Journal of the Egyptian Mathematical Society

ORIGINAL RESEARCH Open Access

Enhanced moving least square method for the solution of volterra integro-differential equation: an interpolating polynomial

O. A. Taiwo[1], M. O. Etuk[2], E. Nwaeze[3] and M. O. Ogunniran[4*]

*Correspondence:
muideen.
ogunniran@uniosun.edu.ng
[4] Department
of Mathematical Sciences,
Osun State University,
Osogbo, Nigeria
Full list of author information
is available at the end of the
article

Abstract

This paper presents an enhanced moving least square method for the solution of volterra integro-differential equation: an interpolating polynomial. It is a numerical scheme that utilizes a modified shape function of the conventional Moving Least Square (MLS) method to solve fourth order Integro-differential equations. Smooth orthogonal polynomials have been constructed and used as the basis functions. A robust and unrestricted trigonometric weight function, along with the basis function, drives the shape function and facilitates the convergence of the scheme. The choice of the support size and some controlling parameters ensures the existence of the moment matrix inverse and the MLS solution. Valid explanation and illustration were made for the existence of the inverse linear operator. To overcome problems of near-singularity, the singular value decomposition rule is used to compute the inverse of the moment matrix. Gauss quadrature rule is used to compute the integral at the initial test points when the exact solution is unknown. Some tested problems were solved to show the applicability of the method. The results obtained compare favourable with the exact solutions. Finally, a highly significant interpolating polynomial is obtained and used to reproduce the solutions over the entire problem domain. The negligible magnitude of the error at each evaluation knot demonstrates the reliability and effectiveness of this scheme.

Keywords: Moving least square, Volterra integro-differential equation, Interpolating polynomial, Shape function, Weight function

Mathematics Subject Classification: 65R20, Secondary 65L02

Introduction

Integro—differential equations (IDEs) are equations that take into account both integral and derivatives of an unknown function [30]. Mathematical modeling of real-life problems usually results in functional equations like ordinary or partial differential equations, integral and integro—differential equations, and stochastic equations. Many mathematical formulations of physical phenomena contain IDEs; those equations pop up in many fields namely physics, Astronomy potential theory, fluid dynamics, biological models, and chemical kinematics.

IDEs are usually difficult to solve analytically and as such, there is a need to obtain an efficient approximate solution. Recently, much interest from researchers in science and engineering has been given to non-traditional methods for non-linear IDEs. The Existence-uniqueness, stability, and application of integro-differential equations were presented by Lakshmikautham and Rao [19]. Armand and Gouyandeh discussed IDE of the first kind in [3] and nonlinear Fredholm Integral Equations of the second kind were discussed by Borzabadi, Kamyad, and Mehne in [7]. A comparison between Adomian Decomposition Method (ADM) and Wavelet-Galerkin Method for solving IDEs was considered in [11] . He's Homotopy Perturbation Method was applied to nth-order IDEs in [12] and [15]. Tau Numerical solution of Fredholm IDEs with arbitrary polynomial bases. Elaborate work on IDEs was discussed in [8, 10, 13, 16, 19, 22–25, 31] and in [21] where Maleknejad and Mahmoudi applied Taylor polynomial to high-order nonlinear Volterra Fredholm Integro-differential Equations. Taylor Collocation Method was applied to linear IDEs in [18] by Karamete and Sezer.

In [2]; Theory, Method, and Application of boundary value problems for higher-order integro-differential equations were considered. Wavelet-Galerkin method and Hybrid Fourier and Block-Pulse Function in [5] and [4] were applied to IDEs respectively. Numerical Approximation of nonlinear Fourth-Order IDEs by Spectral Methods were considered in [34–38] and in [32]. A New Algorithm was utilized in solving a class of nonlinear IDEs in the reproducing kernel space. In [30], a Comparison between Homotopy Perturbation Method and Sine–Cosine Wavelets Method was applied to linear IDEs while in [29], a new Homotopy Method was applied to First and Second Orders IDEs.

The pseudospectral method has been proposed by using shifted Chebyshev nested for solving the IDEs in [28] while [14] applied the Adomian Decomposition Method (ADM) for solving Fourth-Order Integro-differential Equations. In [30], the main objective was only to obtain the exact solution to Fourth—Order Integro—differential equations. The ADM in [14] and the Variational Method in [27] are applied to solve both linear and non-linear boundary value problems of fourth-order Integro—differential equation.

In recent years, meshless methods have gained more attention not only by mathematicians but also by researchers in other fields of sciences and engineering. During the past decades, the moving least square (MLS) method proposed in [20] has now become a very popular approximation scheme, especially when considering a mesh-free approximating function. In [17], MLS and Gauss Legendre were applied to solve Integral Equation of the second kind while [8] utilized MLS with Chebyshev polynomial as a basis function to solve IDEs and the basic MLS was adopted in [9] in the solution of IDEs. The work of [26] and [27] were on the application of a two –dimensional Interpolating Function to Irregular-spaced data. A second kind chebyshev quadrature algorithm was developed for integral equations in [37] while a chebyshev collocation approached was adopted in the solution of IDEs in [33]. Many methodologies of IDEs in literature are popular with the use of regular-spaced data, the disordered-spaced data approach of MLS requires great skill of computations and this has been a source of attraction to researchers over the years.

In this research work, we employ the MLS to solve fourth order integro- differential equation. The method is an effective approach for the approximation of an unknown function by using a set of disordered data. It consists of a local weighted least square fit, valid on a small neighborhood of a point, and does not require information about the

background cell structure. Finally, a representative polynomial is used to generalize the solution to the entire problem domain. It is worthy to note that the MLS do not require a mesh and their approximations are built from the nodes only; an interesting advantage over other methods in the literature. The next section considers the definition of terms. Section two presents the conventional MLS scheme with its convergence description. Section three made a discussion on the scheme. Numerical examples were considered in section four while section five contains the Conclusion and Recommendation.

Definition of relevant terms

Definition 1.1.1 An integro—differential equation is an equation in which the unknown function u(x) appears under an integral sign and contains an ordinary derivative $u^{(n)}(x)$ as well, where n is the order of derivative.

Definition 1.1.2 A Standard integro—differential equation is of the form.

$$u^{(n)}(x) = f(x) + \lambda \int_{g(x)}^{h(x)} k(x,t)u(t)\mathrm{d}t \tag{1}$$

where $g(x)$ and $h(x)$ are the limits of integration, λ is a constant parameter, $k(x,t)$ is the kernel of the integral and $u^{(n)}(x)$ as defined in 1.1.1 above.

