

# Modified least squares method and a review of its applications in machine learning and fractional differential/integral equations

Abhishek Kumar Singh<sup>1,2</sup>, Mani Mehra<sup>\*1</sup>, and Anatoly A. Alikhanov<sup>3</sup>

<sup>1</sup>Department of Mathematics, Indian Institute of Technology Delhi, India

<sup>2</sup>Institute of Mathematics and Computer Science, Universität Greifswald,

Walther-Rathenau-Straße 47, 17489 Greifswald, Germany

<sup>3</sup>North-Caucasus Center for Mathematical Research, North-Caucasus Federal University,

Russia

[assinghabhi@gmail.com](mailto:assinghabhi@gmail.com), [mehra@maths.iitd.ac.in](mailto:mehra@maths.iitd.ac.in), [a.alikhanov@gmail.com](mailto:a.alikhanov@gmail.com)

May 2, 2024

## Abstract

The least squares method provides the best-fit curve by minimizing the total squares error. In this work, we provide the modified least squares method based on the fractional orthogonal polynomials that belong to the space  $M_n^\lambda := \text{span}\{1, x^\lambda, x^{2\lambda}, \dots, x^{n\lambda}\}$ ,  $\lambda \in (0, 2]$ . Numerical experiments demonstrate how to solve different problems using the modified least squares method. Moreover, the results show the advantage of the modified least squares method compared to the classical least squares method. Furthermore, we discuss the various applications of the modified least squares method in the fields like fractional differential/integral equations and machine learning.

**Keywords.** Modified least squares method; Müntz-Legendre polynomials; Machine learning; Fractional differential/integral equations.

## 1 Introduction

The least squares method is one of the oldest methods of modern statistics used to obtain the physical parameters from the experimental data. The first use of the least squares method is generally attributed to Gauss in 1795, although Legendre concurrently and independently used it [12]. Gauss invented the least squares method to estimate planets' orbital motion from telescopic measurements. In modern statistics, Galton [2] was the first to use the least squares method in his work on the heritability of size, which laid down the foundations of correlation and regression analysis. Nowadays, the least squares method is widely used to find the best-fit curve while finding the parameter involved in the curve. There are many versions of the least squares method available in the literature. The simpler version is called the ordinary least squares method, and the more advanced one is the weighted least squares method, which performs better than the ordinary least squares method. The recent version of the least squares method is the moving least squares method [6], and the partial least squares method [27].

One of the areas where the least squares method is frequently used is machine learning, where we analyze data for regression analysis and classification [24]. Machine learning is a field of artificial intelligence that allows computer systems to learn using available data. Recently machine learning algorithms (regression analysis and classification) have become very popular for analyzing data and making predictions. Another application of the least squares method is solving fractional differential/integral equations. Fractional differential/integral equations give an excellent way to deal with complex phenomena in nature, such as biological systems, control theory, finance, signal and image processing, sub-diffusion and

---

\*Corresponding author

super-diffusion process, viscoelastic fluid, electrochemical processes, and so on [3, 13, 28, 26, 18]. The fractional differential equations are equivalent to the Hammerstein form of Volterra's second kind integral equations for the specific choice of kernel (for more details see [9]). Due to the importance of fractional differential/integral equations, people are interested in solving them numerically because of the non-availability of exact solutions. Many numerical methods are available in the literature to solve fractional differential/integral equations, such as finite difference [17, 19], compact finite difference [22, 4], finite element [8, 15], and wavelet methods [16, 25]. Recently, the least squares method based on classical polynomials is also used to solve fractional differential/integral equations [23].

The aforementioned discussion concludes with the importance of the least squares method in various disciplines. In this work, we proposed the modified least squares method based on fractional polynomials. The main contribution of this paper consists of the following aspects:

- A modified least squares method based on fractional orthogonal polynomials belongs to the space  $M_n^\lambda := \text{span}\{1, x^\lambda, x^{2\lambda}, \dots, x^{n\lambda}\}$ ,  $\lambda \in (0, 2]$  has been proposed.
- Applications of the modified least squares method have been discussed in detail, especially in the field of fractional differential/integral equations and machine learning.

The paper has been arranged in the following pattern: Section 2 describes some basic concepts of approximation theory and orthogonal polynomials like Müntz-Legendre polynomials. Section 3 deals with the review of a least squares method. In Section 4, we develop a modified least squares method based on fractional orthogonal polynomials. Section 5 provides the numerical results to validate the modified least squares method. Section 6 summarizes the applications part of the method. Finally, Section 7 gives the brief conclusion.

## 2 Preliminaries

In this Section, some necessary results from approximation theory and orthogonal polynomials are summarized. Let define the space

$$M^\lambda := \text{span}\{x^{\lambda_0}, x^{\lambda_1}, \dots, x^{\lambda_n}, \dots\}, \quad (2.1)$$

and for its subspaces we define

$$M_n^\lambda := \text{span}\{x^{\lambda_0}, x^{\lambda_1}, \dots, x^{\lambda_n}\}, \quad (2.2)$$

with  $0 \leq \lambda_0 < \lambda_1 < \dots \rightarrow \infty$ . The space  $M^\lambda$  is known as Müntz space. Now, we recall one of the fundamental theorems of approximation theory called Müntz-Szász theorem, which is related to the denseness of polynomial belonging to space  $M^\lambda$ .

**Theorem 2.1.** (*Müntz-Szász Theorem*) *The Müntz polynomials of the forms  $\sum_{i=0}^n a_i x^{\lambda_i} \in M_n^\lambda$  with real coefficients are dense in  $L^2[0, 1]$  if and only if*

$$\sum_{i=1}^{\infty} \frac{1}{\lambda_i} = +\infty. \quad (2.3)$$

Moreover, if  $\lambda_0 = 0$ , then Müntz polynomials are dense in  $C[0, 1]$ .

**Proof.** See [7].

In this work, we assume that  $\lambda_i = i\lambda$ ,  $i = 0, 1, \dots, n, \dots$ , where  $\lambda$  is a real constant and  $\lambda \in (0, 2]$ . Now, we review one of the general forms of classical orthogonal polynomials called Jacobi polynomials. Moreover, we also review the Müntz-Legendre polynomials, and because of some advantage in terms of computational accuracy, we define the Müntz-Legendre polynomials with the help of Jacobi polynomials.

### 2.1 Jacobi polynomials

The Jacobi polynomials with parameters  $a, b > -1$ , denoted by  $P_n^{a,b}$ , is defined in the interval  $-1 \leq x \leq 1$  as

$$P_n^{a,b}(x) = \sum_{i=0}^n \frac{(-1)^{n-i}(1+b)_n(1+a+b)_{i+n}}{i!(n-i)!(1+b)_i(1+a+b)_n} \left(\frac{1+x}{2}\right)^i, \quad (2.4)$$

where  $(1+b)_i = (1+b)(2+b)\dots(i+b)$  and  $(1+b)_0 = 1$ .

