Bol. Soc. Paran. Mat.
©SPM – E-ISSN-2175-1188
SPM: www.spm.uem.br/bspm

(3s.) **v. 2026 (44) 16** : 1–25.
ISSN-0037-8712
doi:10.5269/bspm.81705

Numerical Approach for Solving Fredholm Integro-Fractional Differential Equations with Numerous Constant Delays Based on Newton-Cotes Methods

Razaw Salam Rasul and Shazad Shawki Ahmed

ABSTRACT: This article presents strong algorithms for the numerical solution of linear Fredholm integro-fractional differential equations of constant delay type (FIFDEs-Delays) with variable coefficients under boundary conditions with historical functions and the fractional order less than or equal to one in the Caputo sense. With the aid of a finite difference approximation for Caputo derivative utilization collocation points, the approach is based on the Newton-Cotes methods. Our method transforms the original FIFDEs-Dealays equation into a set of algebraic equations with operational matrices for treatments, which are then numerically solved. This approach covers the integral component of the equation by utilizing the power of Newton-Cotes quadrature rules, such as the trapezoidal or Simpson's 1/3 rules, which makes evaluation easy.Furthermore, a number of numerical examples are provided to assess the effectiveness of proposed approach, proving its accuracy and practicality. General programs are written in Python.

Keywords: Fredholm integral equation, Caputo fractional derivatives, constant delay types, Newton–Cotes methods, forward difference approximation, operational matrix.

Contents

1. Introduction

The class of equations investigated in this study are linear Fredholm integro-fractional differential equations of constant delay type (FIDDEs-Delays) with variable coefficients and all arbitrary orders located inside the interval (0,1) in the Caputo definition. For all $a \leq x \leq b$, their general form is as follows:

$$ {}_a^C D_x^{\beta} y(x) + \sum_{\ell=1}^{n} p_\ell(x) {}_a^C D_x^{\alpha_\ell} y(x-\tau_\ell) + p_0(x) y(x-\tau_0) = f(x) + \sum_{j=0}^{m} \lambda_j \int_a^b k_j(x,t) y(t-\tau_j^*) dt \qquad (1.1) $$

Together with boundary condition and historical function:

$$ \underbrace{\{g_{00} y(a) + h_{00} y(b)\} = \vartheta_0}_{\text{Boundary conditions}} \qquad \underbrace{y(x) = \varphi(x)}_{\text{Historical function}} \qquad (1.2) $$

2020 *Mathematics Subject Classification:* 26A33, 65R20, 34K28.
Submitted February 02, 2026. Published May 02, 2026.

for all, $x \in [\bar{a}, a]$ such that $\bar{a} = a - max\{\tau_\ell, \tau_j^* : j = \overline{0:m} \text{ and } \ell = \overline{0:n}\}$ where $y(x)$ is unknown function which is the solution of equation (1.1), the function $k_j : S \times \mathbb{R} \to \mathbb{R}, (S = \{(x,t) : a \le x < t \le b\})$, $j = 0, 1, 2, \ldots, m$ and $f, p_\ell : [a, b] \to \mathbb{R}$; for all $\ell = 0, 1, \ldots, n$ continuous function. In addition $\alpha_\ell \in \mathbb{R}^+$ for all $w^\beta - 1 < \beta \le w^\beta, w^\beta = \lceil \beta \rceil; w_\ell^\alpha - 1 < \alpha_\ell \le w_\ell^\alpha, w_\ell^\alpha = \lceil \alpha_\ell \rceil$ for all $\ell = 1, 2, \ldots, n$ with property $1 > \beta > 0$, $1 > \alpha_n > \alpha_{n-1} > \cdots > \alpha_1 > 0$ and positive constant time-lags (delay), τ_j^*, τ_ℓ for all $j = 0, 1, 2, \ldots, m$ and $\ell = 0, 1, 2, \ldots, n$, while $m \in \mathbb{Z}^+ \cup \{0\}$ and $n \in \mathbb{Z}^+$.
This study aims to solve FIFDEs-Delays (1.1) with boundary conditions and historical function (1.2) using the quadrature rule approach in matrix form. This numerical method for estimating a solution at a certain position can be used to approximate the value of an unknown function at that position.

The most widely used methods are those that fit into the Newton-Cotes classification, like the trapezoidal rule, Simpson's rule and closed. Additionally, these methods are frequently employed in applied mathematics to find numerical solutions for definite integrals that are not analytically solvable [1]. Additionally, it is the foundation of all numerical methods for solving integral equations [2].

Every numerical method for solving integral equations starts with the quadrature techniques: Al-Rawi [3] employed quadrature methods to solve first-kind IEs on convolution types, Al-Nasir [4] used them to solve second-kind VIEs, and Jafar and Mahdi [5] utilized modified trapezoid quadrature methods to solve second-kind FIEs, notwithstanding Rahbar and Hashemizadeh [6]. However, for the second sort Nonlinear Fredholm Integral equation, Emamzadeh and Kajani [7] employed the quadrature technique. Additionally, it was used to solve singular VIE by Samuel and Robert [8]. Additionally, Saadati and Shakeri [9] and Al-Timeme [10] use quadrature techniques to solve linear IDEs, however, Burhan F. Jumah [11] used them to solve non-linear Volterra integral equations. Additionally, Shazad and Shokhan [12] used it to solve the most generic linear VIFDEs numerically.

The paper is organized as follows:

- Section 2 contains the FC's essential definitions, fundamental preliminary information, and other elements needed for the purpose of this research.
- Section 3 discusses quadrature rules and how to handle Fredholm delays.
- In section 4, the trapezoidal and Simpson's 1/3 methods are used to develop the Newton-Cots formulas for solving linear Fredholm integro-fractional differential equations of constant delay type (FIFDEs-Delays) with variable coefficients.
- Our results are illustrated throughout the examples in section 5.
- Finally, section 6 includes a discussion for this method.

2. Preliminaries and Propositions

The definitions and properties of the FCs, the Riemann-Liouville (R-L) integrals and derivative, and the Caputo derivative, as well as information about their relationship and particular properties, are provided in this section. A list of several common definitions, lemmas, and prepositions used in this article is also provided.

Definition 2.1 ([13], [14]) The defined space $C_\sigma[a, b], \sigma \in \mathbb{R}$ contains all σ-weighted contentious real-valued functions,i.e., including y defined on $[a, b]$, where the function $y(x)$ can be written as $(x - a)^s \hat{y}(x)$ if a real number $s > \sigma$ exists and $\hat{y} \in C[a, b]$. $y^{(\upsilon)} \in C_\sigma[a, b]$, $\upsilon \in \mathbb{N}_0$, if and only if it is in the space $C_\sigma^\upsilon[a, b]$.

Definition 2.2 ([15],[16]) The left-sided Riemann-Liouville fractional integral of arbitrary order $\varrho > 0$ for a function $y \in C_\upsilon[a, b], \upsilon \ge -1$ is defined as

$$ {}_aJ_x^\varrho y(x) = \frac{1}{\Gamma(\varrho)} \int_a^x (x - \zeta)^{\varrho - 1} y(\zeta) d\zeta, \quad \varrho \in \mathbb{R}^+, \quad a \le x \le b $$

For $\varrho = 0$, we get the Riemann-Liouville identity operator, ${}_aJ_x^0 y(x) = y(x)$, where $\Gamma(.)$ represents the gamma function.

Definition 2.3 ([14], [17]) For a function $y \in C_{-1}^{\lceil \varrho \rceil}[a,b]$, the operator ${}_a^R D_x^\varrho y(x)$ of order $\varrho \geq 0$ and $x > a$, which are described as:

$$ {}_a^R D_x^\varrho y(x) = D_x^{\lceil \varrho \rceil} {}_a J_t^{\lceil \varrho \rceil - \varrho} y(x) $$

is known as the order ϱ Riemann-Liouville fractional derivative. The ceiling function is denoted by $\lceil . \rceil$, and the Riemann-Liouville identity derivative operator for $\varrho = 0$ is ${}_a^R D_x^0 y(x) = y(x)$.

Definition 2.4 ([16], [17]) For a function $y \in C_{-1}^{\lceil \varrho \rceil}[a,b]$, the operator ${}_a^C D_x^\varrho y(x)$ of order $\varrho \geq 0$ and $x > a$, which are described as:

$$ {}_a^C D_x^\varrho y(x) = {}_a J_x^{\lceil \varrho \rceil - \varrho} [D_x^{\lceil \varrho \rceil} y(x)] = \frac{1}{\Gamma(\lceil \varrho \rceil - \varrho)} \int_a^x (x-\zeta)^{\lceil \varrho \rceil - \varrho - 1} \frac{d^{\lceil \varrho \rceil} y(\zeta)}{d\zeta^{\lceil \varrho \rceil}} d\zeta $$

is known as the order ϱ Caputo fractional differential operator. The Caputo identity derivative operator, ${}_a^C D_x^0 y(x) = y(x)$, is the Caputo way derivative for $\varrho = 0$. The following attributes are true:

(i) ${}_a J_x^{\varrho_1} {}_a J_x^{\varrho_2} y(x) = {}_a J_x^{\varrho_1 + \varrho_2} y(x) = {}_a J_x^{\varrho_2} {}_a J_x^{\varrho_1} y(x)$ for all $\varrho_1, \varrho_2 \geq 0$.

(ii) ${}_a^R D_x^\varrho \mathcal{C} = \mathcal{C} \frac{(x-a)^{-\varrho}}{\Gamma(1-\varrho)}$ and ${}_a^C D_x^\varrho \mathcal{C} = 0; \mathcal{C}$ is any constant; $(\varrho \geq 0,\ \varrho \notin \mathbb{N})$.

(iii) ${}_a^C D_x^\varrho {}_a J_x^\varrho y(x) = y(x)$, for $\lceil \varrho \rceil - 1 < \varrho \leq \lceil \varrho \rceil$, $a \leq x \leq b$.

(iv) ${}_a J_x^\varrho {}_a^C D_x^\varrho y(x) = y(x) - \sum_{\kappa=0}^{\lceil \varrho \rceil - 1} \frac{y^\kappa(x=a)}{\kappa!} (x-a)^\kappa$, for $\lceil \varrho \rceil - 1 < \varrho \leq \lceil \varrho \rceil$.

(vi) ${}_a^C D_x^\varrho\, y(x) = {}_a^R D_x^\varrho \left[y(x) - T_{\lceil \varrho \rceil - 1}[y; a] \right]$, and $T_{\lceil \varrho \rceil - 1}[y; a]$ denotes the Taylor polynomial of degree $\lceil \rho \rceil - 1$ for the function y, centered at a.

Lemma 2.1 *([18]) If we have $x > a$ and for $y(x) = (x-a)^s$, for $s > -1$ and $\varrho \in \mathbb{R}^+$ then the following statement hold:*

$$ {}_a J_x^\varrho y(x) = \frac{\Gamma(s+1)}{\Gamma(s+\varrho+1)} (x-a)^{s+\varrho} $$

Lemma 2.2 *([14]) The function $y(x) = (x-a)^s$, for $s \geq 0$, has a Caputo derivative of order $\varrho \geq 0$, which is formed as: For $s \in \{0, 1, 2, \cdots, \lceil \varrho \rceil - 1\}$: ${}_a^C D_x^\varrho\, y(x) = 0$ and for $s \in \mathbb{N}$ and $s \geq \lceil \varrho \rceil$ or $s \notin \mathbb{N}$ and $s > \lceil \varrho \rceil - 1$:*

$$ {}_a^C D_x^\rho y(x) = \frac{\Gamma(s+1)}{\Gamma(s-\varrho+1)} (x-a)^{s-\rho} $$

Proposition 2.1 *([14]) The Caputo derivative's finite forward difference approximation for $\beta \in (0,1]$ at certain locations $x = x_{r+1}$; $r = 0, 1, \ldots, N-1$ and $h = \frac{b-a}{N}$, $(N \in \mathbb{Z}^+)$ is constructed as follows:*

$$ {}_a^C D_x^\varrho\, y(x)\Big|_{x=x_{r+1}} = \frac{h^{-\varrho}}{\Gamma(2-\varrho)} \sum_{d=0}^{r} \left[y(x_{r-d+1}) - y(x_{r-d}) \right] b_d^\varrho \tag{2.1} $$

where $b_d^\varrho = (d+1)^{1-\varrho} - d^{1-\varrho}$.

Proposition 2.2 *([19], [20]) Let's say $\varrho \geq 0, \varrho \notin \mathbb{N}$. Additionally, assume that $y \in C_{-1}^{\lceil \rho \rceil}[a,b]$. Then $\left[{}_a^C D_x^\varrho y(x) \right]_{x=x_0} = 0$, that is $\lim_{x \to a} \left[{}_a^C D_x^\varrho y(x) \right] = 0$, and the Caputo fractional derivative ${}_a^C D_x^\varrho y(x)$ is continuous on $[a,b]$.*

2.1. Observation (On Parameter Estimation Procedure)

We will use the notation $[x]_h = \left[\!\left[\frac{x}{h}\right]\!\right]h$ where $[\![*]\!]$ is greatest integer part function. It is easy to see that $x-h<[x]_h\leqslant x$, for all $x\in\mathbb{R}$ and $h>0$ therefore $[x]_h$ approach to x as h tends to 0^+ uniformly in x . Here, all arbitrary orders are less than or equal for one, so we have only one historical function that is: $y(x)=\varphi(x)$, for all $x\in[\bar{a},a]$ where $\varphi\in C([\bar{a},a],\mathbb{R})$ and $\bar{a}=a-\tau$ with given positive constants time delays τ. Define h as the grid step, $x_{r+1}=a+(r+1)h,$ for all $r=0,1,2,\ldots,N-1$, and $h=\frac{b-a}{N}$. Now,

1. To calculate $y\,(x-\tau)$ at $x=x_{r+1}$ and $r=0,1,...,N-1$. Thus we have:

$$y(x-\tau)\Big|_{x=x_{r+1}}=y(x_{r+1}-\tau)=y([x_{r+1}-\tau]_h)$$

$$=y\Big(\big[\!\big[\frac{x_{r+1}}{h}-\frac{\tau}{h}\big]\!\big]h\Big)=y\Big(\Big[\!\Big[\frac{x_{r+1}}{h}-\frac{[\tau]_h}{h}\Big]\!\Big]h\Big)=y\Big(\Big[\!\Big[\frac{x_{r+1}}{h}-\frac{1}{h}\big[\!\big[\frac{\tau}{h}\big]\!\big]h\Big]\!\Big]h\Big)=y\Big(\Big[\!\Big[\frac{x_{r+1}}{h}-\big[\!\big[\frac{\tau}{h}\big]\!\big]\Big]\!\Big]h\Big)$$

putting $\overline{\upsilon}=\left[\!\left[\frac{\tau}{h}\right]\!\right]$ so on the other hand, we obtain:-

$$y(x_{r+1}-\tau)=y\Big(\Big[\!\Big[\frac{x_{r+1}}{h}-\overline{\upsilon}\Big]\!\Big]h\Big)=y\Big(\Big[\!\Big[\frac{a+(r+1)h}{h}-\overline{\upsilon}\Big]\!\Big]h\Big)=y\Big(\Big[\!\Big[\frac{a+(r+1-\overline{\upsilon})h}{h}\Big]\!\Big]h\Big)$$

$$=y\Big(\Big[\!\Big[\frac{x_{r+1-\overline{\upsilon}}}{h}\Big]\!\Big]h\Big)=y([x_{r+1-\overline{\upsilon}}]_h)=y(x_{r+1-\overline{\upsilon}})$$

Thus:

$$\begin{gathered}y(x_{r+1}-\tau)=y(x_{r+1-\overline{\upsilon}});\\ \text{and note that}: y(x_k)=\varphi(x_k)\ \text{for}\ \ k=-\overline{\upsilon},-\overline{\upsilon}+1,-\overline{\upsilon}+2,...,-1.\end{gathered}\tag{2.2}$$

