© Universiti Tun Hussein Onn Malaysia Publisher's Office

JST

Journal of Science and Technology

http://penerbit.uthm.edu.my/ojs/index.php/jst
ISSN : 2229-8460 e-ISSN : 2600-7924

Application of Exponentially Fitted Collocation Algorithm for Solving Nth-Order Fredholm Integrodifferential Equations

Falade Kazeem Iyanda[1*], Taiwo Omotayo Adebayo[2]

[1]Department of Mathematics, Faculty of Computing and Mathematical Sciences, Aliko Dangote University of Science and Technology Wudil, P.M.B 3244, Kano State, NIGERIA

[2]Department of Physical Science, Mathematics Programme, College of Applied and Physical Sciences, Landmark University, Omu-Aran, Kwara State NIGERIA

*Corresponding Author

DOI: https://doi.org/10.30880/jst.2023.15.01.004
Received 18 January 2023; Accepted 20 April 2023; Available online 20 July 2023

Abstract: In this paper, we seek to build and apply an exponentially fitted collocation algorithm (EFCA) for the solutions of nth-order Fredholm type integrodifferential equations. For this purpose, an EFCA was formulated and applied to solve four examples from the literature. Numerical experiment was performed and the results were compared with the exact solutions, and some existing methods. From the four examples considered, the results obtained showed that the proposed algorithm is fast, efficient, and reliable.

Keywords: Fredholm integrodifferential equations, exponentially fitted collocation algorithm, four examples, exact solutions

1. Introduction

Integrodifferential equations occur in many areas of applied mathematics such as inventory science, electrical engineering, biomathematical, and solid state physics which can be categorized into two type: Fredholm and Volterra Integrodifferential equations. The Volterra equations possess variable at the upper bound limit while the Fredholm equation possess a fixed bound of limits [1-3]. Generally, the study of integral and integrodifferential equations have an important role in investigating and understanding the physical phenomenon in many fields of applied sciences and engineering which are modeled by partial differential equations, ordinary differential equations, integral equations, and integrodifferential equations of different orders**.**

In this paper, we consider the nth-order Fredholm integrodifferential equation of the form:

$$\frac{d^n y(t)}{dt^n} + f(t)y(t) = \int_a^{\varphi} k(x,t)\frac{d^m y(x)}{dx^m}dx + h(t) \qquad m < n \qquad (1)$$

with initial conditions:

$$\begin{cases} y(a) = \Omega_0 \\ y^{/}(a) = \Omega_1 \\ y^{//}(a) = \Omega_2 \\ \vdots \\ y^{(n-1)}(a) = \Omega_{n-1} \end{cases} \qquad (2)$$

*Corresponding author: *faladekazeem2016@kustwudil.edu.ng*

Where $\varphi, \Omega_0, \Omega_1, \Omega_2 .., \Omega_{n-1}$ are real constants, $n \; and \; m$ are integers, $k(x,t)$ is the kernel function, and $f(t)$, $h(t)$ are smooth functions.

In the last two decades, some authors have proposed and applied several numerical techniques to solve Fredholm type of Integrodifferential equations, among which [4] proposed and used a polynomial method to solve Fredholm integrodifferential equations with coefficient constants, Authors [5] presented a modified homotopy perturbation method for the numerical solutions of the nonlinear Fredholm integral equation, [6] a numerical technique proposed and applied to solve Fredholm type integrodifferential equations, [7] conjugate gradient method was used to solve first-order Fredholm integrodifferential equations, [8] proposed and employed Legendre Galerkin method for the numerical solutions of linear differentials Fredholm integrodifferential equations, numerical solutions for the Fredholm integrodifferential equations with arbitrary polynomial bases by Tau method was presented in [9], Authors [10] used the general minimum remainder method for the numerical solutions of the second-order linear Fredholm integrodifferential equations, [11] built a direct computational algorithm for the numerical solutions of system of Fredholm integrodifferential equations, [12] used trigonometric scaling functions for the numerical solutions of the second-order linear Fredholm integrodifferential equations, and [13] employed power series for the numerical solutions of second-order Fredholm linear integral equations. In this paper, we seek to extend the numerical method proposed by [14] to solve Fredholm integrodifferential equations of nth-order and compare with exact solutions and some available methods in the literatures.

2. Description of Exponentially Fitted Collocation Method (EFCM)

The idea is to consider the power series as a basis function of the form

$$\boldsymbol{y(t) = \sum_{k=0}^{N} z_k \, t^k} \qquad (3)$$

And Exponentially fitted approximate solution of the form

$$\boldsymbol{y(t) \approx \sum_{k=0}^{N} z_k \, t^k \; + e^t} \qquad (4)$$

Taking the nth derivate of equation (3) and substitute into equation (1), we have

$$\sum_{k=1}^{N} \boldsymbol{z_k \, k(k-1)(k-2)\ldots(k-n)t^k} + f(t)\sum_{k=0}^{N} \boldsymbol{z_k \, t^k} \; = \int_a^{\varphi} k(x,t) \sum_{k=1}^{N} \boldsymbol{z_k \, k(k-1)(k-2)\ldots(k-m)x^k} dx + h(t) \quad (5)$$