In practice, one can compute the Jacobi polynomials using the following recurrence relation

$$\begin{aligned} P_0^{a,b}(x) &= 1, \quad P_1^{a,b}(x) = \frac{1}{2}[(a-b)+(a+b+2)x], \\ c_{1,n}^{a,b} P_n^{a,b}(x) &= c_{2,n}^{a,b}(x) P_n^{a,b}(x) - c_{3,n}^{a,b} P_{n-1}^{a,b}(x), \end{aligned} \quad (2.5)$$

where

$$\begin{aligned} c_{1,n}^{a,b} &= 2(n+1)(n+a+b+1)(2n+a+b), \\ c_{2,n}^{a,b}(x) &= (2n+a+b+1)[(2n+a+b)(2n+a+b+2)x + a^2 - b^2], \\ c_{3,n}^{a,b} &= 2(n+a)(n+b)(2n+a+b+2). \end{aligned} \quad (2.6)$$

## 2.2 Müntz Legendre polynomials

One can defined Müntz Legendre polynomials on the interval  $[0, 1]$  as follows

$$L_n(x) = \sum_{i=0}^n \eta_{n,i} x^{\lambda_i}, \quad \eta_{n,i} = \frac{\prod_{k=0}^{n-1} (\lambda_i + \bar{\lambda}_k + 1)}{\prod_{k=0, k \neq i}^n (\lambda_i - \lambda_k)}. \quad (2.7)$$

These polynomials satisfy the following orthogonality condition with respect to weight function  $W(x) = 1$

$$\int_0^1 L_n(x) L_m(x) dx = \frac{\delta_{n,m}}{\lambda_n + \bar{\lambda}_n + 1}. \quad (2.8)$$

Since we assumed that  $\lambda_i = i\lambda$ , where  $\lambda$  is a real constant, then the Müntz-Legendre polynomials on the interval  $[0, 1]$  are represented by the formula

$$L_n(x; \lambda) := \sum_{i=0}^n \eta_{n,i} x^{i\lambda}, \quad \eta_{n,i} = \frac{(-1)^{n-i}}{\lambda^n i! (n-i)!} \prod_{k=0}^{n-1} ((i+k)\lambda + 1). \quad (2.9)$$

From Equation (2.7), one can observe that evaluating Müntz-Legendre polynomials, mainly when  $n$  is vast, and  $x$  is closed to 1, is problematic in finite arithmetic. Milovanovic has addressed these problems [20]. We will use a method for evaluating Müntz-Legendre polynomials, which is based on three-term recurrence relation induced from the accompanying theorem with the help of Jacobi polynomials.

**Theorem 2.2.** *Let  $\lambda > 0$  be a real number and  $x \in [0, 1]$ . Then the representation*

$$L_n(x; \lambda) = P_n^{(0,1/\lambda-1)}(2x^\lambda - 1)$$

holds true.

**Proof.** See [10]. □

So in view of Theorem 2.2 and Equations (2.5) and (2.7), the Müntz-Legendre polynomials  $L_n(x; \lambda)$  can be evaluated by means of three-terms recurrence relation

$$\begin{aligned} L_0(x; \lambda) &= 1, \quad L_1(x; \lambda) = \left(\frac{1}{\lambda} + 1\right) x^\lambda - \frac{1}{\lambda}, \\ d_{1,n} L_{n+1} &= d_{2,n}(t) L_n(x; \lambda) - d_{3,n} L_{n-1}(x; \lambda), \end{aligned} \quad (2.10)$$

where

$$d_{1,n} = c_{1,n}^{0,1/\lambda-1}, \quad d_{2,n}(x) = c_{2,n}^{0,1/\lambda-1} (2x^\lambda - 1), \quad d_{3,n} = c_{3,n}^{0,1/\lambda-1}.$$

### 3 Least squares method

For approximating the continuous function  $y(x)$  defined on the interval  $[a, b]$  with an algebraic polynomial  $P_n(x; 1) = \sum_{i=0}^n a_i x^i \in M_n^1$  using least squares method, we choose the constants  $a_0, a_1, \dots, a_n$ , which minimize the least squares error  $E^C(a_0, a_1, \dots, a_n)$ , where

$$E^C(a_0, a_1, \dots, a_n) = \int_a^b (y(x) - P_n(x))^2 dx.$$

A necessary condition for real coefficient  $a_0, a_1, \dots, a_n$  that minimize the error  $E^C(a_0, a_1, \dots, a_n)$  is that

$$\frac{\partial E^C}{\partial a_i} = 0, \text{ for each } i = 0, 1, \dots, n.$$

Thus, one can get the following linear system of equations

$$A_{LSM}^C X = b^C, \quad (3.1)$$

where  $X = [a_0, a_1, \dots, a_n]^T$ ,  $A_{LSM}^C$  is the  $(n+1) \times (n+1)$  coefficient matrix and  $b^C$  is the  $(n+1) \times 1$  column vector. The  $(i, j)^{th}$  element of the matrix  $A_{LSM}^C$  and  $i^{th}$  element of the column vector  $b^C$  are given by

$$(A_{LSM}^C)_{i,j} = \int_a^b x^{i+j-2} dx, \text{ and, } (b^C)_i = \int_a^b y(x)x^{i-1} dx,$$

respectively. Similarly, in discrete case, for approximating a data set  $(x_k, y_k)$ ,  $k = 1, 2, \dots, m$ , with an algebraic polynomial  $P_n(x; 1) \in M_n^1$  using least squares method, we choose the constants  $a_0, a_1, \dots, a_n$  which minimize the least squares error  $E^D(a_0, a_1, \dots, a_n)$ , where

$$\begin{aligned} E^D(a_0, a_1, \dots, a_n) &= \sum_{k=1}^m (y_k - P_n(x_k))^2, \\ &= \sum_{k=1}^m y_k^2 - 2 \sum_{i=0}^n a_i \left( \sum_{k=1}^m y_k x_k^i \right) + 2 \sum_{i=0}^n \sum_{j=0}^n a_i a_j \left( \sum_{k=1}^m x_k^{i+j} \right). \end{aligned}$$

For  $E^D$  to be minimized it is necessary that  $\frac{\partial E^D}{\partial a_i} = 0$ , for each  $i = 0, 1, \dots, n$ . Thus, one can get following linear system of equations

$$A_{LSM}^D X = b^D, \quad (3.2)$$

where  $A_{LSM}^D$  is the  $(n+1) \times (n+1)$  coefficient matrix and  $b^D$  is the  $(n+1) \times 1$  column vector. The  $(i, j)^{th}$  element of the matrix  $A_{LSM}^D$  and  $i^{th}$  element of the column vector  $b^D$  are given by

$$(A_{LSM}^D)_{i,j} = \sum_{k=1}^m x_k^{i+j-2}, \text{ and, } (b^D)_i = \sum_{k=1}^m y_k x_k^{i-1},$$

respectively.

### 4 Modified least squares method

In this Section, we proposed a modified least squares method based on a fractional polynomials that belongs to the space  $M_n^\lambda$ . Apply a similar technique which is discussed in Section 3 for any algebraic polynomial  $P_n(x; \lambda) = \sum_{i=0}^n a_i x^{i\lambda} \in M_n^\lambda$  to approximating the continuous function  $y(x)$  defined on the interval  $[a, b]$ , we choose the constants  $a_0, a_1, \dots, a_n$  which minimize the least squares error  $E^C(a_0, a_1, \dots, a_n; \lambda)$ , where

$$E^C(a_0, a_1, \dots, a_n; \lambda) = \int_a^b (y(x) - P_n(x; \lambda))^2 dx. \quad (4.1)$$

A necessary condition for real coefficient  $a_0, a_1, \dots, a_n$ , that minimize the error  $E^C(a_0, a_1, \dots, a_n; \lambda)$  is that

$$\frac{\partial E^C}{\partial a_i} = 0, \text{ for each } i = 0, 1, \dots, n.$$

Thus, one can get the following linear system of equations

$$A_{MLSM}^C X = d^C, \quad (4.2)$$

where  $A_{MLSM}^C$  is the  $(n+1) \times (n+1)$  coefficient matrix and  $d^C$  is the  $(n+1) \times 1$  column vector. The  $(i,j)^{th}$  element of the matrix  $A_{MLSM}^C$  and  $i^{th}$  element of the column vector  $d^C$  are given by

$$(A_{MLSM}^C)_{i,j} = \int_a^b x^{\lambda_{i-1} + \lambda_{j-1}} dx, \quad \text{and}, \quad (d^C)_i = \int_a^b y(x) x^{\lambda_{i-1}} dx,$$

respectively. Similarly, in discrete case, for approximating the data set  $(x_k, y_k)$ ,  $k = 1, 2, \dots, m$ , with an algebraic polynomial  $P_n(x; \lambda)$  using least squares method, we choose the constants  $a_0, a_1, \dots, a_n$ , which minimize the least square error  $E^D(a_0, a_1, \dots, a_n; \lambda)$ , where

$$\begin{aligned} E^D(a_0, a_1, \dots, a_n; \lambda) &= \sum_{k=1}^m (y_k - P_n(x_k; \lambda))^2, \\ &= \sum_{k=1}^m y_k^2 - 2 \sum_{k=1}^m P_n(x_k; \lambda) y_k + \sum_{k=1}^m (P_n(x_k; \lambda))^2, \\ &= \sum_{k=1}^m y_k^2 - 2 \sum_{i=0}^n a_i \left( \sum_{k=1}^m y_k x_k^{\lambda_i} \right) + 2 \sum_{i=0}^n \sum_{j=0}^n a_i a_j \left( \sum_{k=1}^m x_k^{\lambda_i + \lambda_j} \right). \end{aligned} \quad (4.3)$$