Also, for calculate the function $y(x-\tau)$ at points $x=x_{r-j}$ where τ is the real positive constant time delay and $r=0,1,...,N-1$ with $j=0,1,...,r$ and by same procedure before we can formulated as follows:

$$\begin{gathered}y(x_{r-j}-\tau)=y(x_{r-j-\overline{\upsilon}})\\ \text{and note that}: y(x_k)=\varphi(x_k)\ \text{for}\ \ k=-\overline{\upsilon},-\overline{\upsilon}+1,-\overline{\upsilon}+2,...,-1.\end{gathered}\tag{2.3}$$

Furthermore,

$$\begin{gathered}y(x_{r-j+1}-\tau)=y(x_{r-j-\overline{\upsilon}+1})\\ \text{and note that}: y(x_k)=\varphi(x_k)\ \text{for}\ \ k=-\overline{\upsilon},-\overline{\upsilon}+1,-\overline{\upsilon}+2,...,-1.\end{gathered}\tag{2.4}$$

2. Now, to display the approximation point along with the historical function, as $h\to 0\,;\ y\cong\widetilde{y}$. Since, For all $k\in\mathbb{Z}$:

$$\widetilde{y}(x_k)=y_h(x_k)=\varphi(x_k)\,;\ k=-\overline{\upsilon},-\overline{\upsilon}+1,-\overline{\upsilon}+2,...,-1\tag{2.5}$$

$$\widetilde{y}(x_k-\tau)=y_h(x_k-\tau)=\widetilde{y}_{k-\overline{\upsilon}}\,;\ k=0,1,\ldots,N\tag{2.6}$$

for all positive real constant time-delay τ be $\overline{\upsilon}=\left[\!\left[\frac{\tau}{h}\right]\!\right]$. Thus, yields

$$y(x_k-\tau)=\begin{cases}\varphi(x_k-\tau) & if\quad x_k-\tau<a\\ \widetilde{y}(x_k-\tau) & if\quad x_k-\tau\geq a\end{cases}=\begin{cases}\varphi(x_k-\tau) & if\quad x_k-\tau<a\\ \widetilde{y}_{k-\overline{\upsilon}} & if\quad x_k-\tau\geq a\end{cases}\tag{2.7}$$

Suppose that:

$$\delta_k=\begin{cases}1 & x_k-\tau<a\\ 0 & o.w.\end{cases}\quad\text{and}\quad\overline{\delta}_k=\begin{cases}1 & x_k-\tau\geq a\\ 0 & o.w.\end{cases}\tag{2.8}$$

So, apply equation (2.8) to (2.7), obtained the following expression:

$$y(x_k-\tau)=\delta_k\,\varphi(x_k-\tau)+\overline{\delta}_k\widetilde{y}(x_k-\tau)=\delta_k\,\varphi(x_k-\tau)+\overline{\delta}_k\,\widetilde{y}_{k-\overline{\upsilon}}\tag{2.9}$$

Proposition 2.3 *(New) The forward finite difference approximation of the Caputo derivative with a constant-time delay ($\tau \in \mathbb{R}^+$) for arbitrary order $\alpha \in (0,1)$ at collocation points $x = x_{r+1}$; $r = 0,1,\ldots,N-1$, where $h = \frac{b-a}{N}$, ($N \in \mathbb{Z}^+$). The following differences construct:*

$$ {}_a^C D_x^\alpha y(x-\tau)\big|_{x=x_{r+1}} = $$
$$ \tfrac{h^{-\alpha}}{\Gamma(2-\alpha)} \sum_{j=0}^{r} \left\{ \left[\delta_{r-j+1}\, \varphi(x_{r-j+1}-\tau) + \overline{\delta}_{r-j+1}\, \widetilde{y}_{r-j+1-\overline{v}} \right] - \left[\delta_{r-j}\, \varphi(x_{r-j}-\tau) + \overline{\delta}_{r-j}\, \widetilde{y}_{r-j-\overline{v}} \right] \right\} b_j^\alpha \tag{2.10} $$

where $y(x) = \varphi(x)$, for all $x \in [\bar{a}, a]$ such that $\bar{a} = a - \tau$; $b_j^\alpha = (1+j)^{1-\alpha} - j^{1-\alpha}$ and $\overline{v} = \left[\!\left[\frac{\tau}{h} \right]\!\right]$. Also, define

$$ \delta_{r-j+1} = \begin{cases} 1 & if \quad x_{r-j+1} - \tau < a \\ 0 & if \quad o.w. \end{cases} \qquad \overline{\delta}_{r-j+1} = \begin{cases} 1 & if \quad x_{r-j+1} - \tau \geq a \\ 0 & if \quad o.w. \end{cases} $$
$$ \delta_{r-j} = \begin{cases} 1 & if \quad x_{r-j} - \tau < a \\ 0 & if \quad o.w. \end{cases} \qquad \overline{\delta}_{r-j} = \begin{cases} 1 & if \quad x_{r-j} - \tau \geq a \\ 0 & if \quad o.w. \end{cases} $$

proof: Recall the definition of α-Caputo fractional derivative, ${}_a^C D_x^\alpha$, in$(0,1)$ for the time delay function $y(x-\tau)$ and using first-order forward difference approximation [22] for each x, that is:

$$ {}_a^C D_x^\alpha y(x-\tau)\big|_{x=x_{r+1}} = \frac{1}{\Gamma(1-\alpha)} \int_a^{x_{r+1}} \frac{\partial y(s-\tau)/\partial s}{(x_{r+1}-s)^\alpha} ds = \frac{1}{\Gamma(1-\alpha)} \sum_{l=0}^{r} \int_{x_l = a+lh}^{x_{l+1}=a+(l+1)h} \frac{\partial y(s-\tau)/\partial s}{(x_{r+1}-s)^\alpha} ds $$

since the finite difference first-order forward method with τ-delay time:

$$ \frac{\partial y(s-\tau)}{\partial s} \cong \frac{y(s_{i+1}-\tau) - y(s_i-\tau)}{h} $$

Thus:

$$ {}_a^C D_x^\alpha y(x_{r+1}-\tau) \cong \frac{1}{\Gamma(1-\alpha)} \sum_{l=0}^{r} \frac{y(s_{l+1}-\tau) - y(s_l-\tau)}{h} \int_{a+lh}^{a+(l+1)h} \frac{ds}{(x_{r+1}-s)^\alpha} $$

Suppose that $\xi = x_{r+1} - s$, so $d\xi = -ds$. The equation became:

$$ {}_a^C D_x^\alpha y(x_{r+1}-\tau) = \frac{1}{\Gamma(1-\alpha)} \sum_{l=0}^{r} \frac{y(s_{l+1}-\tau) - y(s_l-\tau)}{h} \int_{(r-l)h}^{(r-l+1)h} \frac{d\xi}{\xi^\alpha} $$

let $j = r - l$ and $l = r - j$, so we get:

$$ {}_a^C D_x^\alpha y(x_{r+1}-\tau) = \frac{1}{\Gamma(1-\alpha)} \sum_{j=r}^{0} \frac{y(s_{r-j+1}-\tau) - y(s_{r-j}-\tau)}{h} \int_{jh}^{(j+1)h} \xi^{-\alpha} d\xi $$

$$ = \frac{1}{(1-\alpha)\Gamma(1-\alpha)} \sum_{j=0}^{r} \frac{y(s_{r-j+1}-\tau) - y(s_{r-j}-\tau)}{h} \left[(j+1)^{1-\alpha} h^{1-\alpha} - j^{1-\alpha} h^{1-\alpha} \right] $$

Since by property of Gamma function $(1-\alpha)\Gamma(1-\alpha) = \Gamma(2-\alpha)$ and simplify we have

$$ {}_a^C D_x^\alpha y(x_{r+1}-\tau) = \tfrac{h^{-\alpha}}{\Gamma(2-\alpha)} \sum_{j=0}^{r} \{ y(s_{r-j+1}-\tau) - y(s_{r-j}-\tau) \} b_j^\alpha \quad \text{where} \quad b_j^\alpha = (1+j)^{1-\alpha} - j^{1-\alpha}. \tag{2.11} $$

So, by using equation(2.9) with including equations (2.3) and (2.4) in to the equation (2.11), we obtain for each $r = 0, 1, ..., N-1$;

$$ {}_a^C D_x^\alpha y(x-\tau)\big|_{x=x_{r+1}} = $$
$$ \tfrac{h^{-\alpha}}{\Gamma(2-\alpha)} \sum_{j=0}^{r} \left\{ \left[\delta_{r-j+1}\, \varphi(x_{r-j+1}-\tau) + \overline{\delta}_{r-j+1}\, \widetilde{y}_{r-j+1-\overline{v}} \right] - \left[\delta_{r-j}\, \varphi(x_{r-j}-\tau) + \overline{\delta}_{r-j}\, \widetilde{y}_{r-j-\overline{v}} \right] \right\} b_j^\alpha \tag{2.12} $$

where $\overline{\upsilon} = \left[\!\left[\frac{\tau}{h}\right]\!\right]$ with

$$\delta_{r-j+1} = \begin{cases} 1 & \text{if} \quad x_{r-j+1} - \tau < a \\ 0 & \text{if} \quad \text{o.w.} \end{cases} \qquad \overline{\delta}_{r-j+1} = \begin{cases} 1 & \text{if} \quad x_{r-j+1} - \tau \geq a \\ 0 & \text{if} \quad \text{o.w.} \end{cases}$$

and

$$\delta_{r-j} = \begin{cases} 1 & \text{if} \quad x_{r-j} - \tau < a \\ 0 & \text{if} \quad \text{o.w.} \end{cases} \qquad \overline{\delta}_{r-j} = \begin{cases} 1 & \text{if} \quad x_{r-j} - \tau \geq a \\ 0 & \text{if} \quad \text{o.w.} \end{cases} .$$

By same procedure in proposition (2.3) and using (2.2) we can conclude the following proposition:

Proposition 2.4 *Let's say $\varrho \geq 0, \varrho \notin \mathbb{N}$. Additionally, assume that $y \in C_{-1}^{\lceil \rho \rceil}[a,b]$. Then, for each real-positive constant time-delay τ, thus $\left[{}_a^C D_x^{\varrho} y(x-\tau)\right]_{x=x_0} = 0$, that is $\lim_{x \to a} \left[{}_a^C D_x^{\varrho} y(x-\tau)\right] = 0$, and the Caputo fractional derivative ${}_a^C D_x^{\varrho} y(x-\tau)$ is continuous on $[a,b]$.*

3. Newton–Cots Quadrature Methods

This section discusses quadrature rules(Trapezoidal and Simpson's 1/3 rules) and methods for handling Fredholm delays.

3.1. Trapezoidal Rule

The simplest numerical method for evaluating a definite integral is the trapezoidal rule, which is a special case of (unweighted) closed Newton-Cotes formula , and it is based on linear interpolation $y(x)$ at x_r and x_{r+1}, i.e., $y(x)$ approximated by straight line joining (x_r, y_r) and (x_{r+1}, y_{r+1}) , we divide the interval $[a,b]$ into N-subintervals $[x_r, x_{r+1}]$ of equal length $h = \frac{b-a}{N}$ where the sample points can be expressed as $x_r = x_0 + rh$, $(r = 0, 1, \ldots, N)$ with $b = x_0 + Nh$, then the trapezoidal rule's numerical integration of $y(x)$ over the interval $[a,b]$ can therefore be expressed as follows:([21], [22], [24])

$$\int_a^b y(x)dx \cong \frac{h}{2}\Big[y(a) + 2\sum_{s=1}^{N-1} y(x_s) + y(b)\Big] = \sum_{k=0}^{N} w(k) y(x_k) \tag{3.1}$$

where $w(k)$ is weights for trapezoidal rule, where $w(k) = \begin{cases} \frac{h}{2} & ; \quad k = 0 \quad or \quad k = N \\ h & ; \quad o.w. \end{cases}$, with global Error $-\frac{(b-a)}{12} h^2 y''(\theta)$, $a < \theta < b$.

As applicable, the formula can likewise be applied to approximate the definite integral with constant time-delay τ_j^* for each $j = 0, 1, \ldots, m$ by substituting $y(x)$ with $y(x - \tau_j^*)$ in the integrated before using formula (3.1). For each $j = 0, 1, \ldots, m$ the integral term in equation (1.1) is then represented as follows:

$$\int_a^b k_j(x,t) y\left(t - \tau_j^*\right) dt \cong \sum_{z_j=0}^{N} w(z_j) k_j(x, t_{z_j})\ y(t_{z_j} - \tau_j^*) \tag{3.2}$$

$$\text{where } w(z_j) = \begin{cases} \frac{h}{2} & ; \quad z_j = 0 \quad or \quad z_j = N \\ h & ; \quad o.w. \end{cases} .$$

Suppose that

$$y(t_{z_j} - \tau_j^*) = \begin{cases} \varphi(t_{z_j} - \tau_j^*) & if \quad t_{z_j} - \tau_j^* < a \\ \widetilde{y}(t_{z_j} - \tau_j^*) & if \quad t_{z_j} - \tau_j^* \geq a \end{cases} = \begin{cases} \varphi(t_{z_j} - \tau_j^*) & if \quad t_{z_j} - \tau_j^* < a \\ \widetilde{y}_{z_j - \overline{\upsilon}_j^*} & if \quad t_{z_j} - \tau_j^* \geq a \end{cases} \tag{3.3}$$

where, for each constant time-delay τ_j^* let $\overline{\upsilon}_j^*$ be $\left[\!\left[\frac{\tau_j^*}{h}\right]\!\right]$, for all $j = 0, 1, \ldots, m$. Now let

$${}^*\delta_{z_j}^{[j]} = \begin{cases} 1 & \text{if} \quad t_{z_j} - \tau_j^* < a \\ 0 & \text{if} \quad o.w \end{cases} \quad ; \quad {}^*\overline{\delta}_{z_j}^{[j]} = \begin{cases} 1 & \text{if} \quad t_{z_j} - \tau_j^* \geq a \\ 0 & \text{if} \quad o.w \end{cases} \tag{3.4}$$

Thus the equation(3.3) became

$$\begin{aligned} &y(t_{z_j}-\tau_j^*) = {}^*\delta_{z_j}^{[j]}\,\varphi(t_{z_j}-\tau_j^*) + {}^*\overline{\delta}_{z_j}^{[j]}\,\widetilde{y}(t_{z_j}-\tau_j^*) = {}^*\delta_{z_j}^{[j]}\,\varphi(t_{z_j}-\tau_j^*) + {}^*\overline{\delta}_{z_j}^{[j]}\,\widetilde{y}_{z_j-\overline{v}_j^*} \\ &\text{for all constant time-delay } \tau_j^* \text{ put } \overline{v}_j^* = \left[\!\left[\frac{\tau_j^*}{h}\right]\!\right], \quad \forall j = 0,1,...,m \end{aligned} \tag{3.5}$$

Using formula (3.5) in to the equation (3.2). So, for each $j = 0, 1, \ldots, m$ we get :

$$\begin{aligned} &\int_a^b k_j(x,t)y\left(t-\tau_j^*\right)dt \\ &= \sum_{z_j=0}^{N}\left\{w\left(z_j\right)k_j\left(x,t_{z_j}\right)\,{}^*\delta_{z_j}^{[j]}\,\varphi\left(t_{z_j}-\tau_j^*\right) + \; w\left(z_j\right)k_j\left(x,t_{z_j}\right)\,{}^*\overline{\delta}_{z_j}^{[j]}\,\tilde{y}_{z_j-\overline{v}_j^*}\right\} \end{aligned} \tag{3.6}$$

3.2. Simpson's 1/3 Rule

The Simpson's rule is another instance of (unweighted) closed Newton-Cotes. the most important rule for determining bounded integrals is this formula. We use parabolas to numerically approximate each section of the curve in this case. Points $x_r = x_0 + rh, (r = 0, 1, \ldots, N)$ and $b = x_0 + Nh$ can be used to divide the provided integration interval into N- subintervals of equal length $h = \frac{(b-a)}{N}$, $N \geq 2$ ([21], [23], [24]).