Definition 1.1.3 The conventional formula that converts multiple integrals to a single integral is defined as.

$$\int_0^x \int_0^{x_1} \int_0^{x_2} \ldots \int_0^{x_{n-1}} u(x_n)\mathrm{d}x_n\mathrm{d}x_{n-1}\mathrm{d}x_{n-2}\ldots\mathrm{d}x_1 = \frac{1}{(n-1)!}\int_0^x (x-t)^{n-1}u(t)\mathrm{d}t \tag{2}$$

This follows since if

$$\int_0^x \int_0^{x_1} F(t)\mathrm{d}t\mathrm{d}x_1 = \int_0^x (x-t)F(t)\mathrm{d}t \tag{3}$$

then, applying the concept of integration by parts:$\int u\mathrm{d}v = uv - \int v\mathrm{d}u$

$$u(x_1) = \int_0^{x_1} F(t)\mathrm{d}t$$

$$\begin{aligned}&\textstyle\int_0^x \int_0^{x_1} F(t)\mathrm{d}t\mathrm{d}x_1 = x_1 \int_0^{x_1} F(t)\mathrm{d}t|_0^x - \int_0^x x_1F(x_1)\mathrm{d}x_1\\ &\quad = x\textstyle\int_0^x F(t)\mathrm{d}t - \int_0^x x_1F(x_1)\mathrm{d}x_1\\ &\quad = \textstyle\int_0^x (x-t)F(t)\mathrm{d}t; \quad \text{using } x_1 = t.\end{aligned}$$

Definition 1.1.4 The fourth-order integro—differential equation is defined as.

$$\begin{aligned}u^{(4)}(x) =& f(x) + \beta u(x)\\ &+ \int_0^x [g(t)u(t) + h(t)F(u(t))]\mathrm{d}t,\ 0 \le x,t \le 1,\\ &u^{(i)}(0) = \alpha_i,\ i = 0,1,2,\ldots,3.\end{aligned} \tag{4}$$

where F is a real non-linear continuous function, $\beta,\ \alpha_i,\ i = 0,\ 1,\ 2,\ 3$ are real constants, $g(x),\ h(x)$ and $f(x)$ are given.

***Definition** 1.1.5* [6]**:** *The inverse of linear operator exists and it is linear $L : P \to Q$.*

This definition holds since if L^{-1} exists and its domain which is a vector space is Q then for any $P_1,\ P_2 \in P$ whose images are $q_1 = LP_1$ and $q_2 = LP_2$ we have $P_1 = L^{-1}q_1$ and $P_2 = L^{-1}q_2$.L is linear implies that for any scalars α and β we have $\alpha q_1 + \beta q_2 = \alpha LP_1 + \beta LP_2 = L(\alpha P_1 + \beta P_2)$.

Thus, $P_i = L^{-1}q_i$ exists. It follows that $L^{-1}(\alpha q_1 + \beta q_2) = \alpha L^{-1}q_1 + \beta L^{-1}q_2 = \alpha P_1 + \beta P_2$. Thus for $Y \in Q$, there exists X in P such that $L^{-1} : Y \to X$. In this paper, we consider a general n^{th} order Volterra Integro—differential equation of the form:

$$u^{(n)}(x) = f(x) + \beta u(x) + \int_0^x [g(t)u(t) + h(t)F(u(t))]dt,\ 0 \le x, t \le 1,\ u^{(i)}(0) = \alpha_i,\ i = 0,\ 1,\ 2, ...,\ n-1 \tag{5}$$

where F is a real non-linear continuous function, $\beta,\ \alpha_i,\ i = 0,\ 1,\ 2,\ ...,\ n-1$ are real constants, $g(x),\ h(x)$ and $f(x)$ are given and can be approximated by the Taylor series. When $n = 4$ Eq. (5) reduces to fourth-order integro—differential equation with four conditions as proposed in this paper.

The conventional MLS scheme

This research is aimed at obtaining an efficient method for approximating voltterra integro-differential equations. The method was obtained by introducing an interpolation polynomial in the context of the moving least square method, thereby producing an enhanced form of the approach. The absolute difference between the true solutions and the approximated solutions obtained from the new approach was used to check how close the results are to the true solutions. This section comprises the basic idea of the conventional moving least square method and its convergence.

Overview of the conventional MLS

Consider a sub-domain Ω_x, the neighborhood of a point X, and the domain of definition of the MLS approximation for the trial function at X which is located in the problem domain Ω. The approximation of the unknown function, u in Ω_x over some nodes, $x_i,\ i = 0,\ 1,\ 2,\ 3,\ ...,\ n$, is denoted by $\overline{u}(x)\ \forall x \in \Omega_x$ such that

$$\overline{u}(x) = \sum_{j=0}^{m} P_j(x)a_j(x) = P^T(x)a(x),\quad \forall x \in \Omega_x \tag{6}$$

where $P(x)$ is the basis function of the special coordinates, P^T denotes the transpose of P,m is the number of basis function and $a(x)$ is a vector containing coefficients $a_j(x),\ j = 0,\ 1,\ 2,\ ...,\ m$ which are functions of the space coordinate X. Also,'s $a_j(x)'s$ are the unknown coefficients to be determined.

The coefficient vector $a(x)$ is determined by minimizing a weighted discrete $L_2 - norm$, defined as:

$$J(\overline{u}) = \sum_{i=0}^{n} w_i(x)(\overline{u}(x_i) - U_i)^2 \tag{7}$$

where $U = (U_0,\ U_1,\ U_2,\ ...,\ U_n)^T$ is the exact solution and $w_i(x)$ is a new trigonometric weight function associated with the node $i.n$ is the number of nodes Ω for which the weight function, $w_i(x) = \cos(|x - x_i|) + \sin(|x - x_i|)$ is always positive on [0, 1] and |.| denotes absolute value. The stationarity of J with respect to $a_j(x);\ j \geq 0$ gives:

$$\begin{aligned}
\frac{\partial J(\overline{u})}{\partial a_0} &= \sum_{i=0}^{n} 2w_i(x)P_0(x_i)(P(x_i)a_0(x) - U_i) = 0 \\
\frac{\partial J(\overline{u})}{\partial a_1} &= \sum_{i=0}^{n} 2w_i(x)P_1(x_i)(P(x_i)a_1(x) - U_i) = 0 \\
&... \\
\frac{\partial J(\overline{u})}{\partial a_n} &= \sum_{i=0}^{n} 2w_i(x)P_n(x_i)(P(x_i)a_n(x) - U_i) = 0
\end{aligned} \tag{8}$$

Hence, Eq. (8) simplifies to

$$\sum_{i=0}^{n} w_i(x)P_i(x_i)(P(x_i)^T a(x) - U_i)a_i(x) = \sum_{i=0}^{n} w_i(x)(P(x_i)U_i) \tag{9}$$

By setting $A(x) = \sum_{i=0}^{n} w_i(x)P(x_i)P(x_i)^T$ as the $m \times m$ weighted moment matrix and

$$B(x) = [w_0(x)P(x_0),\ w_1(x)P(x_1),\ ...,\ w_n(x)P(x_n)]$$

we have

$$A(x)a(x) = B(x)U \tag{10}$$

Using singular value decomposition (svd) at the known value x, $A = RDV$, the inverse of the diagonal matrix, D^{-1}, contains $\frac{1}{d_{11}}$, $\frac{1}{d_{22}}$, ..., $\frac{1}{d_{mm}}$ elements at the diagonal for all the m nonzero elements in D and zeros elsewhere. Thus $A^{-1} = RD^{-1}V^T$. This procedure simplifies the computation of the inverse when the matrix is large. Selecting the values of x at the nodal points to ensure nonzero determinant of A and using the above inverse at each node, Eq. (10) becomes

$$a(x) = A^{-1}B(x)U \tag{11}$$

Substituting Eq. (11) into (1) gives

$$\overline{u}(x) = P^T(x)A^{-1}(x)B(x)U = \sum_{i=0}^{n} \varphi_i(x)U_i,$$

where $\varphi_i(x) = \sum_{k=0}^{m} P_k(x)[A^{-1}(x)B(x)]_{k_i}$ and $\varphi_i(x)$ are the shape functions of the MLS approximation corresponding to nodal point x. In this research work, a new set of orthogonal polynomials is used as the basis function on [0, 1]. Consider the first m polynomials,$p_m(x)$.