For  $E^D$  to be minimized it is necessary that  $\frac{\partial E^D}{\partial a_i} = 0$ , for each  $i = 0, 1, \dots, n$ . Thus, one can get following linear system of equations

$$A_{MLSM}^D X = d^D, \quad (4.4)$$

where  $A_{MLSM}^D$  is the  $(n+1) \times (n+1)$  coefficient matrix and  $d^D$  is the  $(n+1) \times 1$  column vector. The  $(i,j)^{th}$  element of the matrix  $A_{MLSM}^D$  and  $i^{th}$  element of the column vector  $d^D$  are given by

$$(A_{MLSM}^D)_{i,j} = \sum_{k=1}^m x_k^{\lambda_{i-1} + \lambda_{j-1}}, \quad \text{and}, \quad (d^D)_i = \sum_{k=1}^m y_k x_k^{\lambda_{i-1}},$$

respectively. We know that the large value of  $n$ , the matrix  $A_{MLSM}^C$  and  $A_{MLSM}^D$  become ill-conditioned, which causes significant errors in estimating the parameters  $a_i$ ,  $i = 0, 1, \dots, n$ . This difficulty can be avoided if the functions belonging to the space  $M_n^\lambda$ , denoted by  $L_i(x; \lambda, W)$ ,  $i = 0, 1, \dots, n$ , are so chosen that they are orthogonal with respect to the weight function  $W(x)$  over the interval  $[a, b]$ . In this case the error function in the continuous and discrete case becomes

$$E^C(a_0, a_1, \dots, a_n; \lambda) = \int_a^b W(x) (y(x) - \sum_{i=0}^n a_i L_i(x; \lambda, W))^2 dx, \quad (4.5)$$

and

$$E^D(a_0, a_1, \dots, a_n; \lambda) = \sum_{k=1}^m W(x_k) (y_k - \sum_{i=0}^n a_i L_i(x_k; \lambda, W))^2, \quad (4.6)$$

respectively. The necessary condition for real coefficients  $a_0, a_1, \dots, a_n$  which minimize the error gives the normal equations. The normal equations in the continuous and discrete cases are

$$\int_a^b W(x) (y(x) - \sum_{i=0}^n a_i L_i(x; \lambda, W)) L_j(x; \lambda, W) dx = 0, \quad j = 0, 1, \dots, n, \quad (4.7)$$

and

$$\sum_{k=1}^m W(x_k) (y_k - \sum_{i=0}^n a_i L_i(x_k; \lambda, W)) L_j(x_k; \lambda, W) = 0, \quad j = 0, 1, \dots, n, \quad (4.8)$$

respectively. One can observe from the Equations (4.7) and (4.8) the parameter value can be determined directly. Thus the use of orthogonal functions not only avoids the problem of ill-conditioning but also

determines the constants  $a_i$ ,  $i = 0, 1, \dots, n$ , directly. As discussed in Section 2, the Müntz-Legendre polynomials  $L_i(x; \lambda) \in M_n^\lambda$ ,  $i = 0, 1, \dots, n$ , are orthogonal for the weight function  $W(x) = 1$  on the interval  $[0, 1]$ . If we consider the following error function in the continuous and discrete case

$$E^C(a_0, a_1, \dots, a_n; \lambda) = \int_0^1 W(x)(y(x) - \sum_{i=0}^n a_i L_i(x; \lambda))^2 dx, \quad (4.9)$$

and

$$E^D(a_0, a_1, \dots, a_n; \lambda) = \sum_{k=1}^m W(x_k)(y_k - \sum_{i=0}^n a_i L_i(x_k; \lambda))^2, \quad (4.10)$$

respectively. One can easily observe that if we find the necessary condition for the parameters  $a_0, a_1, \dots, a_n$ , we get the systems of equations. For finding the value of  $a_i$ ,  $i = 0, 1, \dots, n$ , we have to solve the equation system because Müntz-Legendre polynomials are not orthogonal with respect to the weight function  $W(x) \neq 1$ . Therefore, one may face the problem of ill-conditioning for a large value of  $n$ . This problem can be avoided if we generate the orthogonal polynomials with respect to the weight function  $W(x)$ .

**Remark 1.** The modified least squares method is based on the fractional polynomials, which depend on the fractional parameter  $\lambda$ . If we put  $\lambda = 1$  in the Equations (4.2) and (4.4) then we get (3.1) and (3.2) respectively. Therefore, in the modified least squares method, we have one additional degree of freedom compared to the classical least squares method in the form of fractional parameter  $\lambda$ . So, we tune the parameter  $\lambda$  to get the desired results.

**Remark 2.** Since Müntz-Legendre polynomials are orthogonal with respect to weight function  $W(x) = 1$ . We know that the fractional derivative/integral of the polynomial belonging to space  $M_n^\lambda$  is again in space  $M_n^\lambda$  when  $\lambda$  and fractional order are the same. Hence, while solving the fractional differential equations using the least squares method and avoiding the difficulty due to ill-conditioning, we need polynomials belonging to the space  $M_n^\lambda$ , which are orthogonal to weight function  $(b-x)^{-\lambda}$  or  $(x-a)^{-\lambda}$ . Moreover, usually, in applications, only a part of the given data needs more attention; for example, in some cases, the data may have more accuracy in some regions than in others. In such cases, the weight function indicates where data should be given more importance, and it should be chosen accordingly.

Now, we are going to discuss the Theorem, which helps us to generate the fractional orthogonal polynomials belonging to the space  $M_n^\lambda$  with respect to the weight function  $W(x)$ .

**Theorem 4.1.** *The set of fractional polynomial functions  $\{L_0(x; \lambda, W), L_1(x; \lambda, W), \dots, L_n(x; \lambda, W)\}$  defined in the following way is orthogonal on  $[a, b]$  with respect to the weight function  $W(x)$ .*

$$L_0(x; \lambda, W) = 1, \quad L_1(x; \lambda, W) = x^\lambda - B_1,$$

where

$$B_1 = \frac{\int_a^b W(x)x^\lambda(L_0(x; \lambda, W))^2 dx}{\int_a^b W(x)(L_0(x; \lambda, W))^2 dx},$$

and when  $i \geq 2$ ,

$$L_i(x; \lambda, W) = (x^\lambda - B_i)L_{i-1}(x; \lambda, W) - C_i L_{i-2}(x; \lambda, W),$$

where

$$B_i = \frac{\int_a^b W(x)x^\lambda(L_{i-1}(x; \lambda, W))^2 dx}{\int_a^b W(x)(L_{i-1}(x; \lambda, W))^2 dx}, \quad C_i = \frac{\int_a^b W(x)x^\lambda L_{i-1}(x; \lambda, W)L_{i-2}(x; \lambda, W) dx}{\int_a^b W(x)(L_{i-2}(x; \lambda, W))^2 dx}.$$

**Proof.** We prove that the above result holds by mathematical induction; firstly, we will consider  $L_0(x; \lambda, W)$  and  $L_1(x; \lambda, W)$  and show that they are orthogonal.

$$\begin{aligned} \int_a^b W(x)L_0(x; \lambda, W)L_1(x; \lambda, W) dx &= \int_a^b W(x)(x^\lambda - B_1)dx \\ &= \int_a^b W(x)x^\lambda dx - B_1 \int_a^b W(x)dx \\ &= 0. \end{aligned}$$