(i) If N -is even, then the numerical integration of $y(x)$ over $[a, b]$ by Simpson's rule can be expressed as:

$$\int_a^b y(x)dx = \frac{h}{3}\sum_{e=1}^{N/2}\left[y(x_{2e}) + 4y(x_{2e-2})\right] = \sum_{e=1}^{N/2}\sum_{l=0}^{2} w_l^s y(x_{2e-l}) \tag{3.7}$$

(ii) If N-is odd, We developed Simpson's rule as follows:

$$\int_a^b y(x)dx = \sum_{e=1}^{N/2}\sum_{l=0}^{2} w_l^s\, y(x_{2e-l}) + \sum_{l=0}^{1} w_l^t\, y(x_{N-l}) \tag{3.8}$$

The weights for Simpson's rule are w_l^s and w_l^t, where $w_0^s = w_2^s = \frac{h}{3}$, $w_1^s = \frac{4h}{3}$, and $w_0^t = w_1^t = \frac{h}{2}$; also, the set of points $x_r = a + rh(r = \overline{0:N})$ with global error $-\frac{(b-a)}{180}h^4y^{(4)}(\theta)$, $a < \theta < b$.

As applicable, the formula can likewise be applied to approximate the definite integral with constant time-delay τ_j^* for each $j = 0, 1, \ldots, m$ by substituting $y(x)$ with $y(x - \tau_j^*)$ in the integrated before using formulas (3.7 and 3.8). For each $j = 0, 1, \ldots, m$ the integral term in equation (1.1) is then represented as follows:

[(N-is even):] For each $j = 0, 1, \ldots, m$

$$\begin{aligned} &\int_a^b k_j(x,t)y\left(t-\tau_j^*\right)dt \\ &\cong \frac{h}{3}\sum_{d=1}^{N/2}\left[k_j(x,t_{2d})y(t_{2d}-\tau_j^*) + 4k_j(x,t_{2d-1})y(t_{2d-1}-\tau_j^*) + k_j(x,t_{2d-2})y(t_{2d-2}-\tau_j^*)\right] \\ &= \sum_{d=1}^{N/2}\sum_{q=0}^{2} w_{(q)}^{even}\; k_j(x,t_{2d-q})\,y(t_{2d-q}-\tau_j^*) \end{aligned} \tag{3.9}$$

where $w_{(q)}^{even} = \begin{cases} \frac{h}{3} & ; \quad q = 0 \text{ or } 2 \\ \frac{4h}{3} & ; \quad q = 1 \end{cases}$

Simply, assume we use the historical function to present the approximation part, for all constant time-delay τ_j^* let $\overline{v}_j^* = \left[\!\left[\frac{\tau_j^*}{h}\right]\!\right]$, $\forall j = 0, 1, \ldots, m$. So,

$$y(t_{2d-q} - \tau_j^*) = \begin{cases} \varphi(t_{2d-q} - \tau_j^*) & if \quad t_{2d-q} - \tau_j^* < a \\ \widetilde{y}(t_{2d-q} - \tau_j^*) & if \quad t_{2d-q} - \tau_j^* \geq a \end{cases} = \begin{cases} \varphi(t_{2d-q} - \tau_j^*) & if \quad t_{2d-q} - \tau_j^* < a \\ \widetilde{y}_{2d-q-\overline{v}_j^*} & if \quad t_{2d-q} - \tau_j^* \geq a \end{cases} \tag{3.10}$$

Now, define

$${}^*\delta_{2d-q,even}^{[j]} = \begin{cases} 1 & \text{if} \quad t_{2d-q} - \tau_j^* < a \\ 0 & \text{if} \quad o.w \end{cases} \quad ; \quad {}^*\overline{\delta}_{2d-q,even}^{[j]} = \begin{cases} 1 & \text{if} \quad t_{2d-q} - \tau_j^* \geq a \\ 0 & \text{if} \quad o.w \end{cases} \tag{3.11}$$

Thus, the equation(3.10) became

$$\begin{aligned} y(t_{2d-q} - \tau_j^*) &= {}^*\delta_{2d-q,even}^{[j]} \, \varphi(t_{2d-q} - \tau_j^*) + {}^*\overline{\delta}_{2d-q,even}^{[j]} \, \widetilde{y}(t_{2d-q} - \tau_j^*) \\ &= {}^*\delta_{2d-q,even}^{[j]} \, \varphi(t_{2d-q} - \tau_j^*) + {}^*\overline{\delta}_{2d-q,even}^{[j]} \, \widetilde{y}_{2d-q-\overline{v}_j^*} \end{aligned} \tag{3.12}$$

Now applying formula (3.12) in the equation (3.9), yields:

$$\begin{aligned} &\int_a^b k_j(x,t) y\left(t - \tau_j^*\right) dt \\ &= \sum_{d=1}^{N/2} \sum_{q=0}^{2} \left\{ w_{(q)}^{even} \, k_j(x, t_{2d-q}) \, {}^*\delta_{2d-q,even}^{[j]} \, \varphi(t_{2d-q} - \tau_j^*) + w_{(q)}^{even} \, k_j(x, t_{2d-q}) \, {}^*\overline{\delta}_{2d-q,even}^{[j]} \, \widetilde{y}_{2d-q-\overline{v}_j^*} \right\} \end{aligned} \tag{3.13}$$

[(N-is odd):] For each $j = 0, 1, \ldots, m$

$$\begin{aligned} &\int_a^b k_j(x,t) y\left(t - \tau_j^*\right) dt \\ &\cong \sum_{d=1}^{(N-1)/2} \sum_{q=0}^{2} w_{(q)}^{even} \, k_j(x, t_{2d-q}) \, y(t_{2d-q} - \tau_j^*) + \sum_{q=0}^{1} w_{(q)}^{odd} \, k_j(x, t_{N-q}) \, y(t_{N-q} - \tau_j^*) \end{aligned} \tag{3.14}$$

where $w_{(q)}^{even} = \begin{cases} \frac{h}{3} & ; \quad q = 0 \text{ or } 2 \\ \frac{4h}{3} & ; \quad q = 1 \end{cases}$; $w_{(q)}^{odd} = \begin{cases} \frac{h}{2} & ; \quad q = 0 \\ \frac{h}{2} & ; \quad q = 1 \end{cases}$.

Simply, assume we use the historical function to present the approximation part, for all constant time-delay τ_j^* let $\overline{v}_j^* = \left[\!\left[\frac{\tau_j^*}{h}\right]\!\right]$, $\forall j = 0, 1, \ldots, m$. So,

$$y(t_{2d-q} - \tau_j^*) = \begin{cases} \varphi(t_{2d-q} - \tau_j^*) & if \quad t_{2d-q} - \tau_j^* < a \\ \widetilde{y}(t_{2d-q} - \tau_j^*) & if \quad t_{2d-q} - \tau_j^* \geq a \end{cases} = \begin{cases} \varphi(t_{2d-q} - \tau_j^*) & if \quad t_{2d-q} - \tau_j^* < a \\ \widetilde{y}_{2d-q-\overline{v}_j^*} & if \quad t_{2d-q} - \tau_j^* \geq a \end{cases} \tag{3.15}$$

and

$$y(t_{N-q} - \tau_j^*) = \begin{cases} \varphi(t_{N-q} - \tau_j^*) & if \quad t_{N-q} - \tau_j^* < a \\ \widetilde{y}(t_{N-q} - \tau_j^*) & if \quad t_{N-q} - \tau_j^* \geq a \end{cases} = \begin{cases} \varphi(t_{N-q} - \tau_j^*) & if \quad t_{N-q} - \tau_j^* < a \\ \widetilde{y}_{N-q-\overline{v}_j^*} & if \quad t_{N-q} - \tau_j^* \geq a \end{cases} \tag{3.16}$$

Now, define

$$\left.\begin{aligned} &{}^*\delta_{2d-q,even}^{[j]} = \begin{cases} 1 & \text{if} \quad t_{2d-q} - \tau_j^* < a \\ 0 & \text{if} \quad o.w \end{cases} \; ; \; {}^*\overline{\delta}_{2d-q,even}^{[j]} = \begin{cases} 1 & \text{if} \quad t_{2d-q} - \tau_j^* \geq a \\ 0 & \text{if} \quad o.w \end{cases} \\ &\text{and} \\ &{}^*\delta_{N-q,odd}^{[j]} = \begin{cases} 1 & \text{if} \quad t_{N-q} - \tau_j^* < a \\ 0 & \text{if} \quad o.w \end{cases} \; ; \; {}^*\overline{\delta}_{N-q,odd}^{[j]} = \begin{cases} 1 & \text{if} \quad t_{N-q} - \tau_j^* \geq a \\ 0 & \text{if} \quad o.w \end{cases} \end{aligned}\right\} \tag{3.17}$$

Thus the equations (3.15 and 3.16) respectively became:

$$\begin{aligned} y(t_{2d-q}-\tau_j^*) &= {}^*\delta^{[j]}_{2d-q,even}\,\varphi(t_{2d-q}-\tau_j^*)+{}^*\overline{\delta}^{[j]}_{2d-q,even}\,\widetilde{y}(t_{2d-q}-\tau_j^*) \\ &= {}^*\delta^{[j]}_{2d-q,even}\,\varphi(t_{2d-q}-\tau_j^*)+{}^*\overline{\delta}^{[j]}_{2d-q,even}\,\widetilde{y}_{2d-q-\overline{v}_j^*} \end{aligned} \tag{3.18}$$

and

$$\begin{aligned} y(t_{N-q}-\tau_j^*) &= {}^*\delta^{[j]}_{N-q,odd}\,\varphi(t_{N-q}-\tau_j^*)+{}^*\overline{\delta}^{[j]}_{N-q,odd}\,\widetilde{y}(t_{N-q}-\tau_j^*) \\ &= {}^*\delta^{[j]}_{N-q,odd}\,\varphi(t_{N-q}-\tau_j^*)+{}^*\overline{\delta}^{[j]}_{N-q,odd}\,\widetilde{y}_{N-q-\overline{v}_j^*} \end{aligned} \tag{3.19}$$

Apply equations (3.18 and 3.19) in the equation (3.14) so we get :

$$\begin{aligned} &\int_a^b k_j(x,t)y\left(t-\tau_j^*\right)dt \\ &=\sum_{d=1}^{(N-1)/2}\sum_{q=0}^{2}\left\{w_{(q)}^{even}\,k_j(x,t_{2d-q})\,{}^*\delta^{[j]}_{2d-q,even}\,\varphi(t_{2d-q}-\tau_j^*)+w_{(q)}^{even}\,k_j(x,t_{2d-q})\,{}^*\overline{\delta}^{[j]}_{2d-q,even}\,\widetilde{y}_{2d-q-\overline{v}_j^*}\right\} \\ &+\sum_{q=0}^{1}\left\{w_{(q)}^{odd}\,k_j(x,t_{N-q})\,{}^*\delta^{[j]}_{N-q,odd}\,\varphi(t_{N-q}-\tau_j^*)+w_{(q)}^{odd}\,k_j(x,t_{N-q})\,{}^*\overline{\delta}^{[j]}_{N-q,odd}\,\widetilde{y}_{N-q-\overline{v}_j^*}\right\} \end{aligned} \tag{3.20}$$

For each constant time-delay τ_j^* and $\overline{v}_j^*=\left[\!\left[\frac{\tau_j^*}{h}\right]\!\right]$, $\forall j=0,1,\ldots,m$.

4. A Numerical Technique Utilizing the Newton–Cotes Quadrature Rule

This section presents a new two approach that uses quadrature methods with the aid of the finite difference approximation to treating linear Fredholm integro-fractional differential equations of constants delay type (FIFDEs-Delays) with variable coefficients. Recall equation (1.1) for $0<\max_{\ell}\left\{\alpha_\ell,\beta\right\}\le 1$ with strictly decreasing for α_ℓ and β all $\ell=\overline{1:n}$. Thus, for obtaining an approximation of the solution $y(x)$ in a given set of $N+1$-equally spaced grid points $x_r=x_0+rh$, $(r=0,1,\ldots,N)$ with $b=x_0+Nh$ consists in approximating the linear FIFDEs-Delays (1.1) in the discrete equations:

$$\left[{}_a^C D_x^\beta y(x)+\sum_{\ell=1}^{n}p_\ell(x)\,{}_a^C D_x^{\alpha_\ell}y\left(x-\tau_\ell\right)+p_0(x)y\left(x-\tau_0\right)\right]_{x=x_r}=f(x_r)+\sum_{j=0}^{m}\lambda_j\int_a^b k_j(x_r,t)y\left(t-\tau_j^*\right)dt \tag{4.1}$$

This results in a system of $N+1$ linear algebraic equations in $N+1$ unknowns $\widetilde{y}(x_r)=\widetilde{y}_r$, which approximate $y(x_r)(r=0,1,\ldots,N)$. Here, we approximate the fractional derivatives using the forward difference approximation formula, the integrals in (4.1) using quadrature methods, and the parts containing fractional derivatives with constant time-delay parts using formula (2.10) in the new proposition (2.3).

