{x}_{N_B}^B\right)\right)^\top \in \mathbb{R}^{N_B}. \end{aligned}$$

The solution $\hat{u}_m$ can be represented as

$$\hat{u}_m(\boldsymbol{x}) = -\sum_{i=1}^{N_I} \xi_{m,i}\Big(\mu_m K\left(\boldsymbol{x}_i^I, \boldsymbol{x}\right) - \tau \Delta K\left(\boldsymbol{x}_i^I, \boldsymbol{x}\right)\Big) - \sum_{i=1}^{N_B} \eta_{m,i} K\left(\boldsymbol{x}_i^B, \boldsymbol{x}\right) + r_m,$$

or, in the vector form, as

$$\hat{\boldsymbol{u}}_m = -\big[(1+\mu_m)\boldsymbol{K}_0 + \tau \boldsymbol{K}_2\big]\boldsymbol{\xi}_m - \boldsymbol{K}_{0,B}\boldsymbol{\eta}_m + r_m \mathbf{1}_{N_I}. \tag{3.4}$$

Proof. Define the objective functional $L(\boldsymbol{\Theta}_m)$ of the problem (3.1) by

$$\begin{aligned}L(\boldsymbol{\Theta}_m) := &\frac{1}{2}\mathbf{w}_m^\top \mathbf{w}_m + \frac{\lambda}{2}\boldsymbol{e}_m^\top \boldsymbol{e}_m \\ &+ \sum_{i=1}^{N_I} \xi_{m,i}\Big[(1+\mu_m)\big(\mathbf{w}_m^\top \boldsymbol{\phi}\left(\boldsymbol{x}_i^I\right) + r_m\big) - \tau \mathbf{w}_m^\top \Delta\boldsymbol{\phi}\left(\boldsymbol{x}_i^I\right) - \hat{u}_{m-1} \\ &\qquad + \frac{\tau k(t_m)}{\Gamma(2-\alpha(t_m))}\left(\sum_{k=1}^{m-1}(b_{m,k}-b_{m,k+1})\hat{u}_k - b_{m,1}\hat{u}_0\right) - \tau f_m - \tau e_{m,i}\Big] \\ &+ \sum_{i=1}^{N_B} \eta_{m,i}\Big[\mathbf{w}_m^\top \boldsymbol{\phi}\left(\boldsymbol{x}_i^B\right) + r_m - g_m\left(\boldsymbol{x}_i^B\right)\Big],\end{aligned}$$

where $\boldsymbol{\Theta}_m := \{\mathbf{w}_m, r_m, \boldsymbol{e}_m, \boldsymbol{\xi}_m, \boldsymbol{\eta}_m\}$ and $\boldsymbol{\xi}_m$, $\boldsymbol{\eta}_m$ are the Lagrange multipliers. Now we use the KKT optimality condition. Since $\nabla_{\boldsymbol{w}_m} L = \mathbf{0}$, we write

$$\mathbf{w}_m = -\left[\sum_{i=1}^{N_I} \xi_{m,i}\Big((1+\mu_m)\boldsymbol{\phi}\left(\boldsymbol{x}_i^I\right) - \tau\Delta\boldsymbol{\phi}\left(\boldsymbol{x}_i^I\right)\Big) + \sum_{i=1}^{N_B} \eta_{m,i}\boldsymbol{\phi}\left(\boldsymbol{x}_i^B\right)\right]. \tag{3.5}$$

It follows from $\nabla_{\boldsymbol{e}_m} L = \lambda \boldsymbol{e}_m - \tau\boldsymbol{\xi}_m = 0$ that

$$\boldsymbol{e}_m = \frac{\tau}{\lambda}\boldsymbol{\xi}_m. \tag{3.6}$$

Analogously, the relations

$$\begin{aligned}&\frac{\partial L}{\partial \xi_{m,j}} = 0, \quad j = 1,2,\ldots,N_I, \\ &\frac{\partial L}{\partial \eta_{m,j}} = 0, \quad j = 1,2,\ldots,N_B, \\ &\frac{\partial L}{\partial r_m} = 0\end{aligned}$$

yield

$$\begin{aligned}&(1+\mu_m)\big(\mathbf{w}_m^\top \boldsymbol{\phi}(\boldsymbol{x}_j^I) + r_m\big) - \tau \mathbf{w}_m^\top \Delta\boldsymbol{\phi}(\boldsymbol{x}_j^I) - \hat{u}_{m-1}(\boldsymbol{x}_j^I) \\ &\quad + \frac{\tau}{\Gamma(2-\alpha(t_m))}\left(\sum_{k=1}^{m-1}(b_{m,k}-b_{m,k+1})\hat{u}_k - b_{m,1}\hat{u}_0\right) - \tau f_m(\boldsymbol{x}_j^I) - \tau e_{m,j} = 0, \\ &\mathbf{w}_m^\top \boldsymbol{\phi}\left(\boldsymbol{x}_i^B\right) + r_m - g_m(\boldsymbol{x}_j^B) = 0, \quad \sum_{j=1}^{N_I}(1+\mu_m)\xi_{m,j} + \sum_{j=1}^{N_B}\eta_{m,j} = 0.\end{aligned} \tag{3.7}$$

