Applying Cuckoo Search Algorithm to Solve Fractional Differential Equation Based on Cubic Spline Function

Xinming Zhang, He Huang, and Xi Zhang

***Abstract*—In this paper, a swarm intelligence technique is presented for solving the fractional differential equations (FDE) based on the cubic spline function and cuckoo search algorithm. In this technique, we approximate FDE with the cubic spline approximation and use the cuckoo search algorithm as a tool for the accurate and rapid solution. Based on this method, the given problem is transformed into a problem for solving a nonlinear equation system, and by solving this system, we obtain the solution of FDE. Furthermore, special attention is given to the error analysis of this method. The presented scheme is evaluated on two initial value problems of FDE. The numerical simulation results demonstrate that the proposed algorithm has higher accuracy and is feasible and effective in solving initial value problem of FDE.**

***Index Terms*—fractional differential equation, cubic spline approximation, cuckoo search algorithm**

I. INTRODUCTION

WITH the development of science and technology, great breakthroughs have been made in theoretical analysis and numerical algorithm of fractional calculus. Analytical solutions of fractional differential equations are usually represented by some special functions, such as Green function, Mittag-Leffler function, and so on. Up to now, the main methods to solve the analytical solutions of fractional differential equations include Fourier transform, Mellin transform, Laplace transform, etc. However, it is an extremely difficult task to find the analytical solution for general fractional differential equation. Therefore, many researchers tend to use numerical approaches to solve the fractional differential equations in the recent thirty years, including linear multi-step method[1-2], finite difference method[3-4], Adomian decomposition method[5], homotopy perturbation method[6-7], variational iteration method[8-9], artificial neural network method[10], collocation method[11-12] and operation matrix method[13-14], the Legendre wavelet method[15], Chebyshev wavelets method[16], the hybrid Taylor series expansion with MHPM[17] , Stochastic methods[18-20], and so on. However, there is still room for investigation into numerical methods which can improve results in terms of accuracy and reliability, with better convenience, e.g. modern intelligent optimization algorithms. These random search algorithms generally based on biological intelligence or physical phenomena and are not perfect in theory. However, from a practical viewpoint, such algorithms usually do not require the continuity and convexity of the objective function and constraints. Even if there is no analytical expression, it is quite adaptable to the uncertain data in calculation, which can overcome the limitations of traditional methods to some extent.

In this paper, Cuckoo Search (CS) algorithm along with cubic splined approximation is used, for the first time as per our literature survey, to solve fractional differential equation. The CS is a swarm intelligent algorithm developed by Yang and Deb in 2009 that inspired from the nature[21]. Due to its favorable efficiency, CS has been attracting considerable attentions since it was born and has shown promising superiority in many science and engineering fields, such as inverse problems and shape optimization[22], phase equilibrium and stability calculations[23], structural optimization[24], hydraulic parameter estimation problem[25], solution of nonlinear equation system[26], multi-objective optimal power flow[27], hyperspectral image classification[28], and so on. To the best of our knowledge, the application of CS along with cubic splined approximation (CS-CS) to solve fractional differential equation has not been reported, and for the first time, this topic is investigated in the literature. The aim of our study is to identify the relative strengths of the proposed algorithm for the solution of fractional differential equation. This study shows that CS-CS algorithm offers a reliable performance for solving these differential equation of fractional order.

The rest of this paper is organized as follows: In Section 2 some basic definitions and the proposed numerical algorithm (CS-CS) are given. Section 3 shows the error analysis of the presented method. In Section 4, several numerical experiments are conducted to verify the feasibility and effectiveness of the proposed method for solving fractional differential equations, and finally, the conclusions are derived in Section 5.

Manuscript received March 14, 2019; revised September 16, 2019. This work was supported by the Natural Science Foundation of Guangdong Province under Grant No. 2017A030313280

Xinming Zhang (corresponding author) is with the College of Science of Harbin Institute of Technology（Shenzhen）, Shenzhen, Guangdong, 518055 PR China(e-mail: xinmingxueshu@hit.edu.cn).

He Huang is with the Department of Probability and statistics of Harbin Institute of Technology （Shenzhen）, Shenzhen, Guangdong, 518055 PR China(e-mail: isHuanghe@163.com).

Xi Zhang is with the Department of Applied Mathematics of Harbin Institute of Technology （Shenzhen）, Shenzhen, Guangdong, 518055 PR China(e-mail:math_hitsz@163.com).

II. Mathematical Model for solving FRDE

For positive real number v , $0 \le n-1 < v \le n$, the order v Caputo fractional derivative of the function $f(t)$ defined on the interval $[0,T]$ is

$$ {}_{a}^{C}D_{t}^{v} f(t) = \begin{cases} \dfrac{1}{\Gamma(n-v)} \int_0^t \dfrac{f^{(n)}(\tau)\,d\tau}{(t-\tau)^{v-n+1}}, & if\ 0 \le n-1 < v < n, \\ \dfrac{d^n}{dt^n} f(t), & if\ v = n \in N. \end{cases} \tag{1} $$

The generic non-linear quadratic fractional differential equations solved in this article can be written as

$$ \frac{d^v y(t)}{dt^v} = p(t) + q(t)y(t) + r(t)y^2(t), 0 < t \le T \tag{2} $$

with initial condition given as

$$ \frac{d^k}{dt^k} y(0) = c_k, k = 0,1,2,...,n-1, \tag{3} $$

where v is the order which satisfies $v > 0$, $v \in \Re$, $n = \lceil v \rceil$. $y(t)$ is the solution of the fractional differential equation. $p(t), q(t), r(t)$ are known functions, and T, c_k are known parameters.

A. Cubic spline approximation

For the initial value problem of fractional differential equation (2) and (3), in this sub-section, we discretize fractional differential equations into nonlinear algebraic equations by cubic spline function.

Taking $m+1$ nodes on the interval $[0,T]$ and dividing the interval into m subintervals $[t_1,t_2],[t_2,t_3],...,[t_m,t_{m+1}]$ .Using a cubic spline function on each subinterval, that is

$$ y_i(t) = a_i t^3 + b_i t^2 + c_i t + d_i \ , t_i < t < t_{i+1} \ , i = 1,2,...,m. \tag{4} $$

Since the second derivative of the cubic spline is continuous, we have

$$ \hat{y}_i(t)\big|_{t=t_{i+1}} = \hat{y}_{i+1}(t)\big|_{t=t_{i+1}} , i = 1,2,...,m-1 $$

$$ \hat{y}_i^{(1)}(t)\big|_{t=t_{i+1}} = \hat{y}_{i+1}^{(1)}(t)\big|_{t=t_{i+1}} , i = 1,2,...,m-1 $$

$$ \hat{y}_i^{(2)}(t)\big|_{t=t_{i+1}} = \hat{y}_{i+1}^{(2)}(t)\big|_{t=t_{i+1}} , i = 1,2,...,m-1 $$

Since the above equations satisfy the initial condition $\hat{y}(0) = y_0$, we can get $at^3 + bt^2 + ct + d\big|_{t=0} = y_0$, that is $d = y_0$.

Then take l small nodes on each subinterval, and let the function value of each small node approximate the fractional differential equation. So there is

$$ p(t) + q(t)y_i(t) + r(t)y_i^2(t) = y_i^{(v)}(t) , i = 1,2,...,m \cdot l. \tag{5} $$

Thus, the problem is transformed into the following nonlinear algebraic equations with undetermined coefficients.

1) *Case 1.* $0 < v < 1$,

$$ \begin{cases} \hat{y}_i(t)\big|_{t=t_{i+1}} = \hat{y}_{i+1}(t)\big|_{t=t_{i+1}} , i = 1,2,...,m-1 \\ \hat{y}_i'(t)\big|_{t=t_{i+1}} = \hat{y}_{i+1}'(t)\big|_{t=t_{i+1}} , i = 1,2,...,m-1 \\ \hat{y}_i''(t)\big|_{t=t_{i+1}} = \hat{y}_{i+1}''(t)\big|_{t=t_{i+1}} , i = 1,2,...,m-1 \\ d_1 = y_0 \\ \hat{y}_i^{\,v}(t) = p(t) + q(t)\hat{y}(t) + r(t)\hat{y}^2(t), \\ \qquad i = 1,2,...,m \cdot l \end{cases} \tag{6} $$

where

$$ \begin{aligned} \hat{y}_i^{(v)}(t) &= \frac{1}{\Gamma(1-v)} \int_0^t (t-\tau)^{-v} (3a_i\tau^2 + 2b_i\tau + c_i)\,d\tau \\ &= \frac{3a_i}{\Gamma(1-v)} \int_0^t (t-\tau)^{-v} \tau^2 d\tau + \\ &\quad \frac{c_i}{\Gamma(1-v)} \int_0^t (t-\tau)^{-v} d\tau + \frac{2b_i}{\Gamma(1-v)} \int_0^t (t-\tau)^{-v} \tau d\tau \\ &= -\frac{3a_i}{(1-v)\Gamma(1-v)} \int_0^t \tau^2 d(t-\tau)^{1-v} + \\ &\quad \frac{c_i}{(1-v)\Gamma(1-v)} t^{1-v} - \frac{2b_i}{(1-v)\Gamma(1-v)} \int_0^t \tau d(t-\tau)^{1-v} \\ &= \frac{6a_i}{(1-v)\Gamma(1-v)} \int_0^t \tau (t-\tau)^{1-v} d\tau + \\ &\quad \frac{c_i}{(1-v)\Gamma(1-v)} t^{1-v} + \frac{2b_i}{(2-v)(1-v)\Gamma(1-v)} t^{2-v} \\ &= \frac{6a_i}{(3-v)(2-v)(1-v)\Gamma(1-v)} \cdot t^{3-v} + \\ &\quad \frac{c_i}{(1-v)\Gamma(1-v)} t^{1-v} + \frac{2b_i}{(2-v)(1-v)\Gamma(1-v)} t^{2-v}. \end{aligned} \tag{7} $$

