	λ	$P_1(x;\lambda)$	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 λ.

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:

$$dS(x) = rS(x)dx + \sigma S(x)W(x), \tag{5.1}$$

where r and σ 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

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

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^λ. 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 λ.
- 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 λ.

Table 5: Prediction of American put options in Example 6 with different value of λ.

λ	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

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

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

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

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

Example 7. Consider the following fractional differential/integral equation

$$ {}_{\mathrm{C}}D^{\alpha}_{0,x}y(x) = \frac{1}{\Gamma(2-\alpha)}x^{1-\alpha}, \quad 0 < x \leq 1, \tag{6.3} $$

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^{λ}. Let the solution of fractional differential equation $y(x)$ be approximated by the polynomial $a_0 + a_1x^{\lambda} + a_2x^{2\lambda}$. In this case, we define the residual error as $R(x, a_0, a_1, a_2; \lambda) = f(x) - {}_{\mathrm{C}}D^{\alpha}_{0,x}(a_0 + a_1x^{\lambda} + a_2x^{2\lambda})$. So, our error function becomes

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

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 λ. From Table 6, one can observe that when $\lambda = \alpha$, we get the best result. This will happen because fractional derivatives of x^{λ} and $x^{2\lambda}$ belong to the space M_2^{λ} 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 λ.

λ	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

$$ {}_{\mathrm{C}}D^{\alpha_1}_{0,x}y(x) + {}_{\mathrm{C}}D^{\alpha_1}_{0,x}y(x) + y(x) = f(x), \quad 0 < x \leq 1, \tag{6.5} $$

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^{λ}. Let the solution of fractional differential equation $y(x)$ be approximated by the Müntz-Legendre polynomials $\sum_{i=0}^{n} a_iL_i(x; \lambda)$. In this case, we define the residual error as

$$ \begin{aligned} R(x, a_0, a_1, \ldots, a_n; \lambda) = {}_{\mathrm{C}}D^{\alpha_1}_{0,x}\Big(\sum_{i=0}^{n} a_iL_i(x; \lambda)\Big) + {}_{\mathrm{C}}D^{\alpha_2}_{0,x}\Big(\sum_{i=0}^{n} a_iL_i(x; \lambda)\Big) \\ + \sum_{i=0}^{n} a_iL_i(x; \lambda) - f(x) + \sum_{i=0}^{n} a_iL_i(0; \lambda) - y(0). \end{aligned} $$

So, our error function becomes

$$ E^C(a_0, a_1, \ldots, a_n; \lambda) = \int_0^1 (R(x, a_0, a_1, \ldots, a_n; \lambda))^2 dx. \tag{6.6} $$

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 λ 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.