In order to eliminate $\mathbf{w}_m$, $\boldsymbol{e}_m$ in (3.7), we employ the representations of $\mathbf{w}_m$ and $\boldsymbol{e}_m$ from (3.5), (3.6), thus obtaining

$$
\begin{aligned}
&-\left[\sum_{i=1}^{N_I}\xi_{m,i}\left[(1+\mu_m)\boldsymbol{\phi}\big(\boldsymbol{x}_i^I\big)^\top-\tau\Delta\boldsymbol{\phi}\big(\boldsymbol{x}_i^I\big)^\top\right]+\sum_{i=1}^{N_B}\eta_{m,i}\boldsymbol{\phi}(\boldsymbol{x}_i^B)^\top\right]\\
&\quad\times\left[(1+\mu_m)\boldsymbol{\phi}\big(\boldsymbol{x}_j^I\big)-\tau\Delta\boldsymbol{\phi}\big(\boldsymbol{x}_j^I\big)\right]-\frac{\tau^2\xi_{m,j}}{\lambda}+(1+\mu_m)r_m\\
&=\hat{u}_{m-1}\big(\boldsymbol{x}_j^I\big)-\frac{\tau}{\Gamma(2-\alpha(t_m))}\left(\sum_{k=1}^{m-1}(b_{m,k}-b_{m,k+1})\hat{u}_k-b_{m,1}\hat{u}_0\right)+\tau f_m\big(\boldsymbol{x}_j^I\big),\\
&-\left[\sum_{i=1}^{N_I}\xi_{m,i}\left[(1+\mu_m)\boldsymbol{\phi}\big(\boldsymbol{x}_i^I\big)^\top-\tau\Delta\boldsymbol{\phi}\big(\boldsymbol{x}_i^I\big)^\top\right]+\sum_{i=1}^{N_B}\eta_{m,i}\boldsymbol{\phi}\big(\boldsymbol{x}_i^B\big)^\top\right]\cdot\boldsymbol{\phi}\big(\boldsymbol{x}_j^B\big)+r_m=g_m\big(\boldsymbol{x}_j^B\big),\\
&\sum_{j=1}^{N_I}(1+\mu_m)\xi_{m,j}+\sum_{j=1}^{N_B}\eta_{m,j}=0.
\end{aligned}
$$

The system (3.2) appears if we write these equations in the matrix form.

Finally, employing (3.5), we arrive at the representation

$$
\begin{aligned}
\hat{u}_m(\boldsymbol{x}_j)&=\mathbf{w}_m^\top\boldsymbol{\phi}(\boldsymbol{x}_j)+r_m\\
&=-\left[\sum_{i=1}^{N_I}\xi_{m,i}\Big((1+\mu_m)\boldsymbol{\phi}\big(\boldsymbol{x}_i^I\big)-\tau\Delta\boldsymbol{\phi}\big(\boldsymbol{x}_i^I\big)\Big)+\sum_{i=1}^{N_B}\eta_{m,i}\boldsymbol{\phi}\big(\boldsymbol{x}_i^B\big)\right]^\top\boldsymbol{\phi}(\boldsymbol{x}_j)+r_m\\
&=-\sum_{i=1}^{N_I}\xi_{m,i}\Big(\mu_m K\big(\boldsymbol{x}_i^I,\boldsymbol{x}_j\big)-\tau\Delta K\big(\boldsymbol{x}_i^I,\boldsymbol{x}_j\big)\Big)-\sum_{i=1}^{N_B}\eta_{m,i}K\big(\boldsymbol{x}_i^B,\boldsymbol{x}_j\big)+r_m,
\end{aligned}
$$

and the proof is complete. □

Note that the matrices $\boldsymbol{K}_2$, $\boldsymbol{K}_4$, and $\boldsymbol{K}_{2,B}$ in the system (3.2) of the LS-SVM scheme for the tFDE (1.2) involve the dot products of $\Delta\boldsymbol{\phi}(\boldsymbol{x}_1)$ with $\boldsymbol{\phi}(\boldsymbol{x}_2)$ or $\Delta\boldsymbol{\phi}(\boldsymbol{x}_2)$, respectively. For the sake of implementation efficiency of the scheme, one can exploit the kernel method and express these matrices via derivatives of $K(\boldsymbol{x}_1,\boldsymbol{x}_2)$ [13,43]. Since $K(\boldsymbol{x}_1,\boldsymbol{x}_2)=\boldsymbol{\phi}(\boldsymbol{x}_1)^\top\boldsymbol{\phi}(\boldsymbol{x}_2)$, we have

$$
\begin{aligned}
\big(\Delta_{\boldsymbol{x}_1}\boldsymbol{\phi}(\boldsymbol{x}_1)\big)^\top\boldsymbol{\phi}(\boldsymbol{x}_2)&=\Delta_{\boldsymbol{x}_1}K(\boldsymbol{x}_1,\boldsymbol{x}_2),\\
\big(\Delta_{\boldsymbol{x}_1}\boldsymbol{\phi}(\boldsymbol{x}_1)\big)^\top\big(\Delta_{\boldsymbol{x}_2}\boldsymbol{\phi}(\boldsymbol{x}_2)\big)&=\Delta_{\boldsymbol{x}_1}\Delta_{\boldsymbol{x}_2}K(\boldsymbol{x}_1,\boldsymbol{x}_2).
\end{aligned}
$$

