Research Article ISSN Online: 2783-4859 **Computational Mathematics and Computer Modeling with Applications**

Received 27 March 2023
Accepted 6 July 2023

DOI: 10.52547/CMCMA.1.1.95
AMS Subject Classification: 65Yxx ;68Qxx; 34Kxx ; 92Bxx

A distributed approach for a COVID-19 fractional time-delay model

Mahdi Movahedian Moghaddam[a]

After the spread of COVID-19, several attempts were made to model it mathematically. Due to the high power of fractional differential equation modeling, a time delay fractional model was presented for the modeling spread of COVID-19. The solution of these models is done by computer systems in several ways, including the fractional predictor-corrector method, which has many challenges. Among these challenges are execution time, scalability, and memory consumption. In previous research, the shared memory approach was presented to reduce the execution time challenge. Still, because of the challenges of scalability and memory consumption, a coarse-grained distributed approach was presented in this research. The results presented in this research have been compared with sequential approaches and shared memory. These results have been implemented based on the data announced by the city of Wuhan in 2019, and a speedup of 1.704 was achieved per execution on 1000 inputs.Copyright © 2022 Shahid Beheshti University.

Keywords: grained parallelism; distributed computing; domain-specific parallelism; fractional calculus; predictor-corrector method.

1. Introduction

Nowadays, the use of mathematical modeling in biological problems has become very common. Differential equations can be mentioned among these models. In biology, one of the examples of issues whose modeling has been very popular recently is the spread of the COVID-19 pandemic. Among the mathematical models presented to predict the spread of this virus are ordinary differential equations [33, 1, 2, 8, 17].

Recently, fractional differential equations are widely used to model problems in biology and financial mathematics [5]. Fractional differential equations, due to having a non-integer domain on the order of the derivative of the main function, have been more accurate in modeling problems whose domain of the main function is positive [19, 18].

To solve fractional ordinary differential equation (FODE) models, different numerical methods were used. These methods are divided into two categories: direct and indirect [13]. In general, the numerical methods of solving fractional differential equations (whether ordinary or partial) during execution face computational challenges [6, 28, 11, 20].

The more efficient use of hardware resources in the execution platform system is one of the ways to reduce the computational challenge of algorithms. The field of high-performance computing refers to a set of methods for more efficient use of execution platform resources.

One of the available methods in the field of high-performance computing is the use of distributed systems. These systems use a number of autonomous, coherent processing elements to execute the algorithm [26, 10]. This approach has a special place to increase scalability, reliability, efficiency, etc. for the development of numerical algorithms [29].

Distributed systems are free from the problem of race conditions due to the independence of their processing elements. On the other hand, the processing elements of a distributed system use a mechanism called message-passing to transfer data with each other [26, 24, 10, 27]. Due to the high importance of scalability in differential equations, the use of distributed systems is very important [32, 16, 14]. The tool for implementing the proposed method in this research was the message passing interface (MPI) [31].

[a] *Department of Computer and Data Sciences, Faculty of Mathematical Sciences, Shahid Beheshti University, Tehran, Iran. Email: mmovahed02@gmail.com.*
[*] *Correspondence to: M. Movahedian Moghaddam*

The use of distributed computing is more scalable than shared memory and heterogeneous parallelization approaches. On the other hand, the message-passing cost is more for data transfer among its processing elements [26]. The use of coarse granularity in distributed computing is used to minimize the cost of message-passing and increase scalability.

The second section reviews the related works and the expression of our contribution. In the third section, a number of prerequisite concepts are defined for presenting the COVID-19 outbreak model and our distributed method. In the fourth section, the description of the COVID-19 fractional ODE model and its numerical solution was discussed. In section five, we presented sequential and then distributed algorithms to solve our model. In the sixth section, the numerical results obtained from the implementation of our distributed proposed approach versus the parallel shared memory and serial approaches were discussed, and in the final section, the conclusion was presented.

2. Related works

In recent years, fractional differential equations have been widely used in various sciences, including physics, chemistry, biology, engineering, etc. [21, 23, 15, 5, 30].

Among the research that was carried out to model the spread of COVID-19 on differential equations, we can mention [34], this research has done the modeling of the spread of COVID-19 on ordinary differential equations, which solves this model by a non-standard finite difference (NSFD) method. In the method proposed by [19], fractional differential equations were used to model the problem of the spread of COVID-19. This model was associated with a higher accuracy than ordinary differential equations, and the fractional predictor-corrector method was used to numerically solve this model.