Now, For the induction hypothesis, assume that  $\{L_0(x; \lambda, W), L_1(x; \lambda, W), \dots, L_{n-1}(x; \lambda, W)\}$  is orthogonal on  $[a, b]$  with respect to the weight function  $W(x)$ . Consider the following

$$\int_a^b W(x)L_i(x; \lambda, W)L_n(x; \lambda, W)dx, \quad i = 0, 1, \dots, n-1.$$

If we will prove the above integral value is zero then we are done. Take  $i = n - 1$  in the above expression and using the recurrence relation for  $L_n(x; \lambda, W)$ , we get

$$\begin{aligned} \int_a^b W(x)L_{n-1}(x; \lambda, W)L_n(x; \lambda, W)dx &= \int_a^b W(x)x^\lambda L_{n-1}^2(x; \lambda, W)dx - B_n \int_a^b W(x)L_{n-1}^2(x; \lambda, W)dx \\ &= 0. \end{aligned}$$

Similarly, we can show that integral value is zero for  $i = n - 2$ . Now, consider the integral for  $i = n - 3$  and using the recurrence relation for  $L_n(x; \lambda, W)$ , we get

$$\int_a^b W(x)L_{n-3}(x; \lambda, W)L_n(x; \lambda, W)dx = \int_a^b W(x)x^\lambda L_{n-3}(x; \lambda, W)L_{n-1}(x; \lambda, W)dx. \quad (4.11)$$

Using the relation  $x^\lambda L_{n-3}(x; \lambda, W) = L_{n-2}(x; \lambda, W) + B_{n-2}L_{n-3}(x; \lambda, W) + C_{n-2}L_{n-4}(x; \lambda, W)$  and orthogonal property in the Equation (4.11), we get

$$\int_a^b W(x)L_{n-3}(x; \lambda, W)L_n(x; \lambda, W)dx = 0.$$

From the similar argument, we can show for  $i = 0, 1, 2, \dots, n-4$ . Therefore, the set of fractional polynomial functions  $\{L_0(x; \lambda, W), L_1(x; \lambda, W), \dots, L_n(x; \lambda, W)\}$  defined in the following way is orthogonal on  $[a, b]$  with respect to the weight function  $W(x)$ .  $\square$

We can generalize the above Theorem in the discrete case. The following Theorem will help us to generate the fractional orthogonal polynomials over a set of points  $x_k$ ,  $k = 1, 2, \dots, m$  with respect to the weight function  $W(x)$ .

**Theorem 4.2.** *The set of fractional polynomial functions  $\{L_0(x; \lambda, W), L_1(x; \lambda, W), \dots, L_n(x; \lambda, W)\}$  defined in the following way is orthogonal over a set of points  $x_k$ ,  $k = 1, 2, \dots, m$ , with respect to the weight function  $W(x)$ .*

$$L_0(x; \lambda, W) = 1, \quad L_1(x; \lambda, W) = x^\lambda - B_1,$$

where

$$B_1 = \frac{\sum_{k=1}^m W(x_k)x_k^\lambda(L_0(x_k; \lambda, W))^2}{\sum_{k=1}^m W(x_k)(L_0(x_k; \lambda, W))^2},$$

and when  $i \geq 2$ ,

$$L_i(x; \lambda, W) = (x^\lambda - B_i)L_{i-1}(x; \lambda, W) - C_iL_{i-2}(x; \lambda, W),$$

where

$$B_i = \frac{\sum_{k=1}^m W(x_k)x_k^\lambda(L_{i-1}(x_k; \lambda, W))^2}{\sum_{k=1}^m W(x_k)(L_{i-1}(x_k; \lambda, W))^2}, \quad C_i = \frac{\sum_{k=1}^m W(x_k)x_k^\lambda L_{i-1}(x_k; \lambda, W)L_{i-2}(x_k; \lambda, W)}{\sum_{k=1}^m W(x_k)(L_{i-2}(x_k; \lambda, W))^2}.$$

**Proof.** Proof is similar to the proof of the Theorem 4.1, but we take discrete inner product in this case.  $\square$

After generating the fractional orthogonal polynomials with respect to weight function  $W(x)$  using the Theorems 4.1 and 4.2, we can directly compute the value of  $a_i$ . Hence, while minimizing the error defined by the Equations (4.5) and (4.6), we get

$$a_i = \frac{\int_a^b W(x)y(x)L_i(x; \lambda, W)dx}{\int_a^b W(x)(L_i(x; \lambda, W))^2dx}, \quad i = 0, 1, \dots, n, \quad (4.12)$$

and

$$a_i = \frac{\sum_{k=1}^m W(x_k)y(x_k)L_i(x_k; \lambda, W)}{\sum_{k=1}^m W(x_k)(L_i(x_k; \lambda, W))^2}, \quad i = 0, 1, \dots, n, \quad (4.13)$$

respectively.

## 5 Test examples

This Section is devoted to illustrate the accuracy and efficiency of the proposed modified least squares method discussed in Section 4. All the numerical simulation were run on an Intel Core i5 – 1135G7, 2.42GHz machine with 16GB RAM. To demonstrate the efficiency of the proposed method, we consider the following Examples:

**Example 1.** In this Example, we consider the function  $y(x) = x^{0.75} + x^{1.5}$  defined on the interval  $[0, 1]$ .

Here, we approximate  $y(x)$  with the  $P_2(x; \lambda) = a_0 + a_1x^\lambda + a_2x^{2\lambda} \in M_2^\lambda$ . After implementation of the proposed method discussed in Section 4, the results are described in detail as below:

- One can clearly observed that, for choises of  $a_0 = 0$ ,  $a_1 = 1$  and  $a_2 = 1$  with  $\lambda = 0.75$ , we get the  $P_2(x; 0.75)$  exact as a  $y(x)$ .
- Table 1 shows the least squares error  $E^C$  by the proposed method and CPU time with different value of  $\lambda$ .
- Also, Table 1 shows our proposed method's well accurate than the classical least squares method due to the additional parameter  $\lambda$ .

$\lambda$	$y(x) = x^{0.75} + x^{1.5}$	Total Error $E^C$	CPU time (in second)
0.75	$y(x) \approx 0.0000 + 1.0000x^{0.75} + 1.0000x^{1.5}$	$2.70e-24$	0.0558
1	$y(x) \approx 0.0329 + 1.7039x + 0.2597x^2$	$1.40e-5$	0.0702
1.5	$y(x) \approx 0.1388 + 2.5269x^{1.5} - 0.7126x^3$	$8.78e-4$	0.0631

Table 1: Comparison of the least squares error  $E^C$  in Example 1 corresponding to  $n = 2$  and with different value of  $\lambda$ .

**Example 2.** In this Example, we consider the function  $y(x) = x^{0.5} - \frac{\pi}{4}$  defined on the interval  $[0, 1]$ .

Here, we consider the two cases : first we approximate  $y(x)$  with the  $a_0L_0(x; \lambda, W) + a_1L_1(x; \lambda, W)$ , where  $W(x) = (1 - x)^{-\lambda}$ , while in second case we approximate  $y(x)$  with  $b_0L_0(x; \lambda) + b_1L_1(x; \lambda)$ . Both the cases, we will consider the followings residual error

$$E^C(a_0, a_1; \lambda) = \int_0^1 W(x)(y(x) - \sum_{i=0}^1 a_i L_i(x; \lambda, W))^2 dx,$$

and

$$E^C(b_0, b_1; \lambda) = \int_0^1 W(x)(y(x) - \sum_{i=0}^1 b_i L_i(x; \lambda))^2 dx,$$

respectively. After implementation of the proposed method discussed in Section 4, the results are described in detail as below:

- For  $\lambda = 0.5$ , if one can generate the fractional orthogonal polynomials up to order 1 with respect to weight function  $W(x) = (1 - x)^{-0.5}$  using Theorem 4.1, they get  $L_0(x; 0.5, W) = 1$  and  $L_1(x; 0.5, W) = x^{0.5} - \frac{\pi}{4}$ .
- In first case, one can clearly observed that, for choices of  $a_0 = 0$  and  $a_1 = 1$  with  $\lambda = 0.5$ , we get the  $a_0L_0(x; 0.5, W) + a_1L_1(x; 0.5, W)$  exact as a  $y(x)$ , without solving system of equations.
- In second cases, we get the value of  $b_0 = -0.1187$  and  $b_1 = 0.3333$ , with solving the system of equations and  $b_0L_0(x; 0.5) + b_1L_1(x; 0.5)$  exact as a  $y(x)$ .
- From the discussion of the above results, we can conclude that generating the orthogonal polynomial with respect to weight function  $W(x)$  gives the result without solving the system of equations and avoiding the ill-conditioning situation.

