 CMES, vol.73, no.4, pp.395-431, 2011

An Iterative Algorithm for Solving a System of Nonlinear Algebraic Equations, $\mathbf{F}(\mathbf{x})=\mathbf{0}$, Using the System of ODEs with an Optimum α in $\dot{\mathbf{x}}=\lambda[\alpha\mathbf{F}+(1-\alpha)\mathbf{B}^{\mathrm{T}}\mathbf{F}]$; $B_{ij}=\partial F_i/\partial x_j$

Chein-Shan Liu[1] **and Satya N. Atluri**[2]

Abstract: In this paper we solve a system of nonlinear algebraic equations (NAEs) of a vector-form: $\mathbf{F}(\mathbf{x})=\mathbf{0}$. Based-on an invariant manifold defined in the space of $(\mathbf{x},t)$ in terms of the residual-norm of the vector $\mathbf{F}(\mathbf{x})$, we derive a system of nonlinear ordinary differential equations (ODEs) with a fictitious time-like variable t as an independent variable: $\dot{\mathbf{x}}=\lambda[\alpha\mathbf{F}+(1-\alpha)\mathbf{B}^{\mathrm{T}}\mathbf{F}]$, where λ and α are scalars and $B_{ij}=\partial F_i/\partial x_j$. From this set of nonlinear ODEs, we derive a purely iterative algorithm for finding the solution vector $\mathbf{x}$, without having to invert the Jacobian (tangent stiffness matrix) $\mathbf{B}$. Here, we introduce three new concepts of *attracting set*, *bifurcation* and *optimal combination*, which are controlled by two parameters γ and α. Because we have derived all the related quantities explicitly in terms of $\mathbf{F}$ and its differentials, the attracting set, and an optimal α can be derived exactly. When γ changes from zero to a positive value the present algorithms undergo a Hopf bifurcation, such that the convergence speed is much faster than that by using $\gamma=0$. Moreover, when the optimal α is used we can further accelerate the convergence speed several times. Some numerical examples are used to validate the performance of the present algorithms, which reveal a very fast convergence rate in finding the solution, and which display great efficiencies and accuracies than achieved before.

Keywords: Nonlinear algebraic equations, Non-Linear Ordinary Differential Equations, Non-Linear Partial Differential Equations, Optimal Vector Driven Algorithm (OVDA), Iterative algorithm, Attracting set, Hopf bifurcation, Intermittency, Jordan algebra

[1] Department of Civil Engineering, National Taiwan University, Taipei, Taiwan. E-mail: liucs@ntu.edu.tw

[2] Center for Aerospace Research & Education, University of California, Irvine

1 Introduction

In this paper we develop a simple and novel iterative method to solve a system of nonlinear algebraic equations (NAEs): $F_i(x_1,\ldots,x_n)=0,\ i=1,\ldots,n$, or in their vector-form:

$$\mathbf{F}(\mathbf{x})=\mathbf{0}. \tag{1}$$

Different strategies may be used to solve Eq. (1) for $\mathbf{x}$, such as by introducing a fictitious time-like variable t in Eq. (1), and converting it to an initial value problem. Thus, one may obtain the solution $\mathbf{x}$ of Eq. (1) as the steady state solution of the following system of nonlinear ODEs:

$$\dot{\mathbf{x}}=-\mathbf{F}(\mathbf{x}),\ \text{ or }\ \dot{\mathbf{x}}=\mathbf{F}(\mathbf{x}), \tag{2}$$

which in general will lead to a divergence of the solution. Ramm (2007) has proposed such a lazy-bone method, and Deuflhard (2004) a continuation method for solving the second equation in Eq. (2), and called it a pseudo-transient continuation method. Nevertheless, they always lead to the wrong solution of $\mathbf{F}(\mathbf{x})=\mathbf{0}$.