The model presented by [2] investigated the cost of controlling the disease of COVID-19 based on the human-environment-human model. The presented model was based on the data of 2020 in the country of Ghana, which was adjusted using the least square method. They show that strategic safety measures adopted by asymptomatic and symptomatic individuals such as practicing proper coughing etiquette by maintaining a distance, covering coughs and sneezes with disposable tissues or clothing, and washing hands after coughing or sneezing are the most cost-effective strategy. In the research conducted [1], the economic analysis of dealing with the spread of COVID-19 was done based on the data reported from the two countries of Ghana and Egypt. They investigated the occurrence of multiple peaks in this virus based on the model presented on ordinary differential equations. The results of this research suggest strict adherence to health protocols and also, between the two approaches of reducing transmission and case isolation, the approach of reducing transmission is less expensive, and adopting both approaches at the same time is possible but expensive. The authors [17] presented a SIR model based on ordinary differential equations to control the spread of COVID-19 in Iraq. To estimate the parameters, the least square method and the real data of Iraq for 455 days have been used. The result of this research was to provide a kind of vaccination strategy to prevent the spread of this virus in Iraq. In the study [8], the effects of a type of anti-COVID vaccine were investigated not only on the control of the first type of this virus but also on its newer variants. The presented model is based on ordinary differential equations and its results show the effectiveness of discovering a vaccine to control its new variant.

Using the fractional finite difference method is one of the first proposed solutions for solving fractional differential equations. In the research [7], the finite difference method was used to solve Poisson's fractional equation. Among the methods of numerical solution of fractional differential equations are spectral methods. The authors [25] of Bessel and generalized Bessel functions of Kind first and Jacobian free Newton-Krylov method with adaptive preconditioner are used to solve the non-linear time-fractional Burger equation. The numerical solution of fractional differential equations is done today by other approaches, including the pseudo-spectral approach, which was first introduced by Parand and Delkhosh in [4]. This method is based on fractional Lagrange functions (FLF) for the numerical solution of linear and non-linear equations. Another method of solving ordinary differential equations is the predictor-corrector method, which the authors [9] used to solve a pandemic model of COVID-19. Their proposed model included reported and unreported cases.

In the research presented by Lorin [20], a parallel approach in shared memory systems was presented to solve space-time fractional partial differential equations. This approach was based on data parallelization and used time-domain decomposition and space-domain decomposition techniques. This approach used the Fourier-based pseudospectral method to solve this model. In the approach presented in research [12], a fractional reaction-diffusion equation was introduced by the implicit finite difference method. This equation was implemented using the domain decomposition technique in the distributed memory approach. The approach used is data parallelization, which is effective in the finite difference method. This approach was implemented using MPI 4.0.3 in 16 core processor and a speedup of 14.55 was reached. In the research [6], a parallel approach was presented on shared memory and distributed memory systems to increase the efficiency and speed up the fractional second-order Adams-Bashforth-Moulton method for solving fractional differential equations. This approach is presented with fine granularity.

In [22] we presented a coarse-grained parallel approach to shared memory systems for a COVID-19 fractional time-delay model. The numerical method of solving this model was predictor-corrector. Our proposed approach on a 4-core system was associated with a speedup of 2.76 on a 1000-day prediction.

Our contribution

In this research, our attempt is to provide a domain-specific parallel approach based on the fractional time delay model of COVID-19. This approach is on distributed systems with coarse granularity and is based on task parallelism. The proposed distributed approach is very effective to reduce the memory challenge in fractional models [11]. Also, the distributed approach presented in this research has much higher scalability than the shared memory approach, which plays an important role in the model under study.

3. Preliminaries

In this section, we review some prerequisite concepts for presenting the proposed model. First, the definition of the fractional derivative, then the fractional ordinary differential equations, and finally the predictor-corrector method for solving these equations have been discussed.

3.1. Riemann-Liouville definition

Definition 1 *We assume that $y \in C[a, b]$, is the Riemann-Liouville fractional derivative of arbitrary order α, which can be defined as (1):*

$$ {}^{RL}D_t{}^{\alpha} y(t) = \frac{1}{\Gamma(n-\alpha)} \frac{d^n}{xt^n} \int_0^t (t-\theta)^{n-\alpha-1} y(\theta) d(\theta) \tag{1} $$

in this equation, $n = [\alpha] + 1$ and $[\alpha]$ mean the floor of α.

3.2. Caputo definition

Definition 2 *The derivative of non-integer order α for the same function of y (in previous definition) is defined as follows:*

$$ {}^{C}D_t{}^{\alpha} y(t) = \frac{1}{\Gamma(n-\alpha)} \int_0^t (t-\theta)^{n-\alpha-1} y^n(\theta) d(\theta) \tag{2} $$

also in equation (2), we have $n = [\alpha] + 1$.

3.3. Fractional ordinary differential equation

Since the definition given by Caputo (2) provides the possibility to set the initial conditions for initial value problems at the initial point $t = a$ such as $y'(a), y''(a), \ldots$, but the Riemann-Liouville definition does not apply to the initial conditions that include the limit at the initial point $t = a$ Leads. For example :

$$ \lim_{t \to a} [{}_a^{RL}D_t{}^{\alpha-n} y(t)] = b_n $$

where b_n, n are constants.

Although this interpretation is justifiable from a mathematical point of view, it does not have a specific interpretation from a physical point of view, for this reason, the Caputo form is used to present the initial value problems that express mathematical modeling [3].