**Example 3.** In this Example, we consider the function  $y(x) = x^{1.5}$  defined on the interval  $[10, 20]$  to generate the data set of length 20.

Here, we approximate  $y(x)$  with the  $P_1(x; \lambda) = a_0 + a_1x^\lambda \in M_1^\lambda$ . After implementation of the proposed method discussed in Section 4, the results are described in detail as below:

- One can clearly observed that, for choices of  $a_0 = 0$  and  $a_1 = 1$  with  $\lambda = 1.5$ , we get the  $P_1(x; 1.5)$  exact as a  $y(x)$ .
- Table 2 shows the least squares error  $E^D$  by the proposed method with different value of  $\lambda$  and CPU time taken by the proposed method.
- Also, Table 2 displays the results for the noisy data set with generated by the function  $y(x)$  with 5% and 10% noise. Table results show the our proposed method is robust to noise.
- From Table 2, one can observe that, in the case of no noise, our proposed method captures the exact features of data for  $\lambda = 1.5$ , while the classical least squares method does not capture the data exactly. Moreover, when we increase the noise level, our proposed method is also accurate within two points significant digits.

	$\lambda$	$y(x) = x^{1.5}$	Total Error $E^D$	CPU time (in second)
No Noise	1.50	$y(x) \approx 0.0000 + 1.0000x^{1.5}$	8.20e-28	0.0149
	1.25	$y(x) \approx -11.1149 + 2.3609x^{1.25}$	2.02e-0	0.0145
	1.00	$y(x) \approx -27.7817 + 8.1671x$	8.17e-0	0.0143
5% Noise	1.50	$y(x) \approx -0.0001 + 1.0000x^{1.5}$	2.20e-2	0.0135
	1.25	$y(x) \approx -11.1136 + 2.3609x^{1.25}$	2.05e-0	0.0212
	1.00	$y(x) \approx -27.7792 + 5.7898x$	8.19e-0	0.0141
10% Noise	1.50	$y(x) \approx -0.0013 + 1.0000x^{1.5}$	8.80e-2	0.0142
	1.25	$y(x) \approx -11.1162 + 2.3610x^{1.25}$	2.10e-0	0.0142
	1.00	$y(x) \approx -27.7859 + 5.7902x$	8.26e-0	0.0198

Table 2: Comparison of the least squares error  $E^D$  in Example 3 corresponding to  $n = 1$  and with different value of  $\lambda$ .

**Example 4.** In this Example, we consider the data given in the Table 3.

$x$	0.0000	0.2500	0.5000	0.7500	1.0000
$y$ with no noise	0.0000	0.1340	0.3660	0.6589	0.6589
$y$ with 5% noise	-0.0062	0.0745	0.0705	0.0709	0.0336
$y$ with 10% noise	-0.0062	0.0745	0.0705	0.0709	0.0336

Table 3: Data with no noise, 5% noise and 10% noise.

Here, data given in the Table 3 with no noise, 5% noise and 10% noise. For this Example, we search a best fit curve of the form of  $P_1(x; \lambda) = a_0 + a_1x^\lambda \in M_1^\lambda$ . After implementation of the proposed method discussed in Section 4, the results are described in detail as below:

- Table 4 shows the least squares error  $E^D$  by the proposed method for  $n = 1$  and different value of  $\lambda$ .
- Also, Table 4 displays the results for the noisy data set given in the Table 3. Table results show the our proposed method is robust to noise.
- From Table 4, one can observe that, in the case of no noise, we get more accurate results for  $\lambda = 1.5$ , while for noisy data, we get the precise result for a different choice of  $\lambda$ .

	$\lambda$	$P_1(x; \lambda)$	Total Error $E^D$	CPU time (in second)
No Noise	1.50	$0.0071 + 0.9977x^{1.5}$	$1.3042e-4$	0.0123
	1.00	$-0.0732 + 0.0164x$	$1.6400e-2$	0.0109
	0.5	$-0.1409 + 0.9318x^{0.5}$	$1.2320e-1$	0.0115
5% Noise	1.50	$0.0420 + 0.0157x^{1.5}$	$4.7000e-3$	0.0115
	1.00	$0.0335 + 0.0304x$	$4.3000e-3$	0.0107
	0.50	$0.0159 + 0.0533x^{0.5}$	$3.1000e-3$	0.0131
10% Noise	1.50	$0.1789 - 0.2227x^{1.5}$	$3.5000e-2$	0.0111
	1.00	$0.2139 - 0.2596x$	$2.5400e-2$	0.0113
	0.50	$0.2705 - 0.3033x^{1.5}$	$1.1130e-2$	0.0111

Table 4: Comparison of the least squares error  $E^D$  in Example 4 corresponding to  $n = 1$  and with different value of  $\lambda$ .

**Example 5.** In this Example, we consider the function  $y(x) = x^{0.75}$  defined on the interval  $[0, 1]$  to generate the data set of length 30.

To demonstrate the advantage of choosing the weight functions, we consider Example 5. For this, we divide the interval  $[0, 1]$  into two parts  $[0, 0.8]$  and  $[0.8, 1]$ . In  $[0, 0.8]$ , we take the data sets as it is. However, in  $[0.8, 1]$ , we add 10% Gaussian noise in the data set. We generate fractional orthogonal polynomials concerning the two different weight functions. The outcome of these numerical experiments are described below:

- In the first case, if one can generate the fractional orthogonal polynomials up to order 1 with respect to weight function  $W(x) = (1 - x)$  using Theorem 4.1, they get  $L_0(x; 0.75, W) = 1$  and  $L_1(x; 0.75, W) = x^{0.75} - 0.5020$ .
- In the second case, if one can generate the fractional orthogonal polynomials up to order 1 with respect to weight function  $W(x) = 1$ , they get  $L_0(x; 0.75, W) = 1$  and  $L_1(x; 0.75, W) = x^{0.75} - 0.7870$ .
- The least-squares error  $E^D$  in the first case is  $2.020e-2$ , while in the second case, we get  $4.2040e-1$ .
- From the above discussion one can conclude that choosing the appropriate weight function according to the data in the least squares method provides better results.

**Example 6.** In this Example, we will implement our proposed method to predict the value of American put options, where the risk-neutral stock price process satisfies the following stochastic differential equation:

$$dS(x) = rS(x)dx + \sigma S(x)W(x), \quad (5.1)$$

where  $r$  and  $\sigma$  are constant, and  $W(x)$  is the standard Brownian process. Here the variable  $x$  is denoted the time.

The well known solution of the Equation (5.1) is

$$S(x) = S_0 \exp((r - \frac{1}{2}\sigma^2)x + \sigma W(x)), \quad (5.2)$$

where  $S_0$  is the initial stock price. The least square regression analysis is an essential part of machine learning. Therefore, we are interested in the predicted value of the American put option using our proposed method. We assume  $S_0 = 38$ ,  $r = 5\%$ ,  $\sigma = 71\%$ , options strike price is 48 and possible exercise time is 60 days (see [14] for details). Also, in the [14], the authors give the value of the American put option for the same data described above, which is 10.822. After implementation of our proposed method, we get the following results:

- Firstly we generate the data of length 60 from the Equation (5.2) in the interval  $[0, \frac{1}{6}]$  and then we fit the data using modified least square method in the space  $M_2^\lambda$ . The data is stochastic in nature, therefore, we use 10000 simulation for prediction.

- Using the algorithm described in [14] combined with the modified least square method, the value of predicated American put options are shown in Table 5 for different value of  $\lambda$ .
- From Table 5, one can easily observe that the predicted value of American put options in the case of  $\lambda = 0.75$  is much closer to the given value of the put options than other values of  $\lambda$ .

