Contents lists available at ScienceDirect

Computers and Mathematics with Applications

journal homepage: www.elsevier.com/locate/camwa

Numerical algorithm based on Adomian decomposition for fractional differential equations[☆]

Changpin Li[a,*], Yihong Wang[b]

[a] Department of Mathematics, Shanghai University, Shanghai 200444, China

[b] Department of Applied Mathematics, Zhejiang Forestry University, Hangzhou 311300, China

ARTICLE INFO

Article history:
Received 22 July 2008
Received in revised form 15 February 2009
Accepted 3 March 2009

Dedicated to Professor Qishao Lu on the occasion of his 70th birthday

Keywords:
Predictor–Corrector method
Adomian method
Caputo derivative
Fractional Lorenz system
Time-fractional Burgers equation

ABSTRACT

In this paper, a novel algorithm based on Adomian decomposition for fractional differential equations is proposed. Comparing the present method with the fractional Adams method, we use this derived computational method to find a smaller "efficient dimension" such that the fractional Lorenz equation is chaotic. We also apply this new method to the time-fractional Burgers equation with initial and boundary value conditions. Numerical results and computer graphics show that the constructed numerical is efficient.

© 2009 Elsevier Ltd. All rights reserved.

1. Introduction

Factional differential equations are increasingly used to model problems in acoustics and thermal systems, rheology and mechanical systems, signal processing and systems identification, control and robotics, and other areas of applications, for example, see [1,2] and many references cited therein. These applications in interdisciplinary sciences show the importance and necessity of fractional calculus. All this motivates us to construct a variety of efficient numerical methods for fractional differential equations.

Generally speaking, three kinds of definitions of fractional derivatives are widely used, i.e., Grünwald–Letnikov derivative, Riemann–Liouville derivative and Caputo derivative [3–6]. These three definitions are in general non-equivalent. However, the first two definitions defined in the sufficiently smooth spaces are equivalent. So, numerical analysts often use the definition of Grünwald–Letnikov derivative to discretize the fractional differential equations with Riemann–Liouville derivative. The last two definitions are equivalent under their homogenous initial value conditions. If its initial values are non-zero, the Riemann–Liouville derivative definition is used only by pure mathematicians but seldom utilized by applied scientists and engineers since the initial value conditions are not easily measured due to vagueness of physical meanings and geometric interpretation. The alternative definition of the fractional derivative given by Caputo has the advantage of only requiring initial conditions given in terms of integer-order derivatives. In particular, when the Caputo derivative is chosen, it allows us to specify inhomogeneous initial conditions for fractional differential equations with Caputo derivative if it is desired. These initial conditions of integer-order derivatives represent well-understood features of a physical situation.

[☆] The present work was supported in part by National Natural Science Foundation of China under Grant No. 10872119, Shanghai Leading Academic Discipline Project under Grant No. J50101.

[*] Corresponding author.
E-mail address: leecp@online.sh.cn (C. Li).

0898-1221/$ – see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.camwa.2009.03.079

In this paper we would rather use the Caputo derivative, precisely because of its applicability to real world models. The definition of Riemann–Liouville fractional integral of a function y reads as

$$J_{0,t}^{\alpha}y(t)=\frac{1}{\Gamma(\alpha)}\int_0^t(t-\tau)^{\alpha-1}y(\tau)\mathrm{d}\tau,\quad t>0,n-1<\alpha<n\in Z^+,$$

and the Caputo fractional derivative of y is defined as

$${}_CD_{0,t}^{\alpha}y(t)=J_{0,t}^{n-\alpha}y^{(n)}(t)=\frac{1}{\Gamma(n-\alpha)}\int_0^t(t-\tau)^{n-\alpha-1}y^{(n)}(\tau)\mathrm{d}\tau,\quad t>0,n-1<\alpha<n\in Z^+.$$

The nonlinear fractional differential equation with Caputo derivative is in the following form

$${}_CD_{0,t}^{\alpha}x(t)=f(t,x(t)),\qquad x^{(k)}(0)=x_0^{(k)},\quad n-1<\alpha<n\in Z^+,k=0,1,\ldots,\lceil\alpha\rceil-1,\tag{1}$$

where $\lceil\alpha\rceil=n$. It is well known that the initial value problem (1) is equivalent to a Volterra integral equation [7],

$$x(t)=\sum_{k=0}^{\lceil\alpha\rceil-1}x_0^{(k)}\frac{t^k}{k!}+\frac{1}{\Gamma(\alpha)}\int_0^t(t-\tau)^{\alpha-1}f(\tau,x(\tau))\mathrm{d}\tau\tag{2}$$

in the sense that a continuous function solves (2) if and only if it solves (1).

In [8], Diethelm, et al., successfully constructed a Predictor–Corrector method for a fractional differential equation with Caputo derivative, called "fractional Adams method" for brevity. And the error analysis for this method was given in [9]. Their fractional Adams method is introduced below.

Firstly the product trapezoidal quadrature formula is applied to replacing the integrals of (2). Set $h=T/N$, $t_n=nh$, $n=0,1,\ldots,N\in Z^+$. Then (2) can be discretized as follows,

$$x(t_{n+1})=\sum_{k=0}^{\lceil\alpha\rceil-1}x_0^{(k)}\frac{t^k}{k!}+\frac{h^{\alpha}}{\Gamma(\alpha+2)}\sum_{j=0}^{n+1}a_{j,n+1}f(t_j,x(t_j)),\tag{3}$$

where

$$a_{j,n+1}=\begin{cases}n^{\alpha+1}-(n-\alpha)(n+1)^{\alpha}, & j=0,\\ (n-j+2)^{\alpha+1}+(n-j)^{\alpha+1}-2(n-j+1)^{\alpha+1}, & 1\le j\le n,\\ 1, & j=n+1,\end{cases}$$

Eq. (3) can be rewritten as

$$x(t_{n+1})=\sum_{k=0}^{\lceil\alpha\rceil-1}x_0^{(k)}\frac{t^k}{k!}+\frac{h^{\alpha}}{\Gamma(\alpha+2)}f(t_{n+1},x(t_{n+1}))+\frac{h^{\alpha}}{\Gamma(\alpha+2)}\sum_{j=0}^{n}a_{j,n+1}f(t_j,x(t_j)).\tag{4}$$

The right hand side of system (4) contains term $x(t_{n+1})$ in nonlinear function $f(t,x(t))$ so it is an implicit scheme. In order to start the Adams–Moulton iterative method, the solution is accomplished by first "Predicting" ($x^P(t_{n+1})$) by using the explicit Adams–Bashforth formula, and then "Correcting" ($x(t_{n+1})$).