2) *Case 2.* $1 < v < 2$,

$$ \begin{cases} \hat{y}_i(t)\big|_{t=t_{i+1}} = \hat{y}_{i+1}(t)\big|_{t=t_{i+1}} , i = 1,2,...,m-1 \\ \hat{y}_i'(t)\big|_{t=t_{i+1}} = \hat{y}_{i+1}'(t)\big|_{t=t_{i+1}} , i = 1,2,...,m-1 \\ \hat{y}_i''(t)\big|_{t=t_{i+1}} = \hat{y}_{i+1}''(t)\big|_{t=t_{i+1}} , i = 1,2,...,m-1 \\ d_1 = y_0 \\ c_1 = y_0' \\ \hat{y}_i^{\,v}(t) = p(t) + q(t)\hat{y}(t) + r(t)\hat{y}^2(t), \\ \qquad i = 1,2,...,m \cdot l. \end{cases} \tag{8} $$

where

$$ \begin{aligned} \hat{y}_i^{(v)}(t) &= \frac{1}{\Gamma(2-v)} \int_0^t (t-\tau)^{1-v} (6a_i\tau + 2b_i)\,d\tau \\ &= \frac{2b_i}{\Gamma(2-v)} \int_0^t (t-\tau)^{1-v} d\tau + \frac{6a_i}{\Gamma(2-v)} \int_0^t (t-\tau)^{1-v} \tau d\tau \\ &= \frac{2b_i}{(2-v)\Gamma(2-v)} t^{2-v} \\ &\quad + \frac{-6a_i}{(2-v)\Gamma(2-v)} \left[\tau(t-\tau)^{2-v}\big|_0^t - \int_0^t (t-\tau)^{2-v} d\tau \right] \\ &= \frac{2b_i}{(2-v)\Gamma(2-v)} t^{2-v} + \frac{6a_i}{(3-v)(2-v)\Gamma(2-v)} t^{3-v}. \end{aligned} \tag{9} $$

3) Case 3. $2<v<3$,

$$\begin{cases} \hat{y}_i(t)\big|_{t=t_{i+1}} = \hat{y}_{i+1}(t)\big|_{t=t_{i+1}}, i=1,2,...,m-1 \\ \hat{y}_i'(t)\big|_{t=t_{i+1}} = \hat{y}_{i+1}'(t)\big|_{t=t_{i+1}}, i=1,2,...,m-1 \\ \hat{y}_i''(t)\big|_{t=t_{i+1}} = \hat{y}_{i+1}''(t)\big|_{t=t_{i+1}}, i=1,2,...,m-1 \\ d_1 = y_0 \\ c_1 = y_0' \\ 2b_1 = y_0'' \\ \hat{y}_i^{v}(t) = p(t)+q(t)\hat{y}(t)+r(t)\hat{y}^2(t), \\ \qquad i=1,2,...,m\cdot l. \end{cases} \tag{10}$$

where

$$\begin{aligned} \hat{y}_i^{(v)}(t) &= \frac{1}{\Gamma(3-v)}\int_0^t (t-\tau)^{2-v}\,6a_i \mathrm{d}\tau \\ &= \frac{-6a_i}{\Gamma(3-v)}\cdot\frac{1}{3-v}(t-\tau)^{3-v}\Big|_0^t \\ &= 0-\frac{-6a_i}{\Gamma(3-v)}\cdot\frac{1}{3-v}\cdot t^{3-v} \\ &= \frac{6a_i}{\Gamma(3-v)}\cdot\frac{1}{3-v}\cdot t^{3-v}. \end{aligned} \tag{11}$$

To sum up, transforming the initial value problem of fractional differential equation into a nonlinear equation model involves three steps. First, we divide the interval $[0,T]$ into m subintervals , $[t_1,t_2],[t_2,t_3],...,[t_m,t_{m+1}]$, and express the unknown function with cubic spline function in each interval $\hat{y}_i(t)=a_it^3+b_it^2+c_it+d_i$, $t_i<t<t_{i+1}$ $i=1,2,...,m$ .Second, based on the fact that the second derivative of cubic spline function is continuous, we transform the fractional differential equation into a model of nonlinear equations. Third, for the part $\hat{y}_i^{(v)}(t)$ which is difficult to solve in the nonlinear equations model, we discuss it in several cases.

B. Implementation of cuckoo search algorithm

After transforming the fractional differential equation into a nonlinear equations model, common traditional algorithms for solving nonlinear algebraic equations include newton method, conjugate gradient method, least squares method, etc. However, it has been pointed out that these methods require extremely high in the selection of initial points. In recent years, several modern intelligent algorithms have been developed to calculate nonlinear equations, such as Genetic algorithm, particle swarm algorithm, ant colony algorithm, etc. These algorithms not only overcome the problem of selecting the initial points existing in traditional algorithms, but also have strong global optimization ability to some extent. In this section, we will use the cuckoo search algorithm to solve the transformed nonlinear equations. In order to better implement the cuckoo search algorithm, three ideal states are assumed:

(1) Each cuckoo produces only one egg at a time, and randomly chooses a nest to place it;

(2) In the process of searching bird's nest, we calculate the optimal nest position and save it to the next generation;

(3) The probability of host finding cuckoo's eggs and abandoning them is p_a, $p_a\in[0,1]$.

Thus, the update formula of cuckoo search can be expressed as

$$x_i^{(t+1)} = x_i^{(t)} + \alpha\cdot L(\beta), i=1,2,3,...,n, \tag{12}$$

where $x_i^{(t)}$ indicates the position of the i th bird nest in the t th iteration; α represents the step size, usually take $\alpha=1$; $L(\beta)$ obeys Levy distribution

$$L(\beta) = 0.01\times\frac{u}{|v|^{1/\beta}}\times(x_j^{(t)}-x_i^t), 0<\beta\le 2, \tag{13}$$

u,v obeys normal distribution, $u\sim N(0,\delta_u^2), v\sim N(0,\delta_v^2)$,

$$\begin{cases} \delta_u = \left\{\dfrac{\Gamma(1+\beta)\sin(\pi\beta/2)}{\Gamma\left[(1+\beta)/2\right]\times 2^{(\beta-1)/2}\beta}\right\}^{1/\beta} \\ \delta_v = 1. \end{cases} \tag{14}$$

In this way, we can refer to the cuckoo search for the optimal nest and hatching process to implement the cuckoo search algorithm. The specific steps are as follows:

Step 1: Define and initialize the objective function $f(X)$, $X=(x_1,x_2,...,x_d)^T$, d is the dimension of bird's nest, and randomly generate n initial nest position $X_i(i=1,2,...,n)$. Initialize the rejection probability p_a .

Step 2: Calculate the value of objective function of each bird nest position, and select the bird nest with the optimal value.

Step 3: Preserve the optimal nest location of the previous generation, and update the nest location with Levy flight (13).

Step 4: Compare the current value of position function with the previous optimal value. If better, update the value of the current objective function, otherwise, retain the optimal value of the previous generation.

Step 5: After updating the position, generate the random number $r\in[0,1]$. If $r>p_a$, renew $x_i^{(t+1)}$ and compare the new nest, then calculate the global position pb_t^* .

Step 6: Determine whether $f(pb_t^*)$ meets the maximum iterations or minimum error requirement, and if so, the output $f(pb_t^*)$ is the global optimal solution gb .Otherwise, return to step 2.

III. Error analysis

In this section, we will analyze the error of approximating the solution function of the fractional differential equation with cubic spline function, and derive the convergence order.