Slightly perturb and collocate equation (5) respectively, we have

$$\sum_{k=1}^{N} \boldsymbol{z_k \, k(k-1)(k-2)\ldots(k-n)t^k} + f(t)\sum_{k=0}^{N} \boldsymbol{z_k \, t^k} - \int_a^{\varphi} k(x,t) \sum_{k=1}^{N} \boldsymbol{z_k \, k(k-1)(k-2)\ldots(k-m)x^k} dx - \boldsymbol{T_N(t_i)\tau_1} - \boldsymbol{T_N(t_i)\tau_2} - \boldsymbol{T_N(t_i)\tau_3} - \cdots - \boldsymbol{T_N(t_i)\tau_{n-1}} - h(t) = 0 \qquad (6)$$

Here $\boldsymbol{\tau_1, \tau_2, \tau_3, \tau_{n-1}}$ are free tau parameter to be determined, $\boldsymbol{T_N(t_i)}$ are the Chebyshev polynomials of degree N suggested by [16]

$$\boldsymbol{t_i = a + \frac{(b-a)i}{N+2}; t = 1, 2, 3, \ldots, N+1} \qquad (7)$$

Hence, equation (6) gives rise to (N+n+1) algebraic linear system of equations in (N+n+2) unknown constants. The extra equations are obtained from the initial conditions given in equation (2)

$$\begin{cases} y(a) \approx \sum_{k=0}^{N} z_k\, t^k \; + e^{t_0} = \Omega_0 \\ y^{/}(a) = \sum_{k=1}^{N} k z_k\, t^{k-1} \; + e^{t_0} = \Omega_1 \\ y^{//}(a) = \sum_{k=2}^{N} k(k-1) z_k\, t^{k-2} \; + e^{t_0} = \Omega_2 \\ \quad .. \\ \quad .. \\ y^{(n-1)}(a) = \sum_{k=n-1}^{N} k(k-1)(k-n-1) z_k\, t^{k-n-1} \; + e^{t_0} = \Omega_{n-1} \end{cases} \qquad (8)$$

Altogether, we obtained $(N + n + 1)$ algebraic linear equations in $(N + n + 1)$ unknown constants. Thus, we put the $(N + n + 1)$ algebraic equations in matrix form as

$$\mathrm{QY} = \mathrm{H(t)} \qquad (9)$$

Where Q is a square matrix, Y = $[y_0, y_1, y_2, y_3, \ldots y_N, \tau_1, \tau_2, \tau_3, .., \tau_N\ ]^T$ and H(t) = $[h(t_0), h(t_1), h(t_2), h(t_3), .., h(t_N)\ ]^T$

We use Gaussian elimination method to obtain the unknown constants $y_0, y_1, y_2, y_3, \ldots y_N, \tau_1, \tau_2, \tau_3, .., \tau_N$ and substitute into the exponential fitted approximate solution (4) to obtain unknown coefficients.

2.1 Exponentially Fitted Collocation Algorithm (EFCA)

In order to automate the mathematical procedures and reduce the computational length of the description given in section 2.0, we therefore propose a five-step algorithm (EFCA) using the MAPLE 18 software package to solve nth-order Fredholm integrodifferential equation (1) as follows:

```
**Step1:**
withplot:
$Digits := \mathbb{R}^+;$
$\varphi := \mathbb{R}^+;$
        $H := \mathbb{R}^+;$
        $a := [-1, 0];$
$m := order\ \ of\ FIDE;$
        $n := order\ \ of\ FIDE\ ;$
        $\boldsymbol{for\ i\ from\ 1\ to\ \ H\ do}$
        $g[i] := \int_a^{\varphi} k(x,t) * y^{(m)}[i]dt;$
        $a[i] := value(g[i]);$
**end do**

**Step 2:**
$basis\ function := t^i;$
        $v := sum\ (y[i] * basis\ function, i = 0.., H);$
$z[0] := eval(v, t = 0);$
$z[1] := eval(diff(v,t), t = 0);$
$z[2] := eval(diff(v, t\$2), t = 0);$
$z[3] := eval(diff(v, t\$3), t = 0);$
        ⋮        ⋮
$z[n-1] := eval(diff(v, t\$n-1), t = 0);$
        $T[0] := 1;$
        $T[1] := 2 * t + 1;$
        $\boldsymbol{for\ n\ from\ 1\ to\ \ H\ do}$
        $T[n+1] := simplify(2 * (2 * t - 1) * T[n] - T[n-1]);$
**end do;**
```