Now the Predictor–Corrector algorithm for (2) can be described as

$$\begin{cases}x(t_{n+1})=\displaystyle\sum_{k=0}^{\lceil\alpha\rceil-1}x_0^{(k)}\frac{t^k}{k!}+\frac{h^{\alpha}}{\Gamma(\alpha+2)}f(t_{n+1},x^P(t_{n+1}))+\frac{h^{\alpha}}{\Gamma(\alpha+2)}\sum_{j=0}^{n}a_{j,n+1}f(t_j,x(t_j)),\\ x^P(t_{n+1})=\displaystyle\sum_{k=0}^{\lceil\alpha\rceil-1}x_0^{(k)}\frac{t^k}{k!}+\frac{1}{\Gamma(\alpha)}\sum_{j=0}^{n}b_{j,n+1}f(t_j,x(t_j)),\end{cases}$$

where $b_{j,n+1}=\frac{h^{\alpha}}{\alpha}((n+1-j)^{\alpha}-(n-j)^{\alpha}),j=0,1,\ldots,n.$

The error estimate of this fractional Adams method for the fractional differential system is given below.

Lemma 1 (*[9]*). *If ${}_CD_{0,t}^{\alpha}x(t)\in C^2[0,T]$, then the truncated error estimate is*

$$\max_{j=0,1,\ldots,N}|x(t_j)-x_n(t_j)|=\begin{cases}O(h^2), & \text{if } \alpha\ge 1,\\ O(h^{1+\alpha}), & \text{if } 0<\alpha<1.\end{cases}$$

If we do not use the predictor, a somewhat better method is chosen – the Newton iteration method – to numerically solve $x(t_{n+1})$ for Eq. (4). This requires that $f(t,x(t))$ be smooth and that the inverse of the derivative operator f_x exists. For a system of equations, Newton method often needs a lot of time so it is not economical. Luckily, Adomian decomposition method can be used to solve this problem effectively.

We proceed as follows. In Section 2 we first recall the Adomian decomposition method, and then extend the method to solve the fractional differential equations. We apply the derived algorithm to finding a smaller "efficient dimension" of the fractional Lorenz system in Section 3. In Section 4 we use the constructed algorithm for solving the time-fractional Burgers equation.

2. The numerical algorithm based on Adomian decomposition

At the beginning of the 1980's, Adomian proposed a new method for solving nonlinear functional equations of various kinds. The technique is actually a decomposition of the nonlinear operators as a series of functions. Each term of this series is a polynomial called Adomian's polynomial. Let us first recall the basic idea of the decomposition method [10,11]. Consider the general nonlinear equation

$$u = N(u) + f,$$

where N is a nonlinear operator, and where f is supposed to be known. The decomposition method consists in looking for a solution having the series form

$$u = \sum_{i=0}^{\infty} u_i.$$

The nonlinear operator N is decomposed as

$$N(u) = \sum_{n=0}^{\infty} A_n,$$

where A_n's are called Adomian's polynomials. In the first approach given by Adomian [10], A_n's are obtained from the following equalities

$$v = \sum_{i=0}^{\infty} \lambda^i u_i,$$

$$N(v) = N\left(\sum_{i=0}^{\infty} \lambda^i u_i\right) = \sum_{n=0}^{\infty} \lambda^n A_n.$$

We remark that A_n's are formally obtained from the relationship

$$A_n = \frac{1}{n!}\frac{\mathrm{d}^n}{\mathrm{d}\lambda^n}\left[N\left(\sum_{i=0}^{\infty} \lambda^i u_i\right)\right]_{\lambda=0}.$$

The above process leads to the equality

$$\sum_{i=0}^{\infty} u_i = \sum_{n=0}^{\infty} A_n + f,$$

and the Adomian method consists in identifying u_i by means of the formulae below

$$\begin{aligned}
u_0 &= f,\\
u_1 &= A_0(u_0),\\
u_2 &= A_1(u_0, u_1),\\
&\vdots\\
u_n &= A_{n-1}(u_0, u_1 \cdots u_{n-1}),\\
&\vdots
\end{aligned}$$

In practice, Adomian decomposition method gives very good results even if one takes a truncated series with a small number of terms. The reason for such a result is due to the analogy of the Adomian series with the Taylor series.

Assume the following conditions hold.

(i) The solution u can be written as a series of functions u_i, i.e., $u \approx \sum_{i=0}^{\infty} u_i$. Furthermore, this series is absolutely convergent, i.e., $\sum_{i=0}^{\infty} |u_i| < +\infty$.

(ii) The nonlinear function $N(u)$ can be developed in entire series with a convergence radius equal to infinity. In other words, we may write

$$N(u) = \sum_{n=0}^{\infty} N^{(n)}(0)\frac{u^n}{n!}, \quad |u| < \infty.$$

We see that

$$\sum_{i=1}^{\infty} u_i = \sum_{n=0}^{\infty} A_n = \sum_{n=0}^{\infty} \frac{N^{(n)}(0)}{n!}\left(\sum_{i=0}^{\infty} u_i\right)^n.$$

If we consider a truncated series $\sum_{i=1}^{R} u_i$ for approximating u, we can calculate the error as follows:

$$\left|\sum_{i=0}^{\infty} u_i - \sum_{i=0}^{R} u_i\right| = \left|\sum_{i=R+1}^{\infty} u_i\right| = \left|\sum_{n=R}^{\infty} A_n\right| \le \sum_{n=R}^{\infty} \frac{|N^{(n)}(0)|}{n!} U^n,$$

where $U = \sum_{i=0}^{\infty} |u_i|$.