4.1. A Numerical Method Based on the Trapezoidal Rule

Applying quadrature rule (3.6) to evaluate each integral with constant time-delay term in equation (4.1) at each point $x_r=a+rh$ $(r=0,1,2,\cdots,N)$ and take into account the proposition (2.2) that the Caputo fractional order for any defined function are vanishes at the starting points $x=x_0$, i.e., $\left[{}_a^C D_x^\beta y(x)\right]_{x=x_0}=0$ and also by proposition (2.4) that $\left[\sum_{\ell=1}^{n}{}_a^C D_x^{\alpha_\ell}y(x-\tau_\ell)\right]_{x=x_0}=0$, and recalling the historical function here we obtain the following:

$$y(x_r-\tau_0)=\begin{cases}\varphi(x_r-\tau_0) & \text{if} \quad x_r-\tau_0<a\\ \tilde{y}_{r-\overline{v}_0} & \text{if} \quad x_r-\tau_0\ge a\end{cases}\quad;\quad \overline{v}_0=\left[\!\left[\frac{\tau_0}{h}\right]\!\right]$$

Define

$$\delta_r^0=\begin{cases}1 & \text{if} \quad x_r-\tau_0<a\\ 0 & \text{if} \quad \text{o.w.}\end{cases},\quad \overline{\delta}_r^0=\begin{cases}1 & \text{if} \quad x_r-\tau_0\ge a\\ 0 & \text{if} \quad \text{o.w.}\end{cases} \tag{4.2}$$

Thus, for each $r = 0, 1, 2, \cdots, N$:

$$y(x_r - \tau_0) = \delta_r^0 \, \varphi(x_r - \tau_0) + \overline{\delta}_r^0 \, \tilde{y}_{r-\overline{v}_0} \tag{4.3}$$

First, For $r = 0$:

$$p_{0,0}\left[\delta_0^0 \varphi\left(x_0 - \tau_0\right) + \overline{\delta}_0^0 \tilde{y}_{0-\overline{v}_0}\right] = f_0 + \sum_{j=0}^{m} \lambda_j \sum_{z_j=0}^{N} \left\{ {}^{*}\delta_{z_j}^{[j]} \, w\left(z_j\right) k_j^{0,z_j} \, \varphi\left(t_{z_j} - \tau_j^*\right) + {}^{*} \, \overline{\delta}_{z_j}^{[j]} \, w\left(z_j\right) k_j^{0,z_j} \, \tilde{y}_{z_j - \overline{v}_j^*} \right\} \tag{4.4}$$

Second, for $r = 1, 2, \ldots N$ replace it by $\bar{r} = r - 1$ so $\bar{r} = 0, 1, \ldots N-1$. Then evaluate equation (4.1) at points $x = x_{\bar{r}+1}$ also with applying quadrature rule (3.6) to evaluate each integral with delay term in equation (4.1) , we obtain:

$$\begin{aligned} &\left[{}_a^C D_x^{\beta} \, y(x) + \sum_{\ell=1}^{n} p_\ell(x) \, {}_a^C D_x^{\alpha_\ell} y\left(x - \tau_\ell\right) \right]_{x = x_{\bar{r}+1}} + p_0(x_{\bar{r}+1}) y\left(x_{\bar{r}+1} - \tau_0\right) = f(x_{\bar{r}+1}) \\ &+ \sum_{j=0}^{m} \lambda_j \sum_{z_j=0}^{N} \left\{ {}^{*}\delta_{z_j}^{[j]} \, w\left(z_j\right) k_j^{\bar{r}+1,z_j} \, \varphi\left(t_{z_j} - \tau_j^*\right) + {}^{*} \, \overline{\delta}_{z_j}^{[j]} \, w\left(z_j\right) k_j^{\bar{r}+1,z_j} \, \tilde{y}_{z_j - \overline{v}_j^*} \right\} \end{aligned} \tag{4.5}$$

Using equation (4.3) and apply proposition (2.1) with equation (2.10) in the proposition (2.3) then we conclude the equation (4.5) can be written as:

$$\begin{aligned} &\frac{h^{-\beta}}{\Gamma(2-\beta)} \sum_{d=0}^{\bar{r}} \left[\tilde{y}_{\bar{r}-d+1} - \tilde{y}_{\bar{r}-d}\right] b_d^{\beta} + \sum_{\ell=1}^{n} p_{\ell,\bar{r}+1} \frac{h^{-\alpha_\ell}}{\Gamma\left(2-\alpha_\ell\right)} * \\ &\sum_{i=0}^{\bar{r}} \left\{ \left[\delta_{\bar{r}-i+1}^{\ell} \varphi\left(x_{\bar{r}-i+1} - \tau_\ell\right) + \overline{\delta}_{\bar{r}-i+1}^{\ell} \tilde{y}_{\bar{r}-i+1-\overline{v}_\ell}\right] - \left[\delta_{\bar{r}-i}^{\ell} \varphi\left(x_{\bar{r}-i} - \tau_\ell\right) + \overline{\delta}_{\bar{r}-i}^{\ell} \tilde{y}_{\bar{r}-i-\overline{v}_\ell}\right] \right\} b_i^{\alpha_\ell} \\ &+ p_{0,\bar{r}+1} \left[\delta_{\bar{r}+1}^{0} \varphi\left(x_{\bar{r}+1} - \tau_0\right) + \overline{\delta}_{\bar{r}+1}^{0} \, \tilde{y}_{\bar{r}+1-\overline{v}_0}\right] = f_{\bar{r}+1} \\ &+ \sum_{j=0}^{m} \lambda_j \sum_{z_j=0}^{N} \left\{ {}^{*}\delta_{z_j}^{[j]} \, w\left(z_j\right) k_j^{\bar{r}+1,z_j} \, \varphi\left(t_{z_j} - \tau_j^*\right) + {}^{*} \, \overline{\delta}_{z_j}^{[j]} \, w\left(z_j\right) k_j^{\bar{r}+1,z_j} \, \tilde{y}_{z_j - \overline{v}_j^*} \right\} \end{aligned} \tag{4.6}$$

where $k_j(x_r, t_{z_j}^N) = k_j^{r,z_j}$ all kernel's values for each $r, z_j = \overline{0:N}$ and $j = \overline{0:m}$. After some simple manipulation of linear algebraic equations (4.4) and (4.6) we construct a linear system of equations that can be written in matrix form:

$$\begin{aligned} &A_\beta(h) \, \Delta_N(\beta) \, \widetilde{Y}_N + \sum_{\ell=1}^{n} A_{\alpha_\ell}(h) \, P_\ell \, \Delta_N\left(\alpha_\ell\right) \, \overline{\Omega}_\ell \, \widetilde{Y}_N\left(\overline{v}_\ell\right) + P_0 \, \overline{\Omega}_0 \, \widetilde{Y}_N\left(\overline{v}_0\right) - \sum_{j=0}^{m} \lambda_j K_j^N W \overline{\Omega}_j^* \widetilde{Y}_N\left(\overline{v}_j^*\right) \\ &= F_N - \sum_{\ell=1}^{n} A_{\alpha_\ell}(h) P_\ell \Delta_N\left(\alpha_\ell\right) \, \Omega_\ell \Phi_N\left(\tau_\ell\right) - P_0 \, \Omega_0 \Phi_N\left(\tau_0\right) + \sum_{j=0}^{m} \lambda_j K_j^N W \, \Omega_j^* \Phi_N\left(\tau_j^*\right). \end{aligned} \tag{4.7}$$

where $A_\sigma(h)$ for fractional orders $\sigma = \beta$ or α_ℓ $(\forall \ell = \overline{1:n})$ are defined as:

$$A_\sigma(h) = \frac{h^{-\sigma}}{\Gamma(2-\sigma)} \tag{4.8}$$

where $\Delta_N\left(\alpha_\ell\right) = \left[L_{ed}^{\alpha_\ell}\right]_{N+1 \times N+1}$ is a lower triangular matrix and define each element $L_{ed}^{\alpha_\ell}$ for all $e, d = 0, 1, \ldots, N$ as follows:

$$\left.\begin{array}{ll} L_{ed}^{\alpha_\ell} = 0 & \forall e = d = 0 \;\text{ or }\; e < d \\ L_{ed}^{\alpha_\ell} = C_0^{\alpha_\ell} & \text{for all } \; e = d \neq 0 \\ L_{ed}^{\alpha_\ell} = -b_{e-1}^{\alpha_\ell} & \text{for } \; d = 0 \text{ and } e > 0 (e = 1, 2, \ldots, N) \\ L_{ed}^{\alpha_\ell} = C_{e-d}^{\alpha_e} & \forall \, e \geqslant 2 \text{ and } d \geqslant 1 (e > d) \end{array}\right\} \tag{4.9}$$

moreover $\Delta_N(\beta) = \left[L_{ed}^{\beta}\right]_{N+1\times N+1}$ is a lower triangular matrix and define each element L_{ed}^{β} for all $e,d = 0,1,\ldots,N$ as:

$$\left.\begin{array}{ll} L_{ed}^{\beta} = 0 & \forall\, e = d = 0 \text{ or } e < d \\ L_{ed}^{\beta} = C_0^{\beta} & \forall\, e = d \neq 0 \\ L_{ed}^{\beta} = -b_{e-1}^{\beta} & d = 0 \text{ and } e > 0 \\ L_{ed}^{\beta} = C_{e-d}^{\beta} & e \geqslant 2 \text{ and } d \geqslant 1 (e > d) \end{array}\right\} \tag{4.10}$$

For any real number $\sigma \in (0,1]$, $\sigma = \beta$ or $\alpha_\ell\, \forall \ell = \overline{1:n}$), the coefficients b_e^{σ} and C_e^{σ} $(e = \overline{0:N})$ are defined as follows:

$$\left.\begin{array}{ll} b_e^{\sigma} = (1+e)^{1-\sigma} - e^{1-\sigma} & ; \quad b_0^{\sigma} = 1 \\ C_e^{\sigma} = b_e^{\sigma} - b_{e-1}^{\sigma} & ; \quad C_0^{\sigma} = 1 \quad \text{and assume } b_{-1}^{\sigma} = 0 \end{array}\right\} \tag{4.11}$$

Furthermore, the matrix $K_j^N = [K_j^{e,d}]_{N+1\times N+1}$ is a square matrix of dimension $N+1$ and each element are defined for all $e,d = 0,1,\ldots,N$ and for all $j = 0,1,\ldots,m$:

$$K_j^N = \begin{bmatrix} K_j^{0,0} & K_j^{0,1} & \cdots & K_j^{0,N} \\ K_j^{1,0} & K_j^{1,1} & \cdots & K_j^{1,N} \\ \vdots & \vdots & & \vdots \\ K_j^{N,0} & K_j^{N,1} & \cdots & K_j^{N,N} \end{bmatrix}_{N+1\times N+1} ; \; K_j^{e,d} = K_j(x_e, t_d) \tag{4.12}$$

In addition, we have these diagonal matrices of dimension $N+1 \times N+1$, formed

$$\left.\begin{array}{c} W = \operatorname{diag}(w(0), w(1), \ldots, w(N)) \\ P_\ell = \operatorname{diag}\left(p_{\ell,0}, p_{\ell,1}, \ldots, p_{\ell,N}\right) \forall \ell = 0,1,\ldots,n \;\; \text{where}\, p_{\ell,e} = p_\ell\left(x_e\right) \end{array}\right\} \tag{4.13}$$

$$\overline{\Omega}_\ell = \operatorname{diag}\left(\overline{\delta}_0^{\ell}, \overline{\delta}_1^{\ell}, \ldots, \overline{\delta}_N^{\ell}\right) \quad ; \quad \Omega_\ell = \operatorname{diag}\left(\delta_0^{\ell}, \delta_1^{\ell}, \ldots, \delta_N^{\ell}\right) \;\; \forall \ell = 0,1,\ldots,n \tag{4.14}$$

$$\Omega_j^* = \operatorname{diag}\left({}^*\delta_0^{[j]}, {}^*\delta_1^{[j]}, \ldots, {}^*\delta_N^{[j]}\right) \quad ; \quad \overline{\Omega}_j^* = \operatorname{diag}\left({}^*\overline{\delta}_0^{[j]}, {}^*\overline{\delta}_1^{[j]}, \ldots, {}^*\overline{\delta}_N^{[j]}\right) \forall j = 0,1,\cdots,m \tag{4.15}$$

with,

$$\Phi_N(\tau_\ell) = [\varphi(x_0 - \tau_\ell), \varphi(x_1 - \tau_\ell), \ldots, \varphi(x_N - \tau_\ell)]^T \quad \forall \ell = 0,1,\cdots,n \tag{4.16}$$

and,

$$\Phi_N(\tau_j^*) = \left[\varphi(t_0 - \tau_j^*), \varphi(t_1 - \tau_j^*), \cdots, \varphi(t_N - \tau_j^*)\right]^T \quad \forall j = 0,1,\cdots,m \tag{4.17}$$

Moreover,

$$\begin{array}{ll} F_N = [f_0, f_1, \ldots, f_N]^T & ; \quad \widetilde{Y}_N\left(\overline{v}_\ell\right) = [\tilde{y}_{0-\overline{v}_\ell}, \tilde{y}_{1-\overline{v}_\ell}, \ldots, \tilde{y}_{N-\overline{v}_\ell}]^T \; \forall \ell = 0,1,2,\ldots,n \\ \widetilde{Y}_N = [\tilde{y}_0, \tilde{y}_1, \ldots, \tilde{y}_N]^T & ; \quad \widetilde{Y}_N\left(\overline{v}_j^*\right) = \left[\tilde{y}_{0-\overline{v}_j^*}, \tilde{y}_{1-\overline{v}_j^*}, \ldots, \tilde{y}_{N-\overline{v}_j^*}\right]^T, \forall j = 0,1,2,\ldots,m \end{array}$$

Here, $f_k = f(x_k)$ and $\widetilde{y}_k\,(k = 0,1,\cdots,N)$ are the approximate values of $y_k = y(x_k)$.
Assume that V and U appear on the left and right sides of the equation (4.7). The matrix system (4.18) is obtained by simplifying the equation (4.7), formed as

$$V\,\widetilde{Y}_N = U. \tag{4.18}$$

Finally, in this technique, a boundary condition of equation (1.1) is added as a new row in the system (4.7) can be formed in matrix form, this gives:

$$\mathrm{B}\,\widetilde{Y}_N = \mathrm{C} \tag{4.19}$$

$\mathrm{B} = \left[g_{00}\ 0\ \ldots\ 0\ h_{00}\right]_{N+1}$, $\widetilde{Y}_N = [\tilde{y}_0, \tilde{y}_1, \ldots, \tilde{y}_N]^T$ and $\mathrm{C} = [\vartheta_0]$. Adding (4.19) to (4.18) to create a new matrix with a difference value of N using the boundary condition equation in matrix form (4.19) yields:

$$G\,\widetilde{Y}_N = E \tag{4.20}$$

where

$$G = \begin{bmatrix} V \\ \cdots\cdots \\ B \end{bmatrix} \qquad \text{and} \qquad E = \begin{bmatrix} U \\ \cdots\cdots \\ C \end{bmatrix}$$

To find the estimated column vector $\widetilde{Y}_N$, store the matrix G, compute G^TG and G^TE, and then solve $[G^TG]\,\widetilde{Y}_N = [G^TE]$ using any numerical iterative methods. For linear FIFDEs-Delays (1.1), the approximate solution for every $\widetilde{y}_k$ at each point $x_k(k = \overline{0:N})$ is achieved.

The Algorithm (AFDT):
Using the closed Newton-Cotes formula (Trapezoidal Rule) and finite difference approximation, the approximate solution for linear FIFDEs-Delays of Fredholm type with variable coefficients (1.1 and 1.2) can be described as follows:
Step 1:

a. Input $N \in \mathbb{Z}^+$, take $h = \frac{(b-a)}{N}$ and $x_r = t_r = a + rh$.

b. Input the coefficients of boundary conditions g_{00}, h_{00} and ϑ_0.

Step 2: To compute $A_\sigma(h)$ for all $\sigma = \beta$ or α_ℓ and $\ell = n$, applied equation (4.8).
Step 3: Using equation (4.11), determine the constant coefficients (b_e^σ and C_e^σ) for fractional order $\sigma = \beta$ and α, respectively, for all $e = 0, 1, ..., N$.
Step 4: Evaluate each element L_{ed}^{β} and $L_{ed}^{\alpha_\ell}$ for all $e, d = 0, 1 \ldots, N$ using the formulas in equations (4.10) and (4.9), respectively, with step (3). Lastly, build the lower triangular matrix $\Delta_N(\beta) = [L_{ed}^{\beta}]_{N+1\times N+1}$ and $\Delta_N(\alpha_\ell) = [L_{ed}^{\alpha_\ell}]_{N+1\times N+1}$ respectively.
Step 5: Evaluate each element $p_{\ell,e}$ for all $\ell = 0, 1, \ldots, n$ and $e = 0, 1, \ldots, N$ where $p_{\ell,e} = p_\ell(x_e)$ using the formulas in equations (4.13) . Finally, construct the diagonal matrix P_ℓ is an $(N+1 \times N+1)$ square matrix.
Step 6: Evaluate each element $\overline{\delta}_e^{\ell}$ and δ_e^{ℓ} for all $\ell = 0, 1, \ldots, n$ and $e = 0, 1, \ldots, N$ using formula (4.2) and formula (2.8). Finally, equation (4.14)construct the diagonal matrix $\overline{\Omega}_\ell$ and Ω_ℓ where $\ell = 0, 1, \ldots, n$ is an $(N + 1 \times N + 1)$ square matrix.
Step 7: Evaluate each element ${}^*\overline{\delta}_e^{[j]}$ and ${}^*\delta_e^{[j]}$ for all $j = 0, 1, \ldots, m$ and $e = 0, 1, \ldots, N$ using formula (3.4) . Finally, equation (4.15) construct the diagonal matrix $\overline{\Omega}_j^*$ and Ω_j^* where $j = 0, 1, \ldots, m$ is an $(N + 1 \times N + 1)$ square matrix.
Step 8: For every $e = 0, 1, ..., N$, evaluate each element $w(e)$. Equation (4.13) thus creates the diagonal matrix W, which is a square matrix of size $(N + 1 \times N + 1)$.
Step 9: Determine the kernel values at each specified point, $K_j^N = k_j(x_e, t_d)$ for all $j = 0, 1, \ldots, m$ and $e, d = 0, 1, \ldots, N$.