For general nonlinear quadratic fractional differential equation

$$\frac{d^v y(t)}{dt^v} = p(t)+q(t)y(t)+r(t)y^2(t), t=[0,T], v>0, \tag{15}$$

$$\frac{d^k}{dt^k}y(0) = C_k, k=0,1,2,...,n-1, n=\lceil v\rceil, \tag{16}$$

Taking $m+1$ nodes on the interval $[0,T]$ and dividing the interval into m subintervals, $[t_1,t_2],[t_2,t_3],...,[t_m,t_{m+1}]$. Using a cubic spline function on each subinterval, one gets

$$\hat{y}_i(t) = a_it^3+b_it^2+c_it+d_i, t_i<t<t_{i+1}, \quad i=1,2,...,m. \tag{17}$$

and

$$\hat{y}_i'(t) = 3a_i t^2 + 2b_i t + c_i, \hat{y}_i''(t) = 6a_i t + 2b_i, \hat{y}_i'''(t) = 6a_i \tag{18}$$

Using Taylor expansion on the interval $[t_i, t_{i+1}]$, we have

$$\begin{aligned}\hat{y}_i(t) &= a_i t^3 + b_i t^2 + c_i t + d_i \\ &= a_i t_i^3 + b_i t_i^2 + c_i t_i + d_i \\ &\quad + (3a_i t_i^2 + 2b_i t_i + c_i)(t - t_i) \\ &\quad + \frac{(6a_i t_i + 2b_i)}{2!}(t - t_i)^2 + \frac{6a_i}{3!}(t - t_i)^3 + o(h^3) \\ &= \hat{y}_i(t_i) + \hat{y}_i'(t_i)(t - t_i) + \frac{\hat{y}_i''(t_i)}{2!}(t - t_i)^2 \\ &\quad + \frac{\hat{y}_i'''(t_i)}{3!}(t - t_i)^3 + o(h^3),\end{aligned} \tag{19}$$

$$\begin{aligned}\hat{y}_i'(t) &= 3a_i t^2 + 2b_i t + c_i \\ &= 3a_i t_i^2 + 2b_i t_i + c_i + (6a_i t_i + 2b_i)(t - t_i) \\ &\quad + \frac{6a_i}{2!}(t - t_i)^2 + o(h^2),\end{aligned} \tag{20}$$

$$\begin{aligned}\hat{y}_i''(t) &= 6a_i t + 2b_i \\ &= 6a_i t_i + 2b_i + 6a_i (t - t_i) + o(h).\end{aligned} \tag{21}$$

The real analytic solution $y_i(t)$ of (15) and (16) can be expanded to

$$\begin{aligned}y_i(t) &= y_i(t_i) + y_i'(t_i)(t - t_i) + \frac{y_i''(t_i)}{2!}(t - t_i)^2 \\ &\quad + \frac{y_i'''(t_i)}{3!}(t - t_i)^3 + \ldots + \frac{y_i^{n}(t_i)}{n!}(t - t_i)^n + o(h^n) \\ y_i'(t) &= y_i'(t_i) + y_i''(t_i)(t - t_i) + \frac{y_i'''(t_i)}{2!}(t - t_i)^2 \\ &\quad + \ldots + \frac{y_i^{(n)}(t_i)}{(n-1)!}(t - t_i)^{n-1} + o(h^{n-1}) \\ y_i''(t) &= y_i''(t_i) + y_i'''(t_i)(t - t_i) \\ &\quad + \ldots + \frac{y_i^{(n)}(t_i)}{(n-2)!}(t - t_i)^{n-2} + o(h^{n-2})\end{aligned}$$

Then, we get

$$\begin{aligned}&|y_i(t) - \hat{y}_i(t)| \\ &= \Big| y_i(t_i) - \hat{y}_i(t_i) + (y_i'(t_i) - \hat{y}_i'(t_i))(t - t_i) \\ &\quad + (\frac{y_i''(t_i)}{2!} - \frac{\hat{y}_i''(t_i)}{2!})(t - t_i)^2 + (\frac{y_i'''(t_i)}{3!} - \frac{\hat{y}_i'''(t_i)}{3!})(t - t_i)^3 - o(h^3) \\ &\quad + \frac{y_i^{(4)}(t_i)}{4!}(t - t_i)^4 + \ldots + \frac{y_i^{(n)}(t_i)}{n!}(t - t_i)^n + o(h^n) \Big| \\ &|y_i'(t) - \hat{y}_i'(t)| \\ &= \Big| y_i'(t_i) - \hat{y}_i'(t_i) + (y_i''(t_i) - \hat{y}_i''(t_i))(t - t_i) \\ &\quad + (\frac{y_i'''(t_i)}{2!} - \frac{\hat{y}_i'''(t_i)}{2!})(t - t_i)^2 - o(h^2) \\ &\quad + \frac{y_i^{(4)}(t_i)}{3!}(t - t_i)^3 + \ldots + \frac{y_i^{(n)}(t_i)}{(n-1)!}(t - t_i)^{n-1} + o(h^{n-1}) \Big|\end{aligned}$$

$$\begin{aligned}&|y_i''(t) - \hat{y}_i''(t)| \\ &= \Big| y_i''(t_i) - \hat{y}_i''(t_i) + (y_i'''(t_i) - \hat{y}_i'''(t_i))(t - t_i) - o(h) \\ &\quad + \frac{y_i^{(4)}(t_i)}{2!}(t - t_i)^2 + \frac{y_i^{(5)}(t_i)}{3!}(t - t_i)^3 \\ &\quad + \ldots + \frac{y_i^{(n)}(t_i)}{(n-2)!}(t - t_i)^{n-2} + o(h^{n-2}) \Big|\end{aligned}$$

According to (10), fractional differential equation (15) and (16) can be reduced to the following equations

$$\begin{cases} \hat{y}_i(t)\big|_{t=t_{i+1}} = y_{i+1}(t)\big|_{t=t_{i+1}}, i = 1,2,\ldots,m-1 \\ \hat{y}_i'(t)\big|_{t=t_{i+1}} = y_{i+1}'(t)\big|_{t=t_{i+1}}, i = 1,2,\ldots,m-1 \\ \hat{y}_i''(t)\big|_{t=t_{i+1}} = y_{i+1}''(t)\big|_{t=t_{i+1}}, i = 1,2,\ldots,m-1 \\ d_1 = y_0 = C_1 \\ c_1 = y_0' = C_2 \\ 2b_1 = y_0'' = C_3 \\ \hat{y}_i^{v}(t) = p(t) + q(t)y(t) + r(t)y^2(t), \\ \qquad i = 1,2,\ldots,m \cdot l \end{cases} \tag{22}$$

Theorem 1

If the analytical solution $y(t)$ of the problem (15) and (16) has n order continuous derivative on the interval $[0, T]$, then the local truncation error of the approximate solution function $\hat{y}_i(t), i = 1,2,\ldots m$ used to simulate the real analytic solution $y(t)$ is $o(h^3)$.

Proof:

On the interval $[t_1, t_2]$, where $t_1 = 0$,

$$\begin{aligned}&|y_1(t) - \hat{y}_1(t)| \\ &= \Big| y_1(t_1) - \hat{y}_1(t_1) + (y_1'(t_1) - \hat{y}_1'(t_1))(t - t_1) \\ &\quad + (\frac{y_1''(t_1)}{2!} - \frac{\hat{y}_i''(t_1)}{2!})(t - t_1)^2 + (\frac{y_1'''(t_i)}{3!} - \frac{\hat{y}_1'''(t_i)}{3!})(t - t_i)^3 \\ &\quad - o(h^3) + \frac{y_1^{(4)}(t_i)}{4!}(t - t_1)^4 + \ldots + \frac{y_1^{(n)}(t_1)}{n!}(t - t_1)^n + o(h^n) \Big| \\ &= \Big| C_1 - d_1 + (C_2 - c_1)t + (\frac{C_3}{2!} - \frac{2b_1}{2!}) \cdot t^2 + (\frac{y_1'''(0)}{3!} - \frac{6a_1}{3!}) \cdot t^3 \\ &\quad - o(h^3) + \frac{y_1^{(4)}(0)}{4!} \cdot t^4 + \ldots + \frac{y_1^{(n)}(0)}{n!} \cdot t^n + o(h^n) \Big| \\ &= \Big| (\frac{y_1'''(0)}{3!} - \frac{6a_1}{3!}) \cdot t^3 - o(h^3) \\ &\quad + \frac{y_1^{(4)}(0)}{4!} \cdot t^4 + \ldots + \frac{y_1^{(n)}(0)}{n!} \cdot t^n + o(h^n) \Big| \\ &\le \Big| \frac{y_1'''(0)}{3!} - \frac{6a_1}{3!} - 1 \Big| \cdot o(h^3)\end{aligned}$$

$$
\begin{aligned}
&\left|y_1'(t)-\hat{y}_1'(t)\right| \\
&=\left|y_1'(t_1)-\hat{y}_1'(t_1)+(y_1''(t_1)-\hat{y}_1''(t_1))(t-t_1)\right. \\
&+(\frac{y_1'''(t_1)}{2!}-\frac{\hat{y}_1'''(t_1)}{2!})(t-t_1)^2-o(h^2) \\
&\left.+\frac{y_1^{(4)}(t_1)}{3!}(t-t_1)^3+...+\frac{y_1^{(n)}(t_1)}{(n-1)!}(t-t_1)^{n-1}+o(h^{n-1})\right| \\
&=\left|C_2-c_1+(C_3-2b_1)\cdot t+(\frac{y_1'''(0)}{2!}-\frac{6a_1}{2!})\cdot t^2\right. \\
&\left.-o(h^2)+\frac{y_1^{(4)}(0)}{3!}t^3+...+\frac{y_1^{(n)}(0)}{(n-1)!}\cdot t^{n-1}+o(h^{n-1})\right| \\
&=\left|(\frac{y_1'''(0)}{2!}-\frac{6a_1}{2!})\cdot t^2-o(h^2)\right. \\
&\left.+\frac{y_1^{(4)}(0)}{3!}t^3+...+\frac{y_1^{(n)}(0)}{(n-1)!}\cdot t^{n-1}+o(h^{n-1})\right| \\
&\le\left|\frac{y_1'''(0)}{2!}-\frac{6a_1}{2!}-1\right|\cdot o(h^2) \\
&\left|y_1''(t)-\hat{y}_1''(t)\right| \\
&=\left|y_1''(t_1)-\hat{y}_1''(t_1)+(y_1'''(t_1)-\hat{y}_1'''(t))(t-t_1)-o(h)\right. \\
&+\frac{y_1^{(4)}(t_1)}{2!}(t-t_i)^2+\frac{y_1^{(5)}(t_1)}{3!}(t-t_1)^3 \\
&\left.+...+\frac{y_1^{(n)}(t_1)}{(n-2)!}(t-t_1)^{n-2}+o(h^{n-2})\right| \\
&=\left|C_2-2b_1+(y_1'''(t_1)-6a_1)\cdot t-o(h)\right. \\
&\left.+\frac{y_1^{(4)}(0)}{2!}\cdot t^2+\frac{y_1^{(5)}(0)}{3!}\cdot t^3+...+\frac{y_1^{(n)}(0)}{(n-2)!}\cdot t^{n-2}+o(h^{n-2})\right| \\
&\le\left|y_1'''(0)-6a_1-1\right|\cdot o(h)
\end{aligned}
$$