Consequently, the matrices $\boldsymbol{K}_2$, $\boldsymbol{K}_4$, and $\boldsymbol{K}_{2,B}$ in the system (3.2) take the form

$$
\begin{aligned}
\boldsymbol{K}_2&=\Big[\Delta_{\boldsymbol{x}_i^I}K\big(\boldsymbol{x}_i^I,\boldsymbol{x}_j^I\big)\Big]_{N_I\times N_I},\\
\boldsymbol{K}_{2,B}&=\Big[\Delta_{\boldsymbol{x}_i^I}K\big(\boldsymbol{x}_i^I,\boldsymbol{x}_j^B\big)\Big]_{N_I\times N_B},\\
\boldsymbol{K}_4&=\Big[\Delta_{\boldsymbol{x}_i^I}\Delta_{\boldsymbol{x}_j^I}\big(\boldsymbol{x}_i^I,\boldsymbol{x}_j^I\big)\Big]_{N_I\times N_I}.
\end{aligned}
$$

We are now in the position to present an algorithm for the solution of the linear system (3.2) of the LS-SVM scheme for the variable-order tFDE (1.2) in a time-marching fashion from the initial time t_0 to the final time step $t_M = T$.

Algorithm 3.1 LS-SVM scheme for the solution of problem (1.2)

Initial data: $\{\boldsymbol{x}_i^I, u_0(\boldsymbol{x}_i^I)\}_{i=1}^{N_I}$: Boundary information $\{\boldsymbol{x}_i^B, g(\boldsymbol{x}_i^B)\}_{i=1}^{N_B}$.
for $m = 1, \ldots, M$ **do**
1: Compute $\boldsymbol{h}_m$ in (3.3) with the known initial data and the solutions already computed at the previous time steps $(\hat{\boldsymbol{u}}_0, \hat{\boldsymbol{u}}_1, \cdots, \hat{\boldsymbol{u}}_{m-1})$.
2: Assemble matrix $\boldsymbol{A}_m$ in (3.2).
3: Assemble vector $\boldsymbol{F}_m = (\boldsymbol{h}_m^T, \boldsymbol{g}_m^T, 0)^\top$.
4: Solve $\boldsymbol{A}_m \boldsymbol{v}_m = \boldsymbol{F}_m$ in (3.2), where $\boldsymbol{v}_m = (\boldsymbol{\xi}_m^T, \boldsymbol{\eta}_m^T, r_m)^\top$.
5: Compute $\hat{\boldsymbol{u}}_m$ using (3.4).
end for

4. Fast Evaluation of LS-SVM Scheme

Let us discuss the computational complexity of systems (3.2) at any time step t_m. The procedure involves two steps — viz.

(i) **Invert the stiffness matrix $\boldsymbol{A}_m$ to find the solution $\boldsymbol{v}_m$.** Because of the kernel method, the stiffness matrix is dense. Direct solvers usually have computational complexity $\mathcal{O}(N^3)$ to invert the system. Besides, $\mathcal{O}(N^2)$ of memory is needed to store the corresponding matrix $\boldsymbol{A}_m$. As numerical examples show, the kernel method has a high accuracy, so that only a small number of collocation points often suffice.

(ii) **Form the right-hand sides.** Due to the memory effects of tFDE, the construction of the right-hand side of the system (3.2) requires to keep all numerical solutions obtained at the previous time steps. Therefore, at the step t_m one needs $\mathcal{O}(mN)$ memory and $\mathcal{O}(mN)$ computational complexity. Reaching the final time step $t_M = T$ requires $\mathcal{O}(MN)$ memory computational complexity $\mathcal{O}(M^2N)$. This may well surpass the computational complexity of $\mathcal{O}(MN^3)$ for inverting the stiffness matrix, especially in long term simulation [45]. In fact, the current LS-SVM scheme falls into the category where the number N of spatial unknowns is relatively small due to the high-order accuracy of the kernel approximation [13,43]. Hence, the $\mathcal{O}(M^2N)$ computational complexity of forming the right-hand side is far more expensive than the $\mathcal{O}(MN^3)$ computational complexity of inverting the stiffness matrix. This effect is not common in the numerical approximations of the integer-order PDEs.

Extensive research was carried out to develop fast numerical methods of constant-order tFDE, by exploring their convolution [5,26] Toeplitz-like structures [3,6,10,15–17,19], or by utilising rational function approximations to the Laplace transform of the power-law

kernel of the fractional derivative that yields a finite sum-of-exponentials (SOE) approximation of the kernel [14, 18, 25, 35]. These methods have the computational complexity $\mathcal{O}(NM\log^l M)$, $l = 1, 2$. However, they rely on the Laplace transform or Toeplitz-like structure of the discretisation of the fractional derivatives. It is not clear how these methods can be extended to variable-order tFDEs, since the corresponding numerical discretisations do not have a convolution or Toeplitz-like structure. Hence, the Laplace transform of their kernel cannot be written in a closed form.

We extend the shifted binary block partition (SBBP) method [9], coauthored by one of the authors, to the fast evaluation of the LS-SVM scheme and compare with the SOE method [18], where at every time step t_m the time-fractional derivative is split into integrals over the interval $[t_{m-1}, t_m]$ and the interval $[t_0, t_{m-1}]$, which contains historical information of the solution and has no singularity in the kernel. The singular integral of the first-order time derivative over current time interval $[t_{m-1}, t_m]$ is discretised by the $L1$ formula. The corresponding interval over the historical part $[0, t_{m-1}]$ may be reduced to an integral, which can be approximated by a finite sum-of-exponential of $\mathcal{O}(\log M)$ terms. The development heavily relies on the Laplace transform of fractional derivatives. Consequently, the method is not directly applicable to the tFDE (1.2).