For $r = 0,\ 1, ...,\ m;\ f_i(x) = x^{r_i},\ i = 1,\ 2,\ ...,\ m+1$ we have $p_1(x) = f_1(x) = 1$. A simple Gram Schmidth algorithm that generates other polynomials:

for $i = 2$ *to* m

$p_i(x) = f_i(x)$

for $j = 1$ *to* $i - 1$

$$p_i(x) = p_i(x) - p_j(x)\int_0^1 (p_i(x)p_j)dx$$

end

$$p_i(x) = p_i(x)/\left(\int_0^1 (p_i(x)p_i)dx\right)^{0.5}$$

end

It follows that $p_2(x) = 3.4642x - 1.7321;\ p_3(x) = 13.417x^2 - 13.417x + 2.2361;$

$p_4(x) = 52.916x^3 - 79.374x^2 + 31.75x - 2.6458;\ p_5(x) = 210x^4 - 420x^3 + 270x^2 - 60x + 3$

Formulation of the proposed method

We wish to use the MLS method to obtain the numerical solution of (4):

$$Lu(x) = f(x) + \beta u(x) + \int_0^x [g(t)u(t) + h(t)F(u(t))]\mathrm{d}t;\quad L = \frac{\mathrm{d}^4}{\mathrm{d}x^4}. \tag{12}$$

Suppose that the four-fold operator,

$$L^{-1} = \int_0^x \int_0^x \int_0^x \int_0^x (.)\mathrm{d}t\mathrm{d}t\mathrm{d}t\mathrm{d}t = \int_0^x \frac{(x-t)^3}{3!}(.)\mathrm{d}t \tag{13}$$

exists.

By applying (13) on both sides of (12) we have

$$u(x) = a_0 + a_1x + \frac{a_2x^2}{2} + \frac{a_3x^3}{6} + L^{-1}(f(x)) + L^{-1}(\beta u(x)) + L^{-1}[g(t)u(t) + h(t)F(u(t))]\mathrm{d}t$$

and

$$u(x) = \sum_{j=0}^{3} a_j\frac{x^j}{j!} + L^{-1}(f(x)) + L^{-1}(\beta u(x)) + L^{-1}[g(t)u(t) + h(t)F(u(t))]\mathrm{d}t \tag{14}$$

To use the polynomials, we change the integral interval from $[0,\ x]$ to a fixed interval $[0,\ \ 1]$ using the translation $t = xs;\ dt = xds$:

$$u(x) = \sum_{j=0}^{3} a_j\frac{x^j}{j!} + L^{-1}(f(x)) + L^{-1}(\beta u(x)) + x\int_0^1 \frac{(x-xs)^3}{3!}[g(xs)u(xs) + h(xs)F(u(xs))]\mathrm{d}s \tag{15}$$

To apply the method, select the $m+1$ polynomials (basis) with nodal points x_i in $[0,\ 1]$. By using $\sum_{j=0}^{n} U_i\varphi_i(x)$ instead of $\overline{u}(x)$ as the approximation of $u(x)$ in (15) we have

$$\left[\sum_{j=0}^{n}\varphi_j(x)-\sum_{k=0}^{3}a_k\frac{x^k}{k!}-\beta L^{-1}(\varphi_j(x)-x\int_0^1\frac{(x-xs)^3}{3\,!}[g(xs)u(xs)+h(xs)F(u(xs))]\mathrm{d}s\right]U_j\approx L^{-1}(f(x))$$

In compact form we have.

$u(x)=R(x)+\int_0^1 K(x,t)dt$ where $R(x)=\sum_{j=0}^{3}a_j\frac{x^j}{j!}+L^{-1}(f(x))+L^{-1}(\beta u(x))$ and $K(x,t)=\frac{x(x-xt)^3}{3\,!}[g(xt)u(xt)+h(xt)F(u(xt))]$.

Finally, we introduce the use of interpolating polynomial, $up(x)$ at all points in [0, 1]:

$$up(x)=c_0+c_1x+...+c_kx^k \tag{16}$$

It has $N=k+1$ unknowns: $c_i,\ i=0,\ 1,\ ...,\ k$. Calculation of the unknowns requires $2N-1$ knots, $2N-2$ even steps, and MLS solution $u(x)$ at the evaluation points $x_i=\frac{1}{2N-2},\ i=0,\ 1,\ ...,\ 2N-2$. Clearly, $up(x_0)=u(x_0)=c_0$. It follows that

$$up(x_1)=c_0+c_1x_1+...+c_kx_1^k=u(x_1)$$

$$up(x_2)=c_0+c_1x_2+...+c_kx_2^k=u(x_2)$$

$$...\quad =\ ...\ +...\ +\ ...\ +\ ...=...$$

$$up(x_k)=c_0+c_1x_k+...+c_kx_k^k=u(x_k)$$

The above equations constitute a solvable system of k equations in k unknowns.. In general, given z odd knots and N unknowns in Eq. (16), we have $N=\frac{z+1}{2}$ and $k=N-1$. The polynomial $up(x)$, is a perfect fit with unit RSquare. A sample problem at evaluation knots $x_i=\frac{1}{8};\ \ i=0,\ 1,\ ...,\ 8$ requires 9 knots, 8 steps, and $k=4$. The interpolating polynomial becomes

$$up(x)=c_0+c_1x+c_2x^2+c_3x^3+c_4x^4.$$

The application of the new weight function, svd, and orthogonal basis in the implementation of the conventional MLS method constitutes the said enhancement.

Numerical computations

In this section, we use the MLS Method to solve integro-differential equations in the interval [0, 1]. All computations were carried out with scripts written in 2015 MATLAB. The accuracy of this method is directly proportional to the number of basis functions (m) and the nodal points (n). To compute the integral part at the initial nodes, in the absence of an exact solution, we use a six-point Gauss Quadrature Rule (GQR). It involves the Gaussian nodes.

xg = (−0.9324695142031520, −0.6612093864662645, −0.2386191860831969, 0.2386191860831969,

0.6612093864662645, 0.9324695142031520) and the corresponding weights.

C = (0.1713244923791703, 0.3607615730481386, 0.4679139345726910, 0.4679139345 726910,0.3607615730481386, 0.1713244923791703).

GQR requires that.

$\int_a^b f(x)dx = 0.5(b-a)\int_{-1}^{1} f(0.5(b-a)x + 0.5(b+a))\mathrm{d}x$

where $\int_{-1}^{1} f(x)dx = \sum_{i=1}^{6} C_i f(xg_i)$.

Using v as the number of nodes in the given evaluation points (x), initial condition: $u(1) = u_1 = a,\ s = x,\ k(1) = K(x(1), x(1))$ and $j = 2$ we estimate the corresponding values of $u(x)$ through GQR:

For $i = 1$ *to* $v - 1$

$a = x(i);\quad b = x(i+1);\quad an = 0.5(b-a)xg + 0.5(b+a);\ aw = 0.5(b-a)C;$

$\mathrm{int}(j) = Sum(aw \times K(an, an) \times k(j-1);$

$u(j) = R(x(j)) + \mathrm{int}(j);\ k(j) = K(x(i+1),\ x(i+1)) \times u(j);\quad j = j+1;$

end

when the exact solution is unknown at the initial nodes.

The accuracy of MLS increases as the number of basis polynomials and nodal points increases.