On the interval $[t_2,t_3]$,

$$
\begin{aligned}
&\left|y_2(t)-\hat{y}_2(t)\right| \\
&=\left|y_2(t_2)-\hat{y}_2(t_2)+(y_2'(t_2)-\hat{y}_2'(t_2))(t-t_2)\right. \\
&+(\frac{y_2''(t_2)}{2!}-\frac{\hat{y}_2''(t_2)}{2!})(t-t_2)^2+(\frac{y_2'''(t_2)}{3!}-\frac{\hat{y}_2'''(t_2)}{3!})(t-t_2)^3-o(h^3) \\
&\left.+\frac{\hat{y}_2^{(4)}(t_2)}{4!}(t-t_2)^4+...+\frac{\hat{y}_2^{(n)}(t_2)}{n!}(t-t_2)^n+o(h^n)\right| \\
&\le\left|y_1(t_2)-\hat{y}_1(t_2)\right|+\left|y_1'(t_2)-\hat{y}_1'(t_2)\right|\cdot o(h) \\
&+\left|\frac{y_1''(t_2)}{2!}-\frac{\hat{y}_1''(t_2)}{2!}\right|\cdot o(h^2)+(\frac{y_2'''(t_2)}{3!}-\frac{\hat{y}_2'''(t_2)}{3!})(t-t_2)^3-o(h^3) \\
&\left.+\frac{\hat{y}_2^{(4)}(t_2)}{4!}(t-t_2)^4+...+\frac{\hat{y}_2^{(n)}(t_2)}{n!}(t-t_2)^n+o(h^n)\right| \\
&\le\left|\frac{y_1'''(0)}{3!}-\frac{6a_1}{3!}-1\right|\cdot o(h^3)+\left|\frac{y_1'''(0)}{2!}-\frac{6a_1}{2!}-1\right|\cdot o(h^2)\cdot o(h) \\
&+\frac{1}{2!}\cdot\left|y_1'''(0)-6a_1-1\right|\cdot o(h)\cdot o(h^2)+\left|\frac{y_2'''(t_2)}{3!}-\frac{\hat{y}_2'''(t_2)}{3!}-1\right|\cdot o(h^3) \\
&+\left|\frac{\hat{y}_2^{(4)}(t_2)}{4!}(t-t_2)^4\right|+...+\left|\frac{\hat{y}_2^{(n)}(t_2)}{n!}(t-t_2)^n\right|+o(h^n) \\
&\le\left(\left|\frac{y_1'''(0)}{3!}-\frac{6a_1}{3!}-1\right|+\left|\frac{y_1'''(0)}{2!}-\frac{6a_1}{2!}-1\right|\right. \\
&\left.\frac{1}{2!}\cdot\left|y_1'''(0)-6a_1-1\right|+\left|\frac{y_2'''(t_2)}{3!}-\frac{\hat{y}_2'''(t_2)}{3!}-1\right|\right)\cdot o(h^3).
\end{aligned}
$$

In the same way, on each interval $[t_1,t_2],[t_2,t_3],...,[t_m,t_{m+1}]$, we have

$$\left|y_i(t)-\hat{y}_i(t)\right|\le\bar{C}_i\cdot o(h^3)$$

where $\bar{C}_i$ is an appropriate positive constant, $i=1,2,...,m$.

IV. Numerical Examples

In order to verify the feasibility and effectiveness of the new proposed method for solving fractional differential equations. In this section, we will present two numerical experiments based on the previous discussion.

A. Example 1

Consider fractional differential equation

$$\frac{d^v y(t)}{dt^v}=t^2+\frac{2}{\Gamma(3-v)}t^{2-v}-y(t),\quad t>0,\ y(0)=0,\ 0<v\le 1. \tag{23}$$

The exact solution of this equation is

$$y(t)=t^2.$$

According to (6), (23) can be transformed into the following equations:

$$
\begin{cases}
y_i(t)\big|_{t=t_{i+1}}=y_{i+1}(t)\big|_{t=t_{i+1}},\ i=1,2,...m-1\\
y_i^{(1)}(t)\big|_{t=t_{i+1}}=y_{i+1}^{(1)}(t)\big|_{t=t_{i+1}},\ i=1,2,...m-1\\
y_i^{(2)}(t)\big|_{t=t_{i+1}}=y_{i+1}^{(2)}(t)\big|_{t=t_{i+1}},\ i=1,2,...m-1\\
d_1=0\\
0=t^2+\dfrac{2}{\Gamma(3-v)}t^{2-v}-y_i(t)-\dfrac{c_i}{(1-v)\Gamma(1-v)}t^{1-v}\\
\quad-\dfrac{2b_i}{(2-v)(1-v)\Gamma(1-v)}t^{2-v}-\dfrac{6a_i}{(3-v)(2-v)(1-v)\Gamma(1-v)}\cdot t^{3-v},\\
\qquad i=1,2,...,m\cdot l
\end{cases}
$$

For convenience, we first select $v=0.5, m=1, l=20, T=1$. At this time, the error tolerance can reach $Tol=10^{-15}$, which takes 12.313966 seconds. Moreover, in order to increase the credibility of the numerical simulation, the results are averaged by considering 30 different executions. The comparison results are shown in Fig.1. As we can see that the solution obtained by CS-CS is almost close to the real analytical solution. Table I lists the numerical solution and the error comparison of these methods. From Table I, we can find that the accuracy of the cuckoo search algorithm based on cubic spline approximation (CS-CS) is much higher than the particle swarm optimization based on artificial neural network (PSO-ANN) and *Grünwald – Letnikov* classical numerical method[19].

TABLE I
PSO-ANN, GL, CS-CS COMPARISON TABLE AT t = [0,1]

t	Exact solution	Solution of GL	Solution of PSO-ANN	Solution of CS-CS	Error by GL	Error by PSO-ANN	Error by CS-CS
0.0	0.00000	0.00000	5.45e-6	0.00000000000000	0	5.45e-6	5.2e-18
0.2	0.04000	0.04007	0.04045	0.04000000000000	7e-5	4.52e-4	0
0.3	0.09000	0.09011	0.09072	0.09000000000000	1.1e-4	7.18e-4	0
0.4	0.16000	0.16013	0.16045	0.16000000000000	1.3e-4	4.51e-4	0
0.5	0.25000	0.25016	0.24958	0.25000000000000	1.6e-4	4.25e-4	0
0.6	0.36000	0.36019	0.35827	0.36000000000000	1.9e-4	1.73e-3	-5.551e-17
0.7	0.49000	0.49021	0.48693	0.49000000000000	2.1e-4	3.07e-3	-5.551e-17
0.8	0.64000	0.64023	0.63619	0.64000000000000	2.3e-4	3.81e-3	0
0.9	0.81000	0.81026	0.80688	0.81000000000000	2.6e-4	3.11e-3	0
1.0	1.00000	1.00028	1.00004	1.00000000000000	2.8e-4	4.40e-5	0

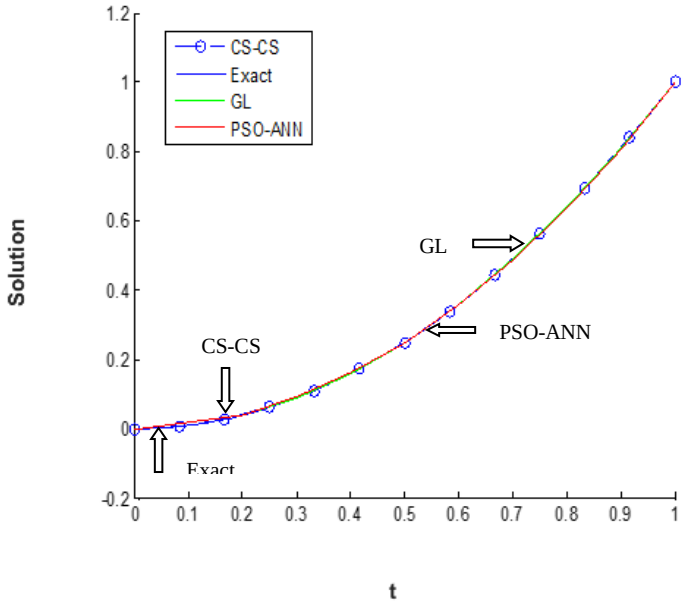

Fig. 1. Comparison of PSO-ANN, GL, CS-CS at t = [0, 1]

