Research Article ISSN Online: 2783-4859 Computational Mathematics and Computer Modeling with Applications

Received 19 August 2025
Accepted 2 October 2025

DOI: 10.48308/CMCMA.4.2.9
AMS Subject Classification: MOS1; MOS2

A novel Müntz polynomial-based least squares support vector regression method for solving fractional optimal control problems

Mitra Bolhasani[a] and Kourosh Parand[b,c]

This paper introduces a novel integration of Müntz polynomials into the Least Squares Support Vector Regression framework for addressing fractional optimal control problems. By utilizing Müntz basis functions as the mapping mechanism to project the problem into a higher-dimensional space, the proposed method reformulates the optimization challenge and resolves it efficiently through Maple's optimization tools. The effectiveness of this technique is validated via numerical experiments on benchmark fractional optimal control cases. Outcomes reveal that the approach delivers high precision in solving these problems, surpassing existing techniques in terms of accuracy and efficiency. Copyright © 2025 Shahid Beheshti University.

Keywords: Least squares support vector machines; Fractional optimal control problems; Müntz polynomials; Regression; Artificial intelligence.

1. Introduction

Optimal control problems (OCPs) play a pivotal role in numerous scientific and engineering disciplines, enabling the determination of optimal strategies for managing dynamic systems. These problems are instrumental in fields such as physics, biology, and engineering, where they facilitate performance enhancement, cost reduction, and efficiency improvement.

Fractional calculus has emerged as a robust framework for modeling intricate phenomena that elude traditional integer-order calculus. By incorporating derivatives and integrals of non-integer orders, it captures systems exhibiting memory and hereditary traits [1]. This paradigm has found applications in diverse domains, including materials science for elucidating long-memory and multiscale behaviors [1], and biology for simulating tissue electrical characteristics [2].

Integrating fractional calculus into OCPs yields substantial advantages, allowing for refined representations of complex dynamics and superior optimization outcomes. For example, in bioengineering, it has fostered innovative models bridging micro- and macro-scale phenomena, thereby augmenting control over biological mechanisms [3]. In engineering contexts, fractional models often surpass conventional ones in precision and efficacy [4].

A variety of numerical techniques have been devised by scholars to tackle fractional optimal control problems (FOCPs), with selected approaches summarized as follows: Among these, methods utilizing Caputo fractional derivatives involve deriving Euler-Lagrange equations through calculus of variations and fractional integration by parts, as explored in references [5] and [6]. Approaches based on Riemann-Liouville fractional derivatives rely on numerical resolution using Grunwald-Letnikov approximations, detailed in [7] and [8]. Finite element methods provide approximations for FOCPs governed by linear elliptic or parabolic equations, as discussed in [9]. Rational approximations recast FOCPs using techniques like Oustaloup's for fractional operators, covered in [10] and [11]. Eigenfunction methods achieve dimensional reduction of FOCPs to enable decoupled solutions, per [12]. Direct methods discretize the problem to form optimization tasks, incorporating tools such as the Clenshaw-Curtis formula or modified Adomian decomposition, as in [13] and [14]. Indirect methods solve nonlinear equations derived from optimality conditions, including Pontryagin's principle, referenced in [15]. Spectral methods employ Galerkin or pseudo-spectral strategies for time-fractional FOCPs, outlined in [16] and [17]. Wavelet-based methods reduce problems to algebraic

[a] *Department of Applied Mathematics, Faculty of Mathematical Sciences, Shahid Beheshti University, Tehran, Iran.*
[b] *Department of Computer and Data sciences, Faculty of Mathematical Sciences, Shahid Beheshti University, Tehran, Iran.*
[c] *Department of Cognitive Modeling, Institute for Cognitive and Brain Sciences, Shahid Beheshti University, Tehran, Iran.*
[*] *Correspondence to: K. Parand. Email: k_parand@sbu.ac.ir*

or programming forms using Bernoulli or biorthogonal wavelets, as in [18] and [19]. In a related approach, operational matrices based on Bernoulli polynomials have been developed to transform the problem into a system of algebraic equations that can be solved numerically [44]. Polynomial approximations apply the Ritz method with a Legendre basis, solved via Newton's iteration, per [20]. Linear programming transforms fractional equations to minimize error, as in [21]. Finally, neural network approaches offer adaptive estimation of Hamiltonian solutions through polynomial expansions, detailed in [22].

Contemporary developments in machine learning for FOCPs encompass innovative strategies. Notable examples include hybrid operational matrix and shooting methods for initial condition challenges [23], differential transforms with Vandermonde matrices for quadratic FOCPs [24], and Lucas polynomial transformations simplifying algebraic systems [25]. Parameter optimization in fractional models further refines control to align predictions with targets [26].

Machine learning paradigms, notably support vector machines (SVMs), exhibit versatility across challenges. Originating from Cortes and Vapnik for classification [27], SVMs convert issues to constrained quadratic programs yielding unique solutions. Suykens et al. advanced this with least squares SVM (LS-SVM) [28], enhancing efficiency through linear equation systems. Subsequently, LS-SVM extended to nonlinear control [29].

This study advances the LS-SVM foundation by integrating least squares support vector regression (LS-SVR) with Müntz polynomials, tailored for FOCPs of the form:

$$\min(\max)J[x,u]=\int_{t_0}^{t_1} f\left(t,x(t),u(t)\right)dt, \tag{1}$$

subject to:

$$D^{\alpha}x(t)=g\left(t,x(t)\right)+b(t)u(t),\quad n-1\le\alpha\le n,\quad n=2,3,\dots \tag{2}$$

with initial conditions:

$$x(t_0)=x_0,\quad \dot{x}(t_0)=x_1,\dots,x^{(\lceil\alpha\rceil-1)}(t_0)=x_{(\lceil\alpha\rceil-1)}. \tag{3}$$

Here, D^{α} signifies the Caputo fractional derivative, $b(t)\neq 0$, and f, g are smooth on $[0,1]$.
Leveraging Suykens et al. (2001) [29], our method incorporates key innovations:

- Müntz polynomials as mapping functions, yielding algebraic equations via their orthogonality, enhancing precision over standard bases.
- Maple's optimization suite for resolving LS-SVR-associated problems, surpassing prior nonlinear techniques.
- Pioneering LS-SVR application to FOCPs with Müntz integration, evidenced by high-accuracy examples.

LS-SVR's breadth spans fields; e.g., Pakniyat et al. combined it with spectral methods for fractional equations [30], Parand et al. addressed Volterra integrals [31, 33], and Rahimkhani and Ordokhani tackled stochastic equations [32], underscoring its adaptability.

The paper structure follows: Section 2 outlines preliminaries on polynomial spaces, fractional calculus, and LS-SVR. Section 3 formulates the LS-SVR adaptation for OCPs. Section 4 evaluates via numerical experiments. Section 5 concludes the paper.

2. Preliminaries

This section outlines essential concepts, encompassing fractional integrals and derivatives, Müntz polynomials, and LS-SVR.