Numerical examples

Example 1. Consider the following nonlinear fourth-order Integro-Differential Equation [1]:

$$U^{(4)}(x) = 1 + \int_0^x e^{-t} U(t)^2 dt$$

with initial conditions: $U^{(i)}(0) = 1,\quad i = 0,\ 1,\ 2,\ 3$. The exact solution is given by $U(x) = e^x$ and using the transformation in (15) with the given initial conditions, we have:

$$U(x) = 1 + x + \frac{1}{2}x^2 + \frac{1}{6}x^3 + \frac{1}{24}x^4 + \frac{1}{24}x\int_0^1 (x - xs)^4 e^{-xs} U(xs)\mathrm{d}s$$

Solution of example 1 with m = 3 and npoints = 4:

For the sake of simplicity, we implement the MLS method for example (1), using three polynomials, $m = 3$, three nodal points $(n = 3)$, and four coordinate points $(npoints = 4)$. Choose the initial nodal point step: $dx = 0.5$. Initial nodal points become $xi = (0,\ 0.5,\ 1)^T$. The corresponding $U(xi) = (1,\ 1.6487,\ 2.7183)^T$ is obtained from the exact solution. A vector of the three basis polynomials is $p = (1,\ 3.4642x - 1.7321,\ 13.417x^2 - 13.417x + 2.2361)^T$. Using the above information, we seek the approximate solution at $x = (0,\ 0.25,\ 0.5,\ 0.75,\ 1)^T$ using the initial nodal points and MLS method at the given evaluation coordinates $x_i,\ i = 0,\ 1,\ 2,\ 3,\ 4$. Thus, $npoints = 4$. This process involves iterating from $j = 1$ to $npoints + 1$ to compute the required solution. At $j = 1$, compute $q = |x(j) - xi(i)|$ and $w = \sin(q) + \cos(q)$ for all $i = 0,\ 1$ and $2:\ w = (1,\ 1.3570,\ 1.3818)^T$.

Using these values in Eqs. (9) to (11) gives

$$p = \begin{pmatrix} 1 & 1 & 1 \\ -1.7321 & 0 & 1.7321 \\ 2.2361 & -1.11815 & 2.2361 \end{pmatrix};$$

$$\overline{\boldsymbol{p}} = (1, \ -1.7321, \ 2.2361)^T;$$

$$A = \begin{pmatrix} 3.7388 & 0.6613 & 3.8085 \\ 0.6613 & 7.1457 & 1.4787 \\ 3.8085 & 1.4787 & 13.6058 \end{pmatrix}$$

$$B = \begin{pmatrix} 1 & 1.357 & 1.3818 \\ -1.7321 & 0 & 2.3934 \\ 2.2361 & -1.5173 & 3.0898 \end{pmatrix}$$

and

$$A^{-1} = \begin{pmatrix} 0.3754 & -0.0133 & -0.1036 \\ -0.0133 & 0.1436 & -0.0119 \\ -0.1036 & -0.0119 & 0.1038 \end{pmatrix}$$

with

$p = (1,$ 3.4642xi - 1.7321, 13.417xi^2−13.417xi+2.2361$)^T$, $\overline{\boldsymbol{p}} = (p_1(x(j), \ p_2(x(j), \ p_3(x(j))^T$

where $p_1(x) = 1, \ p_2(x) = 3.4642x - 1.7321, \ p_3(x) = 13.417x^2 - 13.417x + 2.2361,$
$B = (w; \ w; \ w)^T$ and A^{-1}.
is obtained by singular value decomposition. Other parameters include

$\varphi(j, :) = \overline{p}^T A^{-1} B$:

$$\varphi(j, :) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

$$a(x) = A^{-1} B U^T = (1.71887, \ 0.4960, \ 0.0627)^T$$

$$b^T(j, :) = \overline{\boldsymbol{p}} = \begin{pmatrix} 1 & -1.7321 & 2.2361 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Repeat the above steps to update $\varphi(j, :)$ and $b^T(j, :)$ at $j = 2, \ 3, \ 4$ and 5. At $j = 5$, we obtained

$$a(x) = A^{-1}BU^T = (1.71887,\ \ 0.4960,\ \ 0.0627)^T.$$

$$\boldsymbol{\varphi} = \begin{pmatrix} 1 & 0 & 0 \\ 0.375 & 0.75 & -0.125 \\ 0 & 1 & 0 \\ -0.125 & 0.75 & 0.375 \\ 0 & 0 & 1 \end{pmatrix}$$

$$b^T = \begin{pmatrix} 1 & -1.7321 & 2.2361 \\ 1 & -0.8661 & -0.2796 \\ 1 & 0 & -1.1182 \\ 1 & 0.8661 & -0.2796 \\ 1 & 1.7321 & 2.2361 \end{pmatrix}.$$

Using exact solution at initial nodes, as shown above, we have.

$J(\overline{\boldsymbol{u}}) = (0,\ \ 0.0002042912556,\ \ 0.0002042912556)^T$. The optimal $J(\overline{u})$ is quite close to zero, thus we expect a good approximation:

$$u(x) \cong \boldsymbol{\varphi}(x)U = b^T(x)a(x) = (1,\ \ 1.27175572447,\ \ 1.6487212707,\ \ 2.1308966387,\ \ 2.718281828459)^T$$

Using Gauss six-point quadrature rule at initial nodes will give

$$u(x) \cong \boldsymbol{\varphi}(x)U = b^T(x)a(x) = (1,\ \ 1.27278641,\ \ 1.6484375,\ \ 2.126953268,\ \ 2.708333715)^T$$

The exact solution is

$$U(x) = (1,\ \ 1.28402542,\ \ 1.64872127,\ \ 2.11700002,\ \ 2.7182818285)^T$$

A two-degree polynomial $up(x) = 1 + 0.876603x + 0.841679x^2$, determines other intermediate solutions on [0, 1]. The fit RSquared and Adjusted RSquared are both equal to one. The following table contains the relevant statistics.

Solution of Example 1 with m = 5 and n points = 8:

Select initial nodal points *xi*, using dx = 0.25; and the corresponding approximate solution $U(xi)$. Following the outlined steps, we compute the values of $u(x)$ at $x = 0$ to 1 in steps of 1/8 using the MLS method and five orthogonal polynomials. The following are the obtained results.

$$J(\bar{u}) = 10^{-8} \times (0, 0.2461,\ 0.2461,\ 0.2883,\ 0.2883)^T.$$

The optimal $J(\overline{u})$ is quite close to zero, thus we expect a good approximation:

The exact and Enhanced MLS solutions coincide at the knots (Fig. 1). All the interpolated values are close to the exact solution. An insignificant difference exists as shown in this figure. The next figure highlights this observation.

Figure 2 shows insignificant errors between the exact and the Enhanced MLS solutions. The interpolating polynomial is $up(x) = 1 + 0.998803x + 0.509787x^2 + 0.140276x^3 + 0.0694157x^4$. All the computed coefficients are significant.