Hirsch and Smale (1979) have derived a so-called continuous Newton method, governed by the following nonlinear ODEs:

$$\dot{\mathbf{x}}(t)=-\mathbf{B}^{-1}(\mathbf{x})\mathbf{F}(\mathbf{x}), \tag{3}$$

where $\mathbf{B}$ is the Jacobian matrix with its ij-component being given by $B_{ij}=\partial F_i/\partial x_j$. It can be seen that the ODEs in Eq. (3) are quite difficult to compute, because they involve an inverse matrix $\mathbf{B}^{-1}$. Usually it is not in practical to derive a closed-form solution of Eq. (3); hence a numerical scheme, such as the Euler method, should be employed to integrate Eq. (3).

In order to eliminate the need for inverting the Jacobian matrix in the iteration procedure, Liu and Atluri (2008) have proposed an alternate first-order system of nonlinear ODEs:

$$\dot{\mathbf{x}}=-\frac{\nu}{q(t)}\mathbf{F}(\mathbf{x}), \tag{4}$$

where ν is a nonzero constant and $q(t)$ may in general be a monotonically increasing function of t. In their approach, the term $\nu/q(t)$ plays the major role of being a stabilizing controller to help one obtain a solution even for a bad initial guess of the solution, and speed up the convergence. Liu and his coworkers showed that high performance can be achieved by using the above Fictitious Time Integration Method (FTIM) together with the group-preserving scheme [Liu (2001); Liu (2008,

2009a, 2009b, 2009c); Liu and Chang (2009)]. In spite of its success, the FTIM has only a local convergence, and the solutions depend on different viscous damping coefficients being used for different equations in the same problem. Atluri, Liu and Kuo (2009) have combined the above two methods, and proposed a modification of the Newton method, which does not need the inversion of B_{ij}.

To remedy the shortcoming of the vector homotopy method as initiated by Davidenko (1953), Liu, Yeih, Kuo and Atluri (2009) have proposed a scalar homotopy method with the following evolution equation for $\mathbf{x}$:

$$\dot{\mathbf{x}} = -\frac{\frac{\partial h}{\partial t}}{\|\frac{\partial h}{\partial \mathbf{x}}\|^2}\frac{\partial h}{\partial \mathbf{x}}, \tag{5}$$

where

$$h(\mathbf{x},t) = \frac{1}{2}[t\|\mathbf{F}(\mathbf{x})\|^2 - (1-t)\|\mathbf{x}\|^2], \tag{6}$$

$$\frac{\partial h}{\partial t} = \frac{1}{2}[\|\mathbf{F}(\mathbf{x})\|^2 + \|\mathbf{x}\|^2], \tag{7}$$

$$\frac{\partial h}{\partial \mathbf{x}} = t\mathbf{B}^{\mathrm{T}}\mathbf{F} - (1-t)\mathbf{x}. \tag{8}$$

Ku, Yeih and Liu (2010) combined this idea with an exponentially decaying scalar homotopy function, and developed a manifold-based exponentially convergent algorithm (MBECA):

$$\dot{\mathbf{x}} = -\frac{\nu}{(1+t)^m}\frac{\|\mathbf{F}\|^2}{\|\mathbf{B}^{\mathrm{T}}\mathbf{F}\|^2}\mathbf{B}^{\mathrm{T}}\mathbf{F}. \tag{9}$$

As pointed out by Liu and Atluri (2011), two major drawbacks appeared in the MBECA: irregular bursts and flattened behavior appear in the trajectory of the residual-error.

For the development of stable and convergent numerical algotithms for solving NAEs, it is of utmost importance to build a framework to define the evolution equations on a suitable manifold. Liu, Yeih, Kuo and Atluri (2009) were the first to construct a manifold based on a scalar homotopy function, and a method based on the concept of "normality" and "consistency condition" is derived for solving the NAEs. Unfortunately, even such an algorithm is globally convergent, but its convergence speed is terribly slow. Later, Ku, Yeih and Liu (2010) first constructed a more relevant manifold model directly based on the Euclidean-norm $\|\mathbf{F}\|$ of the residual vector of $\mathbf{F}(\mathbf{x})$ in Eq. (1).

