# Quarter-Sweep Preconditioned Relaxation Method, Algorithm and Efficiency Analysis for Fractional Mathematical Equation

fractal and fractional

MDPI

*Article*

# Quarter-Sweep Preconditioned Relaxation Method, Algorithm and Efficiency Analysis for Fractional Mathematical Equation

**Andang Sunarto [1,*], Praveen Agarwal [2,3,4], Jumat Sulaiman [5], Jackel Vui Lung Chew [6,*] and Shaher Momani [7]**

[1] Tadris Matematika, IAIN Bengkulu, Bengkulu 38211, Indonesia
[2] Department of Mathematics, Anand International College of Engineering, Jaipur 303012, India; goyal.praveen2011@gmail.com
[3] Nonlinear Dynamics Research Center (NDRC), Ajman University, Ajman 346, United Arab Emirates
[4] International Center for Basic and Applied Sciences, Jaipur 302029, India
[5] Faculty of Science and Natural Resources, Universiti Malaysia Sabah, Kota Kinabalu 88400, Malaysia; jumat@ums.edu.my
[6] Faculty of Computing and Informatics, Universiti Malaysia Sabah Labuan International Campus, Labuan 87000, Malaysia
[7] Department of Mathematics, Faculty of Science, University of Jordan, Amman 11942, Jordan; S.Momani@ju.edu.jo
* Correspondence: andang99@gmail.com (A.S.); jackelchew93@ums.edu.my (J.V.L.C.)

**Abstract:** Research into the recent developments for solving fractional mathematical equations requires accurate and efficient numerical methods. Although many numerical methods based on Caputo's fractional derivative have been proposed to solve fractional mathematical equations, the efficiency of obtaining solutions using these methods when dealing with a large matrix requires further study. The matrix size influences the accuracy of the solution. Therefore, this paper proposes a quarter-sweep finite difference scheme with a preconditioned relaxation-based approximation to efficiently solve a large matrix, which is based on the establishment of a linear system for a fractional mathematical equation. The paper presents the formulation of the quarter-sweep finite difference scheme that is used to approximate the selected fractional mathematical equation. Then, the derivation of a preconditioned relaxation method based on a quarter-sweep scheme is discussed. The design of a C++ algorithm of the proposed quarter-sweep preconditioned relaxation method is shown and, finally, efficiency analysis comparing the proposed method with several tested methods is presented. The contributions of this paper are the presentation of a new preconditioned matrix to restructure the developed linear system, and the derivation of an efficient preconditioned relaxation iterative method for solving a fractional mathematical equation. By simulating the solutions of time-fractional diffusion problems with the proposed numerical method, the study found that computing solutions using the quarter-sweep preconditioned relaxation method is more efficient than using the tested methods. The proposed numerical method is able to solve the selected problems with fewer iterations and a faster execution time than the tested existing methods. The efficiency of the methods was evaluated using different matrix sizes. Thus, the combination of a quarter-sweep finite difference method, Caputo's time-fractional derivative, and the preconditioned successive over-relaxation method showed good potential for solving different types of fractional mathematical equations, and provides a future direction for this field of research.

**Keywords:** Caputo's time-fractional derivative; finite difference scheme; iterative relaxation method; time-fractional diffusion equation; preconditioned matrix

check for updates

**Citation:** Sunarto, A.; Agarwal, P.; Sulaiman, J.; Chew, J.V.L.; Momani, S. Quarter-Sweep Preconditioned Relaxation Method, Algorithm and Efficiency Analysis for Fractional Mathematical Equation. *Fractal Fract.* **2021**, *5*, 98. https://doi.org/10.3390/fractalfract5030098

Academic Editor: Paul Eloe

Received: 23 June 2021
Accepted: 10 August 2021
Published: 14 August 2021

## 1. Introduction

Research into the recent developments for solving fractional mathematical equations requires accurate and efficient numerical methods. The accuracy and efficiency of the numerical method used to solve a fractional mathematical equation determine the precision when interpreting the behaviours of physical phenomena. Accurate and efficient methods enable researchers to explore and solve a greater number of complex fractional mathematical models. Many numerical methods based on Caputo's fractional derivative have been proposed to approximate the solution of the modelled fractional mathematical equations. For instance, ref. [1] proposed two numerical methods, the non-standard finite difference method and the generalized Euler method, for solving a mathematical model of drug resistance during treatment for human immunodeficiency virus, using the Caputo approach. Subsequently, ref. [2] studied the conversion of an existing tobacco smoking model into a fractional-order Caputo model, and used the combination of the fourth-order Runge–Kutta and Adams–Bashforth–Moulton methods to solve the developed model. In another article, ref. [3] developed a time–space fractional hyperbolic bioheat transfer model for non-Fourier bioheat transfer in living biological tissues during laser irradiation using Caputo's definition. They applied an L1 finite difference approximation to Caputo's time-fractional derivative and a central difference approximation to Riesz's space-fractional derivative for the solution of the model. In addition, ref. [4] investigated Caputo's fractional Maxwell model of unsteady fluid flow and heat transfer of the natural convection of a viscoelastic non-Newtonian fluid using a finite difference method.

Of the many existing numerical methods, the finite difference method, widely known as FDM, is one of the most popular numerical methods used to solve fractional mathematical equations using the Caputo approach. From our review of several applications of FDM to solve Caputo's time-fractional mathematical equations, ref. [5] utilized the implicit FDM to solve the time-fractional diffusion equation with a time-invariant type variable. Subsequently, ref. [6] developed a fourth-order FDM to solve a time-fractional diffusion equation after transforming the fractional mathematical equation into a Volterra integro-differential equation via a Laplace transform. In addition, ref. [7] proposed a weighted FDM to effectively solve a system of variable-order time-fractional two-dimensional Burgers' equations. FDM can also be combined with spline approximation to solve the time-fractional stochastic advection equation [8]. These studies are examples of the numerous works published by global researchers. The abundance of related research in the literature has motivated numerous researchers to propose more accurate and efficient numerical methods for solving fractional mathematical equations.

In our view, the efficiency of numerical methods used to obtain the solution of a fractional mathematical equation when dealing with a large matrix requires further study because the accuracy of the solution is influenced by the matrix size. Therefore, this paper proposes a quarter-sweep finite difference scheme with a preconditioned relaxation-based approximation to efficiently solve a large matrix, which results from establishing a linear system for a fractional mathematical equation. The quarter-sweep finite difference scheme is suitable for solving a fractional mathematical equation because, among other reasons, the derivation of the scheme is similar to that of an implicit FDM, which makes it compatible with Caputo's time-fractional approximation. The quarter-sweep scheme is derived by modifying an implicit FDM by skipping three grid points for two consecutive unknown grid points. The values of the unknown grid points can be computed using the iterative method until they converge. Then, the remaining grid points resulting from the skipping procedure are computed directly using the approximation function [9–11]. Moreover, due to the combination of iterative and direct computation, the quarter-sweep scheme has a lower computational burden compared to the implicit and high-order FDM. This paper introduces a preconditioned relaxation method to improve the convergence rate of the iterations, thereby resulting in a highly efficient numerical method. The contributions of the paper are the presentation of a new preconditioned matrix to restructure the developed linear system and the derivation of an efficient preconditioned relaxation iterative method for solving a fractional mathematical equation.