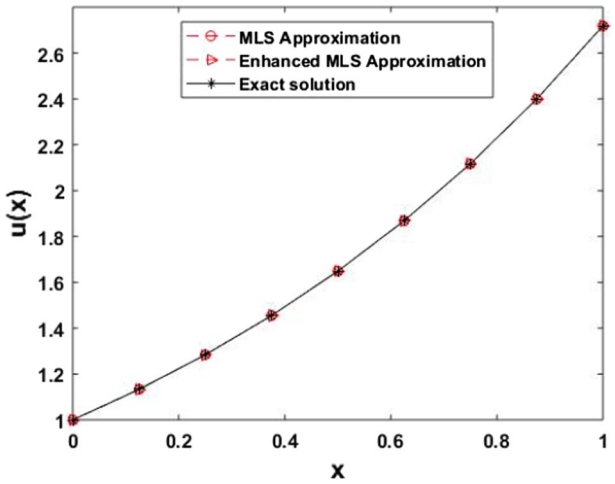

Fig. 1 Exact and MLS solutions of Example (1) at the given knots

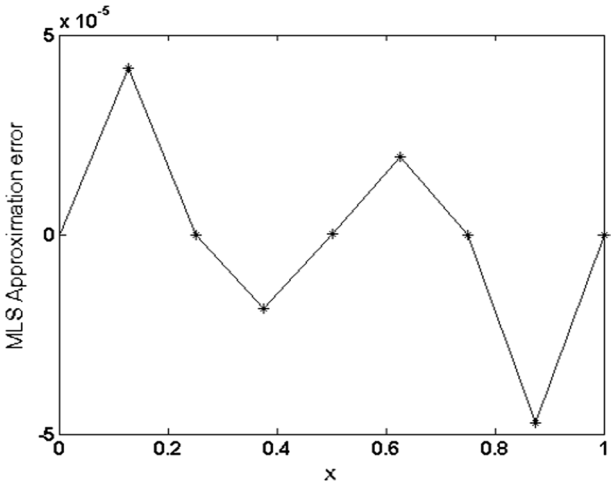

Fig. 2 MLS approximation errors of Example (1) at the given knots

Solution of Example 1 using up(x) on [0, 1] in 15 steps.

Following the outlined steps, compute the values of $up(x)$ at $x = 0$ to 1 in steps of 1/15. The following are the obtained results.

The obtained solutions in Table 4 are very close to the exact solution.

From Fig. 3, the exact and approximate solutions coincide at the knots.

The observed errors are insignificant as shown in Fig. 4. This implies perfect interpolation.

Example 2 Integro-differential equation [1]:

Solve.

$U^{(4)}(x) = \frac{5!}{\Gamma(2)}x + (1 + \frac{1}{7}x^2)x^2 - U(x) + \int_0^x tU(t)dt$ with initial conditions $U^{(i)}(0) = 0, \ \ i = 0, \ 1, \ \ 2, \ \ 3.$

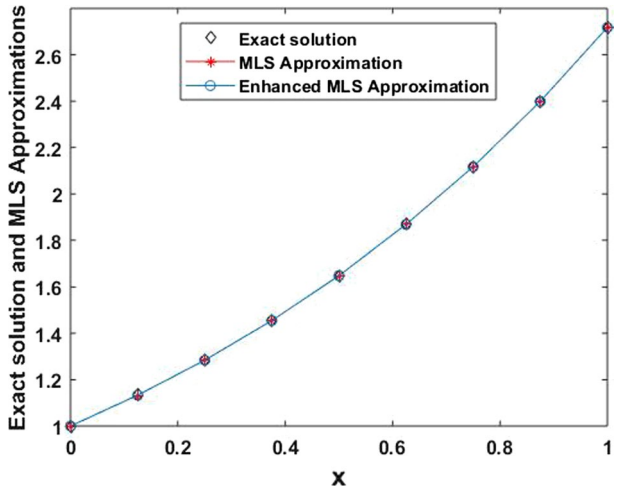

Fig. 3 Exact and MLS solution of Example (1) using up(x) at the given steps

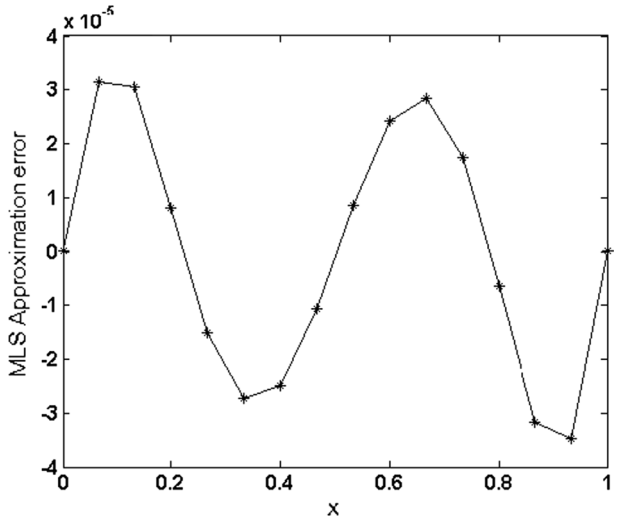

Fig. 4 MLS approximation errors of Example (1) using up(x) at the given steps

Solution of Example 2, using m = 5 polynomials and n = 15 nodes:

The exact solution is $U(x) = x^5$. Applying (12) on Example 2, we have:

$$U(x) = \frac{1}{55440}x^{11} + \frac{1}{3024}x^9 + x^5 - \frac{1}{24}x\int_0^1 (4(x - xs)^3 - (x - xs)^4)xsU(xs)\mathrm{d}s$$

Select initial nodal points and the corresponding approximate solution. Following the outlined steps, compute the values of $u(x)$ in steps of 1/15 using the MLS method. The following are the obtained results.

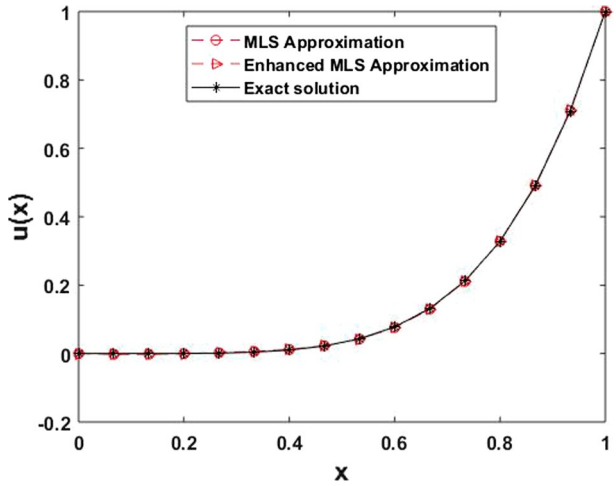

Fig. 5 Exact and MLS solution of Example (2) at the given knots

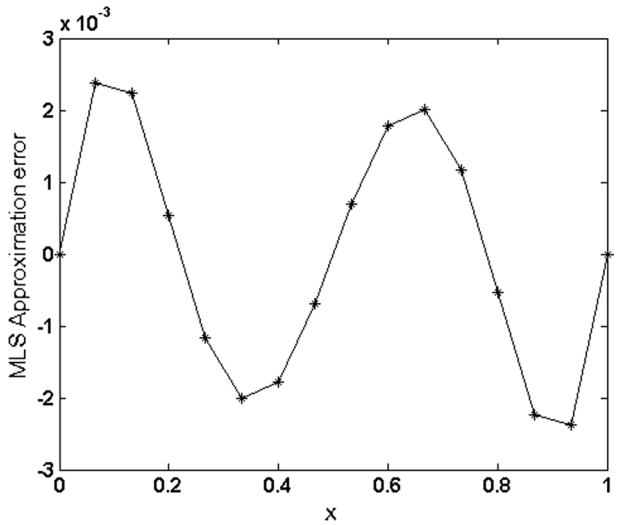

Fig. 6 MLS approximation errors of Example (2) at the given knots

$$J(\bar{u}) = (0, 0.80, 1.51, 1.55, 1.74, 2.28, 2.69, 2.75, 2.81, 3.16, 3.56)^T \times 10^{-5}.$$

The optimal $J(\overline{u})$ is quite close to zero, thus we expect a good approximation: The exact and approximate solutions in Table 5 are very close to each other. From Figs. 5, 6, the exact and approximate solutions coincide at the knots.