Recently, Liu and Atluri (2011) pointed out the limitations of the $\|\mathbf{F}\|$-based algorithms, and proposed three new algorithms which are purely iterative in nature. In

this paper we introduce a novel and very simple iterative algorithm, which can be easily implemented to solve NAEs. The present algorithm can overcome the many drawbacks as observed in all the above algorithms.

The remainder of this paper is arranged as follows. In Section 2 we give a theoretical basis of the present algorithm. We start from a continuous manifold defined in terms of residual-norm, and arrive at a system of ODEs driven by a vector, which is a combination of the vectors $\mathbf{F}$ and $\mathbf{B}^{\mathrm{T}}\mathbf{F}$, where $\mathbf{B}$ is the Jacobian. Section 3 is devoted to deriving a scalar equation to keep the discretely iterative orbit on the manifold, and then we propose two new concepts of *bifurcation* and *optimization* to select the weighting factor and optimal parameter α, which automatically have a convergent behavior of the residual-error curve. In Section 4 we give six numerical examples to test the present algorithms with different weighting factors and combination parameters. Finally, the many advantages of the newly developed algorithms are emphasized in Section 5.

2 New definition for an invariant manifold, and a new evolution equation for $\dot{\mathbf{x}}$, involving both F and $\mathbf{B}^{\mathrm{T}}\mathbf{F}$

For the nonlinear algebraic equations (NAEs) in Eq. (1), we can formulate a scalar Newton homotopy function:

$$h(\mathbf{x},t)=\frac{1}{2}Q(t)\|\mathbf{F}(\mathbf{x})\|^2-\frac{1}{2}\|\mathbf{F}(\mathbf{x}_0)\|^2=0, \tag{10}$$

where, we let $\mathbf{x}$ be a function of a fictitious time-like variable t, and its initial value is $\mathbf{x}(0)=\mathbf{x}_0$.

We expect $h(\mathbf{x},t)=0$ to be an invariant manifold in the space of $(\mathbf{x},t)$ for a dynamical system $h(\mathbf{x}(t),t)=0$ to be specified further. When $Q>0$, the manifold defined by Eq. (10) is continuous, and thus the following operation of differential carried out on the manifold makes sense. As a "consistency condition", by taking the time differential of Eq. (10) with respect to t and considering $\mathbf{x}=\mathbf{x}(t)$, we have

$$\frac{1}{2}\dot{Q}(t)\|\mathbf{F}(\mathbf{x})\|^2+Q(t)(\mathbf{B}^{\mathrm{T}}\mathbf{F})\cdot\dot{\mathbf{x}}=0. \tag{11}$$

We suppose that the evolution of $\mathbf{x}$ is driven by a vector $\mathbf{u}$:

$$\dot{\mathbf{x}}=\lambda\mathbf{u}, \tag{12}$$

where λ in general is a scalar function of t, and

$$\mathbf{u}=\alpha\mathbf{F}+(1-\alpha)\mathbf{B}^{\mathrm{T}}\mathbf{F} \tag{13}$$

is a suitable combination of the residual vector $\mathbf{F}$ as well as the gradient vector $\mathbf{B}^{\mathrm{T}}\mathbf{F}$, and α is a parameter to be optimized below. *As compared to Eqs. (4) and (9), the vector field in Eq. (12) is a compromise between* $\mathbf{F}$ *and* $\mathbf{B}^{T}\mathbf{F}$ as shown in Eq. (13).