To investigate the efficacy of this new numerical method, a well-known fractional mathematical equation, namely, the fractional diffusion equation (FDE), was selected as a mathematical problem. FDE is a fractional mathematical equation that has been successfully used in the mathematical modelling of physical phenomena. Due to the limitation of the classic diffusion equation in the modelling of the anomalous diffusion process, the theory and application of fractional diffusion have evolved. Hence, FDE has been used to successfully model shipping water events [12], image denoising [13], dye-sensitized solar cells [14], diffusion of soluble substances [15], groundwater pollution [16], option pricing and risk calculation [17], and signal smoothing performance analysis [18]. A better understanding of FDE models can be achieved using an efficient numerical method called the quarter-sweep preconditioned relaxation method. It should be noted that this paper extends the works from [19] and [20], which implemented the standard implicit FDM with preconditioned relaxation and the half-sweep difference scheme with preconditioned relaxation, respectively, for solving the time FDE. The work in [19] showed the improvement in the number of iterations and execution time after implementing a preconditioned successive over-relaxation with implicit FDM using the Caputo approach to solve the time FDE. Another article [20] applied a computational complexity reduction technique called the half-sweep iteration to successfully reduce the computational cost using preconditioned successive over-relaxation, and eventually improve the efficiency of Caputo's finite difference scheme. This paper extends these works by employing the quarter-sweep FDM to reduce the computational cost of solving the FDE using a preconditioned successive over-relaxation when dealing with a large matrix.

This paper is organized as follows: Section 2 presents Caputo's fractional operator, one of the important definitions in the fractional derivative theory. In Section 3, an approximation to a general time FDE is formulated using Caputo's fractional operator and the quarter-sweep numerical discretization procedure. Section 4 explains the stability of the approximation to the considered mathematical equation. Section 5 discusses the convergence of the quarter-sweep finite difference approximation in the Caputo sense. In Section 6, the concept of preconditioned successive over-relaxation (PSOR) used to formulate the quarter-sweep PSOR (QSPSOR) iterative method is discussed. Section 7 shows the implementation and application of an algorithm written in C++ programming language for the numerical experiment and simulation with QSPSOR, and provides a comparison with selected existing methods. Finally, a conclusion is given in Section 8.

## 2. Preliminary

Formulating a quarter-sweep approximation to a time FDE requires good knowledge of fractional calculus theory and several useful operators. In the preliminary stage of this paper, Caputo's fractional operator is used to approximate the time-fractional derivative term. The definition of Caputo's derivative with a fractional order is as follows:

**Definition 1 ([21]).** *Let $m-1<\alpha<m$ , where $m$ is an element of the set of natural numbers, and $\alpha$ is a real number, and a function $f(x)$ such that $D_0^\alpha f(x)$ exists. Caputo's fractional operator can be defined in the form of:*

$$D_0^\alpha f(x) = \int_0^x \frac{f^{(m)}(r)}{(x-r)^{\alpha-(m-1)}}\,dr. \tag{1}$$

Using Definition 1, a time-fractional derivative term in a differential equation can be approximated by:

$$D_t^\alpha z(x,t) = \frac{1}{\Gamma(1-\alpha)}\int_0^t \frac{\partial z(x,\tau)}{\partial t}(t-\tau)^{-\alpha}\,d\tau, \tag{2}$$

where $0<\alpha<1$ and $z(x,t)>0$.

The next section discusses the formulation of a quarter-sweep finite difference approximation to the time FDE, which uses Equation (2) to approximate the time-fractional derivative.

## 3. Mixed Caputo Fractional Operator and Quarter-Sweep Discretization Scheme

To demonstrate the formulation of the mixed Caputo fractional operator and quarter-sweep discretization scheme, let us consider a well-known model of the time FDE, namely:

$$\frac{\partial^{\alpha} z(x,t)}{\partial t^{\alpha}} = a(x)\frac{\partial^2 z(x,t)}{\partial x^2} + b(x)\frac{\partial z(x,t)}{\partial x} + c(x)z(x,t), \tag{3}$$

where $a(x),\ b(x)$, and $c(x)$ are either predefined functions, constants, or mixed.

A suitable boundary condition must be imposed on Equation (3) before the solution can be derived using an FDM. In this paper, a Dirichlet-type boundary condition is used to construct a solution domain. Furthermore, the solution domain is restricted to the usual finite domain $0 \leq x \leq \gamma$ because the aim of this paper is to investigate of the efficacy of a new iterative method developed from the use of Equation (2) and a quarter-sweep FDM. Based on Equation (3), Caputo's fractional approximation to the fractional order in time is illustrated as follows:

$$\frac{\partial^{\alpha} z(x_i,t_n)}{\partial t^{\alpha}} = \frac{1}{\Gamma(1-\alpha)}\int_0^{t_n} \frac{\partial z(x_i,\tau)}{\partial t}(t_n-\tau)^{-\alpha}\, d\tau. \tag{4}$$

Equation (4) can be derived further and simplified to:

$$\begin{aligned}\frac{\partial^{\alpha} z(x_i,t_n)}{\partial t^{\alpha}} &= \frac{1}{\Gamma(1-\alpha)(1-\alpha)k^{\alpha}}\sum_{j=1}^{n}\left(Z_{i,j}-Z_{i,j-1}\right)\left[(n-(j-1))^{1-\alpha}-(n-j)^{1-\alpha}\right] \\ &\quad+\frac{1}{\Gamma(1-\alpha)(1-\alpha)}\sum_{j=1}^{n}\left[(n-(j-1))^{1-\alpha}-(n-j)^{1-\alpha}\right]O\left(k^{2-\alpha}\right).\end{aligned} \tag{5}$$

Next, by defining:

$$\sigma_{\alpha,k} = \frac{1}{\Gamma(1-\alpha)(1-\alpha)k^{\alpha}}, \tag{6}$$

where $k$ is a real number, and:

$$\omega_{\alpha,j} = j^{1-\alpha} - (j-1)^{1-\alpha}, \tag{7}$$

a discrete approximation to the fractional order in time becomes:

$$\frac{\partial^{\alpha} z(x_i,t_n)}{\partial t^{\alpha}} = \sigma_{\alpha,k}\sum_{j=0}^{n}\omega_{\alpha,j}\left(Z_{i,n-(j-1)}-Z_{i,n-j}\right) + \frac{1}{\Gamma(1-\alpha)(1-\alpha)}n^{1-\alpha}O\left(k^{2-\alpha}\right), \tag{8}$$

and, finally:

$$\frac{\partial^{\alpha} z(x_i,t_n)}{\partial t^{\alpha}} = \sigma_{\alpha,k}\sum_{j=0}^{n}\omega_{\alpha,j}\left(Z_{i,n-(j-1)}-Z_{i,n-j}\right) + O(k). \tag{9}$$

To discretize the right-hand side part of Equation (3) using a quarter-sweep implicit scheme of FDM, let the solution domain of the problem be partitioned uniformly. Let some integers $M > 0$ and $N > 0$ denote the grid sizes in space and time, respectively. Similar to the standard finite difference framework, we use $h = \gamma/M$ and $k = T/N$, respectively. Then, a uniform mesh of the solution can be formed using the distributed points in the space $[0,\ \gamma]$ as $x_i = ih, i = 1,2,\ldots,M-1$, and the distributed points in the time $[0,\ T]$ are labelled $t_n = nk, n = 1,2,\ldots,N$. All of the unknown values of the solution function $z(x_i,t_n)$ located at these grid points are represented by $Z_{i,n}$.