Due to this, the definition of the fractional ordinary differential equation is presented, and the model studied in this article is expressed in the Caputo form, and for brevity we used the symbol $D_t{}^{\alpha} y(t)$ instead of the symbol ${}_a^C D_t{}^{\alpha} y(t)$ for our definitions.

Definition 3 *A fractional ordinary differential equation (FODE) is defined in the equation (3), the conditions of this equation are ($\alpha > 0, m = [\alpha] + 1$) and α is known as the derivative of the function y.*

$$ D_t{}^{\alpha} y(t) = f(t, y(t)), \quad t \in [0, T]. \tag{3} $$

the initial value conditions (4) must be maintained the existence and uniqueness of equation (3).

$$ D^k y(0) = y^{(k)}(0), \quad k = 0, 1, \ldots, m-1. \tag{4} $$

3.4. Fractional predictor-corrector method

Numerical methods for solving a FODE are divided into two categories, direct and indirect.

Direct methods are used to solve homogeneous equations, but indirect methods do not have such limitations. Among the standard indirect methods, we can mention the fractional predictor-corrector method.

The fractional predictor-corrector method first introduces a grid. This grid has points, as equation (5).

$$ t_j = jh \quad (h = T/N), \quad j = 1, 2, \ldots, N, \tag{5} $$
$$ f_j = f(x_j, y_j). $$

The predictor-corrector method on the n-th iteration includes the following steps:

1- Predict $y^p(t_{n+1})$
2- Compute $f(t_{n+1}, y^p(t_{n+1}))$
3- Correct $y_h(t_{n+1})$
4- Recompute $f(t_{n+1}, y_h(t_{n+1}))$

The recomputation in the last step is used for the next iteration.

4. Model description

Since the beginning of the spread of the coronavirus in 2019, several attempts have been made to find a relatively accurate mathematical model for its spread. Finally, the efforts of the authors [19] led to the presentation of a fractional time delay model with appropriate accuracy for the spread of COVID-19. The above model can be solved by different numerical methods, including the fractional predictor-corrector which was chosen by our research.

The model (6) is a fractional time delay model consisting of 4 equations that $S(t)$ express the number of susceptible cases, $E(t)$ the number of exposed cases, $I(t)$ the number of infectious cases and $R(t)$ the number of recovered cases.

$$\begin{aligned} D_t{}^{\alpha} S(t) &= b - \beta S(t)I(t) - dS(t), \\ D_t{}^{\alpha} E(t) &= \beta S(t)I(t) - \gamma\beta S(t-\tau)I(t-\tau)e^{-d\tau} - dE(t) - \delta E(t), \\ D_t{}^{\alpha} I(t) &= \gamma\beta S(t-\tau)I(t-\tau)e^{-d\tau} - [v_1 + v_2(1-c(t))]I(t) - (\theta_1 + \theta_2 c(t))I(t) - dI(t), \\ D_t{}^{\alpha} R(t) &= (\theta_1 + \theta_2 c(t))I(t) + \delta E(t) - dR(t). \end{aligned} \tag{6}$$

As you can see in the long-term deficit model presented (6), there are several parameters in this model that we depend on. These parameters along with their exact value or their allowed range are introduced in the table 1.

Since the implementation of the mentioned model was based on the parameters of the city of Wuhan, China, the initial values were analyzed for this city. Therefore, the level of accessibility to the health care system is $\frac{321000}{321000+I(t)}$, the initial population is $N(0) = 89,995,587$, and the population birth rate is $b = d \times N(0)$.

It should be noted that the model (6) has initial values that are mentioned in (7).

$$S(0) = 89,985,050, \quad E(0) = 10,050, \quad I(0) = 475. \tag{7}$$

The initial values mentioned in (7) are based on the data reported by Wuhan Municipal Commission on December 31, 2019.

Now we calculate the basic reproduction number $R(0)$ for the model (6) using the next-generation method. Suppose $Y = (I, S)^T$ then we rewrite model (6) as follows

$$D_t{}^{\alpha} Y = F(Y) - V(Y) \tag{8}$$

where

$$F(Y) = \begin{bmatrix} \gamma\beta S(t-\tau)I(t-\tau)e^{-d\tau} \\ 0 \end{bmatrix}$$

and

$$V(Y) = \begin{bmatrix} [v_1 + v_2(1-c(t)) + \theta_1 + \theta_2 c(t) + d]I(t) \\ \beta S(t)I(t) + dS(t) - b \end{bmatrix}.$$

So $Y_0 = (\frac{b}{d}, 0)^T$ is the unique disease-free equilibrium point of the model (6).