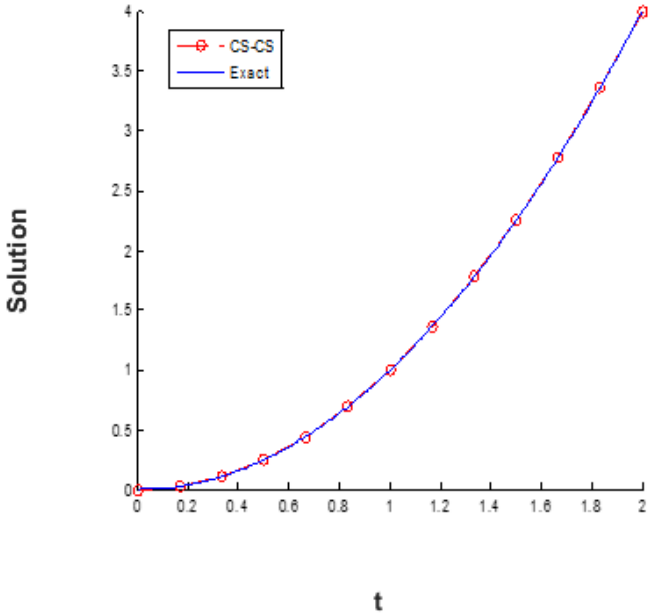

Fig. 2. Simulation diagram of by CS-CS at t = [0, 2]

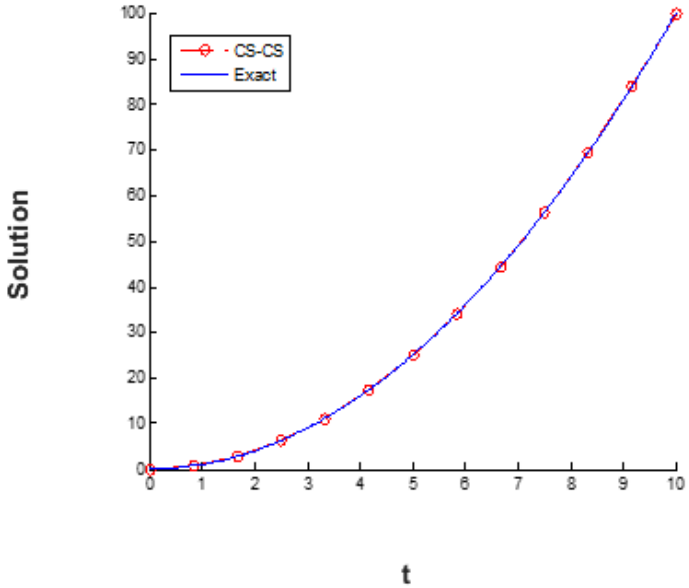

Fig. 3. Simulation diagram of by CS-CS at t = [0, 10]

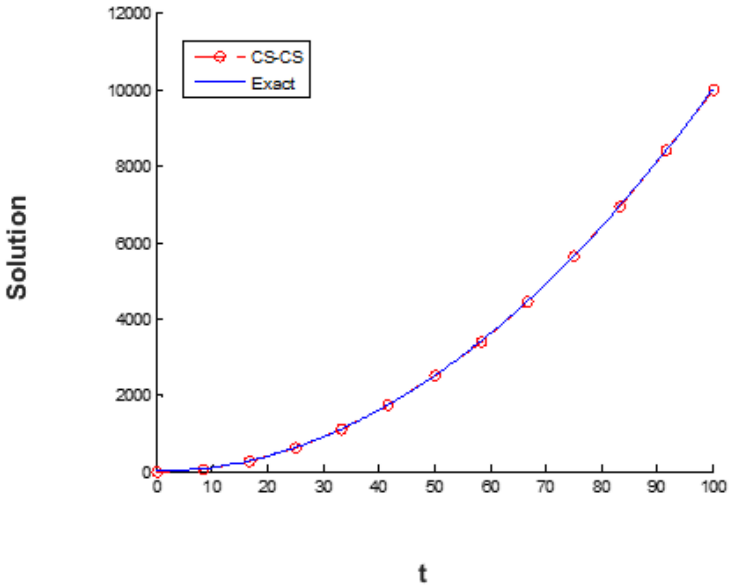

Fig.4. Simulation diagram of by CS-CS at t = [0, 100]

TABLE II
COMPARISON OF RESULTS FOR THE SOLUTION
OF EXAMPLE 1 ($v = 0.25$, $t = [0, 1]$)

t	Exact solution	Error by EOC	Error by FWCW	Mean error by CS-CS
0.09375	0.0087890625	1.15137021309e-5	0.43017672757 e-13	1.243218491116710e-17
0.18750	0.0351562500	1.34008681836e-5	0.53908266740 e-13	1.989149585786740e-17
0.28125	0.0791015625	1.45485346092e-5	0.24466539905 e-13	1.850371707708590e-17
0.37500	0.1406250000	1.53757162245e-5	0.03941291737 e-13	1.757853122323160e-17
0.46875	0.2197265625	1.60223410662e-5	0.64837024638 e-13	2.312964634635740e-17
0.56250	0.3164062500	1.65528139359e-5	0.54012350148 e-13	2.035408878479450e-17
0.65625	0.4306640625	1.70022170524e-5	0.09825473768 e-13	3.515706244646330e-17
0.75000	0.5625000000	1.73918021056e-5	0.17985612999 e-13	1.480297366166880e-17
0.84375	0.7119406250	1.77354280767e-5	0.31863400807 e-13	3.330669073875470e-17
0.93750	0.8789062500	1.80426383865e-5	2.01283434365 e-13	4.810966440042350e-17

TABLE III
COMPARISON OF RESULTS FOR THE SOLUTION
OF EXAMPLE 1 ($v = 0.75$, $t = [0, 1]$)

t	Exact solution	Error by EOC	Error by FWCW	Mean error by CS-CS
0.09375	0.0087890625	0.201409844130e-3	0.347308987125e-13	1.295260195396020e-17
0.18750	0.0351562500	0.312234854242e-3	0.022967738822e-13	2.012279232133100e-17
0.28125	0.0791015625	0.397141377633e-3	0.198174809896e-13	2.359223927328460e-17
0.37500	0.1406250000	0.466219134874e-3	0.293931545769e-13	2.683038976177460e-17
0.46875	0.2197265625	0.524234122143e-3	0.221489493413e-13	2.220446049250310e-17
0.56250	0.3164062500	0.573965121404e-3	0.194289029309e-13	2.775557561562890e-17
0.65625	0.4306640625	0.617222689041e-3	0.170974345793e-13	2.775557561562890e-17
0.75000	0.5625000000	0.655271385754e-3	0.091038288019e-13	2.220446049250310e-17
0.84375	0.7119406250	0.689037602412e-3	0.119904086659e-13	4.810966440042350e-17
0.93750	0.8789062500	0.719224088406e-3	0.275335310107e-13	6.661338147750940e-17

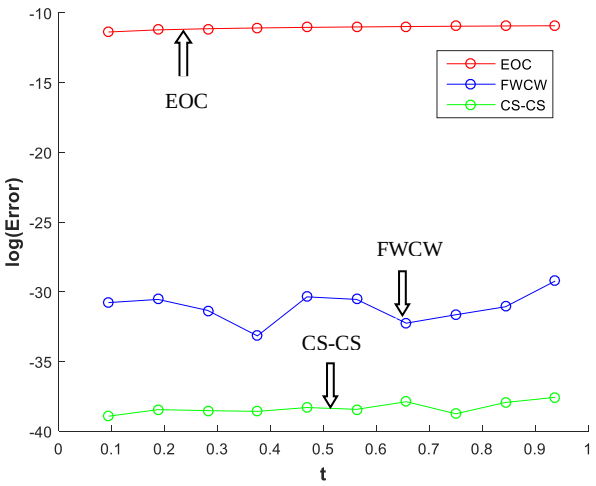

Fig.5. Error comparison of EOC,FWCW,CS-CS at $v = 0.25$, $t = [0, 1]$

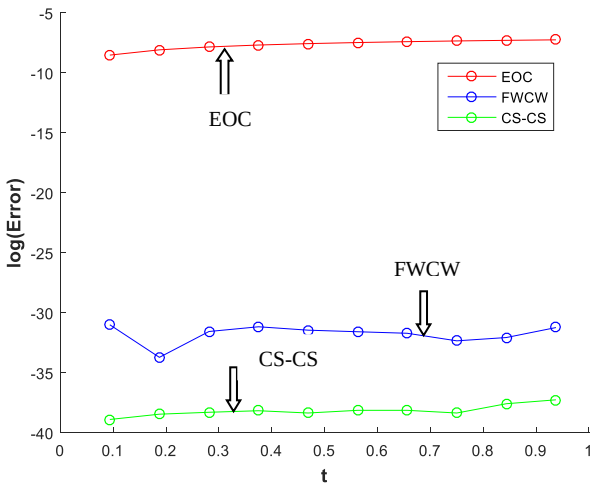

Fig.6. Error comparison of EOC,FWCW,CS-CS at $v = 0.75$, $t = [0, 1]$

In addition, CS-CS not only has high accuracy at interval $t \in [0,1]$, but also maintains high accuracy and stability when the range of T becomes larger. Fig. 2, 3, 4 are the simulated diagram of numerical solutions obtained by

CS-CS at intervals $t \in [0,2]$, $t \in [0,10]$ and $t \in [0,100]$, respectively. These three cases take 15.068018 seconds, 17.216479 seconds and 20.749634 seconds, respectively. Even for the case $t \in [0,100]$, the accuracy of the numerical solution can be as high as 10^{-13}, which shows that the algorithm still has high accuracy and stability when the T value keeps increasing.