The SBBP strategy began with a splitting, similar to the one in [18]. Split the variable-order time derivative as an integral over few current intervals and intervals in the historical part of the time interval. The difference comes from the treatment of the integral on the historical interval, for which the SBBP method introduces an auxiliary shifted binary block partition (SBBP) of the underlying temporal partition. This formats $\mathcal{O}(\log M)$ macro time intervals by clustering the ones, which are away from the current time interval. For the macro time intervals of the SBBP partition, analysis shows that polynomial approximations of a fixed degree can guarantee uniform convergence rates. We underscore that this method does not use the Laplace transform or Toeplitz-like structures and can be applied to the tFDE (1.2). In this work, we combine the LS-SVM scheme with the fast SBBP strategy to develop a fast LS-SVM scheme for the two time-scale variable-order tFDE (1.2), which is demonstrated in Algorithm 4.1.

The method was introduced in [9] for a finite difference approximations of variable-order tFDEs with symmetric positive-definite sparse banded stiffness matrices. The corresponding systems can be solved iteratively by a conjugate gradient method. In the current LS-SVM discretisation, the kernel functions are global, so that the stiffness matrix is dense. Furthermore, due to the introduction of a Lagrange multiplier and the least-square approach, the stiffness matrix reduces to a saddle-point problem and is highly ill-conditioned. All of these issues are among the factors that substantially complicate the fast evaluation algorithm, which have to be handled in this paper.

5. Numerical Experiments

We carry out numerical experiments to investigate the performance of the LS-SVM scheme. All computations are implemented using MATLAB R2018a on a ThinkPad E431 laptop with Inter Core i5 (2.60 GHz) CPU and 8.0G RAM. In all experiments we measure

Algorithm 4.1 A fast LS-SVM scheme for the solution of problem (1.2)

Initial data: $\{\boldsymbol{x}_i^I, u_0(\boldsymbol{x}_i^I)\}_{i=1}^{N_I}$: Boundary information $\{\boldsymbol{x}_i^B, g(\boldsymbol{x}_i^B)\}_{i=1}^{N_B}$.
for $m = 1, \ldots, M$ **do**
1: Compute $\boldsymbol{h}_m$ in (3.3) with the known initial data and the solutions already computed at the previous time steps $(\hat{\boldsymbol{u}}_0, \hat{\boldsymbol{u}}_1, \cdots, \hat{\boldsymbol{u}}_{m-1})$.
2: Assemble matrix $\boldsymbol{A}_m$ in (3.2).
3.1: Calculate historical vector $\boldsymbol{h}_m^T$ by using the fast SBBP algorithm in [9].
3.2: Assemble vector $\boldsymbol{F}_m = (\boldsymbol{h}_m^T, \boldsymbol{g}_m^T, 0)^\top$.
4: Solve $\boldsymbol{A}_m \boldsymbol{v}_m = \boldsymbol{F}_m$ in (3.2), where $\boldsymbol{v}_m = (\boldsymbol{\xi}_m^T, \boldsymbol{\eta}_m^T, r_m)^\top$.
5: Compute $\hat{\boldsymbol{u}}_m$ using (3.4).
end for

the temporal convergence rate κ of the numerical scheme in the discrete L_∞ norm of the global truncation error at the terminal time T as

$$\|\boldsymbol{u}(t_M) - \hat{\boldsymbol{u}}_M\|_\infty = \max_{1\le i\le N} |u(\boldsymbol{x}_i, t_M) - \hat{u}_M(\boldsymbol{x}_i)| \le Q(M^{-\kappa}),$$

where $\boldsymbol{u}(t_M)$ is the exact solution and $\hat{\boldsymbol{u}}_M$ is the approximate solution at the final time step $t_M = T$.

We choose the kernel function $K(\boldsymbol{x}_1, \boldsymbol{x}_2) = \exp(-\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2^2/(2\sigma^2))$ with $\|\cdot\|_2$ referring to the Euclidean norm in $\mathbb{R}^d$ [13,43]. Noting that the penalty parameter λ depends on the number of samples, in all numerical experiments we set $\lambda = 10N_I$ to make the residual error sufficiently small. We use the cross-validation strategy to determine parameter σ. As Fig. 1 shows, one can choose $\sigma = 1.5$.

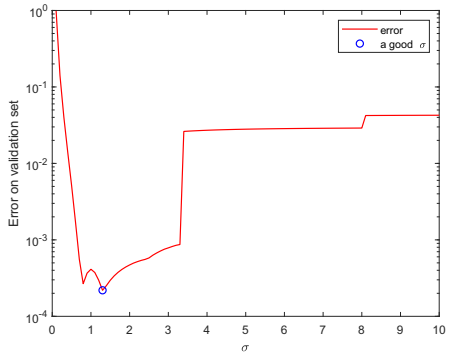

Figure 1: Errors for different values σ on validation set. The circle shows optimal value of σ.