Step 10: Determine every element of column vector F_N at positions x_r using $f_r = f(x_r)$, $x_r = a+rh\,(r = 0, 1, \ldots, N)$.
Step 11: Evaluate the basis functions $\varphi(x_e - \tau_\ell)$ for all $e = 0, 1, \ldots, N$ and $\ell = 0, 1, \ldots, n$ as they appear in the historical function given in (1.2). Subsequently, according to (4.16), these evaluated terms are used to construct the column vector $\Phi_N(\tau_\ell)$ for each $\ell = 0, 1, \ldots, n$.
Step 12: Determine the basis functions $\varphi(x_e - \tau_j^*)$ for every $e = 0, 1, \ldots, N$ and $j = 0, 1, \ldots, m$ in the historical function provided in (1.2). The column vector $\Phi_N(\tau_j^*)$ is then constructed for each $j = 0, 1, ..., m$ using these evaluated terms, as per (4.17).
Step 13: Boundary conditions g_{00}, h_{00}, and ϑ_0 are inserted into matrices B and C to create (4.19).
Step 14: Build the matrices G and E that the system (4.20) represents.
Step 15: After multiplying both sides by G^T in step 15, use the LU-factorization approach for the system to calculate the column-approximate values $\widetilde{Y}_N$ of the precise solution Y.

4.2. A Numerical Method Based on the Simpson's 1/3 Rule

By applying the equations (3.13) or (3.20) for number of sub-intervals even or odd, respectively, to evaluate each integral parts in equation (4.1) with taken proposition (2.1 and 2.2) and proposition (2.3),

also with proposition (2.4) and equation (4.3) , then it results in the following classification:

. For N-is even:
First, for $r = 0$, i.e., enter $x = x_0 = a$ into equation (4.1) and use equation (3.13) with formula (4.3) to obtain:

$$p_{0,0}\left[\delta_0^0\varphi\left(x_0-\tau_0\right)+\overline{\delta}_0^0\tilde{y}_{0-\overline{v}_0}\right]=f_0$$
$$+\sum_{j=0}^{m}\lambda_j\sum_{d=1}^{N/2}\sum_{q=0}^{2}\left\{w_{(q)}^{even}\,k_j^{0,2d-q}*\delta_{2d-q,even}^{[j]}\,\varphi(t_{2d-q}-\tau_j^*)+w_{(q)}^{even}\,k_j^{0,2d-q}*\overline{\delta}_{2d-q,even}^{[j]}\,\widetilde{y}_{2d-q-\overline{v}_j^*}\right\} \tag{4.21}$$

Second, for $r = 1, 2, \ldots, N$ replace it by subsumption ($\bar{r} = 0, 1, \ldots, N-1$), equation (4.1) should therefore be evaluated at positions $x = x_{\overline{r}+1}$. Additionally, by evaluating each integral with a delay term in equation (4.1) using the quadrature rule (3.13), we obtain:

$$\left[{}_a^C D_x^{\beta}y(x)+\sum_{\ell=1}^{n}p_\ell(x)\,{}_a^C D_x^{\alpha_\ell}y\left(x-\tau_\ell\right)\right]_{x=x_{\overline{r}+1}}+p_0(x_{\overline{r}+1})y\left(x_{\overline{r}+1}-\tau_0\right)=f(x_{\overline{r}+1})$$
$$+\sum_{j=0}^{m}\lambda_j\sum_{d=1}^{N/2}\sum_{q=0}^{2}\left\{w_{(q)}^{even}\,k_j^{\overline{r}+1,2d-q}*\delta_{2d-q,even}^{[j]}\,\varphi(t_{2d-q}-\tau_j^*)+w_{(q)}^{even}\,k_j^{\overline{r}+1,2d-q}*\overline{\delta}_{2d-q,even}^{[j]}\,\widetilde{y}_{2d-q-\overline{v}_j^*}\right\} \tag{4.22}$$

We conclude that by utilizing equation (4.3) and applying proposition (2.1) with equation (2.10) in the proposition (2.3), the equation (4.22) became:

$$\frac{h^{-\beta}}{\Gamma(2-\beta)}\sum_{d=0}^{\overline{r}}\left[\tilde{y}_{\overline{r}-d+1}-\tilde{y}_{\overline{r}-d}\right]b_d^{\beta}+\sum_{\ell=1}^{n}p_{\ell,\overline{r}+1}\frac{h^{-\alpha_\ell}}{\Gamma\left(2-\alpha_\ell\right)}*$$
$$\sum_{i=0}^{\overline{r}}\left\{\left[\delta_{\overline{r}-i+1}^{\ell}\varphi\left(x_{\overline{r}-i+1}-\tau_\ell\right)+\overline{\delta}_{\overline{r}-i+1}^{\ell}\tilde{y}_{\overline{r}-i+1-\overline{v}_\ell}\right]-\left[\delta_{\overline{r}-i}^{\ell}\varphi\left(x_{\overline{r}-i}-\tau_\ell\right)+\overline{\delta}_{\overline{r}-i}^{\ell}\tilde{y}_{\overline{r}-i-\overline{v}_\ell}\right]\right\}b_i^{\alpha_\ell}$$
$$+p_{0,\overline{r}+1}\left[\delta_{\overline{r}+1}^{0}\varphi\left(x_{\overline{r}+1}-\tau_0\right)+\overline{\delta}_{\overline{r}+1}^{0}\,\tilde{y}_{\overline{r}+1-\overline{v}_0}\right]=f_{\overline{r}+1}$$
$$+\sum_{j=0}^{m}\lambda_j\sum_{d=1}^{N/2}\sum_{q=0}^{2}\left\{w_{(q)}^{even},k_j^{\overline{r}+1,2d-q}*\delta_{2d-q,even}^{[j]}\,\varphi(t_{2d-q}-\tau_j^*)+w_{(q)}^{even}\,k_j^{\overline{r}+1,2d-q}*\overline{\delta}_{2d-q,even}^{[j]}\,\widetilde{y}_{2d-q-\overline{v}_j^*}\right\} \tag{4.23}$$

where $k_j(x_r, t_{2d-q}) = k_j^{r,2d-q}$ all kernels values for each $r, 2d-q = \overline{0:N}$ and $j = \overline{0:m}$. Following a few straightforward operations on the linear algebraic equations (4.21) and (4.23), we create a linear system of equations that can be expressed as a matrix:

$$A_\beta(h)\,\Delta_N(\beta)\,\widetilde{Y}_N+\sum_{\ell=1}^{n}A_{\alpha_\ell}(h)\,P_\ell\,\Delta_N\left(\alpha_\ell\right)\,\overline{\Omega}_\ell\,\widetilde{Y}_N\left(\overline{v}_\ell\right)+P_0\,\overline{\Omega}_0\,\widetilde{Y}_N\left(\overline{v}_0\right)$$
$$-\sum_{j=0}^{m}\lambda_j K_j^N\,\overline{W}_{even}\,\overline{\Omega}_{j,even}^*\,\widetilde{Y}_N\left(\overline{v}_j^*\right)=F_N-\sum_{\ell=1}^{n}A_{\alpha_\ell}(h)\,P_\ell\,\Delta_N\left(\alpha_\ell\right)\,\Omega_\ell\,\Phi_N\left(\tau_\ell\right)-P_0\,\Omega_0\,\Phi_N\left(\tau_0\right)$$
$$+\sum_{j=0}^{m}\lambda_j K_j^N\,\overline{W}_{even}\,\Omega_{j,even}^*\,\Phi_N\left(\tau_j^*\right). \tag{4.24}$$

moreover, Some matrix of this system are the same as in the trapezoidal case, except for the following changes:

$$w_e=\begin{cases}\frac{h}{3} & if \qquad e=0\,or\,e=N\\ \frac{4h}{3} & if \quad e\neq 0,N\,and\,e-odd\\ \frac{2h}{3} & if \quad e\neq 0,N\,and\,e-even\end{cases},\quad \forall e=0,1,\ldots,N \tag{4.25}$$

Here, we get that

$$\overline{W}_{even} = diag(w_0, w_1, \ldots, w_N) \tag{4.26}$$

and also we have

$$\begin{aligned} \Omega^*_{j,even} &= \text{diag}\left({}^*\delta^{[j]}_{0,even}, {}^*\delta^{[j]}_{1,even}, \ldots, {}^*\delta^{[j]}_{N,even}\right) \\ \overline{\Omega}^*_{j,even} &= \text{diag}\left({}^*\overline{\delta}^{[j]}_{0,even}, {}^*\overline{\delta}^{[j]}_{1,even}, \ldots, {}^*\overline{\delta}^{[j]}_{N,even}\right) \end{aligned} \quad \forall j = 0, 1, \cdots, m \tag{4.27}$$

Furthermore,

$$\begin{aligned} F_N &= [f_0, f_1, \ldots, f_N]^T \quad ; \quad \widetilde{Y}_N(\overline{v}_\ell) = [\tilde{y}_{0-\overline{v}_\ell}, \tilde{y}_{1-\overline{v}_\ell}, \ldots, \tilde{y}_{N-\overline{v}_\ell}]^T \ \forall \ell = 0, 1, 2, \ldots, n \\ \widetilde{Y}_N &= [\tilde{y}_0, \tilde{y}_1, \ldots, \tilde{y}_N]^T \quad ; \quad \widetilde{Y}_N(\overline{v}^*_j) = \left[\tilde{y}_{0-\overline{v}^*_j}, \tilde{y}_{1-\overline{v}^*_j}, \ldots, \tilde{y}_{N-\overline{v}^*_j}\right]^T, \forall j = 0, 1, 2, \ldots, m \end{aligned}$$

since $f_k = f(x_k)$ and $\widetilde{y}_k$ $(k = \overline{0:N})$ are are the approximate values of $y_k = y(x_k)$.
Suppose that in the equation (4.24) U_{even} appears on the left side of the equation, and V_{even} appears on the right side. After simplification in the equation (4.24), obtained the matrix system

$$V_{even}\, \widetilde{Y}_N = U_{even}. \tag{4.28}$$

. For N-is odd:
First, for $r = 0$, i.e. take $x = x_0 = a$ in to equation (4.1) and using formula (4.3) with equation (3.20) , we get:

$$\begin{aligned} & p_{0,0}\left[\delta^0_0 \varphi(x_0 - \tau_0) + \overline{\delta}^0_0 \tilde{y}_{0-\overline{v}_0}\right] = f_0 \\ & + \sum_{j=0}^{m} \lambda_j \left\{ \sum_{d=1}^{(N-1)/2} \sum_{q=0}^{2} \left[w^{even}_{(q)} k_j^{0,2d-q} \, {}^*\delta^{[j]}_{2d-q,even} \varphi(t_{2d-q} - \tau^*_j) + w^{even}_{(q)} k_j^{0,2d-q} \, {}^*\overline{\delta}^{[j]}_{2d-q,even} \, \widetilde{y}_{2d-q-\overline{v}^*_j} \right] \right. \\ & \left. + \sum_{q=0}^{1} \left[w^{odd}_{(q)} \, k_j^{0,N-q} \, {}^*\delta^{[j]}_{N-q,odd} \, \varphi(t_{N-q} - \tau^*_j) + w^{odd}_{(q)} \, k_j^{0,N-q} \, {}^*\overline{\delta}^{[j]}_{N-q,odd} \, \widetilde{y}_{N-q-\overline{v}^*_j} \right] \right\} \end{aligned} \tag{4.29}$$

Second, for $r = 1, 2, \ldots, N$ replace it by subsumption $(\overline{r} = 0, 1, \ldots, N-1)$, equation (4.1) should therefore be evaluated at positions $x = x_{\overline{r}+1}$. Also, with applying quadrature rule (3.20) to evaluate each integral with delay term in equation (4.1) , we obtain:

$$\begin{aligned} & \left[{}^C_a D^\beta_x y(x) + \sum_{\ell=1}^{n} p_\ell(x) \, {}^C_a D^{\alpha_\ell}_x y(x - \tau_\ell) \right]_{x = x_{\overline{r}+1}} + p_0(x_{\overline{r}+1}) y(x_{\overline{r}+1} - \tau_0) = f(x_{\overline{r}+1}) \\ & + \sum_{j=0}^{m} \lambda_j \left\{ \sum_{d=1}^{(N-1)/2} \sum_{q=0}^{2} \left[w^{even}_{(q)} \, k_j^{\overline{r}+1,2d-q} \, {}^*\delta^{[j]}_{2d-q,even} \, \varphi(t_{2d-q} - \tau^*_j) + w^{even}_{(q)} \, k_j^{\overline{r}+1,2d-q} \, {}^*\overline{\delta}^{[j]}_{2d-q,even} \, \widetilde{y}_{2d-q-\overline{v}^*_j} \right] \right. \\ & \left. + \sum_{q=0}^{1} \left[w^{odd}_{(q)} \, k_j^{\overline{r}+1,N-q} \, {}^*\delta^{[j]}_{N-q,odd} \, \varphi(t_{N-q} - \tau^*_j) + w^{odd}_{(q)} \, k_j^{\overline{r}+1,N-q} \, {}^*\overline{\delta}^{[j]}_{N-q,odd} \, \widetilde{y}_{N-q-\overline{v}^*_j} \right] \right\} \end{aligned} \tag{4.30}$$