Lemma 2 ([12]). *Suppose that $\|U\| \le M$, and $\|N^{(n)}(0)\| \le S$ independent of n, $\sum_{i=1}^{R} u_i$ is an approximate solution of equation $u = N(u) + f$. Then the error is $S \sum_{n=R}^{\infty} \frac{M^n}{n!}$.*

Applying Adomian's idea, system (4) can be rewritten as,

$$x(t_{n+1}) = \frac{h^\alpha}{\Gamma(\alpha+2)}(Lx(t_{n+1}) + Nx(t_{n+1})) + \sum_{k=0}^{\lceil\alpha\rceil-1} x_0^{(k)} \frac{t^k}{k!} + \frac{h^\alpha}{\Gamma(\alpha+2)} \sum_{j=0}^{n} a_{j,n+1} f(t_j, x(t_j)), \quad (5)$$

in which $Lx(t_{n+1})$ is the linear terms of $f(t_{n+1}, x(t_{n+1}))$, and $Nx(t_{n+1})$ represents the nonlinear terms of $f(t_{n+1}, x(t_{n+1}))$. Now, $x(t_{n+1})$ can be presented as a series

$$x(t_{n+1}) = \sum_{r=0}^{\infty} x_r(t_{n+1}), \quad (6)$$

with

$$x_0(t_{n+1}) = \sum_{k=0}^{\lceil\alpha\rceil-1} x_0^{(k)} \frac{t^k}{k!} + \frac{h^\alpha}{\Gamma(\alpha+2)} \sum_{j=0}^{n} a_{j,n+1} f(t_j, x(t_j)), \quad (7)$$

and $x_r(t_{n+1})$ is to be determined.

The nonlinear term $Nx(t_{n+1})$ is then decomposed to a series of Adomian polynomials. Substituting (6) and (7) into (5), we obtain

$$\sum_{r=0}^{\infty} x_r(t_{n+1}) = x_0(t_{n+1}) + \frac{h^\alpha}{\Gamma(\alpha+2)} \left(L\left(\sum_{r=0}^{\infty} x_r(t_{n+1})\right) + N\left(\sum_{r=0}^{\infty} x_r(t_{n+1})\right)\right).$$

Consequently, we have

$$\begin{cases} x_0(t_{n+1}) = \sum_{k=0}^{\lceil\alpha\rceil-1} x_0^{(k)} \frac{t^k}{k!} + \frac{h^\alpha}{\Gamma(\alpha+2)} \sum_{j=0}^{n} a_{j,n+1} f(t_j, x(t_j)), \\ x_1(t_{n+1}) = \frac{h^\alpha}{\Gamma(\alpha+2)}(Lx_0(t_{n+1}) + A_0), \\ x_2(t_{n+1}) = \frac{h^\alpha}{\Gamma(\alpha+2)}(Lx_1(t_{n+1}) + A_1), \\ \vdots \\ x_{r+1}(t_{n+1}) = \frac{h^\alpha}{\Gamma(\alpha+2)}(Lx_r(t_{n+1}) + A_r), \\ \vdots \end{cases} \quad (8)$$

where

$$\begin{aligned} A_0 &= N(x_0(t_{n+1})), \\ A_r &= \frac{1}{r!} \frac{\mathrm{d}^r}{\mathrm{d}\lambda^r} \left[N\left(\sum_{i=0}^{\infty} \lambda^i x_i(t_{n+1})\right)\right]_{\lambda=0} \\ &= \sum_{p_1+2p_2+\cdots rp_r = r} \frac{(x_1(t_{n+1}))^{p_1}}{p_1!} \frac{(x_2(t_{n+1}))^{p_2}}{p_2!} \cdots \frac{(x_r(t_{n+1}))^{p_r}}{p_r!} N^{(p_1+p_2+\cdots p_r)}(x_0(t_{n+1})), \quad r = 1, 2, \ldots. \end{aligned}$$

All these $x(t_{n+1})$'s are calculable. Since the series rapidly converges, in practice we use a finite sum $\widetilde{x}(t_{n+1}) = \sum_{r=0}^{R} x_r(t_{n+1})$ to approximate the solution of $x(t_{n+1})$.

According to computational scheme (4)–(8), the errors come from two aspects, one comes from applying the product trapezoidal quadrature formulae to replacing the integrals of (2), see [9], the other one comes from using $\widetilde{x}(t_{n+1}) = \sum_{r=0}^{R} x_r(t_{n+1})$ to approximate the real solution $x(t_{n+1})$ (see Lemma 2).

The above numerical algorithm can be naturally generalized to a system of equations in the following form.

$$\begin{cases} {}_CD_{0,t}^{\alpha_1}x_1(t) = f_1(t, x(t)), & x_1^{(k_1)}(0) = x_{10}^{(k_1)},\ \alpha_1 > 0, k_1 = 0, 1, \ldots, \lceil\alpha_1\rceil - 1, \\ {}_CD_{0,t}^{\alpha_2}x_2(t) = f_2(t, x(t)), & x_2^{(k_2)}(0) = x_{20}^{(k_2)},\ \alpha_2 > 0, k_2 = 0, 1, \ldots, \lceil\alpha_2\rceil - 1, \\ \vdots \\ {}_CD_{0,t}^{\alpha_s}x_s(t) = f_s(t, x(t)), & x_s^{(k_s)}(0) = x_{s0}^{(k_s)},\ \alpha_s > 0, k_s = 0, 1, \ldots, \lceil\alpha_s\rceil - 1, \end{cases} \tag{9}$$

where $x(t) = (x_1(t), x_2(t), \ldots, x_s(t)) \in \Re^n$.

Firstly, applying the trapezoidal quadrature formula to (9) leads to

$$\begin{cases} x_1(t_{n+1}) = \dfrac{h^{\alpha_1}}{\Gamma(\alpha_1+2)}(f_1(t_{n+1}, x(t_{n+1}))) + \displaystyle\sum_{k_1=0}^{\lceil\alpha_1\rceil-1} x_0^{(k_1)}\frac{t^{k_1}}{k_1!} + \frac{h^{\alpha_1}}{\Gamma(\alpha_1+2)}\sum_{j=0}^{n} a_{1,j,n+1}f_1(t_j, x(t_j)), \\ x_2(t_{n+1}) = \dfrac{h^{\alpha_2}}{\Gamma(\alpha_2+2)}(f_2(t_{n+1}, x(t_{n+1}))) + \displaystyle\sum_{k_2=0}^{\lceil\alpha_2\rceil-1} x_0^{(k_2)}\frac{t^{k_2}}{k_2!} + \frac{h^{\alpha_2}}{\Gamma(\alpha_2+2)}\sum_{j=0}^{n} a_{2,j,n+1}f_2(t_j, x(t_j)), \\ \vdots \\ x_s(t_{n+1}) = \dfrac{h^{\alpha_s}}{\Gamma(\alpha_s+2)}(f_s(t_{n+1}, x_s(t_{n+1}))) + \displaystyle\sum_{k_s=0}^{\lceil\alpha_s\rceil-1} x_0^{(k_s)}\frac{t^{k_s}}{k_s!} + \frac{h^{\alpha_s}}{\Gamma(\alpha_s+2)}\sum_{j=0}^{n} a_{s,j,n+1}f_s(t_j, x(t_j)) \end{cases}$$

in which

$$a_{i,j,n+1} = \begin{cases} n^{\alpha_i+1} - (n-\alpha_i)(n+1)^{\alpha_i}, & & j = 0, \\ (n-j+2)^{\alpha_i+1} + (n-j)^{\alpha_i+1} - 2(n-j+1)^{\alpha_i+1}, & i = 1, 2, \ldots, s, & 1 \le j \le n. \end{cases}$$