5.1. Convergence rate and fast computation of LS-SVM scheme

Consider the problem (1.3) on the square domain $\Omega = (0,1)^2$ in the plane and the solution

$$u(x_1, x_2, t) = t^{2-\alpha(t)} \sin(\pi x_1) \sin(\pi x_2).$$

Besides, the time interval is $[0,1]$, $k(t)=1$ and

$$\alpha(t)=\alpha(1)+\big(\alpha(0)-\alpha(1)\big)\bigg((1-t)-\frac{\sin(2\pi(1-t))}{2\pi}\bigg),$$

so that the corresponding source term can be determined. It was shown in [41] that the solution to (1.2) can be sufficiently smooth with respect to time variable, provided that the domain Ω is simply-connected and $\alpha(t)$ has an integer limit as t approaches 0. In particular, if $\alpha(t)$, $k(t)\in C^1[0,T]$, $\alpha(0)=\alpha'(0)=0$, $\lim_{t\to 0^+}\alpha'(t)\ln t$ is finite and the initial value and the source term are regular, then $u(\boldsymbol{x},t)\in C^2[0,T]$ for every $\boldsymbol{x}\in\Omega$.

Since the kernel-based method has a high accuracy, we will use only 11 samples in every spatial direction — i.e. $N_x=N_y=11$ and $N=N_xN_y$ for testing the temporal convergence rate κ. In Table 1, the time step size is reduced from 1/16 to 1/128 in order to check the temporal convergence of the LS-SVM scheme. We observe that the developed LS-SVM scheme has first-order temporal convergence rate, which is consistent with the theoretical results. It is worth noting that the accuracy of the developed method can be improved by using high-order temporal discretisation schemes [7].

We also test the spatial convergence rate for different spatial step sizes under $M=2000$, cf. Table 2. In order to compare the performance of the LS-SVM scheme and the finite difference method (FDM), we list the results of the FDM errors and convergence order in Table 3 for the same data as in the LS-SVM scheme. It is clear that the LS-SVM scheme achieves a higher-order accuracy than the FDM. For example, for $\alpha(0)=0.0$ and $\alpha(1)=0.9$, the numerical error of the LS-SVM method has order $\mathcal{O}(10^{-5})$, whereas the FDM has order $\mathcal{O}(10^{-3})$ only.

Next we consider the performance of fast LS-SVM and LS-SVM schemes with a conventionally formulated right-hand side of (3.2). The consumed CPU time displayed in Ta-

Table 1: Temporal convergence of LS-SVM.

$(\alpha(0),\alpha(1))$	$(0.1,0.9)$		$(0.6,0.8)$		$(0.3,0.7)$	
M	$N_x=N_y=11$	κ	$N_x=N_y=11$	κ	$N_x=N_y=11$	κ
16	5.50E-3		4.86E-3		4.32E-3	
32	2.65E-3	1.05	2.32E-3	1.07	2.06E-3	1.06
64	1.32E-3	1.01	1.15E-3	1.01	1.03E-3	1.00
128	6.83E-4	0.95	6.01E-4	0.93	5.50E-4	0.91

Table 2: Spatial convergence of LS-SVM.

$(\alpha(0),\alpha(1))$	$(0.0,0.9)$		$(0.6,0.8)$		$(0.3,0.7)$	
$N_x=N_y$	$M=2000$	Order	$M=2000$	Order	$M=2000$	Order
6	1.31E-3		1.32E-3		1.32E-3	
8	4.84E-4	2.46	5.06E-4	3.36	5.21E-4	3.30
10	1.42E-4	4.26	1.51E-4	4.22	1.57E-4	4.16
12	7.09E-5	3.12	7.05E-5	3.40	7.10E-5	3.56

Table 3: Spatial convergence of FDM.

$(\alpha(0),\alpha(1))$	$(0.0,0.9)$		$(0.6,0.8)$		$(0.3,0.7)$	
$N_x = N_y$	$M = 2000$	Order	$M = 2000$	Order	$M = 2000$	Order
6	4.33E-2		4.35E-2		4.38E-2	
8	1.90E-2	2.02	1.91E-2	2.02	1.93E-2	2.02
10	1.07E-2	2.00	1.07E-2	2.02	1.08E-2	2.02
12	6.83E-3	2.00	6.87E-3	2.00	6.91E-3	2.00

Table 4: Consumed CPU time (in seconds) of LS-SVM and fast LS-SVM.

M	$N_x = N_y$	LS-SVM	fast LS-SVM
2×10^4	5	3 m 18 s	17 s
4×10^4	5	13 m 6 s	35 s
8×10^4	5	1 h 35 s	1 m 10 s
16×10^4	5	4 h 2 m 16 s	2 m 32 s
32×10^4	5	16 h 13 m 8 s	5 m 39 s

ble 4, shows a significant reduction of the CPU time of the fast LS-SVM. For instance, for $M = 32\times10^4$, the fast LS-SVM scheme needs less than 6 minutes of CPU time to finish the simulation, while the conventional LS-SVM scheme runs more than 16 hours. Another observation is that the doubling the number of time steps quadruples CPU time of the LS-SVM scheme. It is consistent with the computational complexity $\mathcal{O}(M^2N)$ of the standard time-marching method, which dominates the computational complexity $\mathcal{O}(MN^3)$ of the inverting the linear systems for $M > \mathcal{O}(N^2)$. In contrast, the consumed CPU time of the fast LS-SVM scheme is slightly higher than the double of the previous one. This also agrees with the results showing that the fast SBBP method assembles the right-hand side of the linear system (3.2) with the computational complexity $\mathcal{O}(MN\log^2 M)$.