Respectively, F and V, which are the expressions of new infection terms and remaining transfer terms, are obtained as follows

$$F = JF_{1\times1} = \begin{bmatrix} \frac{b\beta\gamma e^{-d\tau}}{d} \end{bmatrix}$$

and

$$V = JV_{1\times1} = \begin{bmatrix} v_1 + v_2(1-c(t)) + \theta_1 + \theta_2 c(t) + d \end{bmatrix}.$$

Now we need to calculate the matrix FV^{-1}, which is obtained as a formula (9).

$$FV^{-1} = \begin{bmatrix} \frac{b\beta\gamma e^{-d\tau}}{d(v_1+v_2(1-c(t))+\theta_1+\theta_2 c(t)+d)} \end{bmatrix} \tag{9}$$

The reproduction number $R(0)$ is actually the spectral radius of the matrix FV^{-1}, which is calculated as (10).

$$R(0) = \rho(FV^{-1}) = \frac{b\beta\gamma e^{-d\tau}}{d(v_1 + v_2(1-c(t)) + \theta_1 + \theta_2 c(t) + d)} \tag{10}$$

Parameter List		
Parameter	Value	Description
b	3210	Birth rate
β	0.62×10^{-8}	Susceptible to infected rate
d	3.57×10^{-5}	Natural death rate
γ	0.143	Exposed to infected rate
δ	0.006	Exposed to removed rate
c	[0,1]	health care systems Availability
θ_1	0.00005	Natural recovery rate
θ_2	0.0667	Recovery rate
v_1	0.01	Min disease
v_2	0.02	Max disease

Table 1. Parameter description of model

In order to apply the fractional predictor-corrector method to solve the above model, we present the equation (11).

$$\begin{aligned} {D_t}^{\alpha} S(t) &= G_1(t, S(t), S(t-\tau)), \\ {D_t}^{\alpha} E(t) &= G_2(t, E(t), E(t-\tau)), \\ {D_t}^{\alpha} I(t) &= G_3(t, I(t), I(t-\tau)), \\ {D_t}^{\alpha} R(t) &= G_4(t, R(t), R(t-\tau)). \end{aligned} \tag{11}$$

In the previous section, we described the fraction predictor-corrector method for solving FODEs. Now we have to use this method in order to solve the mentioned COVID-19 model. Based on the definition of a mesh, this method first obtains an initial solution for a point (prediction) and then tries to correct that solution at the same point (correction).

Numerical methods of solving differential equations, including predictor-corrector and fractional predictor-corrector, solve the model based on the definition of a number of points from the grid. In the previous section, it was mentioned that the equation (5) defines this grid in a way. Based on this, for the model studied in this research, the equation (12) describes this grid.

$$t_m = mh, \quad h = \frac{\tau}{N}, \quad m \in 1, ..., N. \tag{12}$$

Since the predictor-corrector method first requires an initial solution to solve the model, we define the predictor equations as (13) with the help of (11) equations. These equations provide an initial solution (prediction) to solve the model.

$$\begin{aligned} S^p(t_{m+1}) &= S_0 + \frac{1}{\Gamma(\alpha)} \sum_{i=0}^{m} b_{i,m+1} G_1(t_i, S(t_i), S(t_{i-n})), \\ E^p(t_{m+1}) &= E_0 + \frac{1}{\Gamma(\alpha)} \sum_{i=0}^{m} b_{i,m+1} G_2(t_i, E(t_i), E(t_{i-n})), \\ I^p(t_{m+1}) &= I_0 + \frac{1}{\Gamma(\alpha)} \sum_{i=0}^{m} b_{i,m+1} G_3(t_i, I(t_i), I(t_{i-n})), \\ R^p(t_{m+1}) &= R_0 + \frac{1}{\Gamma(\alpha)} \sum_{i=0}^{m} b_{i,m+1} G_4(t_i, R(t_i), R(t_{i-n})). \end{aligned} \tag{13}$$

In equations (13), the value of parameter $b_{i,m+1}$ is calculated from equation (14).

$$b_{i,m+1} = \frac{h^{\alpha}}{\alpha}((m+1-i)^{\alpha} - (m-i)^{\alpha}) \tag{14}$$

After predicting the initial solution of the equation at a certain point, it should be modified. This step takes the predicted answer and improves it. Thus, the solution of equations (13) should be used for this step. The final answer is obtained from the equations (15).

$$
\begin{aligned}
S_h(t_{m+1}) &= S_0 + \frac{h^\alpha}{\Gamma(\alpha+2)} G_1(t_{m+1}, S^p(t_{m+1}), S(t_{m+1-n})) + \frac{h^\alpha}{\Gamma(\alpha+2)} \sum_{i=0}^{m} a_{i,m+1} G_1(t_i, S(t_i), S(t_{i-n})),\\
E_h(t_{m+1}) &= E_0 + \frac{h^\alpha}{\Gamma(\alpha+2)} G_2(t_{m+1}, E^p(t_{m+1}), E(t_{m+1-n})) + \frac{h^\alpha}{\Gamma(\alpha+2)} \sum_{i=0}^{m} a_{i,m+1} G_2(t_i, E(t_i), E(t_{i-n})),\\
I_h(t_{m+1}) &= I_0 + \frac{h^\alpha}{\Gamma(\alpha+2)} G_3(t_{m+1}, I^p(t_{m+1}), I(t_{m+1-n})) + \frac{h^\alpha}{\Gamma(\alpha+2)} \sum_{i=0}^{m} a_{i,m+1} G_3(t_i, I(t_i), I(t_{i-n})),\\
R_h(t_{m+1}) &= R_0 + \frac{h^\alpha}{\Gamma(\alpha+2)} G_4(t_{m+1}, R^p(t_{m+1}), R(t_{m+1-n})) + \frac{h^\alpha}{\Gamma(\alpha+2)} \sum_{i=0}^{m} a_{i,m+1} G_4(t_i, R(t_i), R(t_{i-n})).
\end{aligned}
\tag{15}
$$