We conclude that the equation (4.30) may be written as follows by using equation (4.3) and applying

proposition (2.1) with equation (2.10) in the proposition (2.3):

$$\begin{aligned}
&\frac{h^{-\beta}}{\Gamma(2-\beta)}\sum_{d=0}^{\overline{r}}\left[\tilde{y}_{\overline{r}-d+1}-\tilde{y}_{\overline{r}-d}\right]b_d^{\beta}+\sum_{\ell=1}^{n}p_{\ell,\overline{r}+1}\frac{h^{-\alpha_\ell}}{\Gamma\left(2-\alpha_\ell\right)}*\\
&\sum_{i=0}^{\overline{r}}\left\{\left[\delta^{\ell}_{\overline{r}-i+1}\varphi\left(x_{\overline{r}-i+1}-\tau_\ell\right)+\overline{\delta}^{\ell}_{\overline{r}-i+1}\tilde{y}_{\overline{r}-i+1-\overline{v}_\ell}\right]-\left[\delta^{\ell}_{\overline{r}-i}\varphi\left(x_{\overline{r}-i}-\tau_\ell\right)+\overline{\delta}^{\ell}_{\overline{r}-i}\tilde{y}_{\overline{r}-i-\overline{v}_\ell}\right]\right\}b_i^{\alpha_\ell}\\
&+p_{0,\overline{r}+1}\left[\delta^{0}_{\overline{r}+1}\varphi\left(x_{\overline{r}+1}-\tau_0\right)+\overline{\delta}^{0}_{\overline{r}+1}\,\tilde{y}_{\overline{r}+1-\overline{v}_0}\right]=f_{\overline{r}+1}\\
&+\sum_{j=0}^{m}\lambda_j\left\{\sum_{d=1}^{(N-1)/2}\sum_{q=0}^{2}\left[w^{even}_{(q)}\,k_j^{\overline{r}+1,2d-q}*\delta^{[j]}_{2d-q,even}\,\varphi(t_{2d-q}-\tau_j^*)+w^{even}_{(q)}\,k_j^{\overline{r}+1,2d-q}*\overline{\delta}^{[j]}_{2d-q,even}\,\tilde{y}_{2d-q-\overline{v}_j^*}\right]\right.\\
&\left.+\sum_{q=0}^{1}\left[w^{odd}_{(q)}\,k_j^{\overline{r}+1,N-q}*\delta^{[j]}_{N-q,odd}\,\varphi(t_{N-q}-\tau_j^*)+w^{odd}_{(q)}\,k_j^{\overline{r}+1,N-q}*\overline{\delta}^{[j]}_{N-q,odd}\,\tilde{y}_{N-q-\overline{v}_j^*}\right]\right\}
\end{aligned}\tag{4.31}$$

where $k_j(x_r,t_{2d-q})=k_j^{r,2d-q}$ all kernels values for each $r,2d-q=\overline{0:N}$ and $j=\overline{0:m}$. Following some straightforward manipulation of the linear algebraic equations (4.29) and (4.31), we create a linear system of equations that may be expressed as a matrix:

$$\begin{aligned}
&A_\beta(h)\,\Delta_N(\beta)\,\widetilde{Y}_N+\sum_{\ell=1}^{n}A_{\alpha_\ell}(h)\,P_\ell\,\Delta_N\left(\alpha_\ell\right)\,\overline{\Omega}_\ell\,\widetilde{Y}_N\left(\overline{v}_\ell\right)+P_0\,\overline{\Omega}_0\,\widetilde{Y}_N\left(\overline{v}_0\right)\\
&-\sum_{j=0}^{m}\lambda_j\left\{K_j^N\,W_{even}\,\overline{\Omega}^*_{j,even}+K_j^N\,W_{odd}\,\overline{\Omega}^*_{j,odd}\right\}\widetilde{Y}_N\left(\overline{v}_j^*\right)=F_N-\sum_{\ell=1}^{n}A_{\alpha_\ell}(h)\,P_\ell\,\Delta_N\left(\alpha_\ell\right)\,\Omega_\ell\,\Phi_N\left(\tau_\ell\right)\\
&-P_0\,\Omega_0\,\Phi_N\left(\tau_0\right)+\sum_{j=0}^{m}\lambda_j\left\{K_j^N\,W_{even}\,\Omega^*_{j,even}+K_j^N\,W_{odd}\,\Omega^*_{j,odd}\right\}\Phi_N\left(\tau_j^*\right).
\end{aligned}\tag{4.32}$$

Additionally, some of this system's matrices are identical to those in the trapezoidal case, with the exception of the following modifications:

$$w_e^{even}=\begin{cases}\frac{h}{3} & if \quad e=0\,or\,N-1\\ \frac{4h}{3} & if \quad e\neq 0,N-1,N\,and\,e-odd\\ \frac{2h}{3} & if \quad e\neq 0,N-1,N\,and\,e-even\\ 0 & if \quad e=N\end{cases},\forall e=0,1,\ldots,N\tag{4.33}$$

and

$$w_e^{odd}=\begin{cases}\frac{h}{2} & if \quad e=N-1\,or\,e=N\\ 0 & if \quad o.w\end{cases},\forall e=0,1,\ldots,N\tag{4.34}$$

Here, we get that

$$W_{even}=diag(w_0^{even},w_1^{even},\ldots,w_N^{even})\quad and\quad W_{odd}=diag(w_0^{odd},w_1^{odd},\ldots,w_N^{odd})\tag{4.35}$$

and also we have

$$\begin{array}{l} \Omega^*_{j,even} = \text{diag}\left({}^*\delta^{[j]}_{0,even}, {}^*\delta^{[j]}_{1,even}, \ldots, {}^*\delta^{[j]}_{N,even}\right) \\ \overline{\Omega}^*_{j,even} = \text{diag}\left({}^*\overline{\delta}^{[j]}_{0,even}, {}^*\overline{\delta}^{[j]}_{1,even}, \ldots, {}^*\overline{\delta}^{[j]}_{N,even}\right) \\ \Omega^*_{j,odd} = \text{diag}\left({}^*\delta^{[j]}_{0,odd}, {}^*\delta^{[j]}_{1,odd}, \ldots, {}^*\delta^{[j]}_{N,odd}\right) \\ \overline{\Omega}^*_{j,odd} = \text{diag}\left({}^*\overline{\delta}^{[j]}_{0,odd}, {}^*\overline{\delta}^{[j]}_{1,odd}, \ldots, {}^*\overline{\delta}^{[j]}_{N,odd}\right) \end{array} , \quad \forall j = 0, 1, \cdots, m \tag{4.36}$$

Furthermore,

$$\begin{array}{lll} F_N = [f_0, f_1, \ldots, f_N]^T & ; & \widetilde{Y}_N\left(\overline{v}_\ell\right) = \left[\tilde{y}_{0-\overline{v}_\ell}, \tilde{y}_{1-\overline{v}_\ell}, \ldots, \tilde{y}_{N-\overline{v}_\ell}\right]^T \ \forall \ell = 0, 1, 2, \ldots, n \\ \widetilde{Y}_N = [\tilde{y}_0, \tilde{y}_1, \ldots, \tilde{y}_N]^T & ; & \widetilde{Y}_N\left(\overline{v}^*_j\right) = \left[\tilde{y}_{0-\overline{v}^*_j}, \tilde{y}_{1-\overline{v}^*_j}, \ldots, \tilde{y}_{N-\overline{v}^*_j}\right]^T, \forall j = 0, 1, 2, \ldots, m \end{array}$$

Since, $f_k = f(x_k)$ and $\widetilde{y}_k \ (k = \overline{0:N})$ are are the approximate values of $y_k = y(x_k)$.
Assume that U_{odd} and V_{odd} appear on the left and right sides of the equation (4.32). I acquire the matrix system after simplifying the equation (4.32):

$$V_{odd}\, \widetilde{Y}_N = U_{odd}. \tag{4.37}$$

Finally, from using the boundary equation in matrix form(4.19) and obtaining a new matrix by adding (4.19) to (4.37) or 4.28 for different value of N yields:

$$G\, \widetilde{Y}_N = E \tag{4.38}$$

where

$$\begin{cases} \text{If } N - \text{is even, then} \quad G = \begin{bmatrix} V_{even} \\ \cdots\cdots \\ B \end{bmatrix} \quad \text{and} \quad E = \begin{bmatrix} U_{even} \\ \cdots\cdots \\ C \end{bmatrix} \\ \text{or} \\ \text{If } N - \text{is odd, then} \quad G = \begin{bmatrix} V_{odd} \\ \cdots\cdots \\ B \end{bmatrix} \quad \text{and} \quad E = \begin{bmatrix} U_{odd} \\ \cdots\cdots \\ C \end{bmatrix} \end{cases}$$

To find the estimated column vector $\widetilde{Y}_N$, store the matrix G, compute G^TG and G^TE, and then solve $[G^TG]\, \widetilde{Y}_N = [G^TE]$ using any numerical iterative method. For linear FIFDEs-Delays (1.1), the approximate solution for every $\widetilde{y}_k$ at each point $x_k (k = \overline{0:N})$ is achieved.

The Algorithm (AFDS):
The approximate solution for linear FIFDEs-Delays of Fredholm type with variable coefficients can be described as follows using the finite difference approximation and the closed Newton-Cotes formula (Simpson's 1/3 rule):
Step 1:

a. Input $N \in \mathbb{Z}^+$, take $h = \frac{(b-a)}{N}$ and $x_r = t_r = a + rh$.

b. Input the coefficients of boundary conditions g_{11}, h_{11} and ϑ_1.

Step 2: To compute $A_\sigma(h)$ for all $\sigma = \beta$ or α_ℓ and $\ell = n$, applied equation (4.8).
Step 3: Using equation (4.11), determine the constant coefficients (b^σ_e and C^σ_e) for fractional order $\sigma = \beta$ and α, respectively, for all $e = 0, 1, ..., N$.
Step 4: Evaluate each element L^β_{ed} and $L^{\alpha_\ell}_{ed}$ for all $e, d = 0, 1 \ldots, N$ using the formulas in equations (4.10)

and (4.9), respectively, with step (3). Lastly, build the lower triangular matrix $\Delta_N(\beta) = [L_{ed}^{\beta}]_{N+1\times N+1}$ and $\Delta_N(\alpha_\ell) = [L_{ed}^{\alpha_\ell}]_{N+1\times N+1}$ respectively.
Step 5: Evaluate each element $p_{\ell,e}$ for all $\ell = 0, 1, \ldots, n$ and $e = 0, 1, \ldots, N$ where $p_{\ell,e} = p_\ell(x_e)$ using the formulas in equations (4.13) . Finally, construct the diagonal matrix P_ℓ is an $(N+1 \times N+1)$ square matrix.
Step 6: Evaluate each element $\overline{\delta}_e^{\ell}$ and δ_e^{ℓ} for all $\ell = 0, 1, \ldots, n$ and $e = 0, 1, \ldots, N$ using formula (4.2) and formula (2.8. Finally, equation (4.14)construct the diagonal matrix $\overline{\Omega}_\ell$ and Ω_ℓ where $\ell = 0, 1, \ldots, n$ is an $(N+1 \times N+1)$ square matrix.
Step 7: Evaluate each element $({}^*\overline{\delta}_{e,even}^{[j]}$ and ${}^*\delta_{e,even}^{[j]})$ or $({}^*\overline{\delta}_{e,odd}^{[j]}$ and ${}^*\delta_{e,odd}^{[j]})$ for all $j = 0, 1, \ldots, m$ and $e = 0, 1, \ldots, N$ using formula (3.11 or 3.17) respectively for different value of N . Equation (4.27 or 4.36) construct the diagonal matrix $(\overline{\Omega}_{j,even}^{*}$ and $\Omega_{j,even}^{*})$ or $(\overline{\Omega}_{j,odd}^{*}$ and $\Omega_{j,odd}^{*})$ where $j = 0, 1, \ldots, m$ is an $(N+1 \times N+1)$ square matrix.
Step 8: For each index $e = 0, 1, \ldots, N$, the corresponding weight w_e is computed as defined in equation (4.25). When applicable, the even and odd weights, denoted by w_e^{even} and w_e^{odd}, are evaluated according to equations (4.33) and (4.34), respectively. Using these weights, equation (4.26) is employed to construct the diagonal matrix $\overline{W}_{\text{even}}$, whereas equation (4.35) is used to construct the pair of diagonal matrices $(W_{\text{even}}, W_{\text{odd}})$. Each resulting matrix is a square matrix of dimension $(N+1) \times (N+1)$.
Step 9: Evaluate the kernel function at all prescribed collocation points. Specifically, compute $K_j^N = k_j(x_e, t_d)$ for all $j = 0, 1, \ldots, m$ and $e, d = 0, 1, \ldots, N$.
Step 10: Determine every element of column vector F_N at positions x_r using $f_r = f(x_r)$, $x_r = a+rh$ $(r = 0, 1, \ldots, N)$.
Step 11: Evaluate the basis functions $\varphi(x_e - \tau_\ell)$ for all $e = 0, 1, \ldots, N$ and $\ell = 0, 1, \ldots, n$ as they appear in the historical function given in (1.2). Subsequently, according to (4.16), these evaluated terms are used to construct the column vector $\Phi_N(\tau_\ell)$ for each $\ell = 0, 1, \ldots, n$.
Step 12: Determine the basis functions $\varphi(x_e - \tau_j^*)$ for every $e = 0, 1, \ldots, N$ and $j = 0, 1, \ldots, m$ in the historical function provided in (1.2). The column vector $\Phi_N(\tau_j^*)$ is then constructed for each $j = 0, 1, ..., m$ using these evaluated terms, as per (4.17).
Step 13: Boundary conditions g_{00}, h_{00}, and ϑ_0 are inserted into matrices B and C to create (4.19).
Step 14: Construct the matrices G and E that represent the system (4.38).
Step 15: Compute the column-approximate values $\widetilde{Y}_N$ of the precise solution Y using the numerical approach for the system, which is derived in step 14 after multiplying both sides by G^T.)

5. Numerical Results

The L_2 error norm is used in the numerical part to confirm the accuracy and effectiveness of the suggested strategies. Here, some numerical results for linear FIFDEs-Delays of variable coefficients equations (1.1 and 1.2), were shown using the trapezoidal and Simpson's rules quadrature methods. The algorithms **AFDT** and **AFDS**, respectively, are used for obtaining their results. Python programming was written for all of them.The least square error with running time of programs was used for our comparison between the algorithms.

Example 5.1 *Consider a linear FIFDEs constant multi-time delays containing various fractional orders order lies in* $(0, 1)$ *on the closed interval* $[0, 1]$ *with variable coefficients:*

$$\begin{aligned} &{}_0^C D_x^{0.6} y(x) + x^2\, {}_0^C D_x^{0.5} y(x-0.8) + \frac{1}{2} y(x-0.6) = \frac{2\,x^{1.4}}{\Gamma(2.4)} + \frac{2\,x^{3.5}}{\Gamma(2.5)} - \frac{8\,x^{2.5}}{5\,\Gamma(1.5)} + \frac{x^2}{2} - \frac{87x}{100} + \frac{101}{600} \\ &+ \int_0^1 (x-t) y(t-0.5)dt + \int_0^1 (2x+t^2) y(t-0.4)dt \end{aligned} \tag{5.1}$$

with the boundary condition: $y(0) + y(1) = 1$ *as well as the historical function* $\varphi(x) = x^2$. *While,* $y(x) = x^2$ *is the exact solution. This example obtained the numerical computation at every point by applying the suggested strategy which is a linear FIFDEs-Delays constant multi-time delay containing various fractional orders order lies in* $(0, 1)$*: Take* $N = 10$, $h = 0.1$ *and* $x_r = a + rh$, $r = 0, 1, \ldots, N$.

Since $(n, m) = (1, 1)$ *and the fractional orders are* $\beta = 0.6$ *and* $\alpha_1 = 0.5$ *with boundary coefficients* $g_{00} = h_{00} = 1$ *and* $\vartheta_0 = 1$.