5.2. Simulations for non-convex multiply-connected domains

Consider the variable-order problem (1.2) on the domain bounded by two stars with the polar coordinate representation

$$\Omega := \big\{(r,\theta)\,|\,0.3+0.2\sin(5\theta) \le r \le 0.6+0.2\sin(5\theta),\ 0\le\theta\le 2\pi\big\},$$

cf. Fig. 2. Other data are the same as in Section 5.1. We choose the analytical solution

$$u(x_1,x_2,t) = t^{2-\alpha(t)}\sin\big(x_1^2+x_2^2\big),\quad (x_1,x_2)\in\Omega,\quad t\in[0,T],$$

and compute the corresponding source term $f(x_1,x_2,t)$.

To handle the complex domain, in the numerical experiments we introduce the collocation points as follows. Considering the uniform partition θ by $\theta_i := i\Delta_\theta$, $i = 0,1,\ldots,20$ with the step size $\Delta_\theta := \pi/10$ of the interval $[0,2\pi]$, we then divide the interval $[0.3,0.6]$

in the r direction by the subintervals of the size 0.1. In particular, $r_I = 0.4, 0.5$ and $r_B = 0.3, 0.6$ correspond to the interior nodes and boundary nodes, respectively. After that, we define the collocation points in the physical domain by

$$\begin{aligned} x^I_{1,i} &:= \big(r_I + 0.2\sin(\theta_i)\big)\cos(\theta_i), \\ x^I_{2,i} &:= \big(r_I + 0.2\sin(\theta_i)\big)\sin(\theta_i), \\ r_I &= 0.4, 0.5 \end{aligned}$$

for the interior nodes and by

$$\begin{aligned} x^B_{1,i} &:= \big(r_B + 0.2\sin(\theta_i)\big)\cos(\theta_i), \\ x^B_{2,i} &:= \big(r_B + 0.2\sin(\theta_i)\big)\sin(\theta_i), \\ r_B &= 0.3, 0.6 \end{aligned}$$

for the boundary nodes. In all cases, we let $i = 1, 2, \ldots, 20$, so that $N_I = 40$, $N_B = 40$, and $N = N_I + N_B = 80$. We enforce the LS-SVM scheme at all these collocation points. Fig. 2 demonstrates he domain and the exact solution at terminal time T. Besides, Table 5 shows that the LS-SVM scheme for the tFDE (1.2) on a complex non-convex multiply-connected domain can achieve a high accuracy and the optimal convergence rate with only a few collocation points.

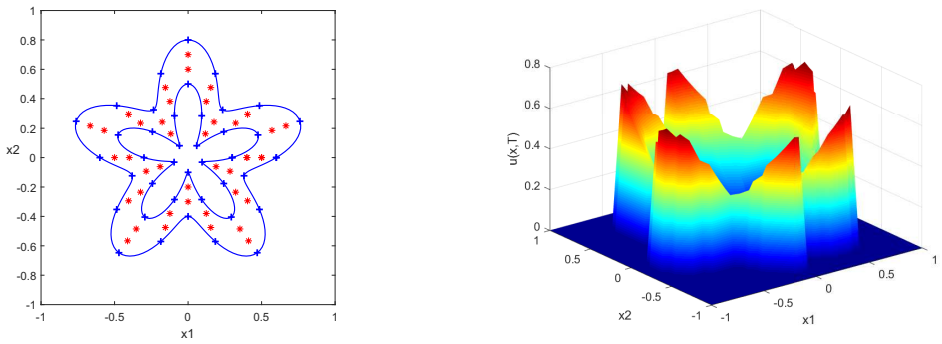

Figure 2: Left: Domain and collocation points; $*$ and $+$ show interior and boundary nodes, respectively. Right: True solution profile at terminal time T.

Table 5: Temporal convergence of LS-SVM scheme on complex, non-convex domain.

$(\alpha(0), \alpha(1))$	$(0.1, 0.9)$		$(0.6, 0.8)$		$(0.3, 0.7)$	
M	$N = 80$	κ	$N = 80$	κ	$N = 80$	κ
16	3.11E-4		2.62E-4		2.28E-4	
32	1.52E-4	1.03	1.27E-5	1.04	1.11E-4	1.04
64	7.56E-5	1.00	6.26E-5	1.02	5.49E-5	1.01
128	3.38E-5	1.16	2.76E-5	1.17	2.43E-5	1.17

5.3. Simulations for five-dimensional spatial domains

In this experiment we investigate the performance of the LS-SVM scheme for the variable-order tFDE in a five spatial dimensions. The order function is the same as in Section 5.1, $\Omega := (0,1)^5$ and $[0,T] = [0,1]$. We assume that the solution of the problem is

$$u = t^{2-\alpha(t)} \prod_{i=1}^{5} \sin(x_i),$$

and determine the corresponding source term. We use only 6 collocation nodes for each coordinate, hence the total number of collocation points is $N = 6^5$. Table 6 shows that although only a few collocation points in spatial space is used, the method has the optimal convergence order and allows to find a highly accurate approximate solution. Thus the LS-SVM method can handle high-dimensional problems and achieve high accuracy and efficiency, which is one of the most important advantages of the LS-SVM scheme.

Table 6: Temporal convergence of LS-SVM for high-dimensional variable-order tFDE.