Step 3:

```
$Y := t^N;$
$\textbf{for } i \textbf{ from } 0 \textbf{ to } H \textbf{ do}$
$P := \left(\frac{d^n y(t)}{dt^n} - f(t) * y(t) - a[i]\right) * y(i);$
$A[i] := eval(P, N = i);$
do
$X := \left(sum(A[j], j = 0 \ldots N) - sum(T[p-k] * \tau[k+1], k = 0 \ldots n-1)\right) = g(t);$
$\textbf{for } m \textbf{ from } 1 \textbf{ to } N+1 \textbf{ do}$
    $eqn[r] := eval\left(T, t = \frac{r}{H+2}\right);$
$\textbf{end do}$
$eqn[H+2] := z[0] + e^{a}\tau[n] = \Omega_0;$
$eqn[H+3] := z[1] + e^{a}\tau[n] = \Omega_1;$
$eqn[H+4] := z[2] + e^{a}\tau[n] = \Omega_2;$
$\vdots \qquad \vdots$
$eqn[H+n+1] := z[n-1] + e^{a}\tau[n] = \Omega_{n-1};$
```

Step 4:

```
$sys := seq(eqn[i], i = 1 \ldots H+n+1);$
$sol := evalf\left(solve(\{sys\})\right);$
$Q := eval\left([seq(y[i], i = 0 \ldots H), \tau[n]], sol\right);$
$for\ r\ from\ 1\ to\ H\ do$
    $y[r] := Q[r+1];$
$end\ do$
$\tau[n] := Q[p+2];$
```

Step 5:

```
$solution := sum(y[j] * t^j, j = 0 \ldots H) + \tau[n] * t;$
$for\ t\ from\ 0\ by\ 0.1\ to\ 1\ do$
    $EFCA[t] := evalf\ (eval(solution, t);$
$end\ do$
$[2Dplot] := plot([Exact, EFCA], t = 0 \ldots 1, color[blue, red], axes = boxed, title = FIDE);$
$[2Dplot] := logplot([Exact, EFCA], t = 0 \ldots 1, color[blue, red], axes = boxed, title = FIDE);$
```

Output: = see Tables 1,2,3,4 and Figs. 1, 2,3,4,5,6,7,8.

2.2 Absolute Errors E_t

To show the effectiveness of the algorithm presented, four examples are considered and the results are compared with the exact solution and available methods. We define the absolute errors as follow:

$$E_t = |y(t_e) - y(t_{numerical})| \qquad (10)$$

Where $y(t_e)$ and $y(t_{numerical})$ are exact and approximate solutions of the given Fredholm integrodifferential equations respectively.

3. Numerical Examples

In this section, four examples are considered to demonstrate the effectiveness of the proposed technique and results obtained are compared with exact solutions by evaluating the maximum absolute errors.

Example 1. Consider the first-order Fredholm integrodifferential equation [15].

$$y^{/}(t) = te^t + e^t - t + \int_0^1 ty(x)dx \qquad (11)$$

with initial condition

$$y(0) = 0 \qquad (12)$$

Exact solution

$$y(0) = te^t \tag{13}$$

Table 1 - Numerical solutions for example 1

t	$y(t_e)$	E_{EFCA}	E_{ADM} **[15]**	E_{VIM} **[15]**
0	0.000000	0.0E-00	0.0E-00	0.0E-00
0.1	0.110517	1.7E-09	3.0E-09	3.0E-09
0.2	0.244280	6.6E-09	1.2E-09	1.2E-09
0.3	0.404957	1.5E-08	2.7E-08	2.7E-08
0.4	0.596729	2.6E-08	4.8E-08	4.8E-08
0.5	0.824360	4.1E-08	7.4E-08	7.4E-08
0.6	1.093271	5.9E-08	1.1E-07	1.1E-07
0.7	1.409626	8.1E-08	1.5E-07	1.7E-07
0.8	1.780432	1.1E-07	1.9E-07	1.9E-07
0.9	2.213643	1.3E-07	2.4E-07	2.4E-07
1.0	2.718282	1.6E-07	2.9E-07	2.9E-07

Example 2. Consider the second-order Fredholm integrodifferential equation [17].

$$y^{//}(t) + ty^{/}(t) - y(t) = e^t - 2\sin(t) + \int_{-1}^{1} e^x \sin(t) y(x) dx \tag{14}$$

with initial conditions

$$\begin{cases} y(0) = 1 \\ y^{/}(0) = 1 \end{cases} \tag{15}$$

Exact solution

$$y(t) = e^t \tag{16}$$

Table 2 - Numerical solutions for example 2

t	E_{EFCA}	E_{MTEM} **[17]**	E_{TEM} **[17]**
-1.0	1.04E-04	8.2E-04	1.1E-03
-0.8	2.42E-05	4.6E-04	4.2E-04
-0.6	3.76E-06	2.1E-04	1.2E-04
-0.4	1.79E-07	6.4E-05	2.1E-05
-0.2	3.62E-08	8.1E-06	4.6E-07
0.0	0.00E-00	----	----
0.2	5.74E-08	8.2E-06	8.2E-06
0.4	3.76E-07	6.4E-05	1.4E-04
0.6	1.18E-06	2.1E-05	9.2E-04
0.8	1.64E-06	4.7E-04	3.9E-03
1.0	4.87E-06	8.7E-04	1.3E-02

--- Not available for comparison

Example 3. Consider the third-order Fredholm integrodifferential equation [19]

$$y^{///}(t) = \sin(t) + t - \int_0^{\frac{\pi}{2}} xty^{/}(x)dx \tag{17}$$

with initial conditions

$$\begin{cases} y(0) = 1 \\ y^{/}(0) = 0 \\ y^{//}(0) = -1 \end{cases} \tag{18}$$