Example 3. Consider the following nonlinear fourth-order Integro-Differential Equation [1]:

$$U^{(4)}(x) = -\frac{1}{2}xe^{-2} + \frac{1}{2}xe^{x^2-2} - \int_0^x xte^{u(t)}dt$$

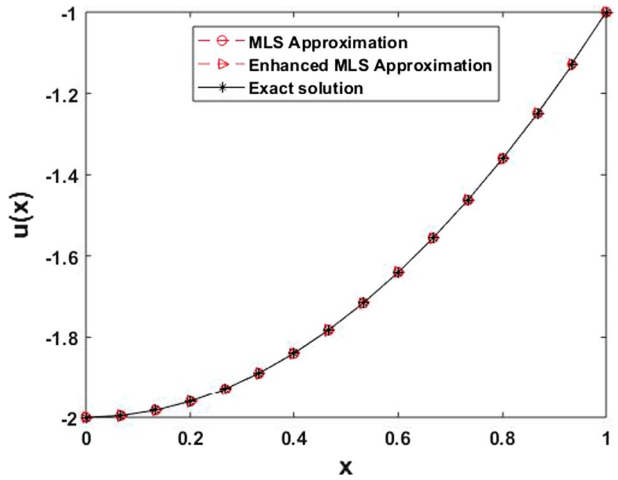

Fig. 7 Exact and MLS solution of Example (3) at the given knots

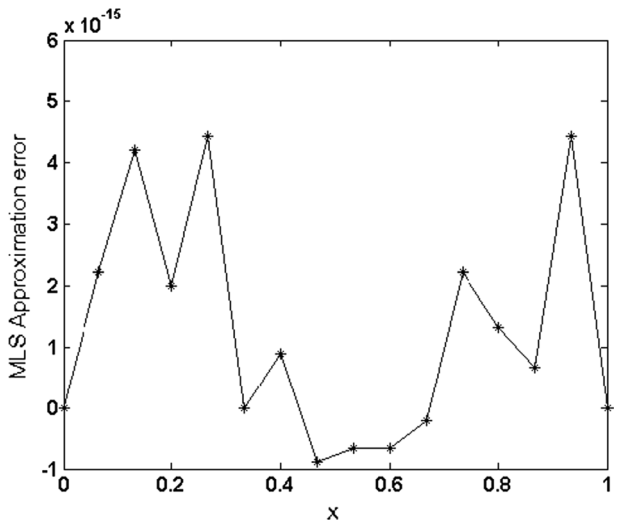

Fig. 8 MLS errors of Example (3) at the given knots

with initial conditions: $U(0) = -2,\ U''(0) = 2,\ \ U'(0) = U'''(0) = 0$. The exact solution is given by $U(x) = x^2 - 2$ and using the transformation in (15) with the given initial conditions, we have

Solution of Example 3, using m = 5 and n = 15:

Applying the same procedure gives

$$J(\bar{u}) = (0, 0.69, 3.21, 3.77, 6.51, 6.51, 6.62, 6.72, 6.77, 6.82, 5.82)^T \times 10^{-19}.$$

The optimal is quite close to zero, thus we expect a good approximation:

Table 1 Parameter results

Coef	Estimate	Standard error	*t*-Statistic	*P* Value
a_0	1	1.68341×10^{-12}	5.94031×10^{11}	0
a_1	0.876603	7.97655×10^{-12}	1.09898×10^{11}	0
a_2	0.841679	7.64892×10^{-12}	1.10039×10^{11}	0

Table 2 Exact and MLS solutions of Example (1) at the given knots, CPU time = 0.75 s

x	Approximate u(*x*)	MLS *u*(*x*)	Exact *u*(*x*)	MLS error	MLS error (enhanced)
0.0000	1.000000000000000	1.006439459167527	1.000000000000000	0.006439459167527	0.000000000000000
0.1250	1.133106724297920	1.126228676652465	1.133148453066826	0.006919776414362	0.000041728768907
0.2500	1.284025416687755	1.272883207934215	1.284025416687741	0.011142208753526	0.000000000000014
0.3750	1.455010035292180	1.447178650505727	1.454991414618201	0.007812764112474	0.000018620673979
0.5000	1.648721270700119	1.648109094186998	1.648721270700128	0.000612176513130	0.000000000000009
0.6250	1.868226545966574	1.875472047307945	1.868245957432222	0.007226089875722	0.000019411465649
0.7500	2.117000016612683	2.128917843084613	2.117000016612675	0.011917826471938	0.000000000000008
0.8750	2.398922570625700	2.407362950738166	2.398875293967098	0.008487656771067	0.000047276658602
1.0000	2.718281828459046	2.711632358267080	2.718281828459046	0.006649470191965	0.000000000000000

Table 3 Parameter results

Coef.	Estimate	Standard error	*t*-Statistic	*P* value
a_0	1	8.5546×10^{-15}	1.16896×10^{14}	3.21329×10^{-56}
a_1	0.998803	1.36072×10^{-13}	7.34026×10^{12}	$2.\ 06{,}683 \times 10^{-51}$
a_2	0.509787	6.0822×10^{-13}	8.38162×10^{11}	1.21573×10^{-47}
a_3	0.140276	9.41689×10^{-13}	1.48962×10^{11}	1.21856×10^{-44}
a_4	0.069416	4.67094×10^{-13}	1.48612×10^{11}	1.2301×10^{-44}

Table 4 Solution of Example (1), using at to 1 in steps of 1/15, CPU time = 0.52 s

x	Approximate *u*(*x*)	Exact *U*(*x*)	MLS error	MLS error (enhanced)
0.000000	1.00000	1.00000	0.0000000	0.000000000
0.0666667	1.06890	1.06894	0.0064395	0.000043585
0.133333	1.14259	1.14263	0.0069198	0.000039754
0.200000	1.22139	1.22140	0.0001114	0.000017405
…	…	…	…	…
1.000000	2.71828	2.71828	0.000066495	0.000000129

From Fig. 7 and Table 7, the exact and approximate solutions coincide at the knots.

The observed errors are insignificant as shown in Fig. 8. This implies perfect interpolation.

Following the procedure in example (1), the interpolating polynomial is Only the first and third coefficients are significant. Others are very close to zero and thus insignificant since their P-Values are greater than 0.05:

Table 5 Exact and MLS solutions at the given knots, CPU time = 0.75 s

x	Approximate $u(x)$	MLS $u(x)$	Exact $U(x)$	MLS error	MLS error (enhanced)
0	0	− 0.024655876676212	0	0.024655876676212	0
0.066666666666667	− 0.002379011893175	0.005537335586241	0.000001316872428	0.005536018713813	0.002380328765603
0.133333333333333	− 0.002200458335326	0.018361645942874	0.000042139917695	0.018319506025179	0.002242598253021
0.200000000000000	− 0.000209882115481	0.018910879661242	0.000320000000000	0.018590879661242	0.000529882115481
0.266666666666667	0.002519511299597	0.012394755114245	0.001348477366255	0.011046277747990	0.001171033933342
0.333333333333333	0.006117640493621	0.004251713602220	0.004115226337449	0.000136487264772	0.002002414156172
0.400000000000000	0.012015681631285	0.000071034716942	0.010240000000000	0.010168965283058	0.001775681631285
0.466666666666667	0.022823533748840	0.005268158970878	0.022132674897119	0.016864515926241	0.000690858851720
0.533333333333333	0.042460416868444	0.025004246618252	0.043151275720165	0.018147029101912	0.000690858851721
0.600000000000000	0.075984318368715	0.064305266155533	0.077760000000000	0.013454733844467	0.001775681631285
0.666666666666667	0.129684828642182	0.128063484798105	0.131687242798354	0.003623758000249	0.002002414156172
0.733333333333333	0.210912587465834	0.220886348193854	0.212083621399177	0.008802726794677	0.001171033933343
0.800000000000000	0.328209882115482	0.347116596621601	0.327680000000000	0.019436596621601	0.000529882115482
0.866666666666667	0.491188112656315	0.511170638717689	0.488945514403292	0.022225124314396	0.002242598253023
0.933333333333333	0.710625925473422	0.717685416786041	0.708245596707819	0.009439820078222	0.002380328765603
1.000000000000000	0.998551256887877	0.971372369628953	1.000000000000000	0.028627630371047	0.001448743112123