2.1. Fractional Integrals and Derivatives

Definition 1: The Riemann-Liouville fractional integral of order $\alpha\ge 0$ for a function $x(t)$ is given by [34]:

$${}_0I_t^{\alpha}x(t)=\frac{1}{\Gamma(\alpha)}\int_0^t (t-\tau)^{\alpha-1}x(\tau)\,d\tau, \tag{4}$$

where the Gamma function $\Gamma(\alpha)$ is expressed as:

$$\Gamma(\alpha)=\int_0^{\infty} t^{\alpha-1}e^{-t}\,dt. \tag{5}$$

Properties Key characteristics of the Riemann-Liouville fractional integral include:

- **Linearity:** For functions $x_1(t)$ and $x_2(t)$, along with constants a and b,

$$I^{\alpha}\left(ax_1(t)+bx_2(t)\right)=aI^{\alpha}x_1(t)+bI^{\alpha}x_2(t). \tag{6}$$

- **Semigroup Property:** When $\alpha,\beta>0$,

$$I^{\alpha}(I^{\beta}x(t))=I^{\alpha+\beta}x(t). \tag{7}$$

- **Reduction to Identity:** In the case where $\alpha=0$,

$$I^{0}x(t)=x(t). \tag{8}$$

Definition 2: The Caputo fractional derivative of order α for a function $x(t)$ is defined as [35]:

$${}_0D_t^{\alpha}x(t):=\left(\frac{d}{dt}\right)^n\left({}_0I_t^{n-\alpha}x\right)(t)=\frac{1}{\Gamma(n-\alpha)}\left(\frac{d}{dt}\right)^n\int_0^t(t-\tau)^{n-\alpha-1}x(\tau)\,d\tau, \tag{9}$$

where $\alpha>0$ is the derivative order, n is the smallest integer exceeding α, and $n-1<\alpha\leq n$.

Properties The Caputo fractional derivative possesses specific attributes that render it effective for problems involving initial conditions:

- **Linearity:** For functions $x_1(t)$ and $x_2(t)$, and constants a and b,

$$D^{\alpha}(ax_1(t)+bx_2(t))=aD^{\alpha}x_1(t)+bD^{\alpha}x_2(t). \tag{10}$$

- **Connection to Riemann-Liouville Derivative:** The Caputo derivative relates to the Riemann-Liouville derivative D_{RL}^{α} through:

$$D^{\alpha}x(t)=I^{n-\alpha}D^{n}x(t), \tag{11}$$

  where n is the smallest integer larger than α.

2.2. Müntz Polynomials

Consider a sequence of exponents that fulfills $0\leq\lambda_0<\lambda_1<\cdots\to\infty$. The well-known MüntzSzsz theorem [36] asserts that Müntz polynomials, which can be represented in the form:

$$M_N(t)=\sum_{k=0}^{N}a_k t^{\lambda_k}, \tag{12}$$

using real coefficients $\{a_k\}_{k=0}^{N}$, constitute a dense subset within $L^2[0,1]$ if and only if the condition

$$\sum_{k=1}^{\infty}\lambda_k^{-1}=\infty \tag{13}$$

holds true.

Moreover, when incorporating the constant function 1 into the framework, namely, $\lambda_0=0$, the density of Müntz polynomials in $C[0,1]$ under the uniform norm is likewise determined by the condition outlined in Eq. (13). For organizing these functions, the standard Müntz basis is specified as

$$\chi_M=(t^{\lambda_0},t^{\lambda_1},\dots)^T. \tag{14}$$

As a result, the related Müntz spaces are defined by

$$M_n(\Lambda)=\operatorname{span}\{t^{\lambda_0},t^{\lambda_1},\dots,t^{\lambda_n}\}. \tag{15}$$

In addition, the overall space encompassing all Müntz polynomials is expressed as

$$M(\Lambda)=\bigcup_{n\in\mathbb{N}_0}M_n(\Lambda)=\operatorname{span}\{t^{\lambda_0},t^{\lambda_1},\dots\}, \tag{16}$$

where $\Lambda=\{\lambda_k\}_{k\in\mathbb{N}_0}$ denotes the collection of exponents.

2.3. LS-SVR Formulations

Support Vector Regression (SVR) is commonly acknowledged in research circles as a powerful method for tackling regression challenges, especially in cases where the connections between inputs and outputs display intricate, mostly nonlinear behaviors. The primary aim of SVR is to create a model that effectively represents the relationships between inputs and outputs, ensuring that forecasts stay within a specified error limit, known as ϵ, compared to the intended values. This is achieved through the optimization of a regularized risk measure, which strikes a balance between the model's intricacy and its forecasting accuracy, thus avoiding overfitting while preserving strong performance. A distinctive feature of SVR is its reliance on a limited group of training data pointscalled support vectorsthat are positioned on or beyond the ϵ-boundary and exclusively shape the ultimate model. This targeted dependence lowers computational demands and boosts the model's precision and speed. For a provided dataset $\{x_i, y_i\}_{i=1}^n$, with $x_i \in \mathbb{R}^m$ as the input vector and $y_i \in \mathbb{R}$ as the corresponding output, the SVR function is given by [37]:

$$y(x) = w^T \varphi(x) + b, \tag{17}$$

where $w \in \mathbb{R}^m$ serves as the weight vector, $\varphi(x)$ is a nonlinear mapping that elevates the input data into a higher-dimensional feature space, and $b \in \mathbb{R}$ is the bias component. The values of w and b are determined by solving this optimization problem:

$$\min_{w,b,\xi,\xi^*} \frac{1}{2} w^T w + \gamma \sum_{i=1}^n (\xi_i + \xi_i^*), \tag{18}$$

under the conditions:

$$y_i - (w^T \varphi(x_i) + b) \le \epsilon + \xi_i, \quad (w^T \varphi(x_i) + b) - y_i \le \epsilon + \xi_i^*, \tag{19}$$

and $\xi_i, \xi_i^* \ge 0$ for $i = 1, \dots, n$, where ξ_i and ξ_i^* are slack variables that handle deviations exceeding the ϵ-margin.