In the equations (15), you see the value of the parameter $a_{i,m+1}$, which is calculated based on the equation (16).

$$
a_{i,m+1} = \begin{cases} m^{\alpha+1} - (m-\alpha)(m+1)^\alpha & i=0 \\ (m-i+2)^{\alpha+1} - 2(m-i+1)^{\alpha+1} & \\ +(m-i)^{\alpha+1} & 0 \le i \le m \\ 1 & i = m+1 \end{cases} \tag{16}
$$

In this section, the main focus was on the studied model. First, the examined COVID-19 model was described along with its related parameters. In the next step, the fractional predictor-corrector approach was used to numerically solve the above model to introduce the proposed method based on this.

5. The proposed method

In the previous section, we explained the model of COVID-19 studied in this research and used the numerical method to solve it, that is, the fractional predictor-corrector method. To solve this model by a computer system, an appropriate algorithm must be provided. To be, The common approach in these algorithms is the sequential approach. In this section, the sequential approach to solving this model on computer systems has been examined first, and then we will discuss its challenges and weaknesses. After stating these challenges in the sequential approach, we introduce the proposed distributed approach.

Algorithm 1 uses the fractional predictor-corrector method to solve the stated model. This algorithm is sequential and in each step, it tries to obtain the solution for a specific point for all model equations. After obtaining the exact value of the answer at one point for one of the equations of the model, the next equation is solved and the answer for the same point is obtained.

Algorithm 1 is an algorithm with a sequential approach to applying the fractional predictor-corrector method to solve the investigated model. The input of the algorithm is the alpha (α) and tau (τ) parameters, which represent the order of the model and the time delay of the model, respectively. Also, the outputs of the algorithm are the solved model. To solve this model, we need the initial solutions of the model, which are initialized in line 1 of the algorithm. The above algorithm should solve the equations in the model for all the points 1 to N and find their solution based on the fractional predictor-corrector method. In each iteration, in lines 2 to 6, the proposed model is solved for a specific point, and in the next iteration of this loop, the equations are solved at the next point. After the execution of this algorithm, the model is completely solved.

Algorithm 1: Sequential algorithm

```
   input  : $\alpha, \tau$
   output : $S, E, I, R$
1  $S[0], E[0], I[0], R[0]$    // Initialize values
2  for $m \leftarrow 1$ to $N$ do
3      $S_h(t_{m+1}) \leftarrow Predictor\_Corrector(m, \alpha, \tau)$
4      $E_h(t_{m+1}) \leftarrow Predictor\_Corrector(m, \alpha, \tau)$
5      $I_h(t_{m+1}) \leftarrow Predictor\_Corrector(m, \alpha, \tau)$
6      $R_h(t_{m+1}) \leftarrow Predictor\_Corrector(m, \alpha, \tau)$
```

The sequential algorithm under review has challenges and weaknesses. Among these weaknesses is its time complexity, especially for solving fractional models. These challenges and weaknesses are not only limited to the execution time of the model solution but also create memory complexity. Among the approaches that were presented to overcome the execution time challenge, the shared memory parallel approach was in the research [22]. In general, shared memory parallel approaches overcome the problem of time complexity to a good extent but do not provide a solution for the complexity of the memory used by the algorithm. The distributed approach not only provides a solution to reduce execution time and memory consumption, but also greatly increases its scalability. Among these reasons, firstly, it is possible and cheaper to add a processing node to the distributed system compared to adding a processing unit to the shared memory system, secondly, there is a separate memory address space for each processing node in the distributed system. That in shared memory systems, the memory space exists in a shared manner and will cause a memory bottleneck in some issues.

Due to the weaknesses expressed in the sequential approach, a new approach based on distribution was presented. A distributed approach is presented to solve the investigated model on a fully connected network. This network can be seen in figure 1. The reason for choosing this topology is the dependence of each equation in the model on other equations.

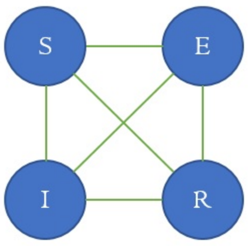

Figure 1. Distributed topology of proposed model

Algorithm 2 is based on the proposed distributed approach. Tau (τ) and alpha (α) are the input parameters of the algorithm, and the solved equations are its output. Line 1 initializes the boundary values of the model. Line 2 specifies the number of system processes, and line 3 specifies the index of the current process. Lines 4 to 17 deal with the solution of the expressed equations for only one specific value m. Lines 5, 8, 11, and 14 deal with the output of the model according to the stated topology 1. Lines 6, 9, 12, and 15 solve each equation using the fractional predictor-corrector method and save the output value for the next iteration. Since the output of one equation may be used for other equations in subsequent iterations, lines 7, 10, 13, and 16 deal with all broadcasts of the obtained value. Line 17 deals with synchronizing different processes in one repetition because a process may need a quantity of process output in a larger repetition number, which is in a lower repetition, but its output is still not obtained. For this reason, the need for this synchronization mechanism is strongly felt.