Inserting Eq. (12) into Eq. (11) we can derive

$$\dot{\mathbf{x}} = -q(t)\frac{\|\mathbf{F}\|^2}{\mathbf{F}^{\mathrm{T}}\mathbf{v}}\mathbf{u}, \tag{14}$$

where

$$\mathbf{A} := \mathbf{B}\mathbf{B}^{\mathrm{T}}, \tag{15}$$

$$\mathbf{v} := \mathbf{B}\mathbf{u} = \mathbf{v}_1 + \alpha\mathbf{v}_2 = \mathbf{A}\mathbf{F} + \alpha(\mathbf{B}-\mathbf{A})\mathbf{F}, \tag{16}$$

$$q(t) := \frac{\dot{Q}(t)}{2Q(t)}. \tag{17}$$

Hence, in our algorithm if $Q(t)$ can be guaranteed to be a monotonically increasing function of t, we may have an absolutely convergent property in solving the NAEs in Eq. (1):

$$\|\mathbf{F}(\mathbf{x})\|^2 = \frac{C}{Q(t)}, \tag{18}$$

where

$$C = \|\mathbf{F}(\mathbf{x}_0)\|^2 \tag{19}$$

is determined by the initial value $\mathbf{x}_0$. We do not need to specify the function $Q(t)$ a priori, but $\sqrt{C/Q(t)}$ merely acts as a measure of the residual error of $\mathbf{F}$ in time. Hence, we impose in our algorithm that $Q(t) > 0$ is a monotonically increasing function of t. When t is large, the above equation will force the residual error $\|\mathbf{F}(\mathbf{x})\|$ to tend to zero, and meanwhile the solution of Eq. (1) is obtained approximately.

3 Dynamics of the present iterative algorithms

3.1 Discretizing, yet keeping $\mathbf{x}$ *on the manifold*

Now we discretize the foregoing continuous time dynamics embodied in Eq. (14) into a discrete time dynamics:

$$\mathbf{x}(t+\Delta t) = \mathbf{x}(t) - \beta\frac{\|\mathbf{F}\|^2}{\mathbf{F}^{\mathrm{T}}\mathbf{v}}\mathbf{u}, \tag{20}$$

where

$$\beta = q(t)\Delta t \tag{21}$$

is the steplength. Eq. (20) is obtained from the ODEs in Eq. (14) by applying the Euler scheme.

In order to keep $\mathbf{x}$ on the manifold defined by Eq. (18), we can consider the evolution of $\mathbf{F}$ along the path $\mathbf{x}(t)$ by

$$\dot{\mathbf{F}} = \mathbf{B}\dot{\mathbf{x}} = -q(t)\frac{\|\mathbf{F}\|^2}{\mathbf{F}^{\mathrm{T}}\mathbf{v}}\mathbf{v}. \tag{22}$$

Similarly we use the Euler scheme to integrate Eq. (22), obtaining

$$\mathbf{F}(t+\Delta t) = \mathbf{F}(t) - \beta\frac{\|\mathbf{F}\|^2}{\mathbf{F}^{\mathrm{T}}\mathbf{v}}\mathbf{v}, \tag{23}$$

Taking the square-norms of both the sides of Eq. (23) and using Eq. (18) we can obtain

$$\frac{C}{Q(t+\Delta t)} = \frac{C}{Q(t)} - 2\beta\frac{C}{Q(t)} + \beta^2\frac{C}{Q(t)}\frac{\|\mathbf{F}\|^2}{(\mathbf{F}^{\mathrm{T}}\mathbf{v})^2}\|\mathbf{v}\|^2. \tag{24}$$

Thus we can derive the following scalar equation:

$$a_0\beta^2 - 2\beta + 1 - \frac{Q(t)}{Q(t+\Delta t)} = 0, \tag{25}$$

where

$$a_0 := \frac{\|\mathbf{F}\|^2\|\mathbf{v}\|^2}{(\mathbf{F}^{\mathrm{T}}\mathbf{v})^2}. \tag{26}$$

As a result $h(\mathbf{x},t) = 0$, $t \in \{0,1,2,\ldots\}$ remains to be an invariant manifold in the space of $(\mathbf{x},t)$ for the discrete time dynamical system $h(\mathbf{x}(t),t) = 0$, which will be explored further in the next two sections. Liu and Atluri (2011) first derived the formula (26) for a purely gradient-vector driven dynamical system.