Within existing studies, Least Squares Support Vector Regression (LS-SVR) presents an enhanced version of SVR by substituting the standard ϵ-insensitive loss with a squared error cost function. This adjustment shifts the optimization from a quadratic programming challenge to a simpler set of linear equations, which notably increases computational reliability and speed, particularly for extensive datasets. The LS-SVR framework is defined as:

$$\min_{w,b,e} \mathcal{J}(w, e) = \frac{1}{2} w^T w + \frac{\gamma}{2} \sum_{i=1}^n e_i^2, \tag{20}$$

constrained by:

$$y_i = w^T \varphi(x_i) + b + e_i, \quad i = 1, \dots, n,$$

where γ functions as a regularization factor, adjusting the equilibrium between model sophistication and the scale of approximation errors, and $e_i \in \mathbb{R}$ indicates the error residual for each data point [38]. To address this optimization, the method of Lagrange multipliers is applied, formulated as:

$$\mathcal{L}(w, b, e, \alpha) = \frac{1}{2} w^T w + \frac{\gamma}{2} \sum_{i=1}^n e_i^2 - \sum_{i=1}^n \alpha_i \left(w^T \varphi(x_i) + b + e_i - y_i \right), \tag{21}$$

with α_i as the Lagrange multipliers. The best solution emerges by identifying the saddle point of the Lagrangian via the Karush-Kuhn-Tucker (KKT) criteria. This leads to a linear equation system, as described in [39]:

$$\left[\begin{array}{c|c} 0 & 1_n{}^T \\ \hline 1_n & \Omega + \gamma^{-1} I \end{array} \right] \left[\begin{array}{c} b \\ \hline \alpha \end{array} \right] = \left[\begin{array}{c} 0 \\ \hline y \end{array} \right], \tag{22}$$

where I denotes the identity matrix, $\Omega_{ij} = \varphi(x_i)^T \varphi(x_j)$ for $i, j = 1, \dots, n$, $y = [y_1, \dots, y_n]^T$ is the output vector, $1_n = [1, \dots, 1]^T$, and $\alpha = [\alpha_1, \dots, \alpha_n]^T$ are the multipliers.

The ideal parameters w, b, and α are extracted from this linear setup and utilized to form the LS-SVR model. A key benefit of LS-SVR compared to conventional SVR is its avoidance of quadratic programming, rendering it exceptionally efficient and appropriate for regression applications.

3. The Proposed Approach

In this section, we explain the suggested computational approach called CA-LSSVR in detail, which combines LS-SVR with Müntz basis functions, specifically tailored to address the FOCPs. In the following, we explained mathematical formulations of the FOCPs. Consider the following dynamical system that describes a process occurring within a given period $[t_a, t_b]$:

$$D^\alpha x(\tau) = g\left(\tau, x(\tau)\right) + b(\tau)u(\tau), \quad \text{for } n-1 \le \alpha \le n \quad \text{and } n = 2, 3, \ldots \tag{23}$$

The system's initial conditions are specified as follows:

$$x(t_a) = x_0, \quad \dot{x}(t_a) = x_1, \ldots, x^{(\lceil\alpha\rceil-1)}(t_a) = x_{(\lceil\alpha\rceil-1)}. \tag{24}$$

The system is defined by the state variable $x(\cdot) : [t_a, t_b] \to \mathbb{R}$, the control variable $u(\cdot) : [t_a, t_b] \to \mathbb{R}$. The objective of solving this problem is to find the control variable $u(\cdot)$ and the state variable $x(\cdot)$ that minimize or maximize the value of the performance functional $J[x, u]$. The problem formulation is as follows

$$\min J[x, u] = \int_{t_a}^{t_b} f\left(\tau, x(\tau), u(\tau)\right) d\tau. \tag{25}$$

If $t_a \neq 0$ or $t_b \neq 1$, we introduce the transformation:

$$\tau = t_a + (t_b - t_a)t. \tag{26}$$

The variable transformation results in t being in the interval $[0, 1]$, which corresponds to τ being in the range $[t_a, t_b]$. By utilizing Equation (26), we can express fractional optimal control problem stated in equations (23)-(25) as follows:

$$D^\alpha x(t) = (t_b - t_a)^\alpha \left[g\left(t_a + (t_b - t_a)t, x(t)\right) + b(t)u(t)\right], \quad \text{for } n-1 \le \alpha \le n \quad \text{and } n = 2, 3, \ldots \tag{27}$$

and the trajectory $x(t)$ that corresponds to the specified initial conditions:

$$x(0) = x_0, \quad \dot{x}(0) = (t_b - t_a)x_1, \ldots, x^{(\lceil\alpha\rceil-1)}(0) = (t_b - t_a)^{(\lceil\alpha\rceil-1)} x_{(\lceil\alpha\rceil-1)}, \tag{28}$$

minimizes:

$$\min J[x, u] = (t_b - t_a) \int_0^1 f\left(t_a + (t_b - t_a)t, x(t), u(t)\right) dt. \tag{29}$$

To address this problem numerically, we approximate the state and control functions using Müntz polynomials, which are fine-tuned to represent non-integer order dynamics. These approximations are formulated as:

$$x_N(t) = \sum_{k=0}^{N} a_k t^{\lambda_k}, \quad u_N(t) = \sum_{k=0}^{N} b_k t^{\lambda_k},$$

where $\{\lambda_k\}_{k=0}^N$ is a sequence of non-negative exponents, a_k and b_k are coefficients to be determined, and N specifies the number of terms, indicating the approximations precision. The approximate performance function is then expressed as:

$$J_N = (t_b - t_a) \int_0^1 f(t_a + (t_b - t_a)t, x_N(t), u_N(t))\, dt.$$

In the following, we utilize the LS-SVR method to reformulate FOCPs into an optimization problem by enforcing the dynamical constraints and initial conditions at a discrete set of collocation points $\{t_i\}$ (e.g., uniformly spaced points $t_i = ih$, where $h = 1/m$ and $i = 0, \ldots, m$), while introducing error terms to accommodate approximations. These constraints are given by:

$$D^\alpha x_N(t_i) = (t_b - t_a)^\alpha \left[g(t_i, x_N(t_i)) + b(t_i)u_N(t_i)\right] + e_i, \quad i = 0, \ldots, m,$$

$$x_N(0) = x_0 + e_{m+1}, \quad x'_N(0) = (t_b - t_a)x_1 + e_{m+2}, \quad \ldots, \quad x_N^{(n-1)}(0) = (t_b - t_a)^{n-1}x_{n-1} + e_{m+n},$$

$$J_N = e_{m+n+1},$$

where e_i (for $i = 1, \ldots, m+n+1$) are error variables introduced to integrate the constraints into a least-squares framework.

To clarify the rationale behind this formulation, it is essential to address how the original optimal control problem is transformed to fit within the LS-SVR framework. This transformation rests on three foundational concepts.

First, the treatment of the performance index J_N as an equality constraint, $J_N = e_{m+n+1}$, is a crucial step to unify all components of the FOCP. In the standard LS-SVR paradigm, the optimization process is driven entirely by minimizing a sum of squared errors derived from a set of equality constraints. The original FOCP, however, contains both an objective function to be minimized (J) and a set of constraints (dynamics and initial conditions). To integrate the objective function into this constraint-based framework, we re-conceptualize it as an additional condition that must be satisfied. By setting J_N equal to an error term, we effectively convert the goal of "minimizing J_N" into the equivalent goal of "driving the error term e_{m+n+1} as close to zero as possible." This allows us to handle both the system's physical constraints and its performance objective within a single, consistent mathematical structure.

Second, minimizing the composite sum of squared errors, $\sum_{i=1}^{m+n+1} e_i^2$, directly leads to the optimal control solution. This objective function acts as a multi-objective loss function where each term corresponds to a specific requirement of the FOCP. Minimizing the error terms $e_0, \dots, e_{m+n}$ forces the approximate solution to satisfy the fractional dynamics and initial conditions at the collocation points. Simultaneously, minimizing the final error term, e_{m+n+1}^2, directly minimizes J_N^2. Since the performance index J in most FOCPs is an integral of non-negative terms, minimizing its squared value is equivalent to minimizing its actual value. Thus, the minimization of this unified sum of squares naturally balances the need to adhere to the system's constraints with the primary goal of optimizing the performance index, guiding the solver toward the true optimal control trajectory.