Algorithm 2: Distributed method

```
    input  : $\alpha, \tau$
    output: $S, E, I, R$
1   $S[0], E[0], I[0], R[0]$   // Initialize values
2   $size \leftarrow MPI\_Comm\_Size()$   // Number of processes
3   $i \leftarrow MPI\_Comm\_Rank()$   // Processes ID
4   for $m \leftarrow 1$ to $N$ do
5       if $i = \frac{size}{4}$ then
6           $S_h(t_{m+1}) \leftarrow Predictor\_Corrector(m, \alpha, \tau)$
7           $MPI\_Broadcast(S_h(t_{m+1}))$
8       if $i = \frac{size}{4} + 1$ then
9           $E_h(t_{m+1}) \leftarrow Predictor\_Corrector(m, \alpha, \tau)$
10          $MPI\_Broadcast(E_h(t_{m+1}))$
11      if $i = \frac{size}{4} + 2$ then
12          $I_h(t_{m+1}) \leftarrow Predictor\_Corrector(m, \alpha, \tau)$
13          $MPI\_Broadcast(I_h(t_{m+1}))$
14      if $i = \frac{size}{4} + 3$ then
15          $R_h(t_{m+1}) \leftarrow Predictor\_Corrector(m, \alpha, \tau)$
16          $MPI\_Broadcast(R_h(t_{m+1}))$
17      $MPI\_Barrier()$
```

In this section, we first described the sequential approach for the numerical solution of the model mentioned in the previous section. Then the challenges of the sequential approach were stated, and we moved towards solving these challenges with the help of distribution. In the next step, we presented an architecture for our distributed approach, and then the distributed algorithm was described.

6. Experimental results

The execution platform that has been used for executing the proposed method is the OpenMPI 4.1.4 library on the processor model "Intel Core i7-3520M" with a main memory of "8192MB".

The numerical results of this research have been implemented for tau (τ) and alpha (α) parameters of 2 and 0.95. According to previous studies [19], these values have better accuracy compared to other values.

Table 2 shows the execution time of solving the time-delay fraction model (6) using the fractional predictor-corrector method. This execution time is expressed in the variable number of days in 3 sequential, parallel (shared memory), and distributed methods. Parallel execution on 4 processing cores and distributed execution on 4 processing nodes. Each row of this table shows the execution time for different input sizes (number of days). Each column represents the execution time in sequential, shared memory, and distributed approaches. It should be noted that the execution time is measured in seconds.

Table 2 shows that in all 3 sequential approaches, the shared memory and distributed increases with the increase of the input size from 250 to 1500 and the execution time is incremental. The execution time of the sequential approach has increased from 42.112 to 253.215. The execution time of the shared memory approach for this input range has changed between 22,050 and 124,565. The implementation time of the distributed approach was between 26,571 and 171,898.

From the data in table 2, it can be concluded that all sequential approaches and shared memory and distributed have positive growth in relation to the increase in the input size. Also, both shared memory and distributed approaches have had lower execution times than the sequential approach. It should be noted that the difference in the execution time of the shared memory and distributed approaches are due to the overhead of data communication among processing nodes, which has increased the execution time compared to the shared memory approach.

	Input size				
-	250	500	750	1000	1500
Sequential	42.112	90.242	141.536	193.238	253.215
Shared memory	22.050	33.141	52.866	69.910	124.565
Distributed	26.571	55.993	87.373	113.354	171.898

Table 2. Execution time of parallel and distributed implementations

Using equation (17) we can calculate the speedup of our approaches.

$$speedup = \frac{T_s}{T_p} \tag{17}$$

In this equation, T_s refers to the execution time of a sequential method and T_p shows the parallel execution time on p processor.

Table 3 shows the speedup obtained from the two parallel approaches of shared memory and distributed versus the sequential method. Each table row displays the speedup expression for a different input size. Each column expresses the speedup resulting from different approaches of parallel shared memory and distributed for specific input.

The data in table 3 shows the speedup of parallel approaches of shared memory and distributed. The speedup achieved per input was 250 days to 1500 days. The shared memory approach has a speedup of 1.909 per input 250, and the distributed approach has a speedup of 1.584 per input. The speedup obtained in both approaches up to 1000 input increases for the growth of the input size and reaches the values of 2.764 and 1.704. As the input increases, the speedup value decreases to 2.032 and 1.473 values.

From the data in table 3, it can be concluded that in the distributed approach, the overhead of data communication has reduced the speedup of this approach. According to the execution platform, the best speedup for the input was 1000 days.