By applying the recommended method to the given situation, we were able to obtain the numerical results. The AFDT and AFDS programs were used to set up the Python application. The outcomes were as follows.From the boundary condition equation, the matrix form is computed as:

$$[\mathrm{B};\mathrm{C}] = [\ 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad ; \quad 1]$$

The augmented matrix is created by replacing the fundamental equation with the matrices mentioned above under the following condition:

$[G; E]_{Trap} =$

$$\left[\begin{array}{cccccccccccc}
0.034 & 0.035 & 0.034 & 0.031 & 0.026 & -0.031 & -0.05 & 0 & 0 & 0 & 0 & ; \ -0.0163 \\
-4.483 & 4.492 & 0.004 & 0.001 & -0.004 & -0.056 & -0.06 & 0 & 0 & 0 & 0 & ; \ 0.02954 \\
-1.4596 & -3.0783 & 4.4609 & -0.029 & -0.034 & -0.081 & -0.07 & 0 & 0 & 0 & 0 & ; \ 0.1166 \\
-1.0985 & -0.4461 & -3.1093 & 4.4279 & -0.064 & -0.106 & -0.08 & 0 & 0 & 0 & 0 & ; \ 0.2282 \\
-0.9352 & -0.2783 & -0.4771 & -3.1423 & 4.3929 & -0.131 & -0.09 & 0 & 0 & 0 & 0 & ; \ 0.3588 \\
-0.8454 & -0.2348 & -0.3093 & -0.5101 & -3.1773 & 4.3309 & -0.1 & 0 & 0 & 0 & 0 & ; \ 0.5054 \\
-0.2922 & -0.2282 & -0.2658 & -0.3423 & -0.5451 & -3.2343 & 4.3769 & 0 & 0 & 0 & 0 & ; \ 0.666 \\
-0.7603 & 0.2631 & -0.2592 & -0.2988 & -0.3773 & -0.5971 & -3.1733 & 4.4869 & 0 & 0 & 0 & ; \ 0.8441 \\
1.5415 & -0.2532 & 0.2321 & -0.2922 & -0.3338 & -0.4243 & -0.5211 & -3.0533 & 4.4869 & 0 & 0 & ; \ 1.044 \\
-2.4264 & 2.6164 & -0.2842 & 0.1991 & -0.3271 & -0.3758 & -0.3333 & -0.3911 & -3.0533 & 4.4869 & 0 & ; \ 1.293 \\
-1.0750 & -2.3874 & 3.2634 & -0.3172 & 0.1641 & -0.3641 & -0.2698 & -0.1933 & -0.3911 & -3.0533 & 4.4869 & ; \ 1.626 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & ; \ 1
\end{array}\right]$$

$[G; E]_{Simp} =$

$$\left[\begin{array}{cccccccccccc}
0.056 & 0.006667 & 0.06933 & -0.012 & 0.07733 & -0.07467 & -0.03333 & 0 & 0 & 0 & 0 & ; \ -0.0165 \\
-4.458 & 4.46 & 0.04267 & -0.04533 & 0.05067 & -0.1047 & -0.04 & 0 & 0 & 0 & 0 & ; \ 0.02913 \\
-1.431 & -3.113 & 4.503 & -0.07867 & 0.024 & -0.1347 & -0.04667 & 0 & 0 & 0 & 0 & ; \ 0.116 \\
-1.066 & -0.4845 & -3.064 & 4.375 & -0.002667 & -0.1647 & -0.05333 & 0 & 0 & 0 & 0 & ; \ 0.2274 \\
-0.8998 & -0.32 & -0.4285 & -3.199 & 4.458 & -0.1947 & -0.06 & 0 & 0 & 0 & 0 & ; \ 0.3578 \\
-0.8067 & -0.2798 & -0.2573 & -0.5698 & -3.109 & 4.262 & -0.06667 & 0 & 0 & 0 & 0 & ; \ 0.5041 \\
-0.2502 & -0.2765 & -0.2105 & -0.4053 & -0.4738 & -3.308 & 4.414 & 0 & 0 & 0 & 0 & ; \ 0.6645 \\
-0.715 & 0.2115 & -0.2005 & -0.3651 & -0.3026 & -0.6758 & -3.133 & 4.487 & 0 & 0 & 0 & ; \ 0.8424 \\
1.59 & -0.3082 & 0.2941 & -0.3618 & -0.2558 & -0.508 & -0.4778 & -3.053 & 4.487 & 0 & 0 & ; \ 1.042 \\
-2.374 & 2.558 & -0.2189 & 0.1261 & -0.2458 & -0.4645 & -0.2866 & -0.3911 & -3.053 & 4.487 & 0 & ; \ 1.291 \\
-1.02 & -2.449 & 3.332 & -0.3935 & 0.2488 & -0.4578 & -0.2198 & -0.1933 & -0.3911 & -3.053 & 4.487 & ; \ 1.624 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & ; \ 1
\end{array}\right]$$

Approximate solutions $\widetilde{y}(x)$ *are produced by solving the three systems above using the approach that* $[G^T G; G^T E]$. *The exact and numerical solutions of the two quadrature methods Trapezoidal and Simpson's rules for* $y(x)$ *depending on running time and least square error (L.S.Error) are contrasted in Table (1).*

Table 1: Exact and Numerical solutions for Example 5.1, by using Trapezoidal, and Simpson's 1/3 methods, with setting h = 0.1 (N = 10)

		Numerical Solution	
x	Exact Solution	Trapezoidal	Simpson's 1/3
0	0	-0.01740637608	-0.0154004748302
0.1	0.01	-0.00278964943	-0.00101469446488
0.2	0.04	0.03041153916	0.0319850463231
0.3	0.09	0.08298526974	0.0843508109323
0.4	0.16	0.1552329748	0.156387329149
0.5	0.25	0.2473103564	0.248235887632
0.6	0.36	0.3612654591	0.361730139738
0.7	0.49	0.4940977903	0.494181216087
0.8	0.64	0.6554023651	0.654114981245
0.9	0.81	0.8265440089	0.824863249613
1	1	1.017466155	1.01544798599
	L.S.Error	$1.472057718 \times 10^{-3}$	$1.150079529 \times 10^{-3}$
	R.Time/Sec	0.022386312	0.041325807

Figure (1) *shows compares the exact and numerical solutions of the two quadrature methods Trapezoidal and Simpson's rules.*

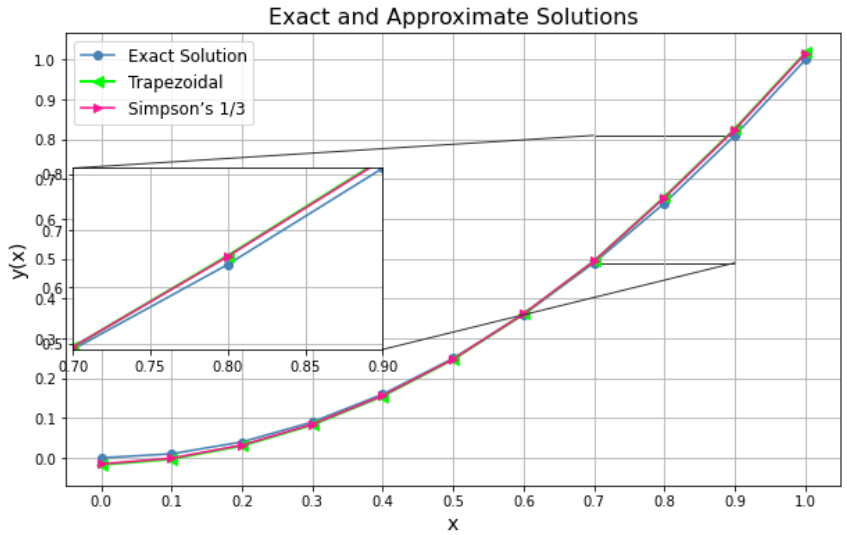

Figure 1: Exact and Numerical solution of Example 5.1

Figure (2) *illustrates the comparison of the absolute errors between the numerical approximations and the exact solution obtained using the AFDT and AFDS methods.*

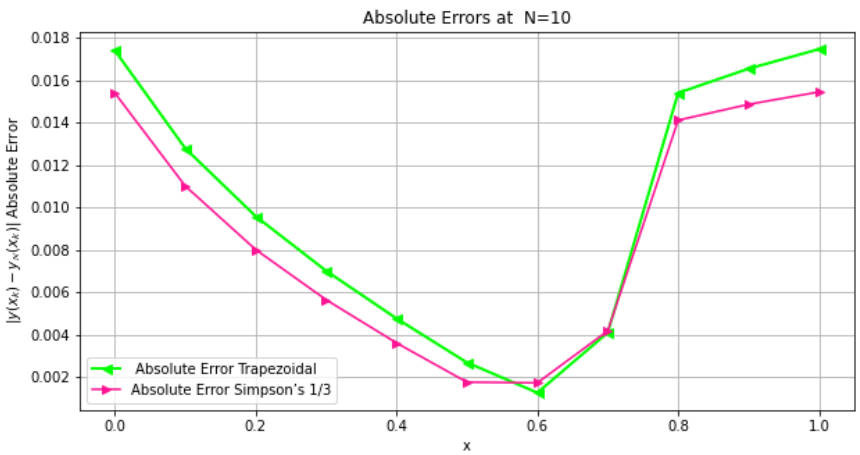

Figure 2: For Example (5.1), the absolute error plot function is $|y(x_k) - y_N(x_k)|$ for $N = 10, h = 0.1$.

The result in Table (2) shows the least square errors and running times for quadrature methods with different values of steps size h.

Table 2: Numerical solutions for Example 5.1, by using Trapezoidal and Simpson's 1/3 for $N = 10, 20, 100$ and 200.

(N,h)	(10,0.1)		(20,0.05)		(100,0.01)		(200,0.005)	
QM	L.S.Error	R.Time/Sec	L.S.Error	R.Time/Sec	L.S.Error	R.Time/Sec	L.S.Error	R.Time/Sec
AFDT	$1.472057718 \times 10^{-3}$	0.022386312	$3.35740570 \times 10^{-4}$	0.129758834	1.4027071×10^{-5}	0.573292493	3.799659×10^{-6}	1.080195188
AFDS	$1.150079529 \times 10^{-3}$	0.041325807	$2.84627348 \times 10^{-4}$	0.108752012	1.3127910×10^{-5}	0.236363410	3.635614×10^{-6}	0.56744503

Example 5.2 *Consider a linear FIFDEs-Delays containing various fractional orders order lies in* $(0,1)$ *on the closed bounded interval* $[0,1]$ *with variable coefficients:*

$$\begin{aligned}
&{}_0^C D_x^{0.7} y(x) + e^x\, {}_0^C D_x^{0.4} y(x-0.9) + x * y(x-0.7) \\
&= \frac{\Gamma(4)}{\Gamma(3.3)} x^{2.3} - 2\frac{\Gamma(2)}{\Gamma(1.3)} x^{0.3} + e^x \Big[\frac{\Gamma(4)}{\Gamma(3.6)} x^{2.6} - \frac{27}{10}\frac{\Gamma(3)}{\Gamma(2.6)} x^{1.6} + \frac{243}{100}\frac{\Gamma(2)}{\Gamma(1.6)} x^{0.6} - 2\frac{\Gamma(2)}{\Gamma(1.6)} x^{0.6} \Big] \\
&- e^x \left[\frac{3e+31}{8} \right] - \frac{6407}{6000} sin(x) + x^4 - \frac{21}{10} x^3 - \frac{797}{250} x^2 + \frac{4057}{1000} x - \frac{15123}{28000} \\
&+ \int_0^1 e^{(x+t)} y(t-0.5)dt + \int_0^1 (x^2+t^3) y(t-0.3)dt + \int_0^1 (sin(x) * t) y(t-0.1)dt
\end{aligned} \tag{5.2}$$

Considering the historical function $\varphi(x) = x(x^2-2)+3$ *and the boundary condition* $y(0)+y(1)=5$. *On the other hand, the precise solution is* $y(x) = x^3 - 2x + 3$. *This example used the recommended method, which is a linear FIFDEs-Delays constant multi-time delay including different fractional orders, to produce the numerical computation at each point. Order is found in* $(0,1)$. *Consider* $x_r = a + rh$, $r = 0, 1, \ldots, N$, $N = 20$, $h = 0.05$. *Given that* $(n,m) = (1,2)$, *the fractional orders are* $\beta = 0.7$ *and* $\alpha_1 = 0.4$, *and the boundary coefficients are* $g_{00} = h_{00} = 1$ *and* $\vartheta_0 = 5$.

Table (3) compares the exact and numerical solutions of the two quadrature methods Trapezoidal and Simpson's rules for $y(x)$ *depending on running time and least square error (L.S.Error). We obtained the numerical results by using the suggested approach to the mentioned situation. The Python application was set up by running the AFDT and AFDS programs.*

Table 3: Exact and Numerical solutions for Example 5.2 , by using Trapezoidal, and Simpson's 1/3 methods, with setting h = 0.05 (N = 20)

		Numerical Solution	
x	Exact Solution	Trapezoidal	Simpson's 1/3
0	3	2.99743701591	2.99793866026
0.05	2.90013	2.8960704935	2.89652179815
0.1	2.801	2.79587233248	2.7962894958
0.15	2.70337	2.69737231707	2.69776088509
0.2	2.608	2.6013215481	2.60168879248
0.25	2.51562	2.50842750984	2.50877919871
0.3	2.427	2.4194152552	2.41975607546
0.35	2.34287	2.33501637646	2.33535069884
0.4	2.264	2.25597575424	2.25630753442
0.45	2.19112	2.18303406645	2.18336768551
0.5	2.125	2.1169344011	2.11727314415
0.55	2.06637	2.05842253489	2.05877035314
0.6	2.016	2.00824551706	2.00860623347
0.65	1.97463	1.96715017044	1.96752746106
0.7	1.943	1.93608180957	1.93644042054
0.75	1.92188	1.91568363218	1.91603743784
0.8	1.912	1.90667853114	1.9070361052
0.85	1.91412	1.90980784916	1.91017488532
0.9	1.929	1.92838797827	1.92826580775
0.95	1.95738	1.96038267687	1.96017621456
1	2	2.00662707268	2.00638200622
L.S.Error		$8.61666384 \times 10^{-4}$	$7.75117677 \times 10^{-4}$
R.Time/Sec		0.14115238189	0.16455769538

The result in Table (4) shows the least square errors and running times for quadrature methods with

different values of steps size h.

Table 4: Numerical solutions for Example 5.2, by using Trapezoidal and Simpson's 1/3 for $N = 10, 100, 200$ and 400.

(N,h)	(10,0.1)		(100,0.01)		(200,0.005)		(400,0.0025)	
QM	L.S.Error	R.Time/Sec	L.S.Error	R.Time/Sec	L.S.Error	R.Time/Sec	L.S.Error	R.Time/Sec
AFDT	$2.602252028 \times 10^{-3}$	0.028834580	6.7883566×10^{-5}	0.644238471	2.2643128×10^{-5}	1.912617921	7.531238×10^{-6}	7.680609703
AFDS	$2.179008843 \times 10^{-3}$	0.067615270	6.5779259×10^{-5}	0.341086387	2.2213984×10^{-5}	1.338727474	7.443529×10^{-6}	4.245245695

Figure (3) *shows compares the exact and numerical solutions of the two quadrature methods Trapezoidal and Simpson's rules.Moreover, the comparison of the absolute errors between the numerical approximations and the actual solution achieved using the AFDT and AFDS methods is shown in* ***Figure (4)***.

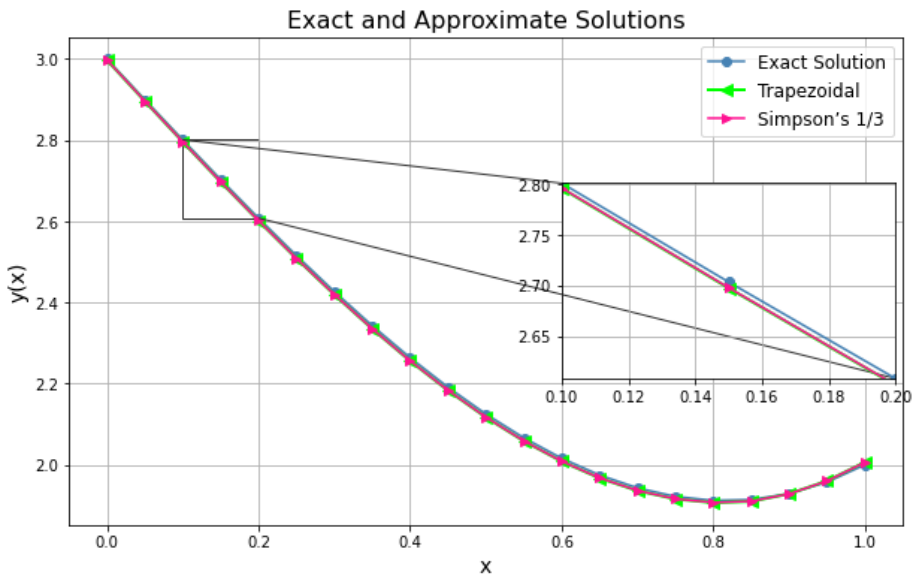

Figure 3: Exact and Numerical solution of Example 5.2

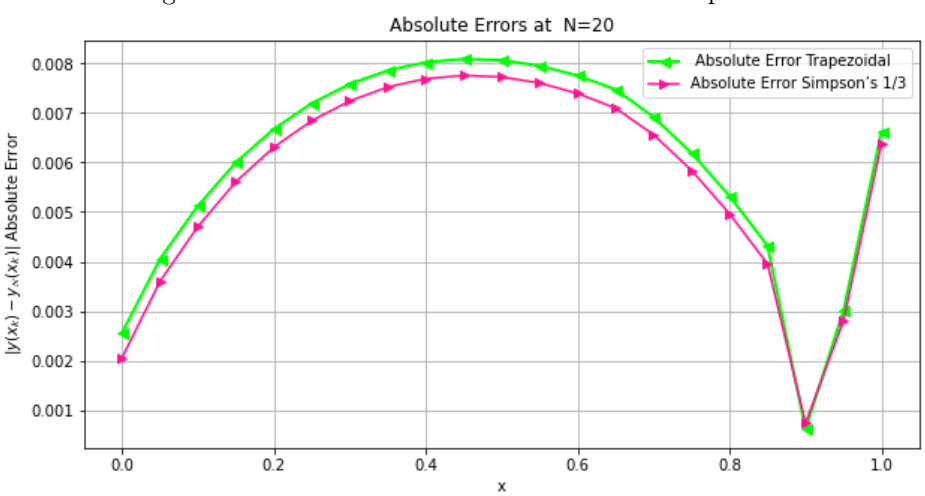

Figure 4: For Example (5.2), the absolute error plot function is $|y(x_k) - y_N(x_k)|$ for $N = 20, h = 0.05$.