3.2 A trial discrete dynamics

Now we specify the discrete time dynamics $h(\mathbf{x}(t),t) = 0$, $t \in \{0,1,2,\ldots\}$, through specifying the discrete time dynamics of $Q(t)$, $t \in \{0,1,2,\ldots\}$. Note that discrete time dynamics is an iterative dynamics, which in turn amounts to an iterative algorithm.

We first try the Euler scheme:

$$Q(t+\Delta t) = Q(t) + \dot{Q}(t)\Delta t. \tag{27}$$

Then from Eq. (17) we have

$$\beta = q(t)\Delta t = \frac{1}{2}[R(t)-1], \tag{28}$$

where the ratio $R(t)$ is defined by

$$R(t) = \frac{Q(t+\Delta t)}{Q(t)}. \tag{29}$$

As a requirement of $\dot{Q}(t) > 0$, we need $R(t) > 1$.
Thus, through some manipulations, Eq. (25) becomes

$$a_0 R^3(t) - (2a_0+4)R^2(t) + (a_0+8)R(t) - 4 = 0, \tag{30}$$

which can be further written as

$$[R(t)-1]^2[a_0 R(t) - 4] = 0. \tag{31}$$

Because $R = 1$ is a double root and does not satisfy $R > 1$, which is not the desired one, we take

$$R(t) = \frac{4}{a_0} = \frac{4(\mathbf{F}^{\mathrm{T}}\mathbf{v})^2}{\|\mathbf{F}\|^2\|\mathbf{v}\|^2}. \tag{32}$$

By using Eq. (28), Eq. (20) can now be written as

$$\mathbf{x}(t+\Delta t) = \mathbf{x}(t) - \frac{1}{2}[R(t)-1]\frac{\|\mathbf{F}\|^2}{\mathbf{F}^{\mathrm{T}}\mathbf{v}}\mathbf{u}. \tag{33}$$

Notice, however, that this algorithm has an unfortunate drawback in that, when the iterated a_0 starts to approach to 4 before it grows up to a large value, the algorithm stagnates at a point which is not necessarily a solution. We will avoid following this kind of dynamics by developing a better dynamics as below. This indicates that the present algorithm will face this fate to lose its dynamic force if we insist the iterative orbit as being located on the manifold defined by Eq. (18).

3.3 A better discrete dynamics

Let

$$s = \frac{Q(t)}{Q(t+\Delta t)} = \frac{\|\mathbf{F}(\mathbf{x}(t+\Delta t))\|^2}{\|\mathbf{F}(\mathbf{x}(t))\|^2}, \tag{34}$$

which is an important quantity to assess the convergence property of numerical algorithm for solving NAEs. *If s can be guaranteed to be $s<1$, then the residual error $\|\mathbf{F}\|$ will be decreased step-by-step.*

From Eqs. (25) and (34) we can obtain

$$a_0\beta^2 - 2\beta + 1 - s = 0, \tag{35}$$

where

$$a_0 := \frac{\|\mathbf{F}\|^2\|\mathbf{v}\|^2}{(\mathbf{F}^{\mathrm{T}}\mathbf{v})^2} \geq 1, \tag{36}$$

by using the Cauchy-Schwarz inequality:

$$\mathbf{F}^{\mathrm{T}}\mathbf{v} \leq \|\mathbf{F}\|\|\mathbf{v}\|.$$

From Eq. (35), we can take the solution of β to be

$$\beta = \frac{1-\sqrt{1-(1-s)a_0}}{a_0}, \text{ if } 1-(1-s)a_0 \geq 0. \tag{37}$$