Finally, this formulation represents a significant generalization of the classic LS-SVR method for regression. In standard regression, LS-SVR is used to fit a function to a static dataset $\{(x_i, y_i)\}$ by satisfying constraints of the form $y_i = f(x_i) + e_i$, where the y_i are known, fixed target values. Our approach extends this concept to a dynamic optimization context where such explicit targets do not exist. Instead, we have operator equations (the system dynamics and initial conditions) that must ideally evaluate to zero. For example, the dynamic constraint can be viewed as forcing the residual, $(D^\alpha x_N(t_i) - g(\dots) - bu_N(t_i))$, to a target of zero. In this sense, we have generalized LS-SVR from a data-fitting tool into a powerful solver for functional equations. The core principle of minimizing a regularized sum of squared residuals is preserved, but its application is elevated to solve a much broader class of problems, namely, constrained dynamic optimization.

The LS-SVR optimization problem is thus defined as:

$$
\begin{aligned}
\min_{a,b,e} \quad & \frac{1}{2}\left(\sum_{k=0}^{N} a_k^2 + \sum_{k=0}^{N} b_k^2\right) + \frac{\gamma}{2}\sum_{i=1}^{m+n+1} e_i^2 \\
\text{subject to:} \quad & D^\alpha x_N(t_i) = (t_b - t_a)^\alpha \left[g(t_i, x_N(t_i)) + b(t_i)u_N(t_i)\right] + e_i, \quad i = 0, \dots, m, \\
& x_N(0) - x_0 = e_{m+1}, \\
& x_N'(0) - (t_b - t_a)x_1 = e_{m+2}, \\
& \vdots \\
& x_N^{(n-1)}(0) - (t_b - t_a)^{n-1}x_{n-1} = e_{m+n}, \\
& J_N = e_{m+n+1},
\end{aligned}
$$

where $\gamma > 0$ is a regularization parameter that balances solution complexity against constraint accuracy. By embedding the performance functional J_N as a constraint with an associated error term e_{m+n+1}, the proposed method optimizes both the control and state trajectories concurrently, while ensuring compliance with the fractional dynamics. While standard (SVM) formulations rely on quadratic programming solvers, our (LS-SVR) approach for (FOCPs) incorporates the performance index J_N, which may render the objective function nonlinear depending on f. To handle this, we employ Maples general optimization packages (e.g., NLPSolve), capable of solving smooth nonlinear programming problems. This ensures applicability even when J_N introduces non-quadratic terms, as validated by the numerical results in Section 4.

4. Numerical results

In this section, we conduct a comprehensive numerical analysis by experimenting with the proposed Müntz Polynomial-Based LS-SVR method on four benchmark examples to evaluate its computational accuracy and efficiency. To achieve these requirements, we performed numerical computations using Maple on a Windows operating system with a computer equipped with a Core i5 processor, with a precision of 60 digits. In each example, the state variable x and the control variable u were approximated using Müntz polynomials with N terms: $x_N(t) = \sum_{k=0}^{N} a_k t^{\lambda_k}$, $u_N(t) = \sum_{k=0}^{N} b_k t^{\lambda_k}$. This approach enables effective control over the approximation error.

To assess the accuracy of the proposed method, the mean absolute error (MAE) is computed by comparing the differences between the exact and approximate solutions. The required CPU time for each simulation is also recorded to demonstrate the computational efficiency of our approach. In the following, Tables 1, 2, 4, and 5 list the obtained results corresponding to each example, highlighting the superiority of the proposed method compared to other state-of-the-art approaches. Furthermore, to address the convergence properties of our method, we present plots of the MAE versus the polynomial order N for each example. These plots visually confirm that the approximation error decreases systematically as N increases, validating the robustness and reliability of the proposed algorithm. As depicted in Algorithm 1, we provide the algorithmic details of the implementation steps.

Algorithm 1 Step-by-step pseudo-code for assembling and solving the FOCP using the CA-LSSVR method.

```
Require: Problem data: $g(t,x)$, $b(t)$, $f(t,x,u)$, initial conditions $\{x_k\}_{k=0}^{n-1}$, interval $[t_a,t_b]$, and order $\alpha$.
    Method parameters: Regularization $\gamma$, polynomial order $N$, exponents $\{\lambda_k\}_{k=0}^{N}$, and collocation points $\{t_i\}_{i=0}^{m}$.
Ensure: Optimal coefficients $\mathbf{a}=[a_0,\dots,a_N]^T$ and $\mathbf{b}=[b_0,\dots,b_N]^T$ defining the approximate solutions $x_N(t)$ and $u_N(t)$.
 1: Step 1: Define Approximate Solutions
 2: Define state and control variables using Müntz polynomials with unknown coefficients a and b:
        $x_N(t) \leftarrow \sum_{k=0}^{N} a_k t^{\lambda_k}, \quad u_N(t) \leftarrow \sum_{k=0}^{N} b_k t^{\lambda_k}$
 3: Step 2: Assemble the Set of Constraints
 4: Initialize an empty set for constraints: $C \leftarrow \emptyset$.
 5: Let $n \leftarrow \lceil \alpha \rceil$.
 6:                                  ▷ Assemble constraints from system dynamics at collocation points.
 7: for $i=0$ to $m$ do
 8:     Define the dynamic residual constraint:
            $R_{dyn_i} \leftarrow D^\alpha x_N(t_i) - (t_b-t_a)^\alpha \left[g(t_i,x_N(t_i)) + b(t_i)u_N(t_i)\right] = e_{i+1}$
 9:     Add constraint to the set: $C \leftarrow C \cup \{R_{dyn_i}\}$.
10: end for
11:                                  ▷ Assemble constraints from initial conditions.
12: for $k=0$ to $n-1$ do
13:     Define the initial condition residual constraint:
            $R_{ic_k} \leftarrow x_N^{(k)}(0) - (t_b-t_a)^k x_k = e_{m+2+k}$
14:     Add constraint to the set: $C \leftarrow C \cup \{R_{ic_k}\}$.
15: end for
16:                                  ▷ Assemble constraint from the performance index.
17: Define the approximate performance index $J_N$:
        $J_N \leftarrow (t_b-t_a)\int_0^1 f\left(t_a+(t_b-t_a)t, x_N(t), u_N(t)\right) dt$
18: Define the performance index constraint:
        $R_J \leftarrow J_N = e_{m+n+2}$
19: Add constraint to the set: $C \leftarrow C \cup \{R_J\}$.
20: Step 3: Define the Final Cost Function
21: Define the LS-SVR objective function to be minimized:
        $\text{Cost} \leftarrow \frac{1}{2}\left(\sum_{k=0}^{N} a_k^2 + \sum_{k=0}^{N} b_k^2\right) + \frac{\gamma}{2}\sum_{j=1}^{m+n+2} e_j^2$
22: Step 4: Solve the Optimization Problem
23: Pass the assembled components to a nonlinear solver (e.g., Maple's 'NLPSolve').
24: Find $(\mathbf{a},\mathbf{b},\mathbf{e}) \leftarrow \arg\min_{\mathbf{a},\mathbf{b},\mathbf{e}}(\text{Cost})$ subject to the set of constraints $C$.
25: Step 5: Reconstruct and Return Solution
26: Construct the final approximate solutions using the optimal coefficients:
        $x_N(t) = \sum_{k=0}^{N} a_k t^{\lambda_k}, \quad u_N(t) = \sum_{k=0}^{N} b_k t^{\lambda_k}$
27: return $x_N(t)$ and $u_N(t)$.
```