	Input size				
-	250	500	750	1000	1500
Shared memory	1.909	2.722	2.677	2.764	2.032
Distributed	1.584	1.611	1.619	1.704	1.473

Table 3. Speedup of parallel and distributed implementations

Using equation (18), we can calculate the efficiency of the proposed approach.

$$E = \frac{S_p}{p} \tag{18}$$

In this equation, S_p represents the speedup of the execution on p processor.

Table 4 shows the performance obtained from the implementation of two shared memory and distributed approaches. The rows of this table examine the efficiency per approach for different inputs, and its columns express the efficiency obtained in a fixed input in different approaches.

The efficiency of the two approaches examined in table 4 has been such that in the input range of 250 to 1500, the shared memory approach has a value of 0.477 when running on the input of 250 days, and the distributed approach on the same input size has the value has had 0.396. The efficiency values of each approach grow by increasing the input size to 1000 days and reach the values of 0.691 and 0.426, respectively. By further increasing the size of the input to more than 1000 days, the value of efficiency decreases and reaches the values of 0.508 and 0.368.

The conclusion that can be drawn from the effectiveness of the investigated approaches mentioned in table 4 is as follows. The lack of efficiency of the distributed approach compared to the shared memory approach is due to the higher cost of data communication in distributed systems. The performance obtained from the distributed approach has a more stable property and does not fluctuate much like the shared memory approach. The reason for the more optimal efficiency in the entry of 1000 days is also due to the execution platform.

	Input size				
	250	500	750	1000	1500
Shared memory	0.477	0.680	0.669	0.691	0.508
Distributed	0.396	0.402	0.404	0.426	0.368

Table 4. Efficiency of parallel and distributed implementations

In this section, the numerical results obtained from the implementation of two shared memory and distributed approaches were presented. These results were evaluated in terms of execution time, speedup, and efficiency. The obtained results show that in terms of speedup and efficiency, the distributed approach is not more optimal than the shared memory approach. The reason for this is the cost of data communication among processing nodes in the distributed approach. Due to the special architecture of distributed systems, in terms of reducing memory complexity in solving fractional differential equations, a distributed approach is suggested. Another reason that makes us suggest the distributed approach is the high importance of scalability in solving differential equations. The importance of scalability in the investigated model is doubled because the presented approaches are completely domain-based for this model.

7. Conclusion

In this research, a domain-specific distributed approach was developed for a model of the spread of COVID-19. The investigated model is a fractional time delay model that has been solved by the predictor-corrector numerical method. The experimental results obtained from the distributed approach of this research were compared with the shared memory approach of the research [22] in terms of execution time, speedup, and efficiency.

The experimental results obtained from this paper showed that the proposed distributed approach works much more optimally than the serial approach in terms of execution time, speedup, and efficiency. It was also mentioned that the distributed approach compared to the shared memory approach proposed by [22] has advantages such as higher scalability and overcoming memory complexity [11]. On the other hand, the distributed approach has data communication overhead compared to the shared memory approach. The obtained experimental results show that this data communication overhead has reduced the speedup and efficiency and increased the execution time compared to the shared memory approach.

Considering the high scalability of a distributed approach compared to the shared memory approach and the speedup and efficiency obtained from the experimental results compared to the serial approach, it shows that the use of the distributed approach is very justified.

References

1. J. K. K. Asamoah, Z. Jin, G.-Q. Sun, B. Seidu, E. Yankson, A. Abidemi, F. Oduro, S. E. Moore, and E. Okyere. Sensitivity assessment and optimal economic evaluation of a new COVID-19 compartmental epidemic model with control interventions. *Chaos, Solitons & Fractals*, 146:110885, 2021.
2. J. K. K. Asamoah, M. A. Owusu, Z. Jin, F. Oduro, A. Abidemi, and E. O. Gyasi. Global stability and cost-effectiveness analysis of COVID-19 considering the impact of the environment: Using data from ghana. *Chaos, Solitons & Fractals*, 140:110103, 2020.
3. M. Caputo. Linear model of dissipation whose Q is almost frequency dependent II, Geophys, JR Ast. *Soc (13)*, pages 529–539, 1967.
4. M. Delkhosh and K. Parand. A new computational method based on fractional lagrange functions to solve multi-term fractional differential equations. *Numerical Algorithms*, pages 1–38, 2021.
5. K. Diethelm. *The analysis of fractional differential equations: An application-oriented exposition using differential operators of Caputo type*, volume 2004. Springer Berlin, 2010.