Table 6 Parameter results

Coef.	Estimate	Standard error	*t*-statistic	*P* value
a_0	1.04754×10^{-16}	2.33136×10^{-15}	0.0449326	0.966315
a_1	− 0.09375	3.70832×10^{-14}	-2.5281×10^{12}	1.46884×10^{-49}
a_2	0.78125	1.65756×10^{-13}	4.71325×10^{12}	1.21582×10^{-50}
a_3	− 2.18756	2.56635×10^{-13}-	8.52377×10^{12}	1.13664×10^{-51}
a_4	2.50000	1.27296×10^{-13}	1.96393×10^{13}	4.03315×10^{-53}

Table 7 Exact and MLS solutions at the given knots, CPU time = 0.5 s

x	Approximate *u*(*x*)	Exact *U*(*x*)	MLS error	MLS error (enchanced)
0	− 2.000000000000000	− 2.00000000000000	0	0
0.066666666666667	− 1.995555555555558	− 1.995555555555556	0.000000107013828	0.000000000000002
0.133333333333333	− 1.982222222222227	− 1.982222222222222	0.000000487796545	0.000000000000004
0.200000000000000	− 1.960000000000002	− 1.960000000000000	0.000000812959607	0.000000000000002
…	…	…	…	…
0.800000000000000	− 1.360000000000001	− 1.360000000000000	0.000048779654539	0.000000000000001
0.866666666666667	− 1.248888888888889	− 1.248888888888889	0.000004877964554	0.000000000000001
0.933333333333333	− 1.128888888888893	− 1.128888888888889	0.000001070138276	0.000000000000004
1.000000000000000	− 1.000000000000000	− 1.000000000000000	0	0

The Rsquare and Adjusted Rsquare are both 1.0. The statistics in Table 8 show that the chosen coefficients are the desired constants in The polynomial is a good fit for the MLS data. Any value, in.

[0, 1] interval can easily be evaluated with high precision.

Discussion of results

It is worthy to note that the computetions were carried out using MATLAB 9.2 on a personal computer of the following specifications. Windows 10 operating system in MATLAB 9.2 environment on 8.00 GB RAM HP Pavilion × 360 Convertible, 64-bits Operating System, × 64-based processor Intel(R) Core(TM) i3-7100U CPU @ 2.40 GHz. All the computed coefficients, in Table 1, are significant. Their P-Values are less than 0.05. Thus represents the generic polynomial which computes the values of on the interval [0, 1]. Throughout the numerical reports in Tables 1, 2, 3, 4, 5, 6, 7, and 8, expect where stated otherwise MLS is taken to mean the conventional moving least square method while MLS (Enhanced) is the new approach.

The magnitude of the computed errors in Table 2 indicates a close proximity between the exact and MLS solutions. The following figure compares the obtained solutions.

All the P-Values, in Table 3, are less than 0.05. The computed Rsquare and Adjusted Rsquare are equal to 1.0. The statistics in Table 3 show that the estimated coefficients are the desired constants in The estimated polynomial is a good fit for the MLS data.

Table 8 Parameter results

Coef.	Estimate	Standard error	*t*-statistic	*P* value
a_0	−2.00000	8.39093×10^{-15}	-2.38353×10^{14}	1.85896×10^{-57}
a_1	-6.31799×10^{-14}	1.33468×10^{-13}	−0.473369	0.660627
a_2	1.00000	5.96583×10^{-13}	1.67621×10^{12}	7.60038×10^{-49}
a_3	-5.15×10^{-13}	9.23672×10^{-13}	−0.557941	0.606636
a_4	2.71×10^{-13}	4.58158×10^{-13}	0.591785	0.585822

The observed errors are insignificant as shown in Fig. 6. This implies perfect interpolation.

Following the given procedure in example (1), the interpolating polynomial is All the computed coefficients are significant except the first which has zero value.

All parameters with a P-Value less than 0.05 are chosen. The Rsquare and Adjusted Rsquare are both one. The statistics in Table 6 show that the estimated coefficients are the desired constants of The polynomial is a good fit for the Enhanced MLS data. Any value of in [0, 1] interval can easily be evaluated with high precision. A high distinction of this method over existing methods is the significant interpolating polynomials obtained as a result of the constructed basis function which was then used to reproduce the solutions over the entire problem domain. The solutions produce a negligible magnitude of the error at each evaluation point and this demonstrates its reliability and effectiveness over existing methods.

Conclusion

An enhanced MLS method with smooth basis polynomials is used to solve the fourth order integro—differential equation of the Volterra type. At any arbitrary point, can be chosen to minimize the weight residual. Based on the results obtained, the value was given as a function of which accounts for the major difference between the Enhanced MLS, MLS method and the popular Least Square Method. Moreso, from the table of results, the error of the Enhanced MLS solution shows a tendency to increase as increases to the end boundary point. This behaviour is expected in any numerical method. Hence, we conclude that the proposed Enhanced Moving Least Square method is good for solving the class of equations described in this paper. Finally, a significant interpolating polynomial could be constructed and used to reproduce the solutions over the entire problem domain. The magnitude of the error at each evaluation knot demonstrates the reliability and effectiveness of this scheme. The application of the new weight function, svd, and orthogonal basis in the implementation of the conventional MLS method constitutes the said enhancement. The determinant of the moment matrix A(x) via SVD minimized the problem of near singularity and improved the accuracy of the results. The study concluded that enhanced MLS provides an alternative and efficient method of finding solutions to Volterra Integro-Differential equations and Fredholm–Volterra Integro-Differential equations. It is therefore recommended that the methods be used in solving the classes of problems considered.

Abbreviations
MLS: Moving least square; IDE: Integro-differential equations; ADM: Adomian decomposition method; GQR: Gauss quadrature rule; SVD: Singular value decomposition.

Acknowledgements
The authors hereby acknowledge with thanks everyone who has contributed to the success of this research

Authors' contributions
OT supervised all processes of development and implementation of the method. Investigated the suitability of the method to problems considered. ME developed and analyzed the method. He also revised the edited manuscript version. EN developed the code for implementation and made original draft preparation. He also revised the edited version. MO validates the software code, also writes and reviews the original manuscript. All authors read and approved the final manuscript.

Funding
Not applicable.

Availability of data and materials
Not applicable.

Declarations

Competing interests
The authors declare that there is no conflict of interest null competing interest.

Author details
[1]Department of Mathematics, University of Ilorin, Ilorin, Nigeria. [2]Department of Maths and Statistics, Federal Polytechnic Bida, Bida, Nigeria. [3]Department of Maths/Comp, Sc/Stats/Info, Alex Ekwueme Federal University Ndufu-Alike, Abakaliki, Nigeria. [4]Department of Mathematical Sciences, Osun State University, Osogbo, Nigeria.