Let

$$1-(1-s)a_0 = \gamma^2 \geq 0, \tag{38}$$

$$s = 1 - \frac{1-\gamma^2}{a_0}. \tag{39}$$

Thus, from Eq. (37) it follows that

$$\beta = \frac{1-\gamma}{a_0}, \tag{40}$$

and from Eqs. (20) and (26) we can obtain the following algorithm:

$$\mathbf{x}(t+\Delta t) = \mathbf{x}(t) - \eta\frac{\mathbf{F}^{\mathrm{T}}\mathbf{v}}{\|\mathbf{v}\|^2}\mathbf{u}, \tag{41}$$

where

$$\eta = 1-\gamma. \tag{42}$$

Here $0 \le \gamma < 1$ is a weighting parameter. Later, in the numerical examples we will explain that γ plays a major role for the *bifurcation* of discrete dynamics. Under the above condition we can prove that the new algorithm satisfies

$$\frac{\|\mathbf{F}(t+\Delta t)\|}{\|\mathbf{F}(t)\|} = \sqrt{s} < 1, \tag{43}$$

which means that the residual error is absolutely decreased.

It is interesting that in the above algorithm no Δt is required. Furthermore, the property in Eq. (43) is very important, since it guarantees the new algorithm to be absolutely convergent to the true solution.

3.4 Optimal value for α

The algorithm (41) does not specify how to choose the parameter α. One way is that α is chosen by the user. Also we can determine a suitable α such that s defined in Eq. (39) is minimized with respect to α, because a smaller s will lead to a faster convergence as shown in Eq. (43).

Thus by inserting Eq. (36) for a_0 into Eq. (39) we can write s as:

$$s = 1 - \frac{(1-\gamma^2)(\mathbf{F}\cdot\mathbf{v})^2}{\|\mathbf{F}\|^2\|\mathbf{v}\|^2}, \tag{44}$$

where $\mathbf{v}$ as defined by Eq. (16) includes a parameter α. Let $\partial s/\partial\alpha = 0$, and through some algebraic operations we can solve α by

$$\alpha = \frac{(\mathbf{v}_1\cdot\mathbf{F})(\mathbf{v}_1\cdot\mathbf{v}_2) - (\mathbf{v}_2\cdot\mathbf{F})\|\mathbf{v}_1\|^2}{(\mathbf{v}_2\cdot\mathbf{F})(\mathbf{v}_1\cdot\mathbf{v}_2) - (\mathbf{v}_1\cdot\mathbf{F})\|\mathbf{v}_2\|^2}. \tag{45}$$

Remark 1: For the usual three-dimensional vectors $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c} \in \mathbb{R}^3$, the following formula is famous:

$$\mathbf{a}\times(\mathbf{b}\times\mathbf{c}) = (\mathbf{a}\cdot\mathbf{c})\mathbf{b} - (\mathbf{a}\cdot\mathbf{b})\mathbf{c}. \tag{46}$$

Liu (2000a) has developed a Jordan algebra by extending the above formula to vectors in n-dimension:

$$[\mathbf{a},\mathbf{b},\mathbf{c}] = (\mathbf{a}\cdot\mathbf{b})\mathbf{c} - (\mathbf{c}\cdot\mathbf{b})\mathbf{a}, \quad \mathbf{a},\mathbf{b},\mathbf{c}\in\mathbb{R}^n. \tag{47}$$

Thus α in Eq. (45) can be expressed as

$$\alpha = \frac{[\mathbf{v}_1,\mathbf{v}_2,\mathbf{F}]\cdot\mathbf{v}_1}{[\mathbf{v}_2,\mathbf{v}_1,\mathbf{F}]\cdot\mathbf{v}_2}. \tag{48}$$

The above parameter α can be called the optimal α, because it brings us a new strategy to select the best orientation to search the solution of NAEs. Furthermore, we have an explicit form to implement it into the numerical program, and thus it is very time-saving for calculating it.