Table 5: Prediction of American put options in Example 6 with different value of  $\lambda$ .

$\lambda$	0.25	0.5	0.75	1.00
Predicated Value	10.743	10.730	10.790	10.714
CPU Time (in seconds)	10.720	10.723	10.756	10.856

## 6 Application of modified least squares method

The modified least squares method will have many practical applications in physics, finance, and other engineering problems. In this Section, we have demonstrated the application of the modified least squares method in particular areas like solving fractional differential/integral equations and in machine learning.

### 6.1 Application in solving fractional differential/integral equations

The theory of non-integer derivatives is an emerging topic of applied mathematics, which attracted many researchers from various disciplines. The non-local properties of fractional operators attract a significant level of intrigue in the area of fractional calculus. It can give an excellent way to deal with complex phenomena in nature, such as biological systems, control theory, finance, signal and image processing, sub-diffusion and super-diffusion process, viscoelastic fluid, electrochemical process, and so on (see [13, 21, 26, 18] and references therein). The main advantage of fractional differential/integral equations is that it provides a powerful tool for depicting the system with memory, long-range interactions and hereditary properties of several materials instead of the classical differential/integral equations in which such effects are difficult to incorporate. The fractional differential equations are equivalent to the Volterra's second kind integral equations for the specific choice of kernel. Consider the following Volterra second kind integral equation of the form

$$y(x) = a + \int_0^x k(x, t)f(t, y(t))dt, \quad (6.1)$$

where  $a \in \mathbb{R}$ ,  $f(x, t)$  to be a continuous function whereas  $k(x, t)$  may be singular. When  $k(x, t) = (x - t)^{\alpha-1}$ ,  $0 < \alpha < 1$ , then Equation (6.1) is equivalent to the following fractional differential equation

$$\begin{aligned} {}_{\text{C}}D_{0,x}^\alpha y(x) &= \Gamma(\alpha)f(x, y(x)), \\ y(0) &= a. \end{aligned} \quad (6.2)$$

However, in many cases, it is not possible to find the exact solution for fractional differential equations. Therefore, it is essential to acquire its approximate solution by using some numerical methods. In the literature there are a paper for solving fractional differential equations using least squares method based on polynomials  $P_n(x; 1) \in M_n^1$  [23]. But the fractional derivative of the polynomials  $P_n(x; 1)$  does not belong to the space  $M_n^1$ . Therefore, we introduce some extra error while solving the fractional differential equations using least squares method based on  $P_n(x; 1)$ . For example,  $n = 2$ , consider the space

$$M_2^\lambda := \{1, x^\lambda, x^{2\lambda}\}.$$

The fractional derivative (in Caputo sense) of order  $\alpha$  of any polynomial  $P_i(x; \lambda) \in M_2^\lambda$ ,  $i = 0, 1, 2$  are also in the space  $M_2^\lambda$  for the choice  $\alpha = \lambda$ . Hence, we feel that when solving the fractional differential using the least squares method in the space  $M_n^\lambda$  is beneficial than the space  $M_n^1$ .

**Example 7.** Consider the following fractional differential/integral equation

$${}_C D_{0,x}^\alpha y(x) = \frac{1}{\Gamma(2-\alpha)} x^{1-\alpha}, \quad 0 < x \leq 1, \quad (6.3)$$

with the analytical solution  $y(x) = x$  when  $y(0) = 0$  and  $f(x) = \frac{1}{\Gamma(2-\alpha)} x^{1-\alpha}$ .

We are using the modified least squares method to solve Example 7 in space  $M_2^\lambda$ . Let the solution of fractional differential equation  $y(x)$  be approximated by the polynomial  $a_0 + a_1 x^\lambda + a_2 x^{2\lambda}$ . In this case, we define the residual error as  $R(x, a_0, a_1, a_2; \lambda) = f(x) - {}_C D_{0,x}^\alpha (a_0 + a_1 x^\lambda + a_2 x^{2\lambda})$ . So, our error function becomes

$$E^C(a_0, a_1, a_2; \lambda) = \int_0^1 (R(x, a_0, a_1, a_2; \lambda))^2 dx. \quad (6.4)$$

For fix  $n = 2$ , the least square error  $E^C$  for Example 7 has been shown in Table 6 for  $\alpha = 0.5$  and various values of  $\lambda$ . From Table 6, one can observe that when  $\lambda = \alpha$ , we get the best result. This will happen because fractional derivatives of  $x^\lambda$  and  $x^{2\lambda}$  belong to the space  $M_2^\lambda$  when  $\lambda = \alpha$ .

Table 6: Comparison of the least square error  $E^C$  in Example 7 corresponding to  $\alpha = 0.5$  and with different value of  $\lambda$ .

$\lambda$	0.5	0.75	1.00	1.25	1.50
$E^C$	0	6.11e-4	5.19e-4	2.70e-3	8.60e-3

**Example 8.** In this Example, we consider the following multi-term fractional/integral differential equation

$${}_C D_{0,x}^{\alpha_1} y(x) + {}_C D_{0,x}^{\alpha_1} y(x) + y(x) = f(x), \quad 0 < x \leq 1, \quad (6.5)$$

with the exact solution  $y(x) = x^{3.5} + x^4$  when  $y(0) = 0$  and  $f(x) = x^4 + x^{3.5} + \frac{\Gamma(5)}{\Gamma(5-\alpha_1)} x^{4-\alpha_1} + \frac{\Gamma(4.5)}{\Gamma(4.5-\alpha_1)} x^{3.5-\alpha_1} + \frac{\Gamma(5)}{\Gamma(5-\alpha_2)} x^{4-\alpha_2} + \frac{\Gamma(4.5)}{\Gamma(4.5-\alpha_2)} x^{3.5-\alpha_2}$ , where  $\alpha_1 = 0.5$  and  $\alpha_2 = 0.25$ .

We are using the modified least squares method to solve Example 8 in space  $M_n^\lambda$ . Let the solution of fractional differential equation  $y(x)$  be approximated by the Müntz-Legendre polynomials  $\sum_{i=0}^n a_i L_i(x; \lambda)$ . In this case, we define the residual error as

$$\begin{aligned} R(x, a_0, a_1, \dots, a_n; \lambda) &= {}_C D_{0,x}^{\alpha_1} \left( \sum_{i=0}^n a_i L_i(x; \lambda) \right) + {}_C D_{0,x}^{\alpha_2} \left( \sum_{i=0}^n a_i L_i(x; \lambda) \right) \\ &\quad + \sum_{i=0}^n a_i L_i(x; \lambda) - f(x) + \sum_{i=0}^n a_i L_i(0; \lambda) - y(0). \end{aligned}$$

So, our error function becomes

$$E^C(a_0, a_1, \dots, a_n; \lambda) = \int_0^1 (R(x, a_0, a_1, \dots, a_n; \lambda))^2 dx. \quad (6.6)$$

The least square error  $E^C$  and absolute error (A.E.) at  $x = 1.00$  for Example 8 and corresponding CPU time have been shown in Table 7 for various values of  $\lambda$  and  $n$ . One can easily observe that for  $\lambda = 1$ , the approximate solution does not capture the exact features involved in the solutions of the Example 8 for any value of  $n$ . However, for some values of  $\lambda \neq 1$ , the approximate solution captures the exact features involved in the solutions of the Example 8. Thus, for the conclusion of the results of this Example, we can say that one can get a good approximation for the solution of fractional differential equations in the case of  $\lambda \neq 1$ . Figure 1 shows the trajectory of the exact solution and modified least square solution (MLSS) for Example 8 with  $\lambda = 0.75$  and different values of  $n$ . Figure 1 shows that our approximate solution converges to the exact solution when we increase the value of  $n$ .

Table 7: Comparison of the least square error  $E^C$  and A.E. in Example 8 corresponding for different values of  $\lambda$  and  $n$ .