Exact solution

$$y(t) = \cos(t) \tag{19}$$

Table 3 - Numerical solutions for example 3

t	$y(t_e)$	E_{EFCA}	E_{VIM} **[19]**
0	1.00000	0.0E-00	0.0E-00
0.1	0.99500	2.5E-17	---
0.2	0.98007	3.9E-16	6.9E-08
0.3	0.95534	1.9E-15	---
0.4	0.92106	6.2E-15	1.1E-06
0.5	0.87758	1.4E-14	---
0.6	0.82534	2.7E-14	5.7E-06
0.7	0.764842	4.1E-14	---
0.8	0.696707	4.5E-14	1.8E-05
0.9	0.621609	1.3E-14	---
1.0	0.540302	9.8E-14	4.4E-05

--- Not available for comparison

Example 4. Consider the eight-order Fredholm integrodifferential equation [19].

$$y^{V///}(t) = -8e^{t} + t^{2} + y(t) + \int_{0}^{1} t^{2} y^{/}(x)dx \tag{20}$$

with initial conditions

$$\begin{cases} y(0) = 1 \\ y^{/}(0) = 0 \\ y^{//}(0) = -1 \\ y^{///}(0) = -2 \\ y^{/V}(0) = -3 \\ y^{V}(0) = -4 \\ y^{V/}(0) = -5 \\ y^{V//}(0) = -6 \end{cases} \tag{21}$$

Exact solution

$$y(t) = (1-t)e^{t} \tag{22}$$

Table 4 - Numerical solutions for example 4

t	$y(t_e)$	E_{EFCA}	E_{VIM} **[19]**
0	1.000000	0.0E-00	0.0E-00
0.1	0.9946534	9.9E-18	---
0.2	0.9771222	1.1E-17	1.1E-16
0.3	0.9449012	1.3E-17	---
0.4	0.8950948	7.5E-18	1.2E-14
0.5	0.8243606	3.9E-17	---
0.6	0.7288475	1.4E-15	6.6E-13
0.7	0.6041258	4.1E-15	---
0.8	0.4451082	1.1E-14	1.2E-11
0.9	0.2459603	2.5E-14	---
1.0	0.0000000	5.6E-14	1.1E-10

--- Not available for comparison

4. Results and Discussions

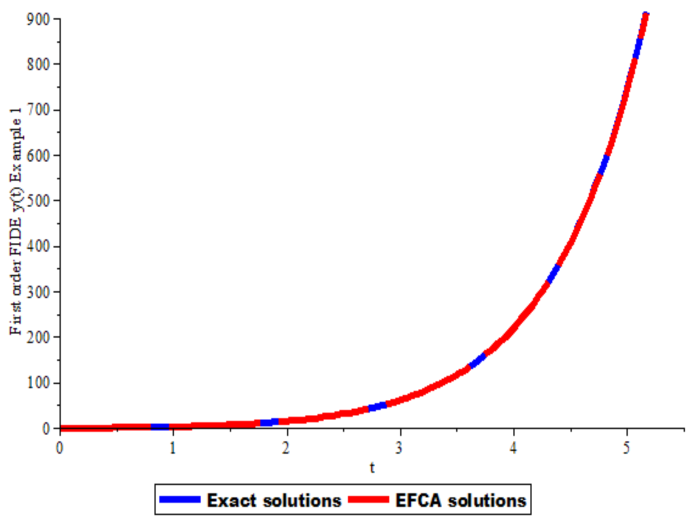

Fig. 1 - Depict comparison between exact and exponentially fitted collocation algorithm solutions on interval $0 \leq t \leq 5$ for example 1

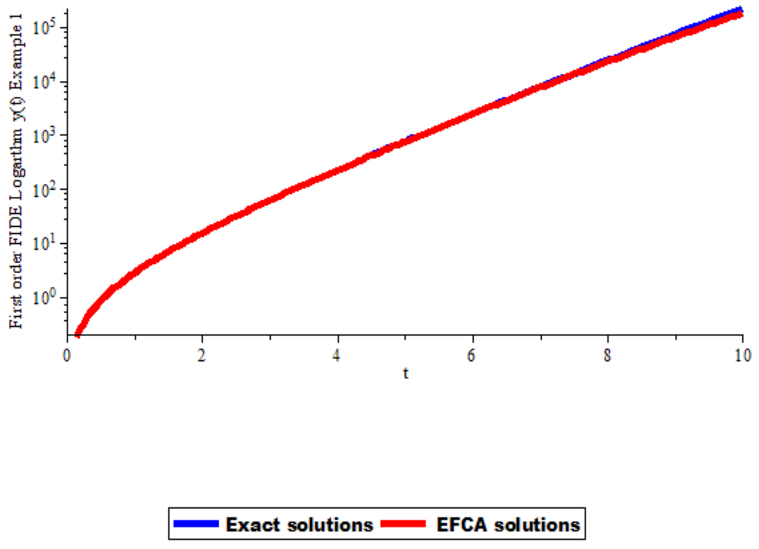

Fig. 2 - Depict logarithm comparison between exact and exponentially fitted collocation algorithm solutions on interval $0 \leq t \leq 5$ for example 1