Using Equation (9), in combination with a quarter-sweep implicit scheme, the quarter-sweep Caputo fractional approximation of Equation (3) to the point centered at $z(x_i, t_n) = z(ih, nk)$ can be formulated and written as:

$$\begin{aligned}&\sigma_{\alpha,k} \sum_{j=0}^{n} \omega_{\alpha,j}\left(Z_{i,n-(j-1)} - Z_{i,n-j}\right) + O(k) \\ &= \tfrac{a_i}{16h^2}\left(Z_{i-4,n} - 2Z_{i,n} + Z_{i+4,n}\right) + \tfrac{b_i}{8h}\left(Z_{i+4,n} - Z_{i-4,n}\right) + c_i Z_{i,n} + O\left(h^2\right),\end{aligned} \tag{10}$$

where $i = 4, 6 \ldots, m-4$.

The approximation structure presented in Equation (10) is similar to a standard implicit approximation equation, with the exception that the distance between any two points is quadrupled. Equation (10) is also consistent with the first-order accuracy in time and second-order accuracy in space. When Equation (10) is written in a particular form, for instance, at time level $n \geq 2$, one can obtain:

$$\sigma_{\alpha,k} \sum_{j=2}^{n} \omega_{\alpha,j}\left(Z_{i,n-(j-1)} - Z_{i,n-j}\right) = a_i^* Z_{i-4,n} + b_i^* Z_{i,n} + c_i^* Z_{i+4,n}, \tag{11}$$

where:

$$a_i^* = \frac{a_i}{16h^2} - \frac{b_i}{8h},\ b_i^* = c_i - \frac{a_i}{8h^2},\ c_i^* = \frac{a_i}{16h^2} + \frac{b_i}{8h}. \tag{12}$$

Moreover, for $n = 1$, one can obtain:

$$F_{i,1} = -a_i^* Z_{i-4,1} + b_i^{**} Z_{i,1} - c_i^* Z_{i+4,1}, \tag{13}$$

where:

$$b_i^{**} = \sigma_{\alpha,k} - b_i^*, F_{i,1} = \sigma_{\alpha,k} Z_{i,0}. \tag{14}$$

Furthermore, using Equation (14), a corresponding linear system can be constructed properly in the form of a matrix as:

$$A\widetilde{Z} = \widetilde{F}, \tag{15}$$

where:

$$A = \begin{bmatrix} b_4^{**} & -c_4^* & & & & \\ -a_8^* & b_8^{**} & -c_8^* & & & \\ & -a_{12}^* & b_{12}^{**} & -c_{12}^* & & \\ & & \ddots & \ddots & \ddots & \\ & & & -a_{M-8}^* & b_{M-8}^{**} & -c_{M-8}^* \\ & & & & -a_{M-4}^* & b_{M-4}^{**} \end{bmatrix}, \tag{16}$$

$$\widetilde{Z} = \left[Z_{4,1},\ Z_{8,1},\ Z_{12,1}, \ldots, Z_{M-8,1}, Z_{M-4,1}\right]^T, \tag{17}$$

and:

$$\widetilde{F} = \left[F_{4,1} + a_4^* F_{0,1},\ F_{8,1},\ F_{12,1}, \ldots, F_{M-8,1}, F_{M-4,1} + c_{M-4}^* F_{M,1}\right]^T. \tag{18}$$

## 4. Stability of Caputo's Fractional Approximation with a Quarter-Sweep Scheme

This section discusses the stability analysis of the quarter-sweep Caputo fractional approximation to the time FDE as shown in Equation (3). In this paper, two common stability verification approaches are applied: Von Neumann's method and the Lax equivalence theorem. The developed theorem for the stability of the quarter-sweep Caputo fractional approximation is presented below.

**Theorem 1.** *The quarter-sweep implicit approximation with Caputo's fractional operator for* $0 < \alpha < 1$, *on the finite space* $0 \leq x \leq 1$ *and all* $t > 0$, *is unconditionally stable.*

**Proof of Theorem 1.** Suppose that a solution function of Equation (3) has the form:

$$Z_{i,n} = \xi_n e^{I(\lambda i h)}, \tag{19}$$

where $I = \sqrt{-1}$ and $\lambda$ is a real number. Using Equation (19), the quarter-sweep Caputo fractional approximation shown by Equation (11) can be rewritten as:

$$\sigma_{\alpha,k}\sum_{j=2}^{n}\omega_{\alpha,j}\left(\xi_{n-(j-1)}e^{I(\lambda i h)} - \xi_{n-j}e^{I(\lambda i h)}\right) = a_i^*\xi_n e^{I(\lambda(i-4)h)} + b_i^*\xi_n e^{I(\lambda i h)} + c_i^*\xi_n e^{I(\lambda(i+4)h)}. \tag{20}$$

Equation (20) can be transformed into:

$$\begin{gathered} -\sigma_{\alpha,k}\xi_{n-1}e^{I(\lambda i h)} + \sigma_{\alpha,k}\sum_{j=2}^{n}\ \omega_{\alpha,j}\left(\xi_{n-(j-1)}e^{I(\lambda i h)} - \xi_{n-j}e^{I(\lambda i h)}\right) = \\ a_i^*\xi_n e^{I(\lambda(i-4)h)} + (b_i^* - \sigma_{\alpha,k})\xi_n e^{I(\lambda i h)} + c_i^*\xi_n e^{I(\lambda(i+4)h)}. \end{gathered} \tag{21}$$

By simplifying and rearranging the distinct terms in Equation (21), one can obtain:

$$-\sigma_{\alpha,k}\xi_{n-1} + \sigma_{\alpha,k}\sum_{j=2}^{n}\omega_{\alpha,j}\left(\xi_{n-(j-1)} - \xi_{n-j}\right) = \xi_n(((a_i^* + c_i^*)\cos 4\lambda h) + (b_i^* - \sigma_{\alpha,k})), \tag{22}$$

which can be reduced to:

$$\xi_n = \frac{\xi_{n-1} + \sum_{j=2}^{n}\ \omega_{\alpha,j}\left(\xi_{n-j} - \xi_{n-(j-1)}\right)}{\left(1 - \frac{\left(a_i^*+c_i^*\right)\cos 4\lambda h + b_i^*}{\sigma_{\alpha,k}}\right)}. \tag{23}$$

From Equation (23), for all values of $j,\ n,\ \omega, \alpha,\ \lambda, h, \sigma$, and $k$, it follows that:

$$\xi_1 \leq \xi_0, \tag{24}$$

and when $n = 2$, we have:

$$\xi_2 \leq \xi_1 + \omega_{\alpha,2}(\xi_0 - \xi_1), \tag{25}$$

which can generally be stated as:

$$\xi_n \leq \xi_{n-1} + \sum_{j=2}^{n}\ \omega_{\alpha,j}\left(\xi_{n-j} - \xi_{n-(j-1)}\right), n \geq 2. \tag{26}$$