6. K. Diethelm. An efficient parallel algorithm for the numerical solution of fractional differential equations. *Fractional Calculus and Applied Analysis*, 14(3):475–490, 2011.
7. S. Duo, H. W. van Wyk, and Y. Zhang. A novel and accurate finite difference method for the fractional Laplacian and the fractional Poisson problem. *Journal of Computational Physics*, 355:233–252, 2018.
8. T. S. Faniran, A. Ali, N. E. Al-Hazmi, J. K. K. Asamoah, T. A. Nofal, and M. O. Adewole. New variant of SARS-CoV-2 dynamics with imperfect vaccine. *Complexity*, 2022.
9. W. Gao, P. Veeresha, C. Cattani, C. Baishya, and H. M. Baskonus. Modified predictor–corrector method for the numerical solution of a fractional-order SIR model with 2019-nCoV. *Fractal and Fractional*, 6(2):92, 2022.
10. B. Gaster, L. Howes, D. R. Kaeli, P. Mistry, and D. Schaa. *Heterogeneous computing with openCL: Revised openCL 1.* Newnes, 2012.
11. C. Gong, W. Bao, G. Tang, Y. Jiang, and J. Liu. Computational challenge of fractional differential equations and the potential solutions: A survey. *Mathematical Problems in Engineering*, 2015, 2015.
12. C. Gong, W. Bao, G. Tang, B. Yang, and J. Liu. An efficient parallel solution for Caputo fractional reaction–diffusion equation. *The Journal of Supercomputing*, 68(3):1521–1537, 2014.
13. B. Guo, X. Pu, and F. Huang. *Fractional partial differential equations and their numerical solutions*. World Scientific, 2015.
14. R. D. Haynes and B. W. Ong. MPI–OpenMP algorithms for the parallel space–time solution of time dependent PDEs. *Domain decomposition methods in science and engineering XXI*, pages 179–187, 2014.
15. R. Hilfer. *Applications of fractional calculus in physics*. World scientific, 2000.
16. R. Keppens, J. Teunissen, C. Xia, and O. Porth. MPI-AMRVAC: A parallel, grid-adaptive PDE toolkit. *Computers & Mathematics with Applications*, 81:316–333, 2021.
17. S. L. Khalaf and H. S. Flayyih. Analysis, predicting, and controlling the COVID-19 pandemic in Iraq through SIR model. *Results in Control and Optimization*, 10:100214, 2023.
18. S. L. Khalaf, M. S. Kadhim, and A. R. Khudair. Studying of COVID-19 fractional model: Stability analysis. *Partial Differential Equations in Applied Mathematics*, 7:100470, 2023.
19. P. Kumar and V. Suat Erturk. The analysis of a time delay fractional COVID-19 model via Caputo type fractional derivative. *Mathematical methods in the applied sciences*, 2020.
20. E. Lorin. A parallel algorithm for space-time-fractional partial differential equations. *Advances in Difference Equations*, 2020(1):1–21, 2020.
21. J. T. Machado, V. Kiryakova, and F. Mainardi. Recent history of fractional calculus. *Communications in nonlinear science and numerical simulation*, 16(3):1140–1153, 2011.
22. M. M. Moghaddam and K. Parand. A parallel approach to the fractional time delay model for predicting the spread of COVID-19. In *2022 13th International Conference on Information and Knowledge Technology (IKT)*, pages 1–6, 2022.
23. K. B. Oldham. Fractional differential equations in electrochemistry. *Advances in Engineering software*, 41(1):9–12, 2010.
24. P. Pacheco. *An introduction to parallel programming*. Elsevier, 2011.
25. K. Parand and M. Nikarya. Application of Bessel functions and Jacobian free Newton method to solve time-fractional Burger equation. *Nonlinear Engineering*, 8(1):688–694, 2019.
26. B. Parhami. *Introduction to parallel processing: Algorithms and architectures*. Springer Science & Business Media, 2006.
27. W. Petersen and P. Arbenz. *Introduction to Parallel Computing: A practical guide with examples in C*, volume 9. OUP Oxford, 2004.
28. M. Saqib, I. Khan, and S. Shafie. Application of fractional differential equations to heat transfer in hybrid nanofluid: Modeling and solution via integral transforms. *Advances in Difference Equations*, 2019(1):1–18, 2019.
29. H. Sarbazi-Azad, L. Mackenzie, M. Ould-Khaoua, and G. Min. An efficient parallel algorithm for Lagrange interpolation and its performance. In *Proceedings Fourth International Conference/Exhibition on High Performance Computing in the Asia-Pacific Region*, volume 2, pages 593–598. IEEE, 2000.
30. L. Suarez and A. Shokooh. An eigenvector expansion method for the solution of motion containing fractional derivatives. *Journal of Applied Mechanics*, 64(3):629, 1997.
31. D. W. Walker and J. J. Dongarra. MPI: A standard message passing interface. *Supercomputer*, 12:56–68, 1996.
32. C. Xia, J. Teunissen, I. El Mellah, E. Chané, and R. Keppens. MPI-AMRVAC 2.0 for solar and astrophysical applications. *The Astrophysical Journal Supplement Series*, 234(2):30, 2018.
33. C. Yang and J. Wang. A mathematical model for the novel coronavirus epidemic in Wuhan, China. *Mathematical biosciences and engineering: MBE*, 17(3):2708, 2020.
34. A. Zeb, E. Alzahrani, V. S. Erturk, and G. Zaman. Mathematical model for coronavirus disease 2019 (COVID-19) containing isolation class. *BioMed research international*, 2020, 2020.