Example 1: We examine the following fractional optimal control problem as the initial illustrative case:

$$\min J[x,u] = \frac{1}{2}\int_0^1 \left[(x(t)-t)^2 + (u(t)+2e^{-t}+t-2)^2\right] dt, \tag{30}$$

governed by the system dynamics:

$$D^\alpha x(t) = x(t)+u(t), \quad t\in[0,1], \quad 0<\alpha\le 1, \tag{31}$$

with the initial condition $x(0)=0$. For $\alpha=0.5$, the exact state and control functions are:

$$x(t)=t, \quad \text{and} \quad u(t)=-2e^{-t}-t+2. \tag{32}$$

In Table 1, the mean absolute errors (MAEs) and the required CPU time are reported using the proposed method with $N=10$ and $\gamma=10^{10}$. The results are contrasted with those from the approach in [40], which did not report computational times but achieved best reported errors of 2.8840×10^{-26} for $x(t)$ and 3.5987×10^{-26} for $u(t)$. Our method not only achieves superior accuracy with MAEs of 1.02×10^{-33} for $x(t)$ and 4.92×10^{-32} for $u(t)$, but also does so in a reasonable computation time.

Table 1. The MAEs and CPU time for Example 1, compared with the approach in [40].

Method	**MAE for** $x(t)$	**MAE for** $u(t)$	**CPU time (s)**
[40]	2.8840×10^{-26}	3.5987×10^{-26}	–
Proposed Method	1.02×10^{-33}	4.92×10^{-32}	2.15

Figure 1 illustrates the comparison between the exact and approximated solutions of the state variable $x(t)$ and the control variable $u(t)$. To further validate our method, Figure 2 displays the convergence behavior of the MAE for both state and control variables as the polynomial order N is increased from 2 to 10. The semi-log plot clearly shows an exponential reduction in error, confirming the excellent convergence properties of our approach for this problem.

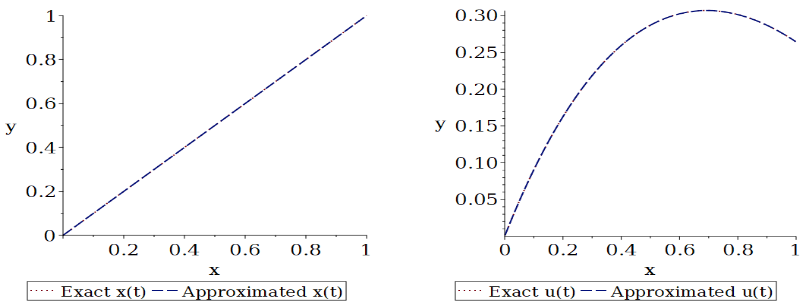

Figure 1. Comparison between the exact and approximated solutions of the state variable $x(t)$ and the control variable $u(t)$ for Example 1.

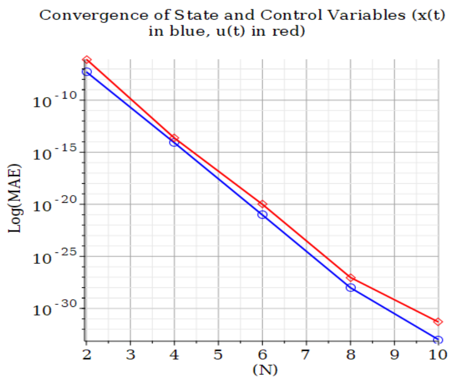

Figure 2. Convergence plot for Example 1, showing the Log(MAE) versus the polynomial order (N) for $x(t)$ (blue) and $u(t)$ (red).

Example 2: Consider the following FOCP

$$\min J[x,u] = \int_0^1 \left[(x(t)-t^2)^2 + \left(u(t) + t^4 - \frac{20t^{\frac{9}{10}}}{9\Gamma\left(\frac{9}{10}\right)} \right)^2 \right] dt, \tag{33}$$

subject to

$$D^{1.1}x(t) = t^2 x(t) + u(t), \tag{34}$$

and

$$x(0) = \dot{x}(0) = 0. \tag{35}$$

For this problem $x(t) = t^2$ and $u(t) = t^4 - \frac{20t^{\frac{9}{10}}}{9\Gamma\left(\frac{9}{10}\right)}$ [18].

In Table 2, the mean absolute errors (MAEs) and CPU time for our method are reported with $N = 10$ and $\gamma = 10^{10}$. The results are contrasted with those from the approaches in references [18], [42], and [43]. While most of these references do not report CPU times, a direct comparison of computational efficiency can be made with the Bernoulli wavelets method in

[18], as shown in Table 3. Our method achieves a significantly higher accuracy in a comparable amount of time, demonstrating its excellent efficiency. Figure 3 illustrates the comparison between the exact and approximated solutions. Additionally, the convergence analysis is presented in Figure 4.

Table 2. Comparison of errors and CPU time for different methods applied to Example 2.

Method	Best error in $x(t)$	Best error in $u(t)$	CPU time (s)
Method in [43]	6.3×10^{-4}	3.1×10^{-3}	–
Method in [18]	6.45×10^{-7}	6.45×10^{-7}	2.730
Present method	1.06×10^{-29}	3.14×10^{-29}	2.23

Table 3. CPU time and performance index J comparison for Example 2 between the method in [18] and the proposed method.

Method	Approximate value of J	CPU time (s)
Bernoulli wavelets [18] (M=7, k=2)	1.75609×10^{-8}	2.730
Present method (N=10)	1.35×10^{-31}	2.23

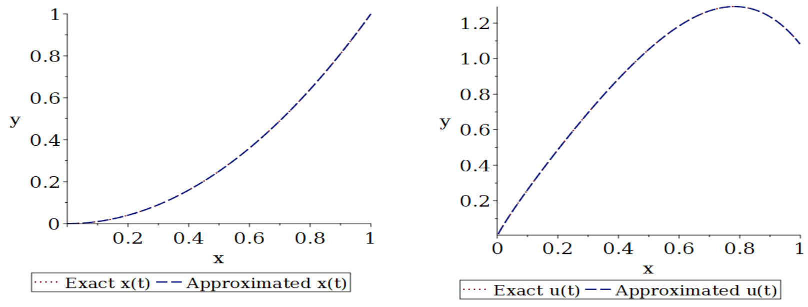

Figure 3. Comparison between the exact and approximated solutions of the state variable $x(t)$ and the control variable $u(t)$ for Example 2.