Since:

$$\xi_n \leq \xi_{n-1} + \sum_{j=2}^{n}\omega_{\alpha,j}\left(\xi_{n-j} - \xi_{n-(j-1)}\right) \leq \xi_{n-j}, \tag{27}$$

the inequalities in Equations (24)–(26) imply:

$$\xi_n \leq \xi_{n-1} \leq \xi_{n-2} \leq \cdots \leq \xi_1 \leq \xi_0. \tag{28}$$

Thus, the stability of the quarter-sweep scheme via Caputo's time-fractional operator is established as:

$$\xi_n = |Z_{i,n}| \leq \xi_0 = |Z_{i,0}|. \tag{29}$$

□

## 5. Convergence of Caputo's Fractional Approximation with a Quarter-Sweep Scheme

This section discusses the convergence analysis of the quarter-sweep Caputo fractional approximation to the time FDE in Equation (3). To begin the analysis, we define $e_{i,n} =$

$z(x_i, t_n) - Z_{i,n}, i = 1, 2, \ldots, M-1, n = 1, 2, \ldots, N$, where $z_{i,n}$ and $Z_{i,n}$ are the exact and approximate solutions, respectively. Substitution of $e_{i,n}$ into Equations (11) and (13) gives:

$$\sigma_{\alpha,k} \sum_{j=2}^{n} \omega_{\alpha,j} \Big( e_{i,n-(j-1)} - e_{i,n-j} \Big) = a_i^* e_{i-4,n} + b_i^* e_{i,n} + c_i^* e_{i+4,n} + R_{i,n}, n \geq 2, \tag{30}$$

where:

$$R_{i,n} = \sigma_{\alpha,k} \sum_{j=2}^{n} \omega_{\alpha,j} \Big( z\Big(x_i, t_{n-(j-1)}\Big) - z\big(x_i, t_{n-j}\big) \Big) - a_i^* z(x_{i-4}, t_n) - b_i^* z(x_i, t_n) - c_i^* z(x_{i+4}, t_n), \tag{31}$$

and:

$$\sigma_{\alpha,k} e_{i,0} = -a_i^* e_{i-4,1} + (\sigma_{\alpha,k} - b_i^*) e_{i,1} - c_i^* e_{i+4,1} + R_{i,1}. \tag{32}$$

Then, we have:

$$\frac{\partial^\alpha z(x_i, t_n)}{\partial t^\alpha} + C_1 k = \sigma_{\alpha,k} \sum_{j=0}^{n} \omega_{\alpha,j} \Big( z\Big(x_i, t_{n-(j-1)}\Big) - z\big(x_i, t_{n-j}\big) \Big), \tag{33}$$

and the quarter-sweep first and second-order central difference operators as follows:

$$\frac{\partial^2 z(x_i, t_n)}{\partial x^2} + C_2 h^2 = \frac{z(x_{i-4}, t_n) - 2z(x_i, t_n) + z(x_{i+4}, t_n)}{16h^2}, \tag{34}$$

and:

$$\frac{\partial z(x_i, t_n)}{\partial x} + C_3 h = \frac{z(x_{i-4}, t_n) - z(x_{i+4}, t_n)}{8h}. \tag{35}$$

Hence:

$$R_{i,n} = \frac{1}{\sigma_{\alpha,k}} \left[ \frac{\partial^\alpha z(x_i, t_n)}{\partial t^\alpha} - a(x) \frac{\partial^2 z(x_i, t_n)}{\partial x^2} - b(x) \frac{\partial z(x_i, t_n)}{\partial x} - c(x) z(x_i, t_n) \right] + C_1 k^{1+\alpha} + k^\alpha \Big( C_2 h^2 + C_3 h \Big), \tag{36}$$

and also:

$$|R_{i,n}| \leq C\Big(k^{1+\alpha} + k^\alpha \Big(h^2 + h\Big)\Big), i = 1, 2, \ldots M-1, n = 1, 2, \ldots, N, \tag{37}$$

where $C$ is an arbitrary constant.

Because $kn \leq N$ is finite, we obtain the following theorem.

**Theorem 2.** *Let $Z_{i,n}$ be the approximation of the exact value $z(x_i, t_n)$, computed by the quarter-sweep implicit approximation with Caputo's fractional operator. Then, there is a positive constant $C$ such that:*

$$|z(x_i, t_n) - Z_{i,n}| \leq C\Big(k + h^2\Big), i = 1, 2, \ldots M-1, n = 1, 2, \ldots, N. \tag{38}$$

## 6. Concept and Formulation of Quarter-Sweep Preconditioned Relaxation Method

The presented PSOR iterative method is a continuation of the derivation of the quarter-sweep finite difference approximation via Caputo's fractional derivative. The formulation of the PSOR method begins with the linear system, as shown by Equation (15). The coefficient matrix considered in Equation (15) has a large scale and is sparse. Thus, the proposed PSOR iterative method solves Equation (15) using the following matrix restructure. Firstly, a preconditioned linear system can be defined as:

$$A' \widetilde{Z} = \widetilde{F}', \tag{39}$$

where the new matrices $A' = PAP^T$, $\widetilde{F}' = P\widetilde{F}$, and $\widetilde{Z} = P^T\psi$. The preconditioned matrix labelled $P$ that is used in this paper is given by:

$$P = I + S, \tag{40}$$

where:

$$S = \begin{bmatrix} 0 & -r_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -r_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & -r_3 & 0 & 0 \\ 0 & 0 & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & 0 & 0 & -r_{m-1} \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{41}$$

Secondly, let us consider that the new coefficient matrix $A'$ in Equation (39) can be defined as the summation of the matrix components of:

$$A' = Di - Lo - Va, \tag{42}$$

where $Di$, $Lo$, and $Va$ are diagonal, lower triangular, and upper triangular matrices, respectively. By combining Equations (39) and (42), the resultant QSPSOR iteration method can be derived as:

$$\psi^{(\eta+1)} = (Di - \omega Lo)^{-1}((1-\omega)Di + \omega Va)\psi^{(\eta)} + (Di - \omega Lo)^{-1}\widetilde{F}', \tag{43}$$

where $\psi^{(\eta+1)}$ represents an unknown vector at the $(\eta+1)$th iteration. The algorithm of the QSPSOR method used to compute the solution of the time FDE is described in the following Table 1.