Secondly, we use the Adomian decomposition method,

$$\begin{cases} x_1(t_{n+1}) = \dfrac{h^{\alpha_1}}{\Gamma(\alpha_1+2)}(L_1\,x(t_{n+1}) + N_1\,x(t_{n+1})) + x_{1,0}(t_{n+1}), \\ x_2(t_{n+1}) = \dfrac{h^{\alpha_2}}{\Gamma(\alpha_2+2)}(L_2\,x_2(t_{n+1}) + N_2\,x(t_{n+1})) + x_{2,0}(t_{n+1}), \\ \vdots \\ x_s(t_{n+1}) = \dfrac{h^{\alpha_s}}{\Gamma(\alpha_s+2)}(L_s\,x_s(t_{n+1}) + N_s\,x(t_{n+1})) + x_{s,0}(t_{n+1}) \end{cases}$$

where $L_i\,x(t_{n+1})$ are the linear terms of $f_i(t_{n+1}, x(t_{n+1}))$, and $N_i\,x(t_{n+1})$ represents the nonlinear terms of $f_i(t_{n+1}, x(t_{n+1}))$. According to (5)–(8), $x_1(t_{n+1}), x_2(t_{n+1}), \ldots, x_s(t_{n+1})$ can be presented as $x_1(t_{n+1}) = \sum_{r=0}^{\infty} x_{1,r}(t_{n+1}), x_2(t_{n+1}) = \sum_{r=0}^{\infty} x_{2,r}(t_{n+1}), \ldots, x_s(t_{n+1}) = \sum_{r=0}^{\infty} x_{s,r}(t_{n+1})$. Here $x_{i,j}(t_{n+1}), i = 1, 2, \ldots, s, j = 1, 2, \ldots$, can be calculated as

$$\begin{cases} x_{1,0}(t_{n+1}) = \displaystyle\sum_{k_1=0}^{\lceil\alpha_1\rceil-1} x_0^{k_1}\frac{t^{k_1}}{k_1!} + \frac{h^{\alpha_1}}{\Gamma(\alpha_1+2)}\sum_{j=0}^{n} a_{1,j,n+1}f_1(t_j, x(t_j)), \\ x_{2,0}(t_{n+1}) = \displaystyle\sum_{k_2=0}^{\lceil\alpha_2\rceil-1} x_0^{(k_2)}\frac{t^{k_2}}{k_2!} + \frac{h^{\alpha_2}}{\Gamma(\alpha_2+2)}\sum_{j=0}^{n} a_{2,j,n+1}f_2(t_j, x(t_j)), \\ \vdots \\ x_{s,0}(t_{n+1}) = \displaystyle\sum_{k_s=0}^{\lceil\alpha_s\rceil-1} x_0^{(k_s)}\frac{t^{k_s}}{k_s!} + \frac{h^{\alpha_s}}{\Gamma(\alpha_s+2)}\sum_{j=0}^{n} a_{s,j,n+1}f_s(t_j, x(t_j)); \end{cases}$$

$$\begin{cases} x_{1,1}(t_{n+1}) = \dfrac{h^{\alpha_1}}{\Gamma(\alpha_1+2)}(L_1\,x_0(t_{n+1}) + A_{1,0}(x_{1,0}(t_{n+1}), x_{2,0}(t_{n+1}), \ldots, x_{s,0}(t_{n+1}))), \\ x_{2,1}(t_{n+1}) = \dfrac{h^{\alpha_1}}{\Gamma(\alpha_2+2)}(L_2\,x_0(t_{n+1}) + A_{2,0}(x_{1,0}(t_{n+1}), x_{2,0}(t_{n+1}), \ldots, x_{s,0}(t_{n+1}))), \\ \vdots \\ x_{s,1}(t_{n+1}) = \dfrac{h^{\alpha_s}}{\Gamma(\alpha_s+2)}(L_s x_0(t_{n+1}) + A_{s,0}(x_{1,0}(t_{n+1}), x_{2,0}(t_{n+1}), \ldots, x_{s,0}(t_{n+1}))); \end{cases}$$

$$\begin{cases} x_{1,2}(t_{n+1}) = \dfrac{h^{\alpha_1}}{\Gamma(\alpha_1+2)}(L_1 x_1(t_{n+1}) + A_{1,1}(x_{1,0}(t_{n+1}), x_{1,1}(t_{n+1}); x_{2,0}(t_{n+1}), x_{2,1}(t_{n+1}); \cdots; x_{s,0}(t_{n+1}), x_{s,1}(t_{n+1}))), \\ x_{2,2}(t_{n+1}) = \dfrac{h^{\alpha_1}}{\Gamma(\alpha_2+2)}(L_2 x_1(t_{n+1}) + A_{2,1}(x_{1,0}(t_{n+1}), x_{1,1}(t_{n+1}); x_{2,0}(t_{n+1}), x_{2,1}(t_{n+1}); \cdots; x_{s,0}(t_{n+1}), x_{s,1}(t_{n+1}))), \\ \vdots \\ x_{s,2}(t_{n+1}) = \dfrac{h^{\alpha_s}}{\Gamma(\alpha_s+2)}(L_s x_1(t_{n+1}) + A_{s,1}(x_{1,0}(t_{n+1}), x_{1,1}(t_{n+1}); x_{2,0}(t_{n+1}), x_{2,1}(t_{n+1}); \cdots; x_{s,0}(t_{n+1}), x_{s,1}(t_{n+1}))); \end{cases}$$