Received: 1 March 2021 Accepted: 13 January 2022
Published online: 25 January 2022

References

1. Abdollahpoor, A.: Moving least square method for treating fourth order integro-differential equations. Commun. Adv. Comput. Sci. Appl. (2014). https://doi.org/10.5899/2014/CACSA-00023
2. Agarwal, R.P.: Boundary value problems for higher order integro-differential equations. Nonlinear Anal. Theory Methods Appl. **7**, 259–270 (1983)
3. Armand, A., Gouyandeh, Z.: Numerical solution of the system of volterra integral equations of the first kind. Int. J. Ind. Math. **6**, 27–35 (2013)
4. Asady, B., Kajani, M.T.: Direct method for solving integro-differential equations using hybrid fourier and block-pulse function. Int. J. Comput. Math. **8**, 888–895 (2007)
5. Avudainayagam, A., Vanci, C.: Wavelet-Galerkin method for integro-differential equations. Appl. Numer. Math. **32**, 247–254 (2000)
6. Gohberg, I., Goldberg, S.: Basic Operator Theory. Birkhäuser, Basel (2001)
7. Borzabadi, A.H., Kamyad, A.V., Mehne, H.H.: A different approach for solving the nonlinear Fredholm Integral equations of the second kind. Appl. Math. Comput. **173**, 724–735 (2006)
8. Bush, A.W.: Perturbation Methods for Engineers and Scientists. CRC, Florida (1992)
9. Dastjerdi, H.L., Ghaini, F.M.: Numerical solution of Voterra-Fredholm integral equations by moving least square method and Chebyshev polynomials. Appl. Math. Model. **36**, 3283–3288 (2012)
10. Darania, P., Ebadian, A.: A method for the numerical solution of the integro-differential equations. Appl. Math. Comput. **188**, 657–668 (2007)
11. El-Sayed, S., Abdel-Azizi, M.R.: A comparison of Adomian's decomposition method and Wavelet-Galerkin Method for solving Integro-differential equations. Appl. Math. Comput. **136**, 151–159 (2003)
12. Golbabai, A., Javidi, M.: Application of He's homotopy perturbation method for nth-order integro-differential equations. Appl. Math. Comput. **190**, 1409–1416 (2007)
13. Han, D.F., Shang, X.F.: Numerical solution of integro-differential equations by using CAS wavelet operational matrix of integration. Appl. Math. Comput. **194**, 460–466 (2007)
14. Hashim, I.: Adomian decomposition method for solving BVPs for fourth-order integro-differential equations. J. Comput. Appl. Math. **193**, 658–664 (2006)
15. Hosseini, S.M., Shahmorad, S.: Tau numerical solution of Fredholm Integro-differential equations with arbitrary polynomial bases. Appl. Math. Model. **27**, 145–154 (2003)
16. Jain, M.K., Iyengar, S.R.R., Jain, R.K.: Numerical Methods for Scientific and Engineering Computation. New Age International, New Delhi (2012)
17. Jid, R.E.: Moving least squares and gauss legendre for solving the integral equations of the second kind. Int. J. Appl. Math. **49**, 1–12 (2019)
18. Karamete, A., Sezer, M.: A taylor collocation method for the solution of linear integro-differential equations. Int. J. Comput. Math. **79**, 987–1000 (2002)
19. Lakshmikautham, V., Rao, M.R.: Theory of Integro-differential Equations. Gordon and Breach Publishers, U.S.A (1995)

20. Lancaster, P., Salkauskas, K.: Surface generated by moving least square method. J. Math. Comput. **37**, 148–158 (1981)
21. Maleknejad, K., Mahmoudi, Y.: Taylor polynomial solution of high-order nonlinear volterra fredholm integro-differential equations. Appl. Math. Comput. **145**, 641–653 (2003)
22. Omran, H.H.: Numerical methods for solving first order linear fredholm volterra integro-differential equations. Al-Nahran J. Sci. **12**, 139–143 (2009)
23. Parandin, N., Chenari, S., Heidari, S.: A numerical method for solving linear fredholm integro-differential equations of the first order. J. Basic Appl. Sci. Res. **3**, 192–195 (2013)
24. Rashed, M.T.: Lagrange interpolation to compute the numerical solutions of differential, integral and integro-differential equations. Appl. Math. Comput. **151**, 869–878 (2004)
25. Rihan, F.A., Doha, E.H., Hassan, M.I., Kamel, N.M.: Numerical treatments for volterra delay integro-differential equations. Comput. Methods Appl. Math. **9**(3), 292–308 (2009)
26. Shepard, D: A two-dimensional interpolation function for irregular-space data. Computer Science, Mathematics, Geograph Proceedings of the ACM National Conference, https://doi.org/10.1145/800186.810616 (1968).
27. Sweilam, N.H.: Fourth order integro-differential equations using variational method. Comput. Math. Appl. **54**, 1086–1091 (2007)
28. Sweilam, N.H., Khader, M.M., Konta, W.Y.: Numerical and analytical study for fourth-order integro-differential equations using a pseudospectral method. Int. J. Sci. Tech. **2**, 328–332 (2013)
29. Taiwo, O.A., Adewumi, A.O., Raji, R.A.: Application of new homotopy analysis method for first and second orders integro-differential equations. Int. J. Sci. Tech. **2**, 328–332 (2012)
30. Tavassoli, M.K., Ghasemi, M., Babolian, E.: Comparison between homotopy perturbation method and sine- cosine wavelets method for solving linear integro-differential equations. Comput. Math. Appl. **54**, 1162–1168 (2007)
31. Wazwaz, A.M.: A reliable algorithm for solving boundary value problems for higher-order Integro-differential equations. Appl. Math. Comput. **118**, 327–342 (2001)
32. Yang, L.H., Cui, M.G.: New algorithm for a class of nonlinear integro-differential equations in the reproducing kernel space. Appl. Math. Comput. **174**, 942–960 (2006)
33. Youssri, Y.H., Hafez, R.M.: Chebyshev collocation treatment of Volterra-Fredholm integral equation with error analysis. Arab. J. Math. **9**(2), 471–480 (2020)
34. Youssri, Y.H., Hafez, R.M.: Spectral Legendre-Chebyshev treatment of 2D linear and nonlinear mixed volterra-fredholm integral equation. Math. Sci. Lett. **9**(2), 37–47 (2020)
35. Doha, E.H., Youssri, Y.H., Zaky, M.A.: Spectral solutions for differential and integral equations with varying coefficients using classical orthogonal polynomials. Bull. Iran. Math. Soc. **45**(2), 527–555 (2019)
36. Doha, E.H., Abd-Elhameed, W.M., Elkot, N.A., Youssri, Y.H.: Integral spectral Tchebyshev approach for solving space Riemann-Liouville and Riesz fractional advection-dispersion problems. Adv. Differ. Equ. **1**, 284 (2017)
37. Abd-Elhameed, W.M.: Numerical solutions for Volterra-Fredholm-Hammerstein integral equations via second kind Chebyshev quadrature collocation algorithm. Adv. Math. Sci. Appl. **24**, 129–141 (2014)
38. Zhuang, Q., Ren, Q.: Numerical approximation of nonlinear fourth-order integro-differential equations by spectral method. Appl. Math. Comput. **232**, 775–783 (2014)

Publisher's Note