3.5 *The present algorithm, driven by the vector* $\mathbf{u}$

Since the fictitious time variable is now discrete, $t \in \{0,1,2,\ldots\}$, we let $\mathbf{x}_k$ denote the numerical value of $\mathbf{x}$ at the k-th step. Thus, we arrive at a purely iterative algorithm through Eqs. (41) and (42):

$$\mathbf{x}_{k+1} = \mathbf{x}_k - (1-\gamma)\frac{\mathbf{F}_k^{\mathrm{T}}\mathbf{v}_k}{\|\mathbf{v}_k\|^2}\mathbf{u}_k. \tag{49}$$

Then, we devise the following algorithm:

(i) Select $0 \le \gamma < 1$, and give an initial guess of $\mathbf{x}_0$.

(ii) For $k = 0,1,2\ldots$ we repeat the following calculations:

$$\mathbf{v}_1^k = \mathbf{A}_k\mathbf{F}_k, \tag{50}$$

$$\mathbf{v}_2^k = (\mathbf{B}_k - \mathbf{A}_k)\mathbf{F}_k, \tag{51}$$

$$\alpha_k = \frac{[\mathbf{v}_1^k, \mathbf{v}_2^k, \mathbf{F}_k]\cdot\mathbf{v}_1^k}{[\mathbf{v}_2^k, \mathbf{v}_1^k, \mathbf{F}_k]\cdot\mathbf{v}_2^k}, \quad (\text{optimal } \alpha_k) \tag{52}$$

$$\mathbf{u}_k = \alpha_k\mathbf{F}_k + (1-\alpha_k)\mathbf{B}_k^{\mathrm{T}}\mathbf{F}_k, \tag{53}$$

$$\mathbf{v}_k = \mathbf{B}_k\mathbf{u}_k, \tag{54}$$

$$\mathbf{x}_{k+1} = \mathbf{x}_k - (1-\gamma)\frac{\mathbf{F}_k\cdot\mathbf{v}_k}{\|\mathbf{v}_k\|^2}\mathbf{u}_k. \tag{55}$$

If $\mathbf{x}_{k+1}$ converges according to a given stopping criterion $\|\mathbf{F}_{k+1}\| < \varepsilon$, then stop; otherwise, go to step (ii).

In summary, we have derived a *thoroughly novel algorithm* for solving NAEs. While the parameter γ is chosen by the user and is problem-dependent, the parameter α_k is exactly given by Eq. (52). Indeed these two parameters play the roles of *bifurcation parameter* and *optimization parameter*, respectively. Up to here we have successfully derived a *drastically novel* algorithm based on the bifurcation and optimal parameter of α_k, which without the help from the formula in Eq. (36), we cannot achieve such a wonderful result. The influences of these two parameters are analyzed below through several numerical examples.

4 Numerical examples

4.1 Example 1: A finite difference method for solving a nonlinear ODE

In this example we apply the new algorithm involving the vector **u** of Eq. (13) to solve the following boundary value problem:

$$u'' = \frac{3}{2}u^2, \tag{56}$$

$$u(0) = 4, \;\; u(1) = 1. \tag{57}$$

The exact solution is

$$u(x) = \frac{4}{(1+x)^2}. \tag{58}$$

By introducing a finite difference discretization of u at the grid points we can obtain

$$F_i = \frac{1}{(\Delta x)^2}(u_{i+1} - 2u_i + u_{i-1}) - \frac{3}{2}u_i^2, \;\; i = 1, \ldots, n, \tag{59}$$

$$u_0 = 4, \;\; u_{n+1} = 1, \tag{60}$$

where $\Delta x = 1/(n+1)$ is the grid length.

First, we fix $n = 39$. Under a convergence criterion $\varepsilon = 10^{-10}$, we find that the algorithm with $\gamma = 0$ and $\alpha = 1$ does not converge within 5000 steps. In Fig. 1 we show a_0, s and residual error with $\gamma = 0$ and $\alpha = 1$. We explain the parameter γ in Eq. (41). From Fig. 1(a) it can be seen that for the case with $\gamma = 0$, the values of a_0 tend to a constant and keep unchanged. By Eq. (26) it means that there exists an attracting set for the iterative orbit of **x** described by the following manifold:

$$\frac{\|\mathbf{F}\|^2\|\mathbf{v}\|^2}{(\mathbf{F}^{\mathrm{T}}\mathbf{v})^2} = \text{Constant}. \tag{61}$$

This manifold is an *attracting set*, which is form invariant for all systems. The constant might be different for different system, but its form is invariant. When the iterative orbit approaches this attracting manifold, it is hard to reduce the residual error as shown in Fig. 1(c), while s is near to 1 as shown in Fig. 1(b).