**Table 1.** QSPSOR computational algorithm.

| | **Set $\psi^{(0)}$ = 0 and the Tolerance Error $\varepsilon$ = $10^{-10}$.** |
|---|---|
| (i) | For $n = 1, 2, \ldots, N$ and for $i = 4,\ 8, \ldots, M-4$, iterate the formula shown in Equation (43), |
| (ii) | Compute $\widetilde{Z}^{(\eta+1)} = P^T\psi^{(\eta+1)}$, |
| (iii) | Convergence criterion $\|\widetilde{Z}^{(\eta+1)} - \widetilde{Z}^{(\eta)}\|_\infty < \varepsilon$, |
| (iv) | If the criterion is achieved, display approximate solutions. |

## 7. Implementation and Application of C++ for Numerical Experiment

In this work, the C++ programming language was applied to implement the designed QSPSOR computational algorithm. C++ is a high-level programming language used in the field of numerical analysis. The C++ language also facilitates low-level coding because it is an extension of the medium-level programming language, C. Because C++ has all of the features and advantages of C, C++ enables the manipulation of low-level memory and the development of numerical iterative formulas to solve mathematical problems. In this work, the C++ simulation code was written to prioritize performance, speed, efficiency, and flexibility of use. Hence, implementing the quarter-sweep approximation based on Caputo's fractional operator via the QSPSOR C++ code can yield important data, such as the number of iterations, execution time, and absolute errors.

For the numerical experiment, two problems of the time-fractional diffusion equation were selected to test the performance of the QSPSOR. For the efficiency comparison and analysis, two existing methods from our previous work were used, abbreviated as FSPSOR [19] and HSPSOR [20]. To compare the performance of these methods, three criteria were considered, namely, $\eta$—representing the number of iterations, *sec.*—representing the execution time of the C++ simulation code, and $\hat{\varepsilon}$—representing the magnitude of the absolute error. The three criteria for comparison and efficiency analysis were observed and taken based on three different values of fractional order, namely, $\alpha = 0.25$, $\alpha = 0.50$, and $\alpha = 0.75$.

Moreover, the simulation considered five different grid points, $M = 128,\ 256,\ 512,\ 1024,$ and 2048. The problems are considered below.

**Example 1 ([22]).** *Let the time-fractional initial boundary value problem be:*

$$\frac{\partial^\alpha z(x,t)}{\partial t^\alpha} = \frac{\partial^2 z(x,t)}{\partial x^2}, 0 < \alpha \le 1, 0 \le x \le \gamma, t > 0, \tag{44}$$

*where the boundary conditions are stated in fractional terms:*

$$z(0,t) = \frac{2kt^\alpha}{\Gamma(\alpha+1)}, z(\gamma,t) = \gamma^2 + \frac{2kt^\alpha}{\Gamma(\alpha+1)}, \tag{45}$$

*and the initial condition is:*

$$z(x,0) = x^2. \tag{46}$$

*The analytical solution of Equation (44) is given by:*

$$z(x,t) = x^2 + \frac{2kt^\alpha}{\Gamma(\alpha+1)}. \tag{47}$$

**Example 2 ([22]).** *Let the next time-fractional initial boundary value problem be given as:*

$$\frac{\partial^\alpha z(x,t)}{\partial t^\alpha} = \frac{1}{2}x^2 \frac{\partial^2 z(x,t)}{\partial x^2}, 0 < \alpha \le 1, 0 \le x \le \gamma, t > 0, \tag{48}$$

*where the boundary conditions are given in fractional terms:*

$$z(0,t) = 0, z(1,t) = e^t, \tag{49}$$

*and the initial condition is given by:*

$$z(x,0) = x^2. \tag{50}$$

*The exact solution of Equation (38) is:*

$$z(x,t) = x^2 e^t. \tag{51}$$

The collected numerical results of the implementation of FSPSOR, HSPSOR, and QSPSOR to solve Examples 1 and 2 are recorded in Tables 2 and 3, respectively. Based on Tables 2 and 3, it can be observed that the QSPSOR method requires fewer iterations to obtain satisfactorily accurate approximate solutions of Examples 1 and 2. This significant improvement in the number of iterations results in a shorter execution time, which indicates QSPSOR can more efficiently solve Examples 1 and 2 than the two tested methods. The accuracy of the solution obtained by the QSPSOR method is comparable and almost equivalent to those of the FSPSOR and HSPSOR methods. A comparison of the different fractional orders of the selected examples shows that the accuracy of the solutions of the three numerical methods, which are based on the implicit finite difference scheme and Caputo's time-fractional approximation, is greatest at $\alpha = 0.50$, followed by $\alpha = 0.75$ and $\alpha = 0.25$. Although the QSPSOR method can solve the selected problems efficiently, the magnitude of the absolute errors of the solutions is slightly bigger than those of the FSPSOR and HSPSOR methods. Thus, the use of the quarter-sweep scheme as the complexity reduction approach for solving time-fractional behavior is a disadvantage due to the less effective direct computation of the remaining points. Future work will further investigate the appropriate treatment to reduce the absolute errors of the quarter-sweep difference scheme in the Caputo sense.

**Table 2.** Comparison between FSPSOR, HSPSOR, and QSPSOR in solving Example 1.

| *M* | Method | $\alpha$ = 0.25 | | | $\alpha$ = 0.50 | | | $\alpha$ = 0.75 | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | $\eta$ | ***sec.*** | $\widehat{\varepsilon}$ | $\eta$ | ***sec.*** | $\widehat{\varepsilon}$ | $\eta$ | ***sec.*** | $\widehat{\varepsilon}$ |
| 128 | **FSPSOR** | 28 | 0.84 | $2.36 \times 10^2$ | 80 | 1.90 | $6.20 \times 10^4$ | 246 | 5.76 | $3.99 \times 10^2$ |
| | **HSPSOR** | 16 | 0.18 | $2.36 \times 10^2$ | 37 | 0.54 | $6.99 \times 10^4$ | 94 | 2.36 | $3.99 \times 10^2$ |
| | **QSPSOR** | 8 | 0.05 | $2.37 \times 10^2$ | 14 | 0.29 | $6.19 \times 10^4$ | 32 | 0.09 | $4.21 \times 10^2$ |
| 256 | **FSPSOR** | 53 | 5.33 | $2.43 \times 10^2$ | 211 | 17.84 | $5.69 \times 10^4$ | 806 | 67.75 | $3.97 \times 10^2$ |
| | **HSPSOR** | 34 | 2.20 | $2.43 \times 10^2$ | 94 | 6.90 | $6.21 \times 10^4$ | 303 | 34.65 | $3.97 \times 10^2$ |
| | **QSPSOR** | 15 | 0.27 | $2.44 \times 10^2$ | 39 | 2.37 | $6.99 \times 10^4$ | 101 | 12.84 | $4.03 \times 10^2$ |
| 512 | **FSPSOR** | 120 | 41.43 | $2.46 \times 10^2$ | 566 | 182.83 | $5.36 \times 10^4$ | 2635 | 843.91 | $3.96 \times 10^2$ |
| | **HSPSOR** | 67 | 21.65 | $2.46 \times 10^2$ | 246 | 86.09 | $5.36 \times 10^4$ | 988 | 421.58 | $3.96 \times 10^2$ |
| | **QSPSOR** | 31 | 5.04 | $2.47 \times 10^2$ | 100 | 40.61 | $6.21 \times 10^4$ | 337 | 198.20 | $3.96 \times 10^2$ |
| 1024 | **FSPSOR** | 250 | 372.35 | $2.48 \times 10^2$ | 1514 | 726.29 | $5.13 \times 10^4$ | 6012 | 1699.87 | $3.95 \times 10^2$ |
| | **HSPSOR** | 141 | 189.58 | $2.48 \times 10^2$ | 655 | 434.72 | $5.13 \times 10^4$ | 2413 | 1003.78 | $3.95 \times 10^2$ |
| | **QSPSOR** | 66 | 47.29 | $2.49 \times 10^2$ | 266 | 214.51 | $5.69 \times 10^4$ | 1095 | 501.76 | $3.95 \times 10^2$ |
| 2048 | **FSPSOR** | 808 | 901.76 | $2.49 \times 10^2$ | 4052 | 3469.73 | $5.13 \times 10^4$ | 35,289 | 7052.28 | $3.93 \times 10^2$ |
| | **HSPSOR** | 305 | 308.80 | $2.49 \times 10^2$ | 1788 | 1956.43 | $5.13 \times 10^4$ | 18,143 | 4025.90 | $3.93 \times 10^2$ |
| | **QSPSOR** | 143 | 95.25 | $2.50 \times 10^2$ | 709 | 365.43 | $5.35 \times 10^4$ | 3574 | 2000.83 | $3.93 \times 10^2$ |