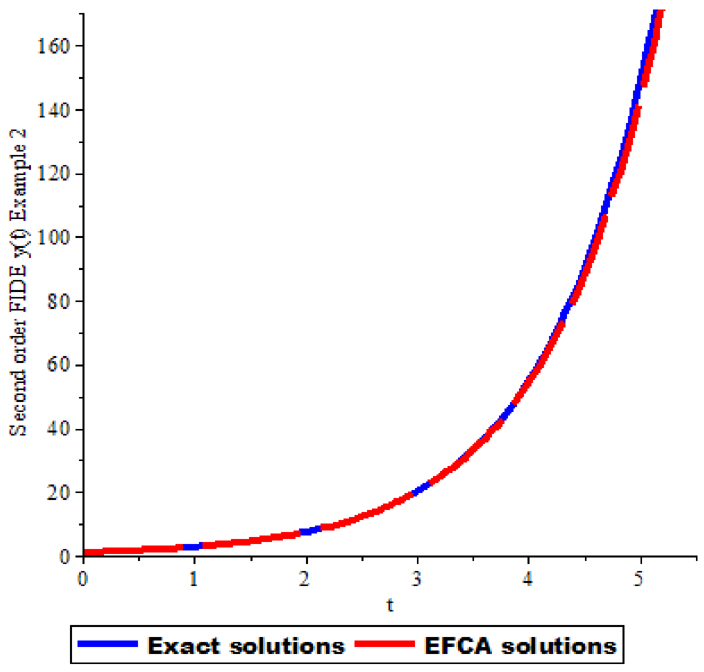

Fig. 3 - Depict comparison between exact and exponentially fitted collocation algorithm solutions on interval $0 \leq t \leq 5$ for example 2

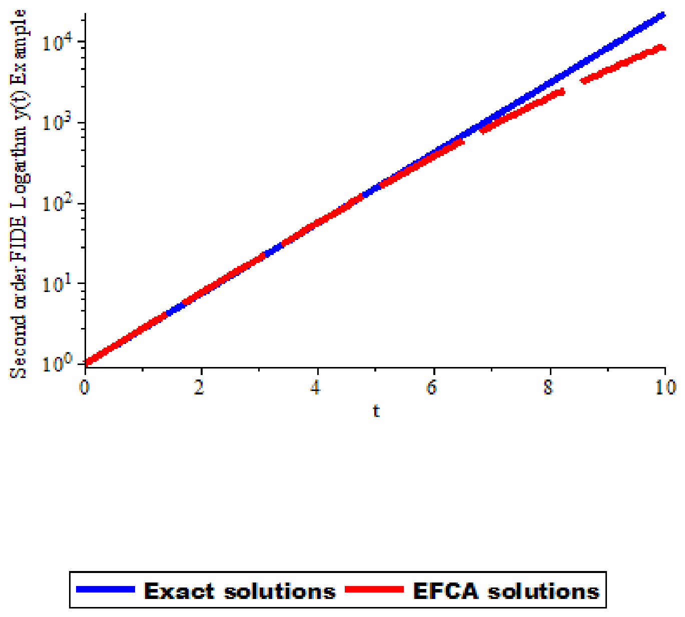

Fig. 4 - Depict logarithm comparison between exact and exponentially fitted collocation algorithm solutions on interval $0 \leq t \leq 5$ for example 2

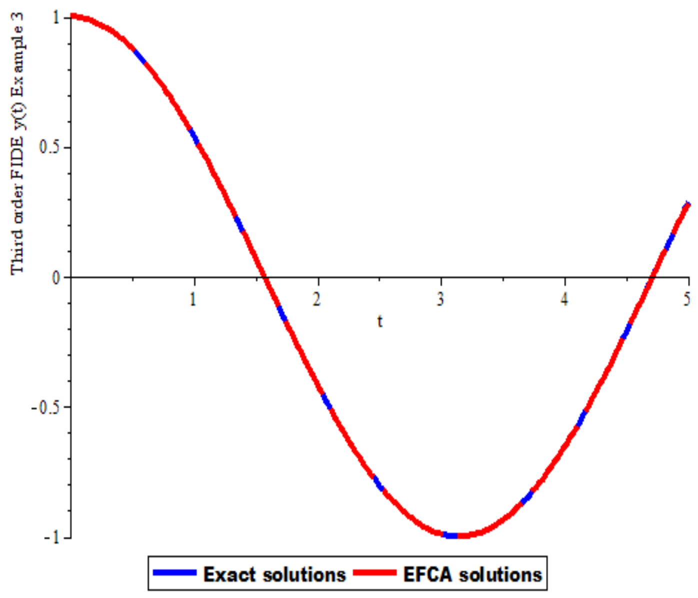

Fig. 5 - Depict comparison between exact and exponentially fitted collocation algorithm solutions Example 3