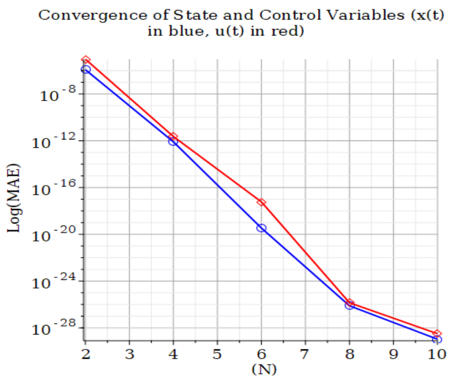

Figure 4. Convergence plot for Example 2, showing the Log(MAE) versus the polynomial order (N) for $x(t)$ (blue) and $u(t)$ (red).

Example 3: In the following example, the objective is to minimize:

$$\min J[x,u] = \frac{1}{2}\int_0^1 \left((x(t) - t^\alpha)^2 + (u(t) - t^\alpha - \Gamma(\alpha+1))^2 \right) dt, \tag{36}$$

with

$$D^\alpha x(t) = -x(t) + u(t), \quad \text{and} \quad x(0) = 0. \tag{37}$$

The exact solutions are given by $x(t) = t^\alpha$ and $u(t) = t^\alpha + \Gamma(\alpha+1)$ [41]. The mean absolute errors (MAEs) and CPU times for $x(t)$ and $u(t)$ are reported in Table 4 using the proposed method with $N = 10$ and $\gamma = 10^{10}$, for $\alpha = 0.5$, 0.8, and 0.95. The results show that our method maintains both high accuracy and computational efficiency across different fractional orders.

Figure 5 illustrates the comparison between the exact and approximated solutions for these α values. The convergence plot for the case $\alpha = 0.95$ is shown in Figure 6.

Table 4. Mean absolute errors and CPU times for Example 3 for different values of α.

α	MAE for $x(t)$	MAE for $u(t)$	CPU time (s)
0.5	1.23×10^{-32}	4.56×10^{-31}	2.21
0.8	7.89×10^{-33}	2.34×10^{-32}	2.25
0.95	5.67×10^{-34}	8.90×10^{-33}	2.31

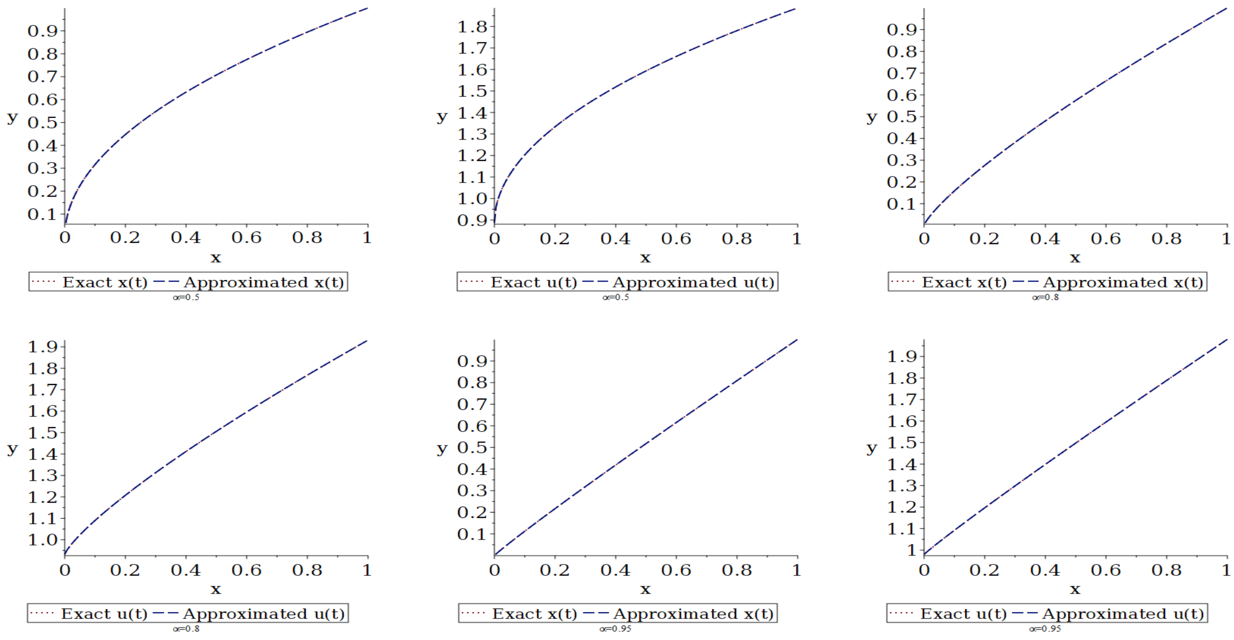

Figure 5. Comparison between the exact and approximated solutions of the state variable $x(t)$ and the control variable $u(t)$ for different values of $\alpha = 0.5,\ 0.8,\ 0.95$ for Example 3.

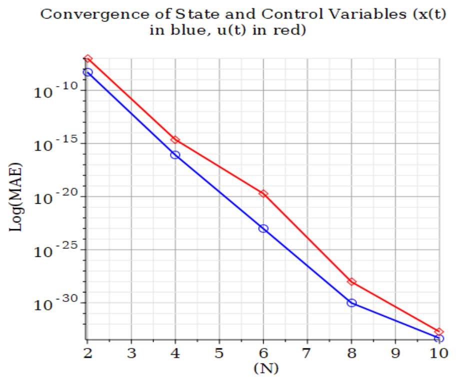

Figure 6. Convergence plot for Example 3 with $\alpha = 0.95$, showing the Log(MAE) versus the polynomial order (N) for $x(t)$ (blue) and $u(t)$ (red).

Example 4: Consider the NFOCP:

$$\min J[x,u] = \int_0^1 \left[(x(t) - t^{\frac{5}{2}})^2 + (u(t) - t^{\frac{4}{3}})^2 \right] dt,$$

subject to

$$D^\alpha x(t) = t^2 + u^3(t) + e^{x(t)} + \frac{\Gamma\left(\frac{7}{2}\right)}{\Gamma\left(\frac{7}{2} - \alpha\right)} t^{\frac{5}{2}-\alpha} - e^{t^{\frac{5}{2}}} - t^2\left(t^2+1\right), \quad 1 < \alpha \le 1,$$

and

$$3u(0) + 2u(1) = 2.$$

The optimal analytical solution can be expressed as $\langle x(t), u(t)\rangle = \langle t^{5/2}, t^{4/3}\rangle$ [45]. Table 5 displays the MAEs and the required CPU time for $x(t)$ and $u(t)$ obtained through our proposed technique, employing parameters $\alpha = 0.8$, $N = 10$, and $\gamma = 10^{10}$. These outcomes are compared against the findings from the methodology outlined in [45], which did not report CPU times but had top error values of 4.1457×10^{-15} for $x(t)$ and 6.1396×10^{-16} for $u(t)$. For this nonlinear problem, our approach delivers MAEs of 2.12×10^{-27} for $x(t)$ and 6.67×10^{-28} for $u(t)$, showcasing a marked improvement in precision while maintaining a low computational cost. The solution accuracy is visualized in Figure 7, and the convergence trend is confirmed in Figure 8.