In order to show the superiority of CS-CS in further, the value of the fractional order derivative v is taken as 0.25 and 0.75. The corresponding results are summarized in Table II, Table III and Fig.5, Fig.6. It also contains reported results with FWCM[29] and EOC[30]. It can be inferred that our algorithm provides an approximate solution to the fractional differential equation more effectively.

B. Example 2

Consider the fractional differential equation

$$\frac{\mathrm{d}^{0.8}y(t)}{\mathrm{d}t^{0.8}} = -y'(t) - (1+t)\cdot y(t) + \frac{14}{\Gamma(3.8)}t^{1.8} + \frac{5}{2}t^2 + \frac{5}{\Gamma(3.8)}(1+t)t^{2.8} \quad (24)$$

The initial condition is

$$y(0) = 0$$

The exact solution of this equation is

$$y(t) = \frac{5}{\Gamma(3.8)}t^{2.8}$$

According to (6), (24) can be transformed into the following equations

$$\begin{cases} \hat{y}_i(t)\big|_{t=t_{i+1}} = \hat{y}_{i+1}(t)\big|_{t=t_{i+1}}, \quad i = 1,2,...,m-1 \\ \hat{y}_i^{(1)}(t)\big|_{t=t_{i+1}} = \hat{y}_{i+1}^{(1)}(t)\big|_{t=t_{i+1}}, \quad i = 1,2,...,m-1 \\ \hat{y}_i^{(2)}(t)\big|_{t=t_{i+1}} = \hat{y}_{i+1}^{(2)}(t)\big|_{t=t_{i+1}}, \quad i = 1,2,...,m-1 \\ d_1 = 0 \\ 0 = -\hat{y}'(t) - (1+t)\cdot\hat{y}(t) + \frac{14}{\Gamma(3.8)}t^{1.8} + \frac{5}{2}t^2 \\ \quad + \frac{5}{\Gamma(3.8)}(1+t)\cdot t^{2.8} - \frac{c_i}{(1-v)\Gamma(1-v)}t^{1-v} \\ \quad - \frac{2b_i}{(2-v)(1-v)\Gamma(1-v)}t^{2-v} - \frac{6a_i}{(3-v)(2-v)(1-v)\Gamma(1-v)}\cdot t^{3-v}, \\ \quad i = 1,2,...,m\cdot l. \end{cases}$$

In the following, we will use cuckoo search algorithm to calculate the above equations and discuss the influence of the parameter l.

(1)Choose $m = 2, l = 10, T = 1, Tol = 1.1227$. It takes 16.439011 seconds. The coefficient solutions are:

$$\begin{cases} a_1 = 0.925561212447776 \\ b_1 = 0.179615628191602 \\ c_1 = -0.015345206321721 \\ d_1 = 0.000000008744638 \end{cases} and \begin{cases} a_2 = 0.738891012901748 \\ b_2 = 0.459623589255583 \\ c_2 = -0.155256770376138 \\ d_2 = 0.023283177842368, \end{cases}$$

We get the approximate solution of the equation:

$$\hat{y}(t) = \begin{cases} y_1(t) = 0.925561212447776t^3 + 0.179615628191602t^2 \\ \quad -0.015345206321721t + 0.000000008744638 \cdots\cdots 0 < t < 0.5 \\ y_2(t) = 0.738891012901748t^3 + 0.459623589255583t^2 \\ \quad -0.155256770376138t + 0.023283177842368 \cdots\cdots 0.5 < t < 1. \end{cases}$$

(2) Choose $m = 2, l = 20, T = 1, Tol = 1.1714$. It takes 27.7729821 seconds. The coefficient solutions are:

$$\begin{cases} a_1 = 0.919498494305771 \\ b_1 = 0.183727300560656 \\ c_1 = -0.015943443439072 \\ d_1 = -0.000000014508539 \end{cases} and \begin{cases} a_2 = 0.750548643123485 \\ b_2 = 0.437107118636707 \\ c_2 = -0.142555742170239 \\ d_2 = 0.021816521494590, \end{cases}$$

The approximate solution of the equation is:

$$\hat{y}(t) = \begin{cases} y_1(t) = 0.919498494305771t^3 + 0.183727300560656t^2 \\ \quad -0.015943443439072t - 0.000000014508539 \cdots\cdots 0 < t < 0.5 \\ y_2(t) = 0.750548643123485t^3 + 0.437107118636707t^2 \\ \quad -0.142555742170239t + 0.021816521494590 \cdots\cdots 0.5 < t < 1. \end{cases}$$

(3) Choose $m = 2, l = 30, T = 1, Tol = 1.2156$. It takes 27.7729821 seconds. The coefficient solutions are:

$$\begin{cases} a_1 = 0.925913752517327 \\ b_1 = 0.178866218377388 \\ c_1 = -0.015223271257590 \\ d_1 = -0.000000025245317 \end{cases} and \begin{cases} a_2 = 0.754732405733558 \\ b_2 = 0.427150121741720 \\ c_2 = -0.134842553318408 \\ d_2 = 0.019562451330709, \end{cases}$$

This yields:

$$\hat{y}(t) = \begin{cases} y_1(t) = 0.925913752517327t^3 + 0.178866218377388t^2 \\ \quad -0.015223271257590t - 0.000000025245317 \cdots\cdots 0 < t < 0.5 \\ y_2(t) = 0.754732405733558t^3 + 0.427150121741720t^2 \\ \quad -0.134842553318408t + 0.019562451330709 \cdots\cdots 0.5 < t < 1. \end{cases}$$

Table IV lists the error comparison between the numerical solutions obtained by CS-CS and the analytical solution when $l = 10, l = 20$, and $l = 30$. The Mean square error(*MSE*) of three cases are *MSE_l10*=2.2772e-5, *MSE_l20*=1.7102e-5, *MSE_l30*=2.1554e-5, respectively. We can find that the error is relatively small at around 10^{-5} for $l = 20$.

Fig.7 shows the numerical solutions obtained with CS-CS ($l = 20$) and difference method(DM) in [31]. From Fig.7, we can observe that the solution obtained by CS-CS is closer to analytical solution than DM. In addition, Table V lists the error comparison between the exact solutions and numerical solutions by CS-CS and DM methods at several points. It is shown that the numerical solution accuracy of CS-CS is around $10^{-4} \sim 10^{-5}$, which is more close to the real analytical solution than DM method. Table VI lists the maximum error, the minimum error and the mean error of CS-CS for 30 experiments.

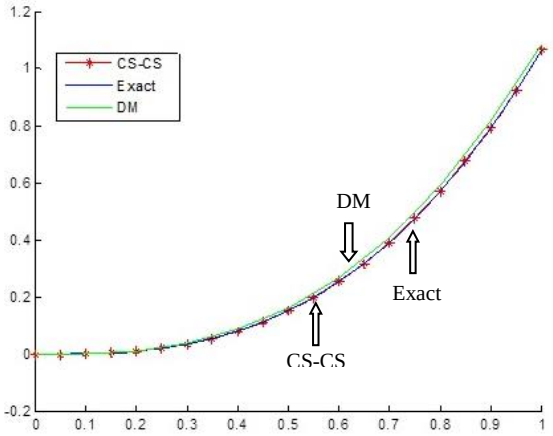

Fig.7. Comparison of CS-CS,DM at t=[0,1],v=0.8

C. Example 3

Consider fractional differential equation

$$\frac{\mathrm{d}^v y(t)}{\mathrm{d}t^v} = \frac{\Gamma(p+1)}{\Gamma(p-v+1)} t^{p-v} + y^2(t) - t^{2p}, \qquad (25)$$

The initial condition is

$$y(0) = 0 \qquad \text{for } 0 < v \le 1$$

$$y(0) = y'(0) = 0 \qquad \text{for } 1 < v \le 2$$

The exact solution of this equation is

$$y(t) = t^p$$

In the following, we will discuss the solution of (25) in different cases.

Case 1. $p = 3, v = 0.5$

According to (7), (25) can be transformed into the following equations

$$\begin{cases} \hat{y}_i(t)\big|_{t=t_{i+1}} = \hat{y}_{i+1}(t)\big|_{t=t_{i+1}}, \quad i = 1,2,...,m-1 \\ \hat{y}_i^{(1)}(t)\big|_{t=t_{i+1}} = \hat{y}_{i+1}^{(1)}(t)\big|_{t=t_{i+1}}, \quad i = 1,2,...,m-1 \\ \hat{y}_i^{(2)}(t)\big|_{t=t_{i+1}} = \hat{y}_{i+1}^{(2)}(t)\big|_{t=t_{i+1}}, \quad i = 1,2,...,m-1 \\ d_1 = 0 \\ 0 = \dfrac{\Gamma(4)}{\Gamma(4-v)} t^{p-v} + y_i^2(t) - t^{2p} - \dfrac{c_i}{(1-v)\Gamma(1-v)} t^{1-v} \\ \quad - \dfrac{2b_i}{(2-v)(1-v)\Gamma(1-v)} t^{2-v} - \dfrac{6a_i}{(3-v)(2-v)(1-v)\Gamma(1-v)} t^{3-v}, \\ \quad i = 1,2,...,m \cdot l. \end{cases}$$

The parameter is chosen as $m = 1,\ T = 1, l = 10, Tol = 0.4182$ .Using cuckoo algorithm to calculate the above equations, and it takes 4.8807 seconds. The simulation results are shown in Fig.8 and Table VII. As we can see that the numerical solution agree well with the analytical solution and the mean error of CS-CS can reach 10^{-7} .

