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

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

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

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

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

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 $x_k, \ k=1,2,\ldots,m$ with respect to the weight function $W(x)$.

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

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

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

and

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

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 λ.
- Also, Table 1 shows our proposed method's well accurate than the classical least squares method due to the additional parameter λ.

λ	$\boldsymbol{y(x) = x^{0.75} + x^{1.5}}$	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 λ.

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

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

and

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

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 $P_1(x;\lambda) = a_0 + a_1 x^{\lambda} \in M_1^{\lambda}$. 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 λ 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.

	λ	$\boldsymbol{y(x) = x^{1.5}}$	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 λ.

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 $P_1(x;\lambda) = a_0 + a_1 x^{\lambda} \in M_1^{\lambda}$. 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 λ.
- 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 λ.