**Table 3.** Comparison between FSPSOR, HSPSOR, and QSPSOR in solving Example 2.

| *M* | Method | $\alpha$ = 0.25 | | | $\alpha$ = 0.50 | | | $\alpha$ = 0.75 | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | $\eta$ | ***sec.*** | $\widehat{\varepsilon}$ | $\eta$ | ***sec.*** | $\widehat{\varepsilon}$ | $\eta$ | ***sec.*** | $\widehat{\varepsilon}$ |
| 128 | **FSPSOR** | 28 | 0.84 | $2.36 \times 10^2$ | 80 | 1.90 | $6.20 \times 10^4$ | 246 | 5.76 | $3.99 \times 10^2$ |
| | **HSPSOR** | 16 | 0.18 | $2.36 \times 10^2$ | 37 | 0.54 | $6.99 \times 10^4$ | 94 | 2.36 | $3.99 \times 10^2$ |
| | **QSPSOR** | 8 | 0.05 | $2.37 \times 10^2$ | 14 | 0.29 | $6.19 \times 10^4$ | 32 | 0.09 | $4.21 \times 10^2$ |
| 256 | **FSPSOR** | 53 | 5.33 | $2.43 \times 10^2$ | 211 | 17.84 | $5.69 \times 10^4$ | 806 | 67.75 | $3.97 \times 10^2$ |
| | **HSPSOR** | 34 | 2.20 | $2.43 \times 10^2$ | 94 | 6.90 | $6.21 \times 10^4$ | 303 | 34.65 | $3.97 \times 10^2$ |
| | **QSPSOR** | 15 | 0.27 | $2.44 \times 10^2$ | 39 | 2.37 | $6.99 \times 10^4$ | 101 | 12.84 | $4.03 \times 10^2$ |
| 512 | **FSPSOR** | 120 | 41.43 | $2.46 \times 10^2$ | 566 | 182.83 | $5.36 \times 10^4$ | 2635 | 843.91 | $3.96 \times 10^2$ |
| | **HSPSOR** | 67 | 21.65 | $2.46 \times 10^2$ | 246 | 86.09 | $5.36 \times 10^4$ | 988 | 421.58 | $3.96 \times 10^2$ |
| | **QSPSOR** | 31 | 5.04 | $2.47 \times 10^2$ | 100 | 40.61 | $6.21 \times 10^4$ | 337 | 198.20 | $3.96 \times 10^2$ |
| 1024 | **FSPSOR** | 250 | 372.35 | $2.48 \times 10^2$ | 1514 | 726.29 | $5.13 \times 10^4$ | 6012 | 1699.87 | $3.95 \times 10^2$ |
| | **HSPSOR** | 141 | 189.58 | $2.48 \times 10^2$ | 655 | 434.72 | $5.13 \times 10^4$ | 2413 | 1003.78 | $3.95 \times 10^2$ |
| | **QSPSOR** | 66 | 47.29 | $2.49 \times 10^2$ | 266 | 214.51 | $5.69 \times 10^4$ | 1095 | 501.76 | $3.95 \times 10^2$ |
| 2048 | **FSPSOR** | 808 | 901.76 | $2.49 \times 10^2$ | 4052 | 3469.73 | $5.13 \times 10^4$ | 35,289 | 7052.28 | $3.93 \times 10^2$ |
| | **HSPSOR** | 305 | 308.80 | $2.49 \times 10^2$ | 1788 | 1956.43 | $5.13 \times 10^4$ | 18,143 | 4025.90 | $3.93 \times 10^2$ |
| | **QSPSOR** | 143 | 95.25 | $2.50 \times 10^2$ | 709 | 365.43 | $5.35 \times 10^4$ | 3574 | 2000.83 | $3.93 \times 10^2$ |

In addition, Figures 1–4 show the performance graphs of the number of iterations and the execution time of the three methods. These graphs provide illustrations of the differences in terms of the efficiency of the combination of Caputo's time-fractional approximation and preconditioned iterations via three different schemes: the standard implicit, half-sweep, and quarter-sweep. The percentage improvement in terms of the number of iterations for the proposed QSPSOR method, at $\alpha = 0.25$, indicated that the number of iterations declined significantly, by 74.64% and 53.18%, compared to the FSPSOR and HSPSOR methods, respectively. Similarly, for execution time, the QSPSOR method was significantly faster, by 90.71% and 76.18%, than the FSPSOR and HSPSOR methods, respectively. At $\alpha = 0.50$, the number of iterations needed by QSPSOR was 82.26% and 59.95% less than that required by the FSPSOR and HSPSOR methods, respectively. As a result, the C++ program execution time was 81.83% and 59.35% faster than that of the FSPSOR and HSPSOR methods, respectively. Finally, for the fractional-order $\alpha = 0.75$, QSPSOR required 86.67% and 66.69% fewer iterations than the FSPSOR and HSPSOR methods, respectively, and the C++ program execution was 79.62% and 62.49% quicker than that of the FSPSOR and HSPSOR methods, respectively.

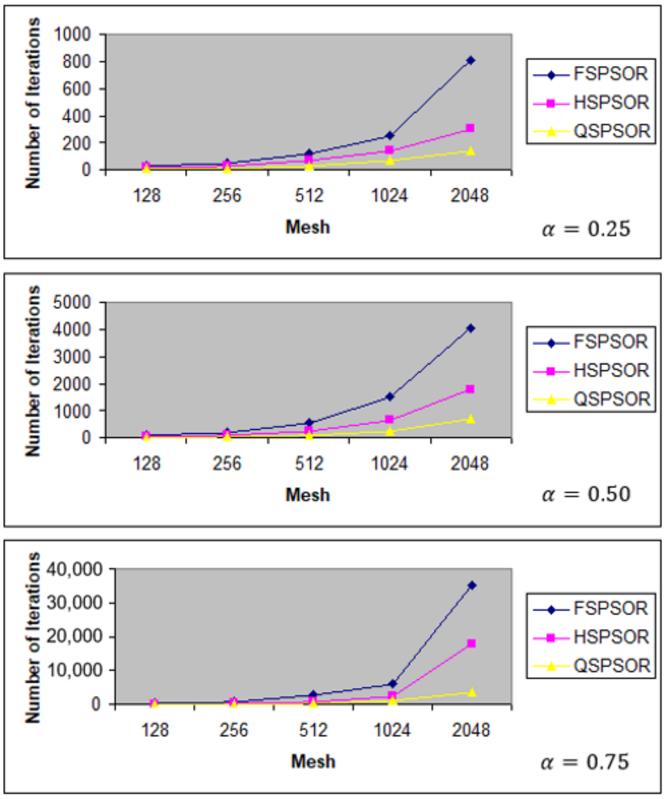

**Figure 1.** Graphs of performance of FSPSOR, HSPSOR, and QSPSOR methods to solve Example 1, showing mesh sizes versus number of iterations for three different fractional orders.