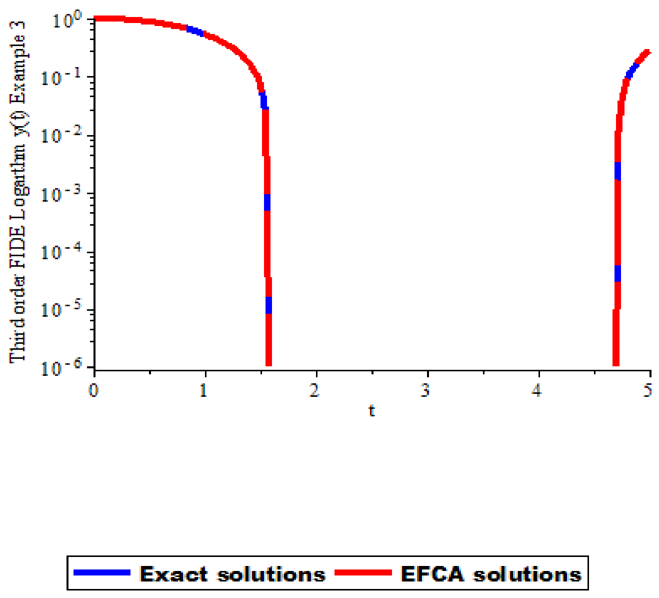

Fig. 6 - Depict logarithm comparison between exact and exponentially fitted collocation algorithm solution on interval $0 \leq t \leq 5$ for example 3

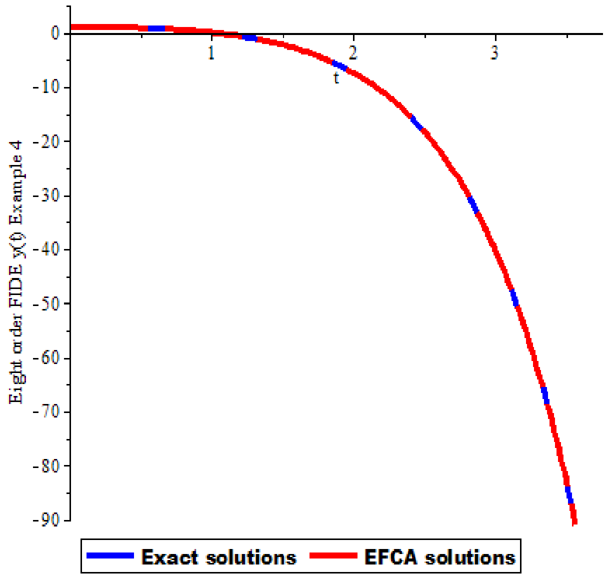

Fig. 7 - Depict comparison between exact and exponentially fitted collocation algorithm solutions on interval $0 \leq t \leq 5$ for example 4

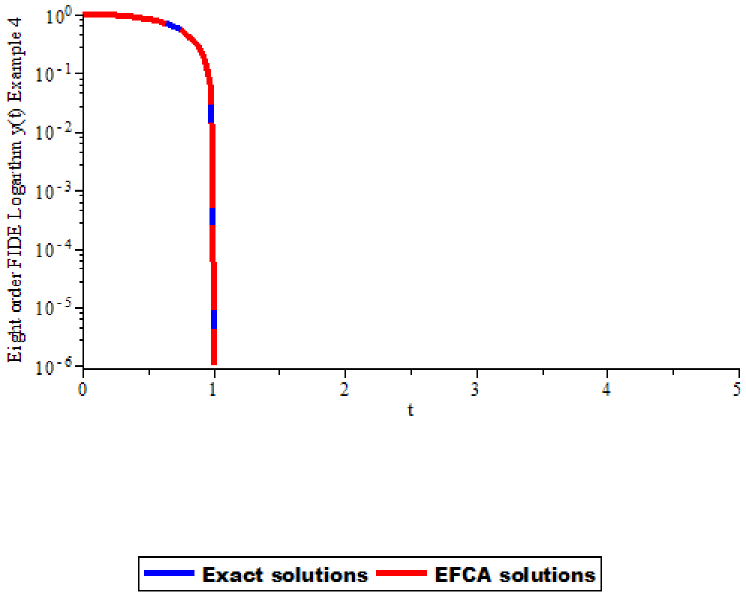

Fig. 8 - Depict logarithm comparison between exact and exponentially fitted collocation algorithm solutions on interval $0 \leq t \leq 5$ for example 4

5. Conclusion

In this paper, we have demonstrated the feasibility of the newly formulated exponentially collocation algorithm for the numerical solutions of the nth-order Fredholm's integrodifferential equations. Comparing the exact solutions and some of the techniques available in the literatures with EFCA indicated that the presented algorithm is efficient and simple. Furthermore, approximate solutions are obtained and presented in graphical form (see Tables 1, 2, 3, 4) and Figs. (1, 2, 3, 4, 5, 6, 7, 8) which demonstrated that the proposed algorithm gives solutions that comprisable with exact solutions and some existing methods available in literatures. All computations work was performed using the MAPLE 18 software package.

Acknowledgement

The author would like to thank the Aliko Dangote University of Science and Technology Wudil and Landmark University for their support on this study.