$\cdots\cdots$

$$\begin{cases} x_{1,r+1}(t_{n+1}) = \dfrac{h^{\alpha_1}}{\Gamma(\alpha_1+2)}(L_1 x_r(t_{n+1}) + A_{1,r}(x_{1,0}(t_{n+1}), x_{1,1}(t_{n+1}), \ldots, x_{1,r}(t_{n+1}); \\ \qquad x_{2,0}(t_{n+1}), x_{2,1}(t_{n+1}), \ldots, x_{2,r}(t_{n+1}); \cdots; x_{s,0}(t_{n+1}), x_{s,1}(t_{n+1}) \ \ldots, x_{s,r}(t_{n+1}))), \\ x_{2,r+1}(t_{n+1}) = \dfrac{h^{\alpha_1}}{\Gamma(\alpha_2+2)}(L_2 x_r(t_{n+1}) + A_{2,r}(x_{1,0}(t_{n+1}), x_{1,1}(t_{n+1}), \ldots, x_{1,r}(t_{n+1}); \\ \qquad x_{2,0}(t_{n+1}), x_{2,1}(t_{n+1}), \ldots, x_{2,r}(t_{n+1}); \cdots; x_{s,0}(t_{n+1}), x_{s,1}(t_{n+1}) \ \ldots, x_{s,r}(t_{n+1}))), \\ \vdots \\ x_{s,r+1}(t_{n+1}) = \dfrac{h^{\alpha_s}}{\Gamma(\alpha_s+2)}(L_s x_r(t_{n+1}) + A_{s,r}(x_{1,0}(t_{n+1}), x_{1,1}(t_{n+1}), \ldots, x_{1,r}(t_{n+1}); \\ \qquad x_{2,0}(t_{n+1}), x_{2,1}(t_{n+1}), \ldots, x_{2,r}(t_{n+1}); \cdots; x_{s,0}(t_{n+1}), x_{s,1}(t_{n+1}) \ \ldots, x_{s,r}(t_{n+1}))); \end{cases}$$

$\cdots\cdots$

where

$$\begin{aligned} & L_i x_0(t_{n+1}) = L_i(x_{1,0}(t_{n+1}), x_{2,0}(t_{n+1}), \ldots, x_{s,0}(t_{n+1})), \\ & L_i x_r(t_{n+1}) = L_i(x_{1,r}(t_{n+1}), x_{2,r}(t_{n+1}), \ldots, x_{s,r}(t_{n+1})), \\ & A_{i,0}(x_{1,0}(t_{n+1}), x_{2,0}(t_{n+1}), \ldots, x_{s,0}(t_{n+1})) = N_i(x_{1,0}(t_{n+1}), x_{2,0}(t_{n+1}), \ldots, x_{s,0}(t_{n+1})), \\ & A_{i,r}(x_{1,0}(t_{n+1}), x_{1,1}(t_{n+1}), \ldots, x_{1,r}(t_{n+1}); x_{2,0}(t_{n+1}), x_{2,1}(t_{n+1}), \ldots, x_{2,r}(t_{n+1}); \cdots; \\ & \quad x_{s,0}(t_{n+1}), x_{s,1}(t_{n+1}) \ \ldots, x_{s,r}(t_{n+1})) \\ & \quad = \sum_{\bigwedge_r} \frac{(x_{1,1}(t_{n+1}))^{k_{1,1}}}{k_{1,1}!} \cdots \frac{(x_{1,r}(t_{n+1}))^{k_{1,r}}}{k_{1,r}!} \cdots \frac{(x_{s,1}(t_{n+1}))^{k_{s,1}}}{k_{s,1}!} \cdots \frac{(x_{s,r}(t_{n+1}))^{k_{s,r}}}{k_{s,r}!} \\ & \quad \times \frac{\partial^{(k_{1,1}+\cdots+k_{1,r})+\cdots+(k_{s,1}+\cdots+k_{s,,r})}}{\partial x_1^{k_{1,1}+\cdots+k_{1,r}} \cdots \partial x_s^{k_{s,1}+\cdots+k_{s,r}}} N_i(x_{1,0}(t_{n+1}), x_{2,0}(t_{n+1}), \ldots, x_{s,0}(t_{n+1})), \quad i = 1, \ldots, s, \\ & \bigwedge_r = (k_{1,1} + 2k_{1,2} + \cdots + rk_{1,r}) + \cdots + (k_{s,1} + 2k_{s,2} + \cdots + rk_{s,r}) = r, \quad r = 1, 2, \ldots. \end{aligned}$$

Finally, we use $\widetilde{x}_1(t_{n+1}) = \sum_{r=0}^{R} x_{1,r}(t_{n+1}), \widetilde{x}_2(t_{n+1}) = \sum_{r=0}^{R} x_{2,r}(t_{n+1}), \ldots, \widetilde{x}_s(t_{n+1}) = \sum_{r=0}^{R} x_{s,r}(t_{n+1})$, to approximate $x_1(t_{n+1}), x_2(t_{n+1}), \ldots, x_s(t_{n+1})$.

3. Numerical tracking the smallest "efficient dimension" of the fractional Lorenz system

The definition of "efficient dimensions" was defined for a fractional dynamical system [13–15], where the efficient dimension characterizes the damping property for the stable limit set of a fractional differential system of equations [16].

In this section, we study the fractional Lorenz system in the following form

$$\begin{cases} {}_C D_{0,t}^{p_1} x(t) = (25\gamma + 10)(y - x), \\ {}_C D_{0,t}^{p_2} y(t) = (28 - 35\gamma)x - xz + (29\gamma - 1)y, \\ {}_C D_{0,t}^{p_3} x_3(t) = xy - \dfrac{\gamma + 8}{3} z, \end{cases} \tag{10}$$

where $\gamma \in [0, 0.8)$, and its efficient dimension $q = p_1 + p_2 + p_3$. If $p_1 = p_2 = p_3 = 1$ and $\gamma = 0$, then it is the classical Lorenz system. Here we use our algorithm to numerically detect the smallest efficient dimension q such that system (10) is chaotic.

The computational procedure is given below. Throughout the section, we always let $\gamma = 0.78$ for comparison.