Example 5.3 *Consider a linear FIFDEs constant multi-time delays containing various fractional orders order lies in* $(0,1)$ *on the closed interval* $[0,1]$ *with variable coefficients:*

$$
\begin{aligned}
&{}_0^C D_x^{0.9} y(x) + \frac{e^x}{8}\, {}_0^C D_x^{0.8} y(x-0.8) + \frac{sinh(x)}{6} y(x-0.76) = \frac{\Gamma(5)}{\Gamma(4.1)} x^{3.1} \\
&+ \frac{e^x}{8}\Big[\frac{\Gamma(5)}{\Gamma(4.2)} x^{3.2} - \frac{16}{5}\frac{\Gamma(4)}{\Gamma(3.2)} x^{2.2} + \frac{96}{25}\frac{\Gamma(3)}{\Gamma(2.2)} x^{1.2} - \frac{256}{125}\frac{\Gamma(2)}{\Gamma(1.2)} x^{0.2}\Big] \\
&+ \frac{sinh(x)}{6}\Big[x^4 - \frac{76}{25}x^3 + \frac{2166}{625}x^2 - \frac{27436}{15625}x + \frac{3\,776\,943}{3\,125\,000}\Big] - cosh(x)\Big[\frac{89\,096\,881}{200\,000\,000}\Big] \\
&- sin(x)\Big[\frac{4463}{5000}\Big] - e^x\Big[\frac{7377}{28000}\Big] + \Big[\frac{30841}{105000}\Big] + \int_0^1 (t * cosh(x))\, y(t-0.47)\, dt \\
&- \int_0^1 (t^2 - sin(x))\, y(t-0.6)\, dt + \int_0^1 (e^x * t^3)\, y(t-0.2)\, dt
\end{aligned}
\tag{5.3}
$$

Considering the historical function $\varphi(x) = x^4 + \frac{7}{8}$ *and the boundary condition* $2y(0) + y(1) = \frac{29}{8}$. *On the other hand, the precise solution is* $y(x) = x^4 + \frac{7}{8}$. *This example used the recommended method, which is a linear FIFDEs-Delays constant multi-time delay including different fractional orders, to produce the numerical computation at each point. Order is found in* $(0,1)$. *Consider* $x_r = a + rh$, $r = 0, 1, \ldots, N$, $N = 20$, $h = 0.05$. *Given that* $(n, m) = (1, 2)$, *the fractional orders are* $\beta = 0.9$ *and* $\alpha_1 = 0.8$, *and the boundary coefficients are* $g_{00} = 2, h_{00} = 1$ *and* $\vartheta_0 = \frac{29}{8}$.

Table (5) compares the exact and numerical solutions of the two quadrature methods Trapezoidal and Simpson's rules for $y(x)$ *depending on running time and least square error (L.S.Error). We obtained the numerical results by using the suggested approach to the mentioned situation. The Python application was set up by running the AFDT and AFDS programs.*

Table 5: Exact and Numerical solutions for Example 5.3 , by using Trapezoidal, and Simpson's 1/3 methods, with setting h = 0.05 (N = 20)

x	Exact Solution	Numerical Solution	
		Trapezoidal	Simpson's 1/3
0	0.875	0.849302539161	0.850115151655
0.05	0.87500625	0.849865173466	0.85060620892
0.1	0.8751	0.850478674832	0.851149293117
0.15	0.87550625	0.851437255102	0.852036118721
0.2	0.8766	0.853177148771	0.853702201243
0.25	0.87890625	0.856280750739	0.856728879868
0.3	0.8831	0.861478497765	0.861846627622
0.35	0.89000625	0.869649531244	0.86993388216
0.4	0.9006	0.88182225154	0.882019054999
0.45	0.91600625	0.899174607071	0.899279325733
0.5	0.9375	0.923034373276	0.923042204586
0.55	0.96650625	0.954879401425	0.954785040201
0.6	1.0046	0.996337789542	0.99613576748
0.65	1.05350625	1.04918800143	1.04887265583
0.7	1.1151	1.11535896849	1.11492411598
0.75	1.19140625	1.19693027988	1.19636949428
0.8	1.2846	1.30186086441	1.30098646954
0.85	1.39700625	1.42164855689	1.42062903523
0.9	1.5311	1.56382155461	1.56265661252
0.95	1.68950625	1.7311398471	1.72982617094
1	1.875	1.92644822458	1.92498134101
L.S.Error		$1.1874384378 \times 10^{-2}$	$1.1245288735 \times 10^{-2}$
R.Time/Sec		0.1752452850341797	0.1515958309173584

The result in Table (6) shows the least square errors and running times for quadrature methods with different values of steps size h.

Table 6: Numerical solutions for Example 5.3, by using Trapezoidal and Simpson's 1/3 for $N = 20, 50, 100$ and 150.

(N,h)	(20,0.05)		(50,0.02)		(100,0.001)		(150,0.00$\overline{6}$)	
QM	L.S.Error	R.Time/Sec	L.S.Error	R.Time/Sec	L.S.Error	R.Time/Sec	L.S.Error	R.Time/Sec
AFDT	$1.1874384378 \times 10^{-2}$	0.175245285	$3.495595560 \times 10^{-3}$	0.181525230	$1.363315189 \times 10^{-3}$	0.592600584	$8.82322880 \times 10^{-4}$	1.536804437
AFDS	$1.1245288735 \times 10^{-2}$	0.151595830	$3.417110428 \times 10^{-3}$	0.146749019	$1.345427611 \times 10^{-3}$	0.48625016	$8.74437224 \times 10^{-4}$	1.431174278

Figure (5) *shows compares the exact and numerical solutions of the two quadrature methods Trapezoidal and Simpson's rules. Moreover,* ***Figure (6)*** *illustrates the comparison of the absolute errors between the numerical approximations and the exact solution obtained using the AFDT and AFDS methods.*

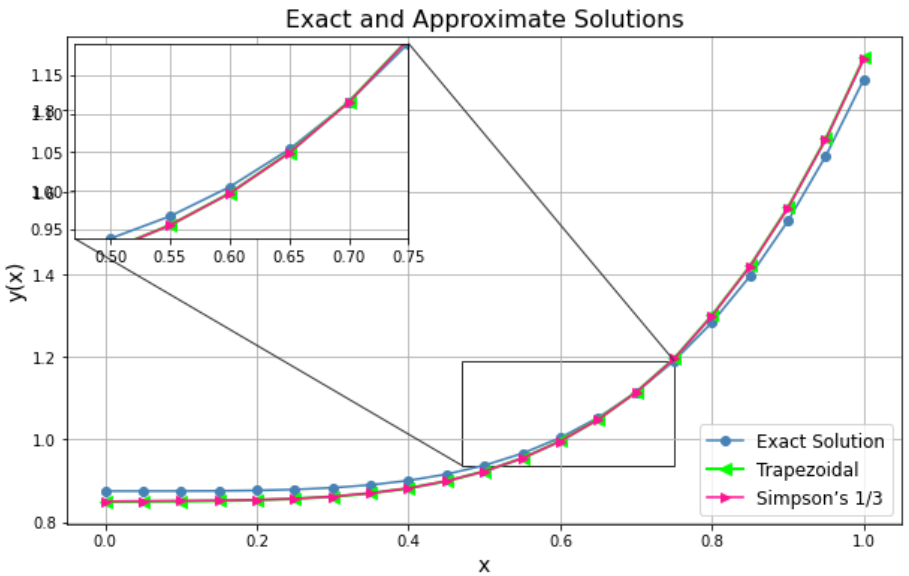

Figure 5: Exact and Numerical solution of Example 5.3

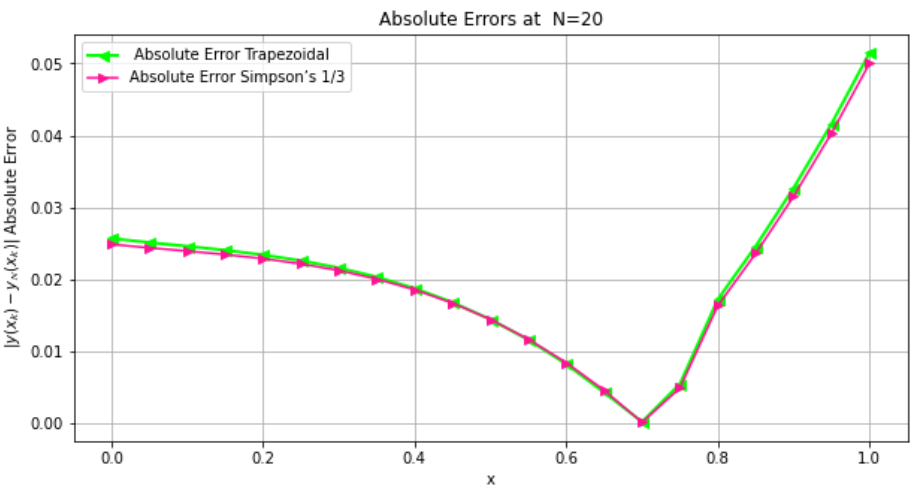

Figure 6: For Example (5.3), the absolute error plot function is $|y(x_k) - y_N(x_k)|$ for $N = 20, h = 0.05$.

6. Discussion

In this study, a forward finite difference approach (FFDA) for a Caputo derivative sense is used to approximate the solution of linear FIFDEs-Delays with constant coefficients, including Trapezoidal and Simpson rules. A Python computer program was written to create the new algorithms **AFDT** and **AFDS**, with multiple examples provided for demonstration. In order to compare computation speed and accuracy, the least square error and running time are also given in tabular form. We get to the following conclusions:

1. The AFDS method provides greater accuracy than the AFDT technique when the step sizes are equal.
2. Both the procedure and the step length h affect how accurate the findings are; that is, the accuracy increases as h decreases (see tables (2), (4) and (6)).
3. We can classify these two algorithms as multi- step methods.

References

1. Mathews, J. H. and Fink, K. D., *Numerical Methods Using MATLAB*, Pearson Prentice Hall, Upper Saddle River, NJ, (2004).
2. Delves, L. M. and Mohamed, J. L., *Computational Methods for Integral Equations*, Cambridge University Press, (1985).
3. Al-Rawi, S. N., *Numerical Solutions of First Kind Integral Equations of Convolution Type*, M.Sc. Thesis, University of Technology, Baghdad, (1995).
4. Al-Nasir, R. H., *Numerical Solution of Volterra Integral Equations of the Second Kind*, M.Sc. Thesis, University of Technology, (1999).
5. Saberi-Nadjafi, J. and Heidari, M., *Solving linear integral equations of the second kind with repeated modified trapezoid quadrature method*, Appl. Math. Comput. 189, 980–985, (2007).
6. Rahbar, S. and Hashemizadeh, E., *A computational approach to the Fredholm integral equation of the second kind*, Proc. World Congress on Engineering, Vol. 2, 933–937, (2008).
7. Jafari, M. E. and Tavassoli Kajani, M., *Nonlinear Fredholm integral equation of the second kind with quadrature methods*, J. Math. Extension, (2010).
8. Isaacson, S. A. and Kirby, R. M., *Numerical solution of linear Volterra integral equations of the second kind with sharp gradients*, J. Comput. Appl. Math. 235, 4283–4301, (2011).
9. Saadati, R., Raftari, B., Adibi, H., Vaezpour, S. M., and Shakeri, S., *A comparison between the variational iteration method and trapezoidal rule for solving linear integro-differential equations*, World Appl. Sci. J. 4, 321–325, (2008).
10. Al-Jawary, M. A. W., *Numerical Methods for System of Integral Equations*, M.Sc. Thesis, University of Baghdad, (2005).
11. Al-Salhi, B. F. J., *Numerical Solution of Non-Linear Volterra Integral Equations of the Second Kind*, M.Sc. Thesis, University of Technology, (2000).
12. Salih, S. A. H. and Ahmed, S. S., *Numerical treatment of the most general linear Volterra integro-fractional differential equations with Caputo derivatives by quadrature methods*, J. Math. Comput. Sci. 2, 1293–1311, (2012).
13. Podlubny, I., *Fractional Differential Equations*, Academic Press, San Diego, (1999).
14. Ahmed, S. S., *On System of Linear Volterra Integro-Fractional Differential Equations*, Ph.D. Thesis, University of Sulaimani, (2009).
15. Kilbas, A. A., Srivastava, H. M., and Trujillo, J. J., *Theory and Applications of Fractional Differential Equations*, Elsevier, (2006).
16. Ahmed, M. R., Ahmed, S. S., and Sabir, P. O., *Approximate solution to the system of nonlinear Volterra integro-fractional differential equations with variable coefficients using linear B-spline functions*, J. Southwest Jiaotong Univ. 59, (2024).
17. Miller, K. S. and Ross, B., *An Introduction to the Fractional Calculus and Fractional Differential Equations*, Wiley–Interscience, New York, (1993).
18. Ahmed, S. S. and MohammedFaeq, S. J., *Bessel collocation method for solving Fredholm–Volterra integro-fractional differential equations of multi-high order in the Caputo sense*, Symmetry 13, 2354, (2021).
19. Zhou, Y., *Basic Theory of Fractional Differential Equations*, World Scientific, (2023).
20. Weilbeer, M., *Efficient Numerical Methods for Fractional Differential Equations and Their Analytical Background*, Clausthal-Zellerfeld, Germany, (2006).

21. Ibrahim, G. H., *Numerical Treatments of System of Fredholm Integral Equations*, M.Sc. Thesis, University of Technology, (2006).

22. Atkinson, K. E., *An Introduction to Numerical Analysis*, 2nd ed., John Wiley & Sons, (1989).

23. De Meyer, H., Bocher, P., Fack, V., and Vanden Berghe, G., *A parallel fifth-order algorithm for the numerical solution of Volterra equations*, J. Comput. Appl. Math. 71, 115–124, (1996).

24. Dashne, Z. C., *Numerical solutions for the most general multi-higher fractional order linear integro-differential equations of Fredholm type in Caputo sense*, M.Sc. Thesis, University of Sulaimani, (2017).

Razaw Salam Rasul,
Department of Mathematics,
College of Science, University of Sulaimani,
Sulaymaniyah, 46001, Iraq.
E-mail address: razaw.ghafoor@univsul.edu.iq

and

Shazad Shawki Ahmed,
Department of Mathematics,
College of Science, University of Sulaimani,
Sulaymaniyah, 46001, Iraq.
E-mail address: shazad.ahmed@univsul.edu.iq