$(\alpha(0),\alpha(1))$	$(0.1,0.9)$		$(0.6,0.8)$		$(0.4,0.8)$	
M	$N=6^5$	κ	$N=6^5$	κ	$N=6^5$	κ
16	8.82E-5		6.04E-5		7.71E-5	
32	4.34E-5	1.02	2.84E-5	1.08	3.60E-5	1.09
64	2.11E-5	1.04	1.19E-5	1.24	1.74E-5	1.05
128	9.31E-6	1.18	4.63E-6	1.37	7.13E-6	1.28

6. Concluding Remarks

We developed an efficient LS-SVM scheme for the two time-scale variable-order tFDE (1.2) with meshless kernel functions used in the spatial discretisation and $L1$-formula in the temporal discretisation. Reformulating the temporal semi-discrete scheme as a minimisation problem, we derived the KKT optimality condition, so that the numerical solution can be obtained by solving a linear system [40]. The scheme enjoys both the flexibility and accuracy of the kernel approach and the efficiency of the recently developed SBBP method for evaluating the variable-order fractional derivative [9, 13, 29, 43]. Numerical experiments show that the LS-SVM scheme has the optimal convergence rate in time and generates accurate numerical solution of the equation considered on complex non-convex multiply-connected domains or in high space dimensions. These features can play an important role in various applications.

Acknowledgments

The authors would like to express their most sincere thanks to the referees for their very helpful comments and suggestions, which greatly improved the quality of this paper.

This work was funded in part by the ARO MURI Grant W911NF-15-1-0562, by the National Science Foundation under Grants DMS-1620194 and DMS-2012291, by the National Natural Science Foundation of China under Grant 11971272, and by the China Postdoctoral Science Foundation under Grants 2020M681136 and 2021T140129.

References

[1] P. Baveye, P. Vandevivere, B.L. Hoyle, P.C. DeLeo and D.S. de Lozada, *Environmental impact and mechanisms of the biological clogging of saturated soils and aquifer materials*, Crit. Rev. Env. Sci. Tech. **28**, 123–191 (2006).

[2] J. Bear, *Dynamics of Fluids in Porous Media*, Elsevier (1972).

[3] R.H. Chan and M.K. Ng, *Conjugate gradient methods for Toeplitz systems*, SIAM Rev. **38**, 427–482 (1996).

[4] C. Cortes and V. Vapnik, *Support-vector networks*, Mach. Learn. **20**, 273–297 (1995).

[5] K. Diethelm, N. Ford, A. Freed and Y. Luchko, *Algorithms for the fractional calculus: a selection of numerical methods*, Comput. Meth. Appl. Mech. Engrg. **194**, 743–773 (2005).

[6] N. Du and H. Wang, *A fast finite element method for space-fractional dispersion equations on bounded domains in* $\mathbb{R}^2$, SIAM J. Sci. Comput. **37**, A1614–A1635 (2015).

[7] R. Du, A. Alikhanov and Z. Sun, *Temporal second order difference schemes for the multi-dimensional variable-order time fractional sub-diffusion equations*, Comput. Math. Appl. **10**, 2952–2972 (2020).

[8] H. Fan, Y. Zhao, F. Wang, Y, Shi and Y. Tang, *A superconvergent nonconforming mixed FEM for multi-term time-fractional mixed diffusion and diffusion-wave equations with variable coefficients*, East Asian J. Appl. Math. **11**, 63–92 (2021).

[9] Z. Fang, H. Sun and H. Wang, *A fast method for variable-order Caputo fractional derivative with applications to time-fractional diffusion equations*, Comput. Math. Appl. **80**,1443–1458 (2020).

[10] H. Fu, M. Ng and H. Wang, *A divide-and-conquer fast finite difference method for space-time fractional partial differential equation*, Comput. Math. Appl. **73**, 1233–1242 (2017).

[11] L. Gandossi, *An overview of hydraulic fracturing and other formation stimulation technologies for shale gas production*, Eur. Commisison Jt. Res. Cent. Tech. Reports. 26347 (2013).

[12] G. Gao and Q. Yang, *Fast evaluation of linear combinations of Caputo fractional derivatives and its applications to multi-term time-fractional sub-diffusion equations*, Numer. Math. Theor. Meth. Appl. **13**, 433–451 (2020).

[13] I. Goodfellow, Y. Bengio and A. Courville, *Deep Learning*, The MIT Press (2016).

[14] L. Greengard and J. Strain, *A fast algorithm for the evaluation of heat potentials*, Commun. Pure Appl. Math. **43**, 949–963 (1990).

[15] J. Jia and H. Wang, *A preconditioned fast finite volume scheme for a fractional differential equation discretized on a locally refined composite mesh*, J. Comput. Phys. **299**, 842–862 (2015).

[16] J. Jia and H. Wang, *A fast finite volume method on locally refined meshes for fractional diffusion equations*, East Asian J. Appl. Math. **9**, 755–779 (2019).

[17] J. Jia and H. Wang, *A fast finite volume method for conservative space-time fractional diffusion equations discretized on space-time locally refined meshes*, Comput. Math. Appl. **78**, 1345–1356 (2019).

[18] S. Jiang, J. Zhang, Q. Zhang Q, Z. Zhang, *Fast evaluation of the Caputo fractional derivative and its applications to fractional diffusion equations*, Commun. Comput. Phys. **21**, 650–678 (2017).