1^0 First, fix $p_2 = p_3 = 1$, let p_1 decrease with step-length $\Delta s = 0.1$. After k steps, $p_1(k) = 1 - 0.1k$. If system (10) has no chaotic attractor for $p_1(k)$ and $p_2 = p_3 = 1$ but has a chaotic attractor for $p_1(k-1) = 1 - 0.1(k-1)$ and the same p_2, p_3 values. Next we decrease $p_1(k-1)$ with a new step-length $\Delta' s = 0.01$. After k' steps, p_1 changes into $p_1(k-1) - 0.01k'$. If system (10) has no chaotic attractor for $p_1(k-1) - 0.01k'$ and $p_2 = p_3 = 1$ but has one for $p_1(k-1) - 0.01(k'-1)$ and the same

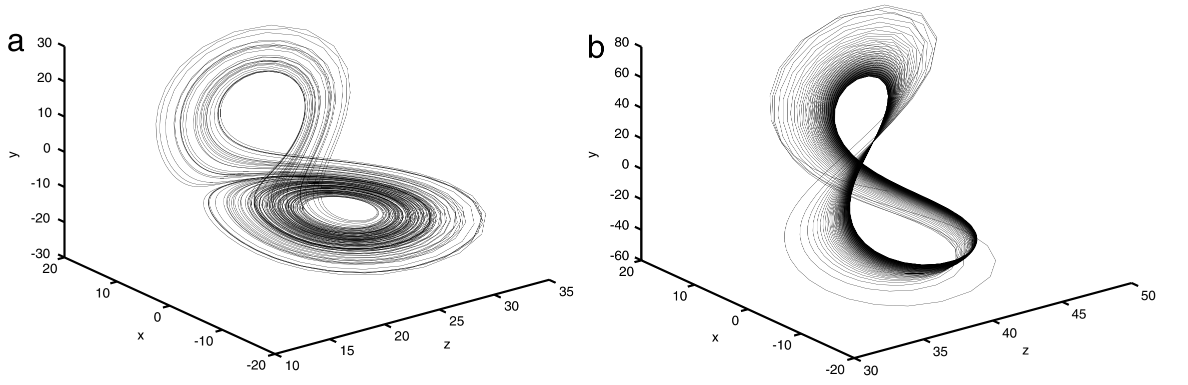

Fig. 1. Chaotic attractors of the fractional Lorenz system. Initial value $(x_{10}, x_{20}, x_{30}) = (-15.5, -17.48, 35.64)$, $\gamma = 0.78$, $R = 8$ (the meaning of R can be found from the computational scheme for system (9)). (a) $(p_1, p_2, p_3) = (0.8, 0.7, 0.7)$, $N = 5200$, time step-length $h = 0.004$; (b) $(p_1, p_2, p_3) = (0.45, 0.27, 0.35)$, $N = 5200$, time step-length $h = 0.002$, the first 100 points are removed.

p_2, p_3 values. We stop here and think in this situation the smaller efficient dimension $|q| = p_2+p_3+p_1(k-1)-0.01(k'-1)$ such that system (10) has a chaotic attractor.

2^0 Fix $p_1 = p_1(k-1) - 0.01(k'-1)$, let p_2, p_3 decrease as above.

It is surprisingly found that chaos is generated for $\gamma = 0.78$ when $p_1 = 0.45, p_2 = 0.27, p_3 = 0.35$. The smallest efficient dimension is $|q| = p_1 + p_2 + p_3 = 1.07$. For the same parameter γ value and the same tracking procedure, we use the fractional Adams method but find that the smallest efficient dimension is $q = p_1 + p_2 + p_3 = 1.32$ [17] such that system (10) is chaotic. See Fig. 1. Comparing between the two methods, our algorithm spends less time and finds a smaller efficient dimension.

4. Numerical algorithm for the time-fractional Burgers equation

In the present section, we adopt the derived algorithm for nonlinear fractional partial differential equation. Here we numerically study the time-fractional Burgers equation of the form below

$$\begin{cases} {}_CD_{0,t}^{\alpha}u + u\dfrac{\partial u}{\partial x} = \dfrac{1}{Re}\dfrac{\partial^2 u}{\partial x^2}, \quad 0 < x < 1, t > 0, \\ \begin{cases} u(x,0) = \sin(\pi x), \quad 0 \le x \le 1, \text{if } 0 < \alpha \le 1, \\ u(x,0) = \sin(\pi x), \qquad u_t(x,0) = 0, \quad 0 \le x \le 1, \text{if } 1 < \alpha \le 2, \end{cases} \\ u(0,t) = u(1,t) = 0, \quad t > 0, \end{cases} \tag{11}$$

where the constant Re is the Reynolds number.

Now we set $\Delta x = h, h = 1/N, x_i = ih, i = 1, 2, \ldots, N-1$, then

$$\begin{cases} \left.\dfrac{\partial u}{\partial x}\right|_{x=x_i} = \dfrac{u_{i+1} - u_{i-1}}{2h}, \\ \left.\dfrac{\partial^2 u}{\partial x^2}\right|_{x=x_i} = \dfrac{u_{i+1} - 2u_i + u_{i-1}}{h^2}. \end{cases} \tag{12}$$

According to (11) and (12), the partial differential equation (11) can be decomposed into an ordinary differential system of equations with time-fractional derivative.

$${}_CD_{0,t}^{\alpha}u_i + u_i\frac{u_{i+1} - u_{i-1}}{2h} = \frac{1}{Re}\frac{u_{i+1} - 2u_i + u_{i-1}}{h^2}, \tag{13}$$

where $u_0 = u_N = 0$ and

$$\begin{cases} u(x,0)|_{x=x_i} = \sin(\pi x_i), \quad 0 \le x \le 1, \text{if } 0 < \alpha \le 1, \\ u(x,0)|_{x=x_i} = \sin(\pi x_i), \qquad u_t(x,0)|_{x=x_i} = 0, \quad 0 \le x \le 1, \text{if } 1 < \alpha \le 2. \end{cases}$$

Suppose $U = (u_1, u_2, \ldots u_{N-1})^{\mathrm{T}}$, then (11) reads as a compact form,

$${}_CD_{0,t}^{\alpha}U = B(t, U)U, \tag{14}$$

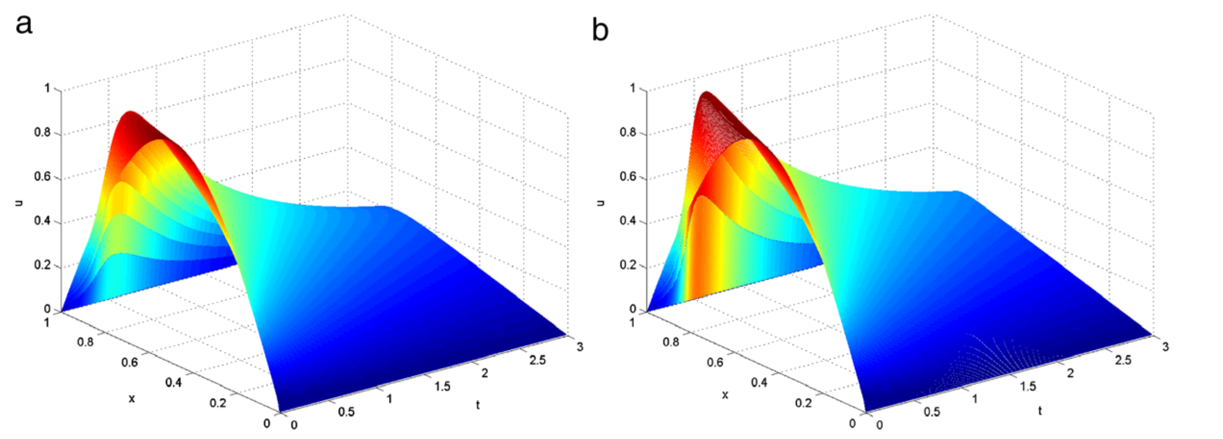

Fig. 2. Figures of the classical Burgers equation ($\alpha = 1$). The space step-length is $h = 0.01$. (a) $(\alpha, Re, \omega) = (1.0, 50, 0.0025)$. (b) $(\alpha, Re, \omega) = (1.0, 100, 0.005)$.