2) Case 2. $p = 3, v = 1.5$

According to (9), (25) can be transformed into the following equations

$$\begin{cases} \hat{y}_i(t)\big|_{t=t_{i+1}} = \hat{y}_{i+1}(t)\big|_{t=t_{i+1}}, i = 1,2,\cdots,m-1 \\ \hat{y}_i^{(1)}(t)\big|_{t=t_{i+1}} = \hat{y}_{i+1}^{(1)}(t)\big|_{t=t_{i+1}}, i = 1,2,\cdots,m-1 \\ \hat{y}_i^{(2)}(t)\big|_{t=t_{i+1}} = \hat{y}_{i+1}^{(2)}(t)\big|_{t=t_{i+1}}, i = 1,2,\cdots,m-1 \\ d_1 = 0 \\ c_1 = 0 \\ 0 = \dfrac{\Gamma(4)}{\Gamma(4-v)} t^{p-v} - \dfrac{2b_i}{(2-v)\Gamma(2-v)} t^{2-v} \\ \quad - \dfrac{6a_i}{(3-v)(2-v)\Gamma(1-v)} t^{3-v} + \hat{y}_i^2(t) - t^{2p}, \\ \quad i = 1,2,\cdots,m-1. \end{cases}$$

Similarly, the parameter is chosen as $m = 1, T = 1, l = 10, Tol = 10^{-15}$. It takes 10.7952 seconds for cuckoo algorithm to calculate the above equations. Fig.9 and Table VIII give the simulation results. We can conclude that the numerical solution are found in well agreement with the analytical solution and the mean error of CS-CS can reach 10^{-16} .

V. Conclusion

In this paper, we propose a new solution scheme for the initial value problem of fractional differential equations, which is solved by cuckoo search algorithm based on cubic spline (CS-CS). A cubic spline function was introduced to transform the fractional differential equations into nonlinear equations and the Cuckoo search algorithm was applied to solve the nonlinear equation system. Furthermore, we derive the convergence order which proves the theoretical feasibility of the proposed method. By using CS-CS algorithm to solve specific examples, we find that the new method has the characteristics of high precision and fast convergence speed. Therefore, the cuckoo algorithm based on cubic spline (CS-CS) presented in this paper is feasible and effective in solving the initial value problem of fractional differential equations.

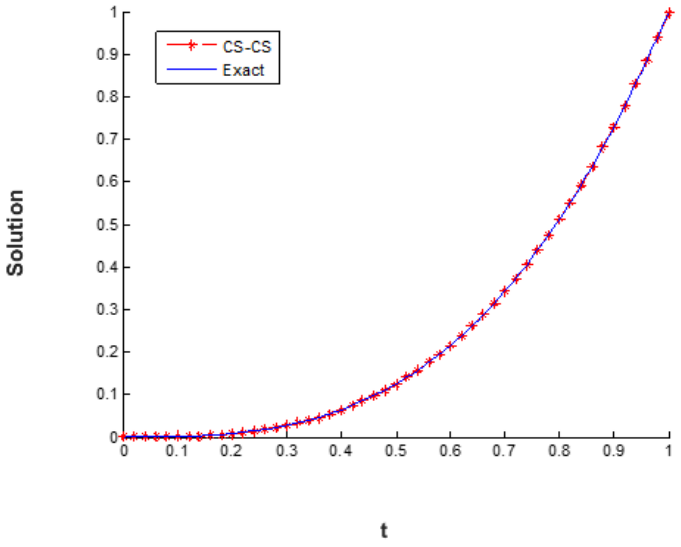

Fig. 8. Numerical results of CS-CS for p = 3, v = 0.5

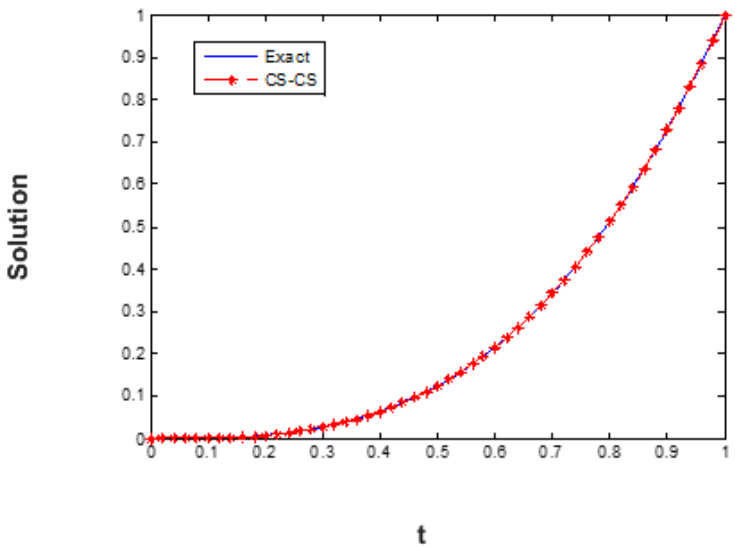

Fig. 9. Numerical results of CS-CS for p = 3, v = 1.5

TABLE IV
THE ERROR COMPARISON OF CS-CS (t=[0,1] , v = 0.8)

t	Exact Solution	Solution of CS-CS			Error by CS-CS		
		l=10	l=20	l =30	l=10	l=20	l =30
0	0	0.000000008744638	-0.000000014508539	-0.000000025245317	0.0000000087446 38	0.0000000145085 39	0.0000000252453 17
0.1	0.001688149100350	0.001187205606829	0.0011624126474 66	0.0011922235652 15	0.0005009434935 20	0.0005257364528 84	0.0004959255351 34
0.2	0.011756953201898	0.010694555759719	0.0115163767805 19	0.0115172792583 99	0.0002368708943 59	0.0002405764213 80	0.0002396739434 99
0.3	0.036588981023069	0.036552006121455	0.0365788688564 54	0.0365306243493 39	0.0000369749016 14	0.0000101121666 15	0.0000583566737 30
0.4	0.081880177860470	0.081836344323263	0.0818668798411 06	0.0817877413531 380	0.0000438335372 07	0.0000132980193 64	0.0000924365073 32
0.5	0.152942017148660	0.152926464187650	0.1528974007003 10	0.1528441127849 01	0.0000155529610 10	0.0000446164483 49	0.0000979043637 58
0.6	0.254820464320514	0.255375732989301	0.2551874223999 00	0.2552552211597 32	0.0005552686687 87	0.0003669580793 86	0.0004347568392 18
0.7	0.392360531068552	0.394737518002905	0.3942539359057 11	0.3945765489927 34	0.0023769869343 53	0.0018934048371 59	0.0022160179241 81
0.8	0.570246679673755	0.576565186503147	0.5756139321835 78	0.5763635787990 11	0.0063185068293 93	0.0053672525098 23	0.0061168991252 57
0.9	0.793030385781202	0.806412105764715	0.8047844021993 34	0.8061717930936 68	0.0134730381732 07	0.0117540164181 33	0.0131414073124 67

TABLE V
THE ERROR COMPARISON OF CS-CS, DM (t=[0,1] , v = 0.8, l=20)

t	Exact Solution	Mean solution of CS-CS	Solution of DM	Mean error by CS-CS	Error by DM
0	0	0.000000000140022	0.00000000	9.013090719625970e-09	0
0.1	0.001688149100350	0.001166142476867	0.00194987	5.220066234831710e-04	0.00026172
0.2	0.011756953201898	0.011516154287996	0.01284495	2.407989139025410e-04	0.00108709
0.3	0.036588981023069	0.036571078568839	0.03926037	1.790245423084970e-05	0.00267139
0.4	0.081880177860470	0.081851958314823	0.08701887	2.821954564697870e-05	0.00513869
0.5	0.152942017148660	0.152879836521379	0.16142110	6.218062728019690e-05	0.00847908
0.6	0.254820464320514	0.255175756183935	0.26732269	3.552918634213640e-04	0.01250222
0.7	0.392360531068552	0.394260760297919	0.40916953	1.900229229367280e-03	0.01680899
0.8	0.570246679673755	0.575655891858762	0.59101864	5.409212185006990e-03	0.02077195
0.9	0.793030385781202	0.804882193861890	0.81655453	1.185180808068810e-02	0.02395547

TABLE VI
THE ERROR OF CS-CS (t=[0,1],v=0.8,l=20)

t	Min error by CS-CS	Max error by CS-CS	Mean error by CS-CS
0	1.702279956823930e-09	3.888834435686740e-10	9.013090719625970e-09
0.1	5.265364140468620e-04	5.191041327268500e-04	5.220066234831710e-04
0.2	2.437314167992880e-04	2.409312625101370e-04	2.407989139025410e-04
0.3	1.818830810562670e-05	2.247311334883930e-05	1.790245423084970e-05
0.4	2.988866827845220e-05	3.409837630968800e-05	2.821954564697870e-05
0.5	7.434221558075270e-05	6.170388040974140e-05	6.218062728019690e-05
0.6	3.184493334056350e-04	3.743215470149440e-04	3.552918634213640e-04
0.7	1.819438002456540e-03	1.954542818986270e-03	1.900229229367280e-03
0.8	5.260125226887680e-03	5.520074260065730e-03	5.409212185006990e-03
0.9	1.160499907791320e-02	1.204501683071290e-02	1.185180808068810e-02