$\lambda$		0.50	0.75	1.00	1.25
$n = 2$	$E^c$	5.96e-1	3.19e-1	1.29e-1	4.31e-2
	A.E.	2.22e-0	2.08e-1	3.28e-2	2.67e-4
	CPU Time (in seconds)	2.69	3.08	3.28	3.16
$n = 4$	$E^c$	7.93e-3	2.33e-4	5.10e-7	1.85e-9
	A.E.	2.09e-0	2.19e-2	4.27e-4	3.45e-5
	CPU Time (in seconds)	5.89	9.33	10.67	13.90
$n = 6$	$E^c$	4.59e-6	2.57e-11	9.79e-10	3.05e-10
	A.E.	1.72e-1	8.81e-6	3.27e-5	1.18e-5
	CPU Time (in seconds)	16.55	31.06	34.29	37.73
$n = 8$	$E^c$	3.08e-45	6.61e-14	1.53e-11	2.16e-11
	A.E.	4.40e-16	1.34e-6	5.79e-6	3.98e-6
	CPU Time (in seconds)	39.55	75.15	77.97	81.87
$n = 10$	$E^c$	2.69e-47	2.27e-15	6.45e-13	2.35e-12
	A.E.	4.40e-16	3.12e-7	1.53e-6	1.59e-6
	CPU Time (in seconds)	80.23	149.29	183.23	241.30

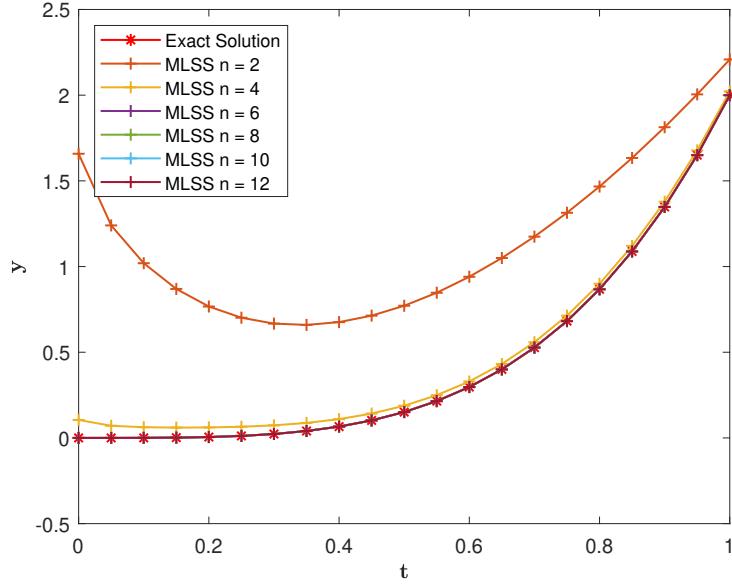


Figure 1: Exact solution and approximate solution for Example 8 with  $\lambda = 0.75$  and different values of  $n$ .

## 6.2 Application in machine learning

Machine learning is a field of artificial intelligence which gives computer systems the ability to learn using available data. Recently machine learning algorithms become very popular for analyzing of data and make predictions. The least-squares method is widely used in machine learning to analyze data for regression analysis and classification. In particular, in the regression analysis, the goal of people is to plot a best fit curve or line between data. If someone is interested in discovering the best fit line for one variable using the least squares method for the data, then, in this case, the form of polynomials

(hypothesis/model) is

$$P_1(x; 1) = a_0 + a_1 x.$$

We have already shown the advantage of the modified least squares method in Section 5 while searching the best fit curve between data. Also, some data need to have features vector like  $x^\lambda$ , for example  $x^{0.5}$ . Therefore, we must choose the polynomials (hypothesis/model) of the form

$$P_2(x; 0.5) = a_0 + a_1 x^{0.5}.$$

**Example 9.** Consider the data in Table 8, which is related to pharmaceutical sales of some company.

Table 8: Pharmaceutical sales of some company [1]

Years	2014	2015	2016	2017	2018
Sales	10000	21000	50000	70000	71000

For Example 9, we fit the data with the  $P_1(x; \lambda) = a_0 + a_1 x^\lambda \in M_1^\lambda$ . To simplify the calculation, the years in Table 8 are replaced by the coded values. For example, 2014 is 1, 2015 is 2, and so forth on. We use the data from 2014 to 2017 to fit the curve  $a_0 + a_1 x^\lambda$  and predict the sales in 2018. After implementation of the proposed method discussed in Section 4, the results are described in the Table 9. From the Table 9, one can observe that for  $\lambda = 0.5$ , the predicted sales in 2018 is close to the real value in 2018.

Table 9: Prediction of pharmaceutical sales of some company in the year 2018 with different values of  $\lambda$

$\lambda$	0.5	0.75	1.00	1.25	1.5
Predicted sales in 2018	69692	80546	90000	98307	105870

**Example 10.** In this Example, we will fit the data generated by the solution of the fractional differential equation, which describes the dynamics of the world population.

$$\begin{aligned} {}_C D_{0,x}^\alpha y(x) &= P y(x), \quad 0 < x \leq 1, \\ y(0) &= y_0 \end{aligned} \tag{6.7}$$

The solution of the fractional differential equations is given by  $y(x) = y_0 E_\alpha(Px^\alpha)$ , where  $y_0$  is the population of the world at initial time,  $E_\alpha(\cdot)$  is the Mittage-Leffler function,  $P$  is the production rate, and  $\alpha$  is the fractional order of the model.

In [5], authors are finding the values of  $P$  and  $\alpha$  using the world population data from 1910 to 2010, which are  $1.3502e - 2$  and 1.39, respectively. For this Example, we generating the data of  $y(x)$  with the help of the exact solution of the fractional differential equation (6.7) at 11 equispaced points on the interval  $[0, 1]$  for the values of  $P$  and  $\alpha$ . Here, we consider the two cases: first, we approximate  $y(x)$  with the  $\sum_{i=0}^n a_i x^{i\lambda}$ , while in the second case, we approximate  $y(x)$  with the  $\sum_{i=0}^n b_i P_i(x; \lambda)$ , where  $P_i(x; \lambda)$ ,  $i = 0, 1, \dots, n$ , are the orthogonal fractional polynomials with respect to data  $\{x_j, j = 0, 1, \dots, 10\}$ , and we generated  $P_i(x; \lambda)$ ,  $i = 0, 1, \dots, n$ , with help of the Theorem 4.2. In the first case for finding the value of parameter  $a_i$ ,  $i = 0, 1, \dots, n$ , we have to solve the system of equations because  $\{1, x^\lambda, \dots, x^{n\lambda}\}$  are non-orthogonal fractional polynomials with respect to data  $\{x_j, j = 0, 1, \dots, 10\}$ , and for a large value of  $n$ , we may end with the ill-conditioned coefficient matrix. In the second case, we are directly finding the parameter values  $b_i$ ,  $i = 0, 1, \dots, n$  using Equation (4.13). Table 10 demonstrates the absolute error (A.E.) at  $x = 0.55$  with different values of  $\lambda$  and  $n$  for both cases. From Table 10, one can observe that in the case of  $\lambda = 1.39$ , we get the better results compared to other values of  $\lambda$  for each  $n$ . This happens because the exact data is generated for  $\alpha = 1.39$ .

Table 10: A.E. at  $x = 0.55$  for Example 10 with different values of  $\lambda$  and  $n$ .