References

[1] Jerri, A.J. (1999). Introduction to integral equations with applications, Seconded, *Wiley-Interscience*, pp1-10.

[2] Golbabai, A and Javidi, M. (2007). Application of He's homotopy perturbation method for nth-order integrodifferential equations, *Appl. Math. Comput.* 190, pp.1409-1416.

[3] Taiwo, O.A and Ishola, C.Y. (2009). Collocation approximation methods for the numerical solution of integral equations, *International Journal Pure and Applied Science,* 2, pp.29-37.

[4] Kurt, N, Sezer, M. (2008). Polynomial solution of high-order linear Fredholm integrodifferential equations with constant coefficients, *Journal of the Franklin Institute*, 345, pp839-850.

[5] Avidi, M., Golbabai, A. (2007). Modified homotopy perturbation method for solving non-linear Fredholm integral equations, *Chaos, Solutions and Fractals*, 09. 026.

[6] Heidari, S.C.S and Parandin, N. (2013). A numerical method for solving linear Fredholm Integrodifferential equations, *Appl. Math. Comput.* vol. 3, pp.192-195.

[7] Aruchunan, E. (2009). Numerical solution of first-order linear Fredholm integrodifferential equations using Conjugate gradient method, *ISG 1-2009*, pp. 11-13.

[8] Fathy, M, El-gamel, M.M, El-azab, M.S. (2014). Legendre Galerkin method for the linear Fredholm Integrodifferential Equations, *Appl. Math. Comput.* vol. 243 pp. 789-800.

[9] Hosseini, S,M, Shahmorad, M.S. (2003). Tau numerical solution of Fredholm integrodifferential equations with arbitrary polynomial bases, *Appl. Math. Comput.* vol. 27, pp. 145-154.

[10] Aruchunan, E, Sulaiman, J. (2016). Numerical solution of second-order linear Fredholm Integrodifferential equation using generalized minimal residual method, *American Journal of Applied Sciences* 7 (6) pp. 780-783.

[11] Falade, K. I and Mustaphar, M. (2020). Direct computational algorithm for solving systems of Fredholm integrodifferential equations, *International Journal of Engineering Applied Sciences and Technology.* Vol. 5, Issue 1, ISSN No. 2455-2143, pp. 30-35.

[12] Safdari, H and Aghdam. Y.E. (2015). Numerical solution of second-order linear Fredholm Integrodifferential equations by trigonometric scaling functions, *Open Journal of Applied Sciences* 5. pp. 135-144.

[13] Al-rammahi, A. (2014). Power series form for solving linear Fredholm integral equations of second order via Banach fixed point theorem, vol. 8 (no.1), pp. 177-179.

[14] Tiamiyu, A, T, Falade, K.I and Abubakar, A,S. (2021). Computational assessment of external force acting on beam elastic foundation, *Pamukkale University Journal of Engineering Sciences*, 28(3), pp. 401-403.

[15] Alao, S, Akinboro, F.S and Akinpelu, F.O, R.A. Oderinu. (2014). Numerical solution of integrodifferential equation using anomia decomposition and variational iteration methods, *IOSR Journal of Mathematics,* volume 10, issue 4, pp18-22.

[16] Falade K.I, Abubakar A.S. (2019). Solving Bessel differential equation of order zero using exponentially fitted collocation approximation method, *Research Journal of Mathematical and Statistical Sciences* Vol. 7(2), 21-26.

[17] Rashidinia, J and Tahmasebi, A. (2013). Approximate solution of linear integrodifferential equations by using modified Taylor expansion method, *World Journal of Modelling and Simulation* Vol. 9, No. 4, pp. 289-301.

[18] Huang, Y, and Li, X. (2010). Approximate solution of a class of linear integrodifferential equations by Taylor expansion method, *International Journal of Computer Mathematics*, 87(6): pp. 1277–1288.

[19] Shang, X and Han, D. (2010). Application of the variational iteration method for solving nth-order Integrodifferential equations, *Journal of Computational and Applied Mathematics* 234, pp.1442–1447.

Appendix:
Example 4

Step 1:

```
restart : with(plots); Digits := 20;
[animate, animate3d, animatecurve, arrow, changecoords, complexplot, complexplot3d,
    conformal, conformal3d, contourplot, contourplot3d, coordplot, coordplot3d, densityplot,
    display, dualaxisplot, fieldplot, fieldplot3d, gradplot, gradplot3d, implicitplot,
    implicitplot3d, inequal, interactive, interactiveparams, intersectplot, listcontplot,
    listcontplot3d, listdensityplot, listplot, listplot3d, loglogplot, logplot, matrixplot, multiple,
    odeplot, pareto, plotcompare, pointplot, pointplot3d, polarplot, polygonplot, polygonplot3d,
    polyhedra_supported, polyhedraplot, rootlocus, semilogplot, setcolors, setoptions,
    setoptions3d, spacecurve, sparsematrixplot, surfdata, textplot, textplot3d, tubeplot]

Digits := 20
H := 15 :


 for l from 0 to H do
  g[l] := Int( x^2·(l·t^(l − 1)), t = 0 ..1) :

   a[l] := value( g[l]) :
  end do:
```