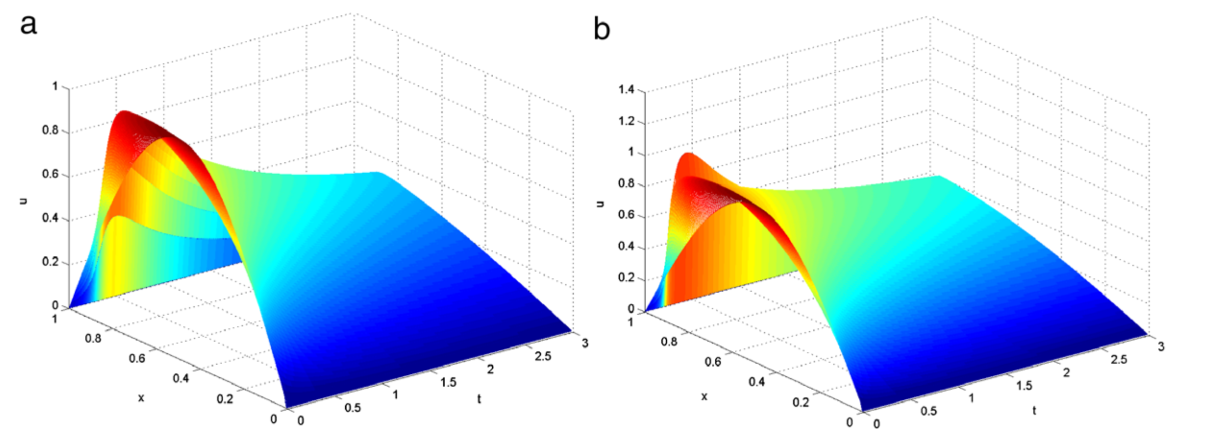

Fig. 3. Figures of the time-fractional Burgers equation ($0 < \alpha < 1$). The space step-length is $h = 0.01$. (a) $(\alpha, Re, \omega) = (0.9, 100, 0.0025)$. (b) $(\alpha, Re, \omega) = (0.8, 300, 0.0025)$.

$$B(t, U) = \frac{1}{Re\, h^2}\begin{pmatrix} -2 & 1 & 0 & \cdots & \cdots & 0 & 0 \\ 1 & -2 & 1 & \cdots & \cdots & 0 & 0 \\ 0 & 1 & -2 & \cdots & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \cdots & -2 & 1 \\ 0 & 0 & 0 & \cdots & \cdots & 1 & -2 \end{pmatrix} + \frac{1}{2h}\begin{pmatrix} u_2 & 0 & 0 & \cdots & \cdots & 0 & 0 \\ 0 & c_2 & 0 & \cdots & \cdots & 0 & 0 \\ 0 & 0 & c_3 & \cdots & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \cdots & 0 & c_{N-2} \\ 0 & 0 & 0 & \cdots & \cdots & 0 & c_{N-2} \end{pmatrix}.$$

Here $c_k = u_{k+1} - u_{k-1}, k = 2, 3, \ldots, N-2, c_{N-2} = -u_{N-2}$.

Next, set $\Delta t = \omega, t_{n+1} = (n+1)\omega, u(x_i, t_{n+1}) \approx u_i(t_{n+1}) \approx u_i^{n+1}, i = 0, 1, 2, \ldots, N-1, n = 1, 2, \ldots, M-1$. According to the computational scheme of system (9), we can easily write out the computational scheme for system (14) so the scheme is omitted here. By numerical experiments, we may take $R = 6$, the meaning of R is in the computational scheme of system (9).

If $\alpha = 1$, then system (11) is the classical Burgers equation. Here we apply our numerical algorithm to compute the classical Burgers equation. In this case, for a fixed space step-length h, Re relates to the time step-length ω, i.e., smaller Re, smaller ω. The numerical results obtained by our algorithm are the same as those of Sun and Qin's [18]. In the present paper we focus on studying the time-fractional Burgers equation. For a given space step-length h, during our numerical simulations we find that the time step-length ω depends on parameter Re for $0.5 < \alpha \leq 1$, i.e, the smaller Re, the smaller ω. Furthermore, when α decreases from $\alpha = 1$, Re should be more than 100 and the time step-length ω should be less than 0.005. But it is different for the case with $1 < \alpha < 2$. For a given space step-length h, when α becomes larger from $\alpha = 1$, Re should be less than 100 and ω should be less than 0.005. The computer graphics for classical/fractional Burgers equation are in Figs. 2–5. These numerical results show that our algorithm works well. This numerical method can be certainly applied to other fractional differential equations, for example, the known fractional KdV equation [19].