[19] R. Ke, M.K. Ng and H.-W. Sun, *A fast direct method for block triangular Toeplitz-like with tridiagonal block systems from time-fractional partial differential equations*, J. Comput. Phys. **303**, 203–211 (2015).
[20] Z. Li, H. Wang, R. Xiao and S. Yang, *A variable-order fractional differential equation model of shape memory polymers*, Chaos, Solitons Fract. **102**, 473–485 (2017).
[21] Y. Lin and C. Xu, *Finite difference/spectral approximations for the time-fractional diffusion equation*, J. Comput. Phys. **225**, 1552–1553 (2007).
[22] X. Liu and M. Stynes, *An alternative finite difference stability analysis for a multiterm time-fractional initial-boundary value problem*, East Asian J. Appl. Math. **10**, 427–436 (2020).
[23] Z. Liu, X. Li and X. Zhang, *A fast high-order compact difference method for the fractal mobile/immobile transport equation*, Int. J. Comput. Math. **97**, 1860–1883 (2020).
[24] C.F. Lorenzo and T.T. Hartley, *Variable order and distributed order fractional operators*, Nonlinear Dynamics **29**, 57–98 (2002).
[25] C. Lubich, *Convolution quadrature and discretized operational calculus. I*, Numer. Math. **52**, 129–145 (1988).
[26] W. McLean, *Fast summation by interval clustering for an evolution equation with memory*, SIAM J. Sci. Comput. **34**, A3039–A3056 (2012).
[27] M.M. Meerschaert and A. Sikorskii, *Stochastic Models for Fractional Calculus*, De Gruyter Studies in Mathematics (2011).
[28] S. Mehrkanoon, T. Falck and J.A. Suykens, *Approximate solutions to ordinary differential equations using least squares support vector machines*, IEEE Trans. Neural Networks Learn. Syst. **23**, 1356–1367 (2012).
[29] S. Mehrkanoon and J.A. Suykens, *Learning solutions to partial differential equations using LS-SVM*, Neurocomputing. **159**, 105–116 (2015).
[30] R. Metzler and J. Klafter, *The random walk's guide to anomalous diffusion: A fractional dynamics approach*, Phys. Rep. **339**, 1–77 (2000).
[31] I. Podlubny, *Fractional Differential Equations*, Academic Press (1999).
[32] H. Qiao and A. Cheng, *Convergence of finite difference method in positive time for multi-term time fractional differential equations*, East Asian J. Appl. Math. **10**, 774–785 (2020).
[33] K. Sakamoto and M. Yamamoto, *Initial value/boundary value problems for fractional diffusion-wave equations and applications to some inverse problems*, J. Math. Anal. Appl. **382**, 426–447 (2011).
[34] S. Samko and B. Ross, *Integration and differentiation to a variable fractional order*, Integral Transform. Spec. Funct. **1**, 277–300 (1993).
[35] A. Schädle, M. López-Fernández, C. Lubich, *Fast and oblivious convolution quadrature*, SIAM J. Sci. Comput. **28**, 421–438 (2006).
[36] R. Schumer, D.A. Benson, M.M. Meerschaert and B. Baeumer, *Fractal mobile/immobile solute transport*, Water Resour. Res. **39**, 1296 (2003).
[37] M. Stynes, E. O'Riordan and J.L. Gracia, *Error analysis of a finite difference method on graded mesh for a time-fractional diffusion equation*, SIAM Numer. Anal. **55**, 1057–1079 (2017).
[38] H. Sun, A. Chang, Y. Zhang, W. Chen, *A review on variable-order fractional differential equations: mathematical foundations, physical models, numerical methods and applications*, Fract. Calc. Appl. Anal. **22**, 27–59 (2019).
[39] Z. Sun and X. Wu, *A fully discrete difference scheme for a diffusion-wave system*, Appl. Numer. Math. **56**, 193–209 (2006).
[40] J.A.K. Suykens, *Least Squares Support Vector Machines*, World Scientific (2002).
[41] H. Wang and X. Zheng, *Wellposedness and regularity of the variable-order time-fractional diffusion equations*, J. Math. Anal. Appl. **475**, 1778–1802 (2019).

[42] X. Wang, S. Xu, S. Zhou, W. Xu, M. Leary, P. Choong, M. Qian, M. Brandt and Y.M. Xie, *Topological design and additive manufacturing of porous metals for bone scaffolds and orthopedic implants*, Biomaterials **83**, 127–141 (2016).
[43] C.K. Williams and C.E. Rasmussen, *Gaussian Processes for Machine Learning*, MIT Press (2006).
[44] Z. Yang, X. Zheng, Z. Zhang and H. Wang, *A variably distributed-order time-fractional diffusion equation: analysis and approximation*, Comput. Methods Appl. Mech. Eng. **367**, 113118 (2020).
[45] M. Zhao and H. Wang, *Fast finite difference methods for space-time fractional partial differential equations in three space dimensions with nonlocal boundary conditions*, Appl. Numer. Math. **145**, 411–428 (2019).
[46] X. Zheng and H. Wang, *Optimal-order error estimates of finite element approximations to variable-order time-fractional diffusion equations without regularity assumptions of the true solutions*, IMA J. Numer. Anal. **41**, 1522–1545 (2021).
[47] X. Zheng and H. Wang, *A hidden-memory variable-order fractional optimal control model: analysis and approximation*, SIAM J. Control Optim. **59**, 1851–1880 (2021).