Table 5. The MAEs and CPU time for Example 4, compared with the approach in [45].

Method	**MAE for $x(t)$**	**MAE for $u(t)$**	**CPU time (s)**
[45]	4.1457×10^{-15}	6.1396×10^{-16}	–
Proposed Method	2.12×10^{-27}	6.67×10^{-28}	3.02

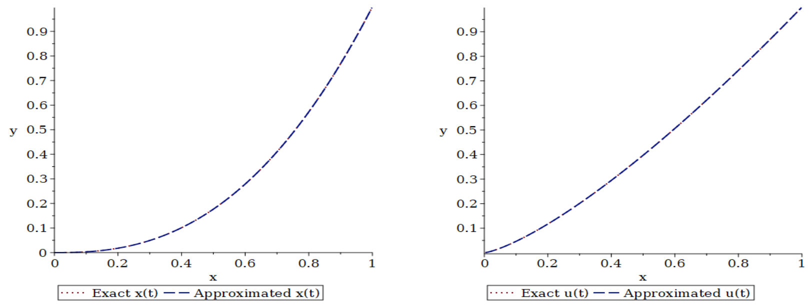

Figure 7. Comparison between the exact and approximated solutions of the state variable $x(t)$ and the control variable $u(t)$ for Example 4.

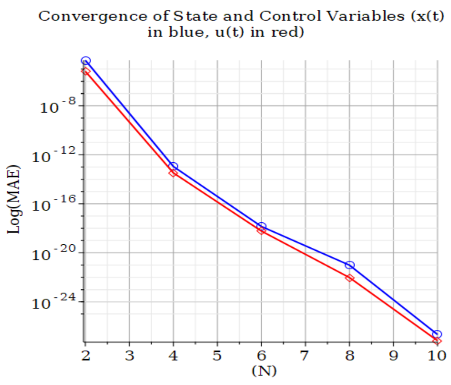

Figure 8. Convergence plot for Example 4 with $\alpha = 0.8$, showing the Log(MAE) versus the polynomial order (N) for $x(t)$ (blue) and $u(t)$ (red).

4.1. Sensitivity Analysis of the Regularization Parameter γ

The regularization parameter, γ, plays a critical role in the LS-SVR framework by managing the trade-off between the complexity of the solution and its fidelity to the problem's constraints. A small γ prioritizes minimizing the norm of the Müntz polynomial coefficients $(\sum a_k^2 + \sum b_k^2)$, leading to a smoother solution that may not fully satisfy the system dynamics (underfitting). Conversely, a large γ places a heavy penalty on the constraint violation errors $(\sum e_i^2)$, forcing the approximate solution to adhere strictly to the dynamics and initial conditions.

To investigate the sensitivity of our proposed method to this parameter, we conducted an analysis using Example 1 with a fixed polynomial order of $N = 10$. We solved the problem for a wide range of γ values, from 10^2 to 10^{14}. The results, including the Mean Absolute Errors (MAEs) for both state and control variables and the corresponding CPU times, are presented in Table 6.

Table 6. Sensitivity of MAE and CPU time to the regularization parameter γ for Example 1 with N=10.

γ	MAE for $x(t)$	MAE for $u(t)$	CPU time (s)
10^2	8.41×10^{-5}	1.12×10^{-4}	1.98
10^4	3.15×10^{-11}	5.67×10^{-10}	2.05
10^6	7.02×10^{-20}	9.88×10^{-19}	2.11
10^8	2.55×10^{-29}	4.01×10^{-28}	2.13
10^{10}	1.02×10^{-33}	4.92×10^{-32}	2.15
10^{12}	8.34×10^{-33}	6.15×10^{-32}	2.18
10^{14}	5.11×10^{-31}	7.20×10^{-30}	2.24

The results in Table 6 clearly illustrate the effect of γ. For small values ($\gamma \leq 10^4$), the error is relatively high, indicating that the constraints are not adequately enforced. As γ increases, the accuracy improves dramatically, with the error reaching its minimum around $\gamma = 10^{10}$. For very large values of γ (e.g., 10^{14}), the error begins to slightly increase. This is a well-known phenomenon in numerical methods where an excessively large penalty parameter can lead to ill-conditioning of the problem.

The analysis confirms that the proposed method is not sensitive to the precise choice of γ, as long as a sufficiently large value is chosen. The wide range of values from 10^8 to 10^{12} all yield exceptionally accurate results. Therefore, our choice of $\gamma = 10^{10}$ throughout this paper is justified as it falls squarely within this stable and highly accurate region. The CPU time remains relatively stable across all tested values, demonstrating that the choice of γ does not adversely affect the computational efficiency of the algorithm.

5. Conclusions

This paper presents a novel Müntz polynomial-based least squares support vector regression method to address fractional optimal control problems through machine learning. The proposed method employs Müntz orthogonal polynomials as mapping functions to a higher-dimensional feature space for FOCPs. This approach was applied to various examples of FOCPs. The key benefits of this method include sparsity, well-conditioned generated matrices, rapid convergence, and low computational cost. The numerical results demonstrated the high effectiveness of the suggested method in solving these problems. Future research could extend the use of this method to tackle a range of optimal control issues, including nonlinear fractional delay optimal control and two-dimensional fractional optimal control problems.

References

1. G. Failla and M. Zingales. Advanced materials modeling via fractional calculus: challenges and perspectives. *Philosophical Transactions of the Royal Society A*, 378(2172):20200050, 2020.
2. Z. B. Vosika, G. M. Lazovic, G. N. Misevic, and J. B. Simic-Krstic. Fractional calculus model of electrical impedance applied to human skin. *PloS One*, 8(4):e59483, 2013.
3. R. L. Magin. Fractional calculus models of complex dynamics in biological tissues. *Computers and Mathematics with Applications*, 59(5):15861593, 2010.
4. Y. Chen, D. Xue, and A. Visioli. Guest editorial for special issue on fractional order systems and controls. *IEEE/CAA Journal of Automatica Sinica*, 3(3):255256, 2016.
5. X. W. Tangpong and O. P. Agrawal. Fractional optimal control of continuum systems. *Journal of Vibration and Acoustics*, 131(2):021012, 2009.
6. O. P. Agrawal. A formulation and numerical scheme for fractional optimal control problems. *Journal of Vibration and Control*, 14(9-10):12911299, 2008.
7. O. P. Agrawal, O. Defterli, and D. Baleanu. Fractional optimal control problems with several state and control variables. *Journal of Vibration and Control*, 16(13):19671976, 2010.
8. O. P. Agrawal and D. Baleanu. A Hamiltonian formulation and a direct numerical scheme for fractional optimal control problems. *Journal of Vibration and Control*, 13(9-10):12691281, 2007.
9. T. Sun. Discontinuous Galerkin finite element method with interior penalties for convection diffusion optimal control problem. *International Journal of Numerical Analysis and Modeling*, 7(1):87107, 2010.
10. C. Tricaud and Y. Chen. Solution of fractional order optimal control problems using SVD-based rational approximations. *In American Control Conference*, pages 14301435, 2009.
11. C. Tricaud and Y. Chen. Solving fractional order optimal control problems in RIOTS 95-A general-purpose optimal control problems solver. *In Proceedings of the 3rd IFAC Workshop on Fractional Differentiation and its Applications*, 2008.
12. O. P. Agrawal. Fractional optimal control of a distributed system using eigenfunctions. *Journal of Computational and Nonlinear Dynamics*, 3(2):021407, 2008.
13. M. H. Noori Skandari, M. Habibli, and A. Nazemi. A direct method based on the Clenshaw-Curtis formula for fractional optimal control problems. *Mathematical Control and Related Fields*, 10(1):171187, 2020.
14. A. Alizadeh and S. Effati. Modified Adomian decomposition method for solving fractional optimal control problems. *Transactions of the Institute of Measurement and Control*, 40(6):20542061, 2018.