For non-orthogonal fractional polynomials				
$n$	$\lambda = 0.50$	$\lambda = 1.00$	$\lambda = 1.50$	$\lambda = 1.39$
2	1.96e-4	4.16e-5	4.78e-7	6.68e-10
3	8.79e-7	1.61e-5	1.50e-5	6.86e-13
4	4.19e-7	7.01e-6	1.62e-6	5.10e-15
5	2.64e-8	2.00e-6	4.79e-6	5.11e-15
6	3.61e-9	1.35e-6	8.31e-7	2.21e-13
For orthogonal fractional polynomials				
$n$	$\lambda = 0.50$	$\lambda = 1.00$	$\lambda = 1.50$	$\lambda = 1.39$
2	4.61e-4	2.01e-3	1.08e-4	8.36e-9
3	1.05e-5	9.55e-5	8.68e-5	4.76e-12
4	1.98e-6	6.29e-5	1.00e-4	3.18e-15
5	2.63e-7	4.93e-5	1.54e-4	4.83e-15
6	5.57e-8	4.24e-5	3.12e-4	2.09e-16

## 7 Conclusions and future work

The introductory Section shows the demands of the least squares method in various fields. So the modification in the least squares method is the demand of time due to its application. The main idea of this work has been to modify the least squares method using the space  $M_n^\lambda$ . The numerical results for test Examples have been reported to show the efficiency of the modified least squares method over the classical least squares method. We can use the current work in the support vector machines in the future. Support vector machines are part of machine learning to analyze data for classification and regression analysis. In most cases, data are non-linear. So, we find some non-linear transformation  $\phi$  that can be mapped the data onto high-dimensional feature space. The transformation is chosen in such a way that their dot product leads the kernel style function

$$K(x, x_i) = \phi(x) \cdot \phi(x_i).$$

If we choose the polynomial classifiers [11] of degree 2 and we have  $n$  data set  $x_i$ ,  $i = 1, \dots, n$ , in this case

$$\phi(x) = (1 \ x_1 \ x_2 \dots x_n \ x_1^2 \ x_1 x_2 \dots x_2^2 \dots x_n^2)^T.$$

However, one can use fractional polynomial classifiers instead of classical polynomial classifiers for more accurate results. Further, one can also use fractional orthogonal polynomials as an activation function in the neural networks to avoid the vanishing gradient problem.

## Acknowledgements

The First author acknowledges the support provided by University Grants Commission (UGC), India, under the grant number 20/12/2015(ii)EU-V. The second author acknowledges the support provided by the SERB India, under the grant number SERB/F/3060/2021 – 2022. The third author acknowledges the financial support from the North-Caucasus Center for Mathematical Research under agreement number 075 – 02 – 2021 – 1749 with the Ministry of Science and Higher Education of the Russian Federation.

## Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## References

- [1] Mohammad Abazid and Duaa Alkoud. A least-squares approach to prediction the future sales of pharmacy. *Int. J. Innov. Technol. Explor. Eng*, 7:1–4, 2018.
- [2] Hervé Abdi et al. The method of least squares. *Encyclopedia of Measurement and Statistics*. CA, USA: Thousand Oaks, 2007.
- [3] Anatoly A Alikhanov. A time-fractional diffusion equation with generalized memory kernel in differential and difference settings with smooth solutions. *Computational methods in applied mathematics*, 17(4):647–660, 2017.
- [4] Anatoly A Alikhanov, Murat Beshtokov, and Mani Mehra. The crank-nicolson type compact difference schemes for a loaded time-fractional hallaire equation. *Fractional Calculus and Applied Analysis*, 24(4):1231–1256, 2021.
- [5] Ricardo Almeida, Nuno RO Bastos, and M Teresa T Monteiro. Modeling some real phenomena by fractional differential equations. *Mathematical Methods in the Applied Sciences*, 39(16):4846–4855, 2016.
- [6] Rahul Bale, Amneet Pal Singh Bhalla, Boyce E Griffith, and Makoto Tsubokura. A one-sided direct forcing immersed boundary method using moving least squares. *Journal of Computational Physics*, 440:110359, 2021.
- [7] Peter Borwein, Tamás Erdélyi, and John Zhang. Müntz systems and orthogonal müntz-legendre polynomials. *Transactions of the American Mathematical Society*, 342(2):523–542, 1994.
- [8] Weihua Deng. Finite element method for the space and time fractional fokker–planck equation. *SIAM journal on numerical analysis*, 47(1):204–226, 2009.
- [9] Kai Diethelm and Neville J Ford. Volterra integral equations and fractional calculus: do neighboring solutions intersect? *The Journal of Integral Equations and Applications*, pages 25–37, 2012.
- [10] Shahrokh Esmaeili, Mostafa Shamsi, and Yury Luchko. Numerical solution of fractional differential equations with a collocation method based on müntz polynomials. *Computers & Mathematics with Applications*, 62(3):918–929, 2011.
- [11] Ingo Graf, Ulrich Kreßel, and Jürgen Franke. Polynomial classifiers and support vector machines. In *International Conference on Artificial Neural Networks*, pages 397–402. Springer, 1997.
- [12] W Leon Harter. The method of least squares and some alternatives: Part i. *International Statistical Review/Revue Internationale de Statistique*, pages 147–174, 1974.
- [13] Richard Herrmann. Folded potentials in cluster physics—a comparison of yukawa and coulomb potentials with riesz fractional integrals. *Journal of Physics A: Mathematical and Theoretical*, 46(40):405203, 2013.
- [14] Xuejun Huang and Xuewen Huang. The least-squares method for american option pricing, 2009.
- [15] Bangti Jin, Raytcho Lazarov, Yikan Liu, and Zhi Zhou. The galerkin finite element method for a multi-term time-fractional diffusion equation. *Journal of Computational Physics*, 281:825–843, 2015.
- [16] Nitin Kumar and Mani Mehra. Legendre wavelet collocation method for fractional optimal control problems with fractional bolza cost. *Numerical Methods for Partial Differential Equations*, 37(2):1693–1724, 2021.
- [17] Mark M Meerschaert, Hans-Peter Scheffler, and Charles Tadjeran. Finite difference methods for two-dimensional fractional dispersion equation. *Journal of Computational physics*, 211(1):249–261, 2006.

- [18] Vaibhav Mehandiratta, Mani Mehra, and Günter Leugering. Existence and uniqueness results for a nonlinear caputo fractional boundary value problem on a star graph. *Journal of Mathematical Analysis and Applications*, 477(2):1243–1264, 2019.
- [19] Vaibhav Mehandiratta, Mani Mehra, and Gunter Leugering. Optimal control problems driven by time-fractional diffusion equations on metric graphs: Optimality system and finite difference approximation. *SIAM Journal on Control and Optimization*, 59(6):4216–4242, 2021.
- [20] Gradimir V Milovanović. Müntz orthogonal polynomials and their numerical evaluation. In *Applications and computation of orthogonal polynomials*, pages 179–194. Springer, 1999.
- [21] Tatiana Odzijewicz, Agnieszka Malinowska, and Delfim Torres. Noether’s theorem for fractional variational problems of variable order. *Open Physics*, 11(6):691–701, 2013.
- [22] Kuldeep Singh Patel and Mani Mehra. Fourth order compact scheme for space fractional advection-diffusion reaction equations with variable coefficients. *Journal of Computational and Applied Mathematics*, page 112963, 2020.
- [23] Parisa Rahimkhani, Yadollah Ordokhani, and Esmail Babolian. Fractional-order bernoulli functions and their applications in solving fractional fredholem–volterra integro-differential equations. *Applied Numerical Mathematics*, 122:66–81, 2017.
- [24] Maziar Raissi and George Em Karniadakis. Hidden physics models: Machine learning of nonlinear partial differential equations. *Journal of Computational Physics*, 357:125–141, 2018.
- [25] Abhishek Kumar Singh and Mani Mehra. Uncertainty quantification in fractional stochastic integro-differential equations using legendre wavelet collocation method. In Krzhizhanovskaya V. et al. (eds) *Computational Science – ICCS 2020. Lecture Notes in Computer Science*, volume 12138, pages 58–71. Springer, 2020.
- [26] Abhishek Kumar Singh and Mani Mehra. Wavelet collocation method based on legendre polynomials and its application in solving the stochastic fractional integro-differential equations. *Journal of Computational Science*, 51:101342, 2021.
- [27] Brian V Smoliak, John M Wallace, Mark T Stoelinga, and Todd P Mitchell. Application of partial least squares regression to the diagnosis of year-to-year variations in pacific northwest snowpack and atlantic hurricanes. *Geophysical Research Letters*, 37(3), 2010.
- [28] Nikhil Srivastava, Aman Singh, Yashveer Kumar, and Vineet Kumar Singh. Efficient numerical algorithms for riesz-space fractional partial differential equations based on finite difference/operational matrix. *Applied Numerical Mathematics*, 161:244–274, 2021.