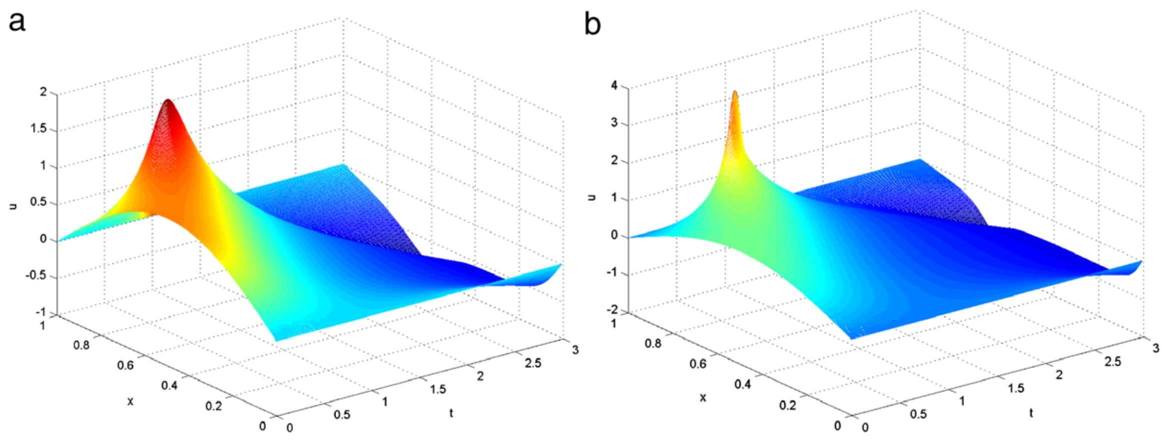

Fig. 4. Figures of the time-fractional Burgers equation ($1 < \alpha < 2$). The space step-length is $h = 0.01$. (a) $(\alpha, Re, \omega) = (1.5, 50, 0.005)$. (b) $(\alpha, Re, \omega) = (1.5, 100, 0.005)$.

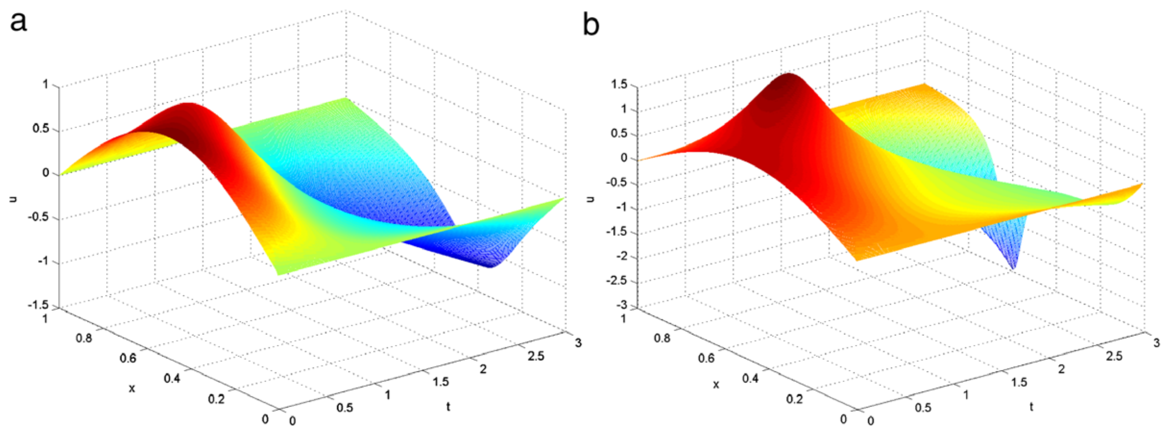

Fig. 5. Figures of the time-fractional Burgers equation ($1 < \alpha < 2$). The space step-length is $h = 0.01$. (a) $(\alpha, Re, \omega) = (1.8, 10, 0.005)$. (b) $(\alpha, Re, \omega) = (1.8, 20, 0.005)$.

5. Conclusion

In this paper, we derive a novel algorithm based on Adomian decomposition for fractional differential equations. Comparing the fractional Adams method with our numerical method, the latter needs less time due to less iteration time. By using this method, we can find a smaller "efficient dimension" such that the fractional Lorenz system is chaotic. And we also successfully compute the fractional Burgers equation. From the experiments, we find that the Reynolds number *Re* relates to the time step-length for given α and space step-length.

References

[1] G.M. Zaslavsky, Chaos, fractional kinetics, and anomalous transport, Phys. Rep. 371 (2002) 461–580.
[2] D. Baleanu, K. Diethelm, E. Scalas, J.J. Trujillo, Fractional Calculus Models and Numerical Methods, World Scientific, Singapore, 2009.
[3] K.S. Miller, B. Ross, An Introduction to the Fractional Calculus and Fractional Differential Equations, John Wiley & Sons, New York, 1993.
[4] A.A. Kilbas, H.M. Srivastava, J.J. Trujillo, Theory and Applications of Fractional Differential Equations, Elsevier, Amsterdam, 2006.
[5] C.P. Li, W.H. Deng, Remarks on fractional derivative, Appl. Math. Comput. 187 (2007) 774–784.
[6] C.P. Li, X.H. Dao, P. Guo, Fractional derivatives in complex plane. Nonlinear Anal. TMA, 2009, doi:10.1016/j.na.2009.01.021 (in press).
[7] K. Diethelm, N.J. Ford, Analysis of fractional differential equations, J. Math. Anal. Appl. 265 (2002) 229–248.
[8] K. Diethelm, N.J. Ford, A.D. Freed, A Predictor–Corrector approach for the numerical solution of fractional differential equations, Nonlinear Dynam. 29 (2002) 3–22.
[9] K. Diethelm, N.J. Ford, A.D. Freed, Detailed error analysis for a fractional Adams method, Numer. Algorithms 36 (2004) 31–52.
[10] G. Adomian, A review of the decomposition method in applied mathematics, J. Math. Anal. Appl. 135 (1988) 501–544.
[11] G. Adomian, Solving Frontier Problems of Physics: The Decomposition Method, Kluwer, 1995.
[12] Y. Cherruault, G. Adomian, Decomposition methods: A new proof of convergence, Math. Comput. Model. 18 (12) (1993) 103–106.
[13] I. Grigorenko, E. Grigorenko, Chaotic dynamics of the fractional Lorenz system, Phys. Rev. Lett. 91 (2003) 034101-1.
[14] C.P. Li, G.J. Peng, Chaos in Chen's system with a fractional order, Chaos Solitons Fractals 22 (2004) 443–450.
[15] Y.H. Wang, C.P. Li, Does the fractional Brusselator with efficient dimension less than 1 have a limit cycle? Phys. Lett. A 363 (2007) 414–419.

[16] W.H. Deng, C.P. Li, The evolution of chaotic dynamics for fractional unified system, Phys. Lett. A 372 (2008) 401–407.
[17] T.C. Hu, Y.H. Wang, Numerical detection of the lowest "efficient dimensions" for chaotic fractional differential systems, Open Mathe. J. 1 (2008) 11–18.
[18] J.Q. Sun, M.Z. Qin, A kind of explicit stable method to solve the Burgers equation, Math. Numer. Sinica 29 (2007) 67–72.
[19] S. Momani, An explicit and numerical solutions of the fractional KdV equation, Math Comput Simulation 70 (2005) 110–118.