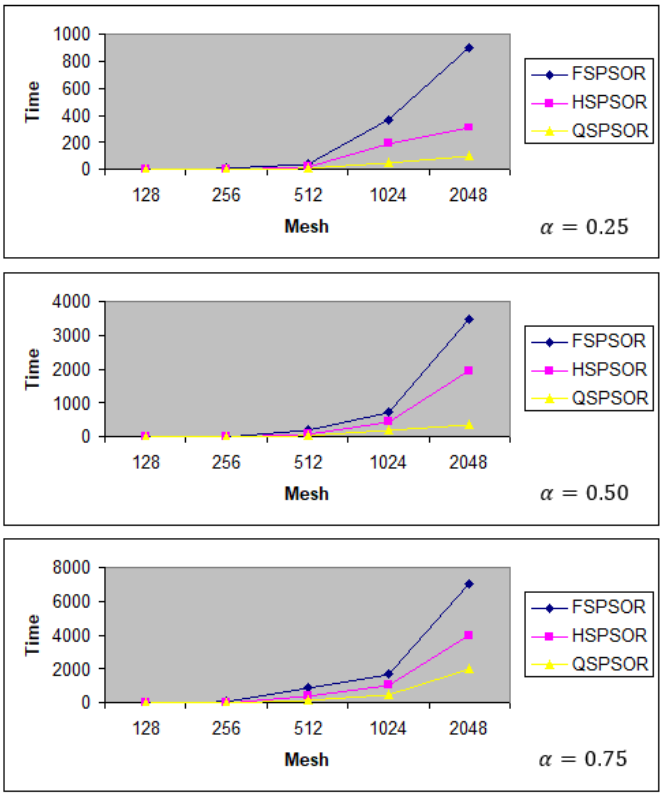

**Figure 2.** Graphs of performance of FSPSOR, HSPSOR, and QSPSOR methods to solve Example 1, showing mesh sizes versus execution time for three different fractional orders.

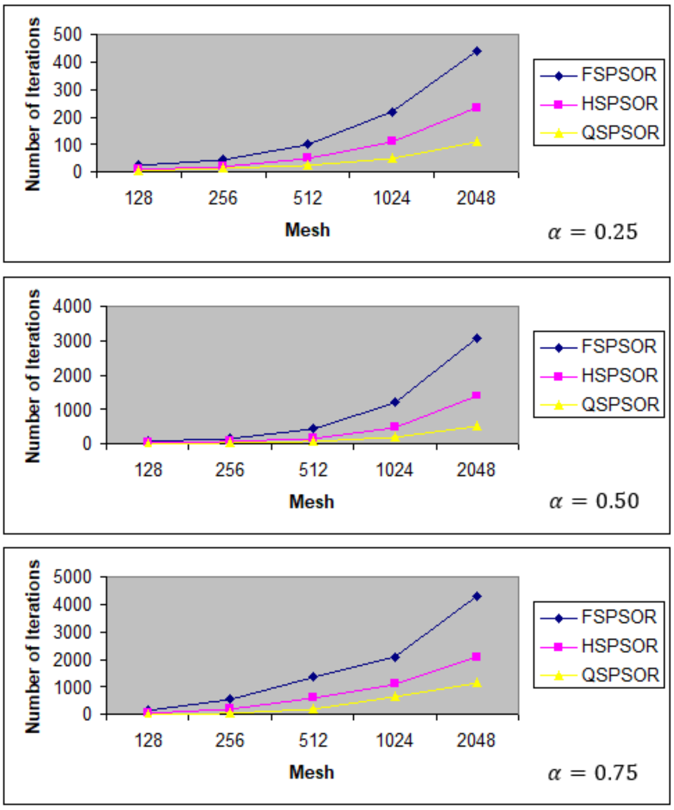

**Figure 3.** Graphs of performance of FSPSOR, HSPSOR, and QSPSOR methods to solve Example 2, showing mesh sizes versus number of iterations at three different fractional orders.

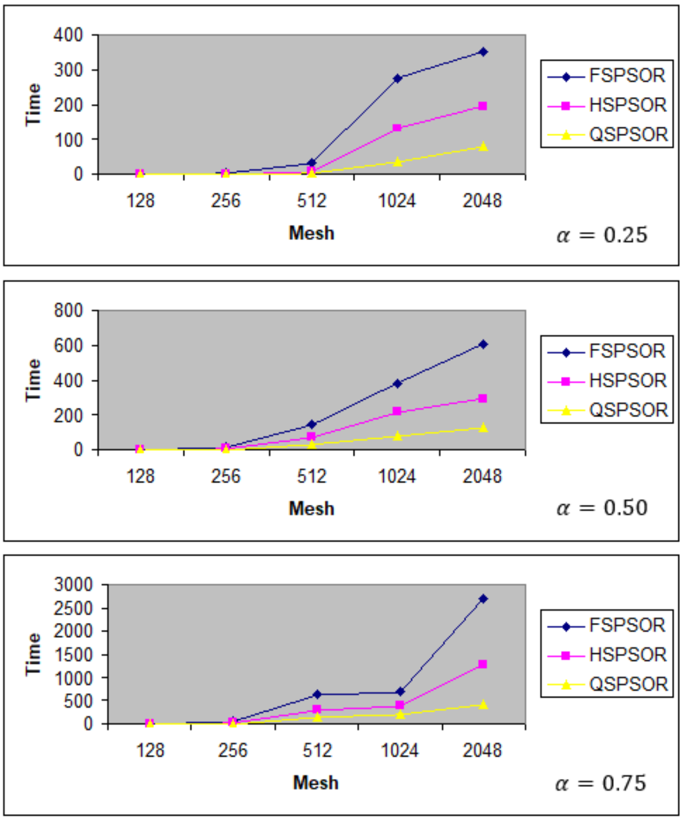

**Figure 4.** Graphs of performance of FSPSOR, HSPSOR, and QSPSOR methods to solve Example 2, showing mesh sizes versus execution time at three different fractional orders.