Step 2:

```
 basisfunction := x^i :
  v := sum(y[i]·basisfunction, i = 0 ..H) :
 z[0] := eval(v, x = 0) :
 z[1] := eval( diff(v, x), x = 0) :
 z[2] := eval( diff(v, x$2), x = 0) :
 z[3] := eval( diff(v, x$3), x = 0) :
 z[4] := eval( diff(v, x$4), x = 0) :
 z[5] := eval( diff(v, x$5), x = 0) :
 z[6] := eval( diff(v, x$6), x = 0) :
 z[7] := eval( diff(v, x$7), x = 0) :
```

Step 3:

```
T[0] := 1 :
T[1] := 2·x − 1 :
for n from 1 to H do T[n + 1] := simplify(2·(2·x − 1)·T[n] − T[n − 1]) : end do:
Y := x^N : n := 8 :
for i from 0 to H doP := (diff(Y, x$n) − Y)·y[i] : A[i] := eval(P, N = i) : od:
X := sum(A[j], j = 0 ..H) - sum(T[H - k]·τ[k + 1], k = 0 ..n − 1) =-8·e^x + x^2 :

for m from 1 to H + 1 do eqn[m] := eval(L, x = m/(H + 2)) : end do :
```

```
eqn[ H + 2] := z[0] + e^0·τ[n] = 1 :
eqn[ H + 3] := z[1] + e^0·τ[n] = 0 :
eqn[ H + 4] := z[2] + e^0·τ[n] =−1 :
eqn[ H + 5] := z[3] + e^0·τ[n] =−2 :
eqn[ H + 6] := z[4] + e^0·τ[n] =−3 :
eqn[ H + 7] := z[5] + e^0·τ[n] =−4 :
eqn[ H + 8] := z[6] + e^0·τ[n] =−5 :
eqn[ H + 9] := z[7] + e^0·τ[n] =−6 :
```

Step 4:

```
sys := seq(eqn[i], i = 1 ..H + n + 1) :
sol := evalf(solve({sys})) :
 Q := eval([seq(y[i], i = 0 ..H), τ[n]], sol) :
  for r from 0 to H do
  y[r] := Q[r + 1] :
  end do:
 τ[n] := Q[H + 2] :
```

Step 5:

```
sol1 := sum(y[j]·t^j, j = 0 ..H) + τ[n]·e^t :
for x from 0 by 0.1 to 1 doEFCA[x] := evalf(eval(sol1, t = x)) end do:
Exact := evalf(1 − t)·e^t :
EFCA := (sol1) :
 y := sol1 :
 for n from 0 by 0.1 to 1 do
 E[n] := evalf(eval(Exact, t = n));
  Y[n] := evalf(eval(y, t = n));
  Error[n] := abs(E[n]−Y[n]);
  end do;
```

$E_0 := 1.$

$Y_0 := 1.0000000000000000099$

$Error_0 := 9.9\ 10^{-18}$

$E_{0.1} := 0.99465382626808286232$

$Y_{0.1} := 0.99465382626808287299$

$Error_{0.1} := 1.067\ 10^{-17}$

$E_{0.2} := 0.97712220652813586712$

$Y_{0.2} := 0.97712220652813587991$

$Error_{0.2} := 1.279\ 10^{-17}$

$E_{0.3} := 0.94490116530320217280$

$Y_{0.3} := 0.94490116530320218025$

$Error_{0.3} := 7.45\ 10^{-18}$

$E_{0.4} := 0.89509481858476219068$

$Y_{0.4} := 0.89509481858476212634$

$Error_{0.4} := 6.434\ 10^{-17}$

$E_{0.5} := 0.82436063535006407340$

$Y_{0.5} := 0.82436063535006368284$

$Error_{0.5} := 3.9056\ 10^{-16}$

```
E[0.6] := 0.72884752015620358996
Y[0.6] := 0.72884752015620217762
Error[0.6] := 1.41234 10^-15
E[0.7] := 0.60412581224114295648
Y[0.7] := 0.60412581224113885570
Error[0.7] := 4.10078 10^-15
E[0.8] := 0.44510818569849352092
Y[0.8] := 0.44510818569848296664
Error[0.8] := 1.055428 10^-14
E[0.9] := 0.24596031111569496638
Y[0.9] := 0.24596031111566987981
Error[0.9] := 2.508657 10^-14
E[1.0] := 0.
Y[1.0] := -5.6102412445748 10^-14
Error[1.0] := 5.6102412445748 10^-14
plot([Exact, sol1], t = 0 ..5, color = [blue, red], axes = BOXED, title = " FIDE", labels = ["t",
      "  Eight order FIDE y(t) Example 4 "], labeldirections = [HORIZONTAL,
VERTICAL]);
 logplot([Exact, sol1], t = 0 ..5, color = [blue, red], axes = BOXED, title = " FIDE", labels
       = ["t"," Eight order FIDE Logarthm y(t) Example 4 "], labeldirections
       = [HORIZONTAL,
VERTICAL])
```