TABLE VII
THE ERROR OF CS-CS FOR $p = 3$, $v = 0.5$

t	Exact Solution	Mean solution of CS-CS	Min error by CS-CS	Max error by CS-CS	Mean error by CS-CS
0.0	0.00000000	0.000000002488263	3.575699808794530e-10	7.543891859534880e-10	6.147292989429410e-09
0.1	0.00100000	0.001000004272664	1.726569095941020e-08	2.181228103091940e-07	7.953910263397570e-08
0.2	0.00800000	0.008000007493398	1.878053475928840e-08	3.178512749980880e-07	1.153399363086850e-07
0.3	0.02700000	0.027000008015559	9.453083996135980e-09	3.339483128535210e-07	1.241476972414320e-07
0.4	0.06400000	0.064000006680766	6.165678703706770e-09	3.004224534697290e-07	1.177731709646930e-07
0.5	0.12500000	0.125000004330639	2.352477077027790e-08	2.512822264294500e-07	1.062012284040270e-07
0.6	0.21600000	0.216000001806796	3.807320952953220e-08	2.205361613605290e-07	1.076849300165240e-07
0.7	0.34300000	0.342999999950857	4.526001246007990e-08	2.421927878248910e-07	1.310367756746090e-07
0.8	0.51200000	0.511999999604442	4.053419666583120e-08	3.502606354954810e-07	1.755128204328220e-07
0.9	0.72900000	0.729000001609168	1.934478000009680e-08	5.787482337815680e-07	2.677919390126070e-07

TABLE VIII
THE ERROR OF CS-CS FOR $p = 3$, $v = 1.5$

t	Exact Solution	Mean solution of CS- CS	Min error by CS-CS	Max error by CS-CS	Mean error by CS-CS
0.0	0.00000000	0.001000000000000	2.168404344971009e-19	6.505213034913030e-19	5.348730717595160e-19
0.1	0.00100000	0.008000000000000	0	3.469446951953610e-18	1.272130549049660e-18
0.2	0.00800000	0.027000000000000	0	6.938893903907230e-18	2.775557561562890e-18
0.3	0.02700000	0.064000000000000	0	1.387778780781450e-17	4.625929269271490e-18
0.4	0.06400000	0.125000000000000	0	1.387778780781450e-17	6.476300976980080e-18
0.5	0.12500000	0.216000000000000	0	2.775557561562890e-17	1.110223024625160e-17
0.6	0.21600000	0.343000000000000	0	5.551115123125780e-17	5.551115123125780e-18
0.7	0.34300000	0.512000000000000	0	0	0
0.8	0.51200000	0.729000000000000	0	1.110223024625160e-16	7.401486830834380e-18
0.9	0.72900000	1.000000000000000	0	1.110223024625160e-16	1.110223024625160e-17

REFERENCES

[1] C. Lubich, "Discretized fractional calculus," *SIAM Journal on Mathematical Analysis*, vol.17, no.3, pp. 704–719, 1986.

[2] N. J. Ford, J. A. Connolly , "Systems-based decomposition schemes for the approximate solution of multi-term fractional differential equations," *Journal of Computational and Applied Mathematics*, vol.229, no. 2, pp.382–391, 2009.

[3] I. Podlubny, *Fractional differential equations* (Book style), 1999.

[4] Z. M. Odibat, "Computational algorithms for computing the fractional derivatives of functions," *Elsevier Science Publishers B. V*, 2009.

[5] Z. Odibat, S. Momani, "Numerical methods for nonlinear partial differential equations of fractional order," *Applied Mathematical Modelling*, vol.32, no.1, pp. 28–39, 2008.

[6] Z. Odibat, S. Momani, "Modified homotopy perturbation method: application to quadratic differential equation of fractional order," *Chaos, Solitons Fractal* , vol.36, no.1, pp.167–174, 2008.

[7] Y. Tan, S. Abbasbandy, "Homotopy analysis method for quadratic Riccati differential equation," *Communications in Nonlinear Science and Numerical Simulation*, vol.13, no.3, pp. 539–546, 2008.

[8] G. C. Wu, E. W. M. Lee , "Fractional variational iteration method and its application," *Physics Letters A*, vol.374, no.25, pp. 2506–2509, 2010.

[9] S. Abbasbandy, "A new application of He's variational iteration method for quadratic Riccati differential equation by using Adomian's polynomials," *Journal of Computational & Applied Mathematics*, vol.207, no.1, pp. 59–63, 2007.

[10] H. S. Yazdi, R. Pourreza, "Unsupervised adaptive neural-fuzzy inference system for solving differential equations," *Applied Soft Computing*, vol.10, no.1, pp.267–275, 2010.

[11] A. Pedas, E. Tamme, "Spline collocation methods for linear multi-term fractional differential equations," *Journal of Computational & Applied Mathematics*, vol.236, no.2, pp. 167–176, 2011.

[12] E.A.Rawashdeh, "Numerical solution of fractional integro–differential equations by collocation method," *Applied Mathematics and Computation*, vol.176, no.1, pp.1–6, 2006.

[13] A. Saadatmandi, M. Dehghan, "A new operational matrix for solving fractional-order differential equations," *Computers & Mathematics with Applications*, vol.59, no.3, pp. 1326–1336, 2010.

[14] M. M. Khader, "Numerical solution of nonlinear multi-order fractioanl defferential equations by implementation of the operatioanl matrix of fractional derivative," *Stud Nonlinear Sci* , vol.2, no.1, pp. 5–12, 2011.

[15] F. Mohammadi, M. M. Hosseini, "A comparative study of numerical methods for solving quadratic differential equations," *Journal of the Franklin Institude*, vol.348, no.8, pp. 1787–1796, 2011.
[16] M. L. Li, L. F. Wang, "Numerical solution of fractional partial differential equation of parabolic type using Chebyshev wavelets method," *Engineering Letters*, vol. 26, no. 2, pp. 224–227, 2018.
[17] N. A. Khan, A. Ara, M. Jamil , "An efficient approach for solving the Riccati equation with fractional orders," *Computers and Mathematics with Applications*, vol.61, no.9, pp. 2683–2689, 2011.
[18] M. A. Z. Raja, J. A. Khan, I. M. Qureshi, "Evolutionary computation technique for solving Riccati differential equation of arbitrary order," *World Academy of Science*, vol.58, no.58, pp.531–536, 2009.
[19] M. A. Z. Raja, "A new stochastic approach for solution of differential Riccati equation of fractional order," *Annals of Mathematics & Artificial Intelligence*, vol.60, no.3–4, pp. 229–250, 2010.
[20] M. A. Z. Raja, M. A. Manzar, R. Sama, "An efficient computational intelligence approach for solving fractional order Riccati equations using ANN and SQP," *Applied Mathematical Modelling*, vol.39, no.10–11, pp.3075–3093, 2015.
[21] X. S. Yang, S. Deb , "Engineering optimization by cuckoo search," *International Journal of Mathematical Modelling and Numerical Optimisation*, vol.1, no.4, pp.330–343, 2010.
[22] X. S. Yang, "Cuckoo search for inverse problems and simulated-driven shape optimization," *Journal of Computational Methods in Sciences and Engineering*, vol.12, no.1–2, pp. 129–137, 2012.
[23] V. Bhargava , S. E. K. Fateen, A. Bonillapetriciolet , "Cuckoo Search: A new nature-inspired optimization method for phase equilibrium calculations," *Fluid Phase Equilibria*, vol.337, no.337, pp. 191–200, 2013.
[24] A. H. Gandomi, X. S. Yang, A. H. Alavi , "Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems," *Engineering With Computers*, vol.29, no.1, pp.17–35, 2013.
[25] X. M. Zhang, "Parameter estimation of shallow wave equation via cuckoo search," *Neural Computing & Applications*, vol.28, no.12, pp.4047–4059, 2017.
[26] X. M. Zhang, Q. Wan, Y. H. Fan, "Applying modified cuckoo search algorithm for solving systems of nonlinear equations," *Neural Computing and Applications*, vol.31, no.2, pp.553–576, 2019.
[27] G. Chen, S. Qiu, Z. Zhang, Z. Sun, "Quasi-oppositional Cuckoo Search Algorithm for Multi-objective Optimal Power Flow," *IAENG International Journal of Computer Science*, vol.45, no. 2, pp. 255–266, 2018.
[28] S. A. Medjahed, T. A. Saadi, A. Benyettou, M. Ouali, "Binary cuckoo search algorithm for band selection in hyperspectral image classification," *IAENG International Journal of Computer Science*, vol. 42, no.3, pp.183–191, 2015.
[29] X. Li, "Numerical solution of fractional differential equations using cubic B-spline wavelet collocation method," *Communications in Nonlinear Science & Numerical Simulation*, vol.17, no.10, pp. 3934–3946, 2012.
[30] D. Kai, "An algorithm for the numerical solution of differential equations of fractional order," *Electron. Trans. Numer. Anal*, vol.5, no.1, pp.1–6, 1997.
[31] X. Cai, J. H. Chen, "Numerical method for the fractional order ordinary differential equation," *Journal of Jimei Uiversity (Natural Science)*, vol.12, no.4, pp.367–370, 2007.