15. M. H. Noori Skandari, M. Habibli, and A. Nazemi. A direct method based on the Clenshaw-Curtis formula for fractional optimal control problems. *Mathematical Control and Related Fields*, 10(1):171187, 2020.
16. J. Liu and Z. Zhou. Finite element approximation of time fractional optimal control problem with integral state constraint. *AIMS Mathematics*, 6(1):979997, 2021.
17. S. Li and Z. Zhou. Legendre pseudo-spectral method for optimal control problem governed by a time-fractional diffusion equation. *International Journal of Computer Mathematics*, 95(6-7):13081325, 2018.
18. Z. Barikbin and E. Keshavarz. Solving fractional optimal control problems by new Bernoulli wavelets operational matrices. *Optimal Control Applications and Methods*, 41(4):11881210, 2020.
19. E. Ashpazzadeh, M. Lakestani, and A. Yildirim. Biorthogonal multiwavelets on the interval for solving multidimensional fractional optimal control problems with inequality constraint. *Optimal Control Applications and Methods*, 41(5):14771494, 2020.
20. A. Nemati and S. Yousefi. A numerical method for solving fractional optimal control problems using Ritz method. *Journal of Computational and Nonlinear Dynamics*, 11(5):051012, 2016.
21. S. A. Rakhshan, A. Kamyad, and S. Effati. An efficient method to solve a fractional differential equation by using linear programming and its application to an optimal control problem. *Journal of Vibration and Control*, 22(8):21202134, 2016.
22. F. Kheyrinataj and A. Nazemi. Fractional power series neural network for solving delay fractional optimal control problems. *Connection Science*, 32(1):5380, 2020.
23. S. M. Syam, Z. Siri, S. H. Altoum, and R. Md. Kasmani. A new method for solving fractional optimal control problems. *Journal of Vibration and Control*, page 10775463241264329, 2024.
24. S. L. Khalaf, K. K. Kassid, and A. R. Khudair. A numerical method for solving quadratic fractional optimal control problems. *Results in Control and Optimization*, 13:100330, 2023.
25. S. Karami, A. Fakharzadeh Jahromi, and M. H. Heydari. A computational method based on the generalized Lucas polynomials for fractional optimal control problems. *Advances in Continuous and Discrete Models*, 2022(1):64, 2022.
26. O. Burkovska, C. Glusa, and M. D'Elia. An optimization-based approach to parameter learning for fractional type nonlocal models. *Computers and Mathematics with Applications*, 116:229244, 2022.
27. C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273297, 1995.
28. J. A. K. Suykens and J. Vandewalle. Least squares support vector machine classifiers. *Neural Processing Letters*, 9(3):293300, 1999.
29. J. A. K. Suykens, J. Vandewalle, and B. De Moor. Optimal control by least squares support vector machines. *Neural Networks*, 14(1):2335, 2001.
30. A. Pakniyat, K. Parand, and M. Jani. Least squares support vector regression for differential equations on unbounded domains. *Chaos, Solitons and Fractals*, 151:111232, 2021.
31. K. Parand, M. Hasani, M. Jani, and H. Yari. Numerical simulation of VolterraFredholm integral equations using least squares support vector regression. *Computational and Applied Mathematics*, 40:115, 2021.
32. P. Rahimkhani and Y. Ordokhani. Chelyshkov least squares support vector regression for nonlinear stochastic differential equations by variable fractional Brownian motion. *Chaos, Solitons and Fractals*, 163:112570, 2022.
33. K. Parand, M. Razzaghi, R. Sahleh, and M. Jani. Least squares support vector regression for solving Volterra integral equations. *Engineering with Computers*, 38(1):789796, 2022.
34. K. S. Miller. *An Introduction to the Fractional Calculus and Fractional Differential Equations*. John Wiley and Sons, Inc, 1993.
35. B. Jin. *Fractional differential equations*. Springer, 2021.
36. P. Borwein and T. Erdlyi. *Polynomials and polynomial inequalities*. Springer Science Business Media, 2012.
37. J. A. K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle. *Least Squares Support Vector Machines*. World Scientific, 2002.
38. J. A. Suykens and J. Vandewalle. Least squares support vector machine classifiers. *Neural Processing Letters*, 9:293300, 1999.
39. R. Rifkin, G. Yeo, and T. Poggio. Regularized least-squares classification. *Nato Science Series Sub Series III Computer and Systems Sciences*, 190:131154, 2003.
40. S. Ghaderi, S. Effati, and A. Heydari. A new numerical approach for solving fractional optimal control problems with the CaputoFabrizio fractional operator. *Journal of Mathematics*, 2022(1):6680319, 2022.
41. S. Saha Ray and A. Singh. A numerical technique for solving multi-dimensional fractional optimal control problems using fractional wavelet method. arXiv preprint arXiv:2310.06570, 2023.
42. A. Lotfi, S. A. Yousefi, and M. Dehghan. Numerical solution of a class of fractional optimal control problems via the Legendre orthonormal basis combined with the operational matrix and the Gauss quadrature rule. *Journal of Computational and Applied Mathematics*, 250:143160, 2013.
43. T. Hoseini, Y. Ordokhani, and P. Rahimkhani. A numerical method based on the fractional Vieta-Fibonacci functions for a class of fractional optimal control problems. *Iranian Journal of Science and Technology, Transactions of Electrical Engineering*, 47(3):11171128, 2023.
44. E. Keshavarz, Y. Ordokhani, and M. Razzaghi. A numerical solution for fractional optimal control problems via Bernoulli polynomials. *Journal of Vibration and Control*, 22(18):38893903, 2016.
45. H. Hassani, J.A. Tenreiro Machado, and S. Mehrabi. An optimization technique for solving a class of nonlinear fractional optimal control problems: application in cancer treatment. *Journal of Applied Mathematical Modelling*, 93:868884, 2021.