## 8. Conclusions

This paper discusses and presents the numerical solution of a fractional mathematical equation, namely, the time FDE, using a quarter-sweep mixed Caputo time-fractional approximation, and a C++ program to implement the QSPSOR iterative method. A preconditioned matrix was successfully implemented to modify the matrix structure to improve the convergence rate, and the QSPSOR method was formulated to efficiently solve the matrix. A numerical experiment using two alternative methods, namely FSPSOR and HSPSOR, showed the superiority of the proposed QSPSOR method in terms of the efficiency of solving two time-fractional initial boundary value problems of the diffusion equation. The findings of this paper can be highlighted as follows:

- The QSPSOR method significantly reduced the number of iterations and execution time compared to the existing FSPSOR and HSPSOR methods. On average, QSPSOR reduced the number of iterations and execution time by 81.19% and 84.06%, respectively, compared to FSPSOR. Moreover, QSPSOR reduced the number of iterations and execution time compared to HSPSOR by 59.94% and 66.00%, respectively. The result also showed that using the quarter-sweep scheme and PSOR iteration can reduce the computational complexity for solving the time FDE with a large matrix size.
- Observations regarding the accuracy of all of the implemented numerical methods indicated that their numerical solutions were in good agreement. Furthermore, the accuracy of the solutions of the time-FDE problems obtained by each of the three numerical methods was greater at $\alpha = 0.50$, followed by $\alpha = 0.75$ and $\alpha = 0.25$. The combination of the quarter-sweep implicit finite difference scheme and Caputo's time-fractional derivative enabled an accurate solution for the time FDE to be computed.
- However, a disadvantage of the quarter-sweep difference scheme is that the magnitude of the absolute errors is slightly larger than that of the two previous methods. The accuracy of the quarter-sweep scheme can be improved by applying a suitable treatment.

Thus, the combination of a quarter-sweep finite difference method, Caputo's time-fractional derivative, and the preconditioned successive over-relaxation method showed good potential to solve different types of fractional mathematical equations, and provides a future direction for this field of research.

**Author Contributions:** Writing—original draft preparation, A.S.; writing—review and editing, J.V.L.C.; supervision, P.A., J.S. and S.M. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Not applicable.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Sweilam, N.; Kareem, W.A.; Al-Mekhlafi, S.; Nabih, Z. Numerical treatments for a complex order fractional HIV infection model with drug resistance during therapy. *Prog. Fract. Differ. Appl.* **2021**, *7*, 163–176. [CrossRef]
2. Alrabaiah, H.; Zeb, A.; Alzahrani, E.; Shah, K. Dynamical analysis of fractional-order tobacco smoking model containing snuffing class. *Alex. Eng. J.* **2021**, *60*, 3669–3678. [CrossRef]
3. Wang, X.; Qi, H.; Yang, X.; Xu, H. Analysis of the time-space fractional bioheat transfer equation for biological tissues during laser irradiation. *Int. J. Heat Mass Transf.* **2021**, *177*, 121555. [CrossRef]
4. Moosavi, R.; Moltafet, R.; Shekari, Y. Analysis of viscoelastic non-Newtonian fluid over a vertical forward-facing step using the Maxwell fractional model. *Appl. Math. Comput.* **2021**, *401*, 126119. [CrossRef]
5. Gu, X.-M.; Sun, H.-W.; Zhao, Y.-L.; Zheng, X. An implicit difference scheme for time-fractional diffusion equations with a time-invariant type variable order. *Appl. Math. Lett.* **2021**, *120*, 107270. [CrossRef]

6. Davis, W.; Noren, R.; Shi, K. A C$^\alpha$ finite difference method for the Caputo time-fractional diffusion equation. *Numer. Methods Part. Differ. Equ.* **2021**, *37*, 2261–2277. [CrossRef]
7. Hosseininia, M.; Heydari, M.H.; Maalek Ghaini, F.M. A numerical method for variable-order fractional version of the coupled 2D Burgers equations by the 2D Chelyshkov polynomials. *Math. Methods Appl. Sci.* **2021**, *44*, 6482–6499. [CrossRef]
8. Mirzaee, F.; Sayevand, K.; Rezaei, S.; Samadyar, N. Finite difference and spline approximation for solving fractional stochastic advection-diffusion equation. *Iran. J. Sci. Technol. Trans. A Sci.* **2021**, *45*, 607–617. [CrossRef]
9. González-Olvera, M.A.; Torres, L.; Hernández-Fontes, J.V.; Mendoza, E. Time fractional diffusion equation for shipping water events simulation. *Chaos Solitons Fractals* **2021**, *143*, 110538. [CrossRef]
10. Liao, X.; Feng, M. Time-fractional diffusion equation-based image denoising model. *Nonlinear Dyn.* **2021**, *103*, 1999–2017. [CrossRef]
11. Maldon, B.; Thamwattana, N. A fractional diffusion model for dye-sensitized solar cells. *Molecules* **2020**, *25*, 2966. [CrossRef] [PubMed]
12. Bohaienko, V.; Bulavatsky, V. Simplified mathematical model for the description of anomalous migration of soluble substances in vertical filtration flow. *Fractal Fract.* **2020**, *4*, 20. [CrossRef]
13. Li, L.; Jiang, Z.; Yin, Z. Compact finite-difference method for 2D time-fractional convection–diffusion equation of groundwater pollution problems. *Comput. Appl. Math.* **2020**, *39*, 142. [CrossRef]
14. Aguilar, J.-P.; Korbel, J.; Luchko, Y. Applications of the fractional diffusion equation to option pricing and risk calculations. *Mathematics* **2019**, *7*, 796. [CrossRef]
15. Li, Y.; Liu, F.; Turner, I.W.; Li, T. Time-fractional diffusion equation for signal smoothing. *Appl. Math. Comput.* **2018**, *326*, 108–116. [CrossRef]
16. Sunarto, A.; Sulaiman, J. Preconditioned SOR method to solve time-fractional diffusion equations. *J. Phys. Conf. Ser.* **2019**, *1179*, 012020. [CrossRef]
17. Sunarto, A.; Sulaiman, J. Application half-sweep preconditioned SOR method for solving time-fractional diffusion equations. In Proceedings of the International Conference on Industrial Engineering and Operations Management, Detroit, MI, USA, 10–14 August 2020; pp. 3414–3420.
18. Lung, J.C.V.; Sulaiman, J. On quarter-sweep finite difference scheme for one-dimensional porous medium equations. *Int. J. Appl. Math.* **2020**, *33*, 439–450. [CrossRef]
19. Muhiddin, F.A.; Sulaiman, J.; Sunarto, A. Numerical evaluation of quarter-sweep KSOR method to solve time-fractional parabolic equations. *Int. J. Eng. Trends Technol.* **2020**, *2020*, 63–69. [CrossRef]
20. Suardi, M.N.; Radzuan, N.Z.F.M.; Sulaiman, J. Performance analysis of quarter-sweep Gauss-Seidel iteration with cubic B-spline approach to solve two-point boundary value problems. *Adv. Sci. Lett.* **2018**, *24*, 1732–1735. [CrossRef]
21. Atangana, A. Fractional Operators and Their Applications. In *Fractional Operators with Constant and Variable Order with Application to Geo-Hydrology*; Atangana, A., Ed.; Academic Press: Cambridge, MA, USA, 2018; Chapter 5; pp. 79–112.
22. Demir, A.; Erman, S.; Özgür, B.; Korkmaz, E. Analysis of fractional partial differential equations by Taylor series expansion. *Bound. Value Probl.* **2013**, *2013*, 68. [CrossRef]