However, with the optimal α it can converge with 1182 steps as shown in Fig. 2 for displaying a_0, s, α and residual error. From Fig. 2(a) it can be seen that for the case with $\gamma = 0$ even with optimal α, the value of a_0 also tends to a constant and keeps unchanged. It means that the attracting set is also existent for the present algorithm. Because its value of a_0 is smaller than that in Fig. 1(a) and thus a smaller s due to Eq. (39), the present algorithm can converge much faster than the algoritm with a

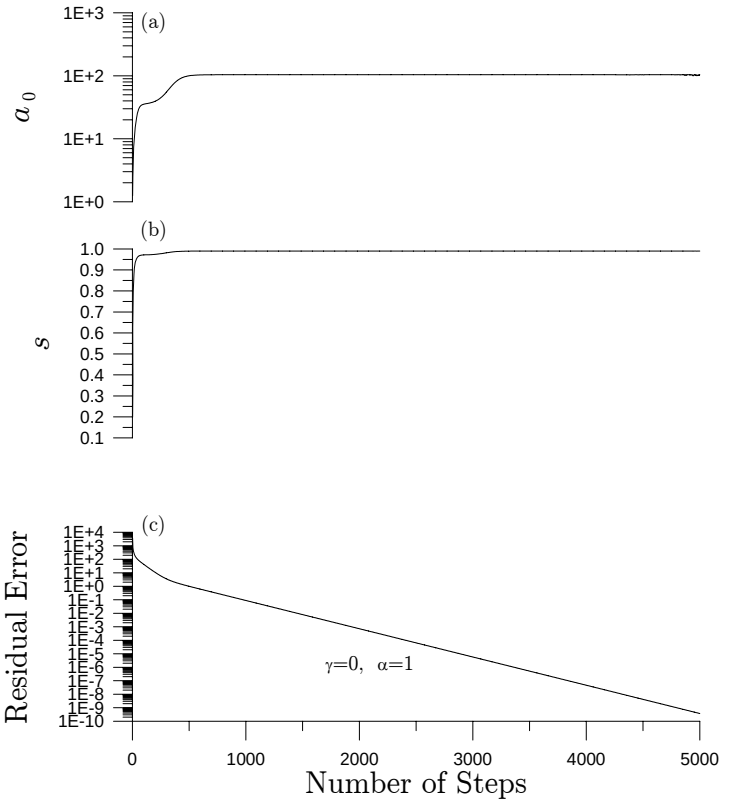

Figure 1: For example 1 using a finite difference approximation, by fixing $\gamma = 0$ showing a_0, s and residual error calculated by the new algorithm with $\alpha = 1$.

fixed $\alpha = 1$ and $\gamma = 0$. As shown in Fig. 2(c) we can observe that the behavior of α exhibits a beat oscillation within a narrow range from 0.9997 to 0.9999, which near to 1.

In summary, when $\gamma = 0$ the attracting sets are existent, which cause a slow convergence of the iterations. If the optimal value of α is used, it can reduce the values of a_0 and s, and thus increase the convergence speed.

By fixing $\gamma = 0.15$, in Fig. 3 we compare a_0, s and residual errors for the two cases with $\alpha = 1$ and optimal α. While the first case is convergent with 794 steps, the second case is convergent with 329 steps. Again, the algorithm with optimal α can significiantly reduce the value of a_0 as shwon in Fig. 3(a) by the red solid line, which is smaller than that as shown by the blue dashed line for the algorithm with a fixed $\alpha = 1$. This effect is very important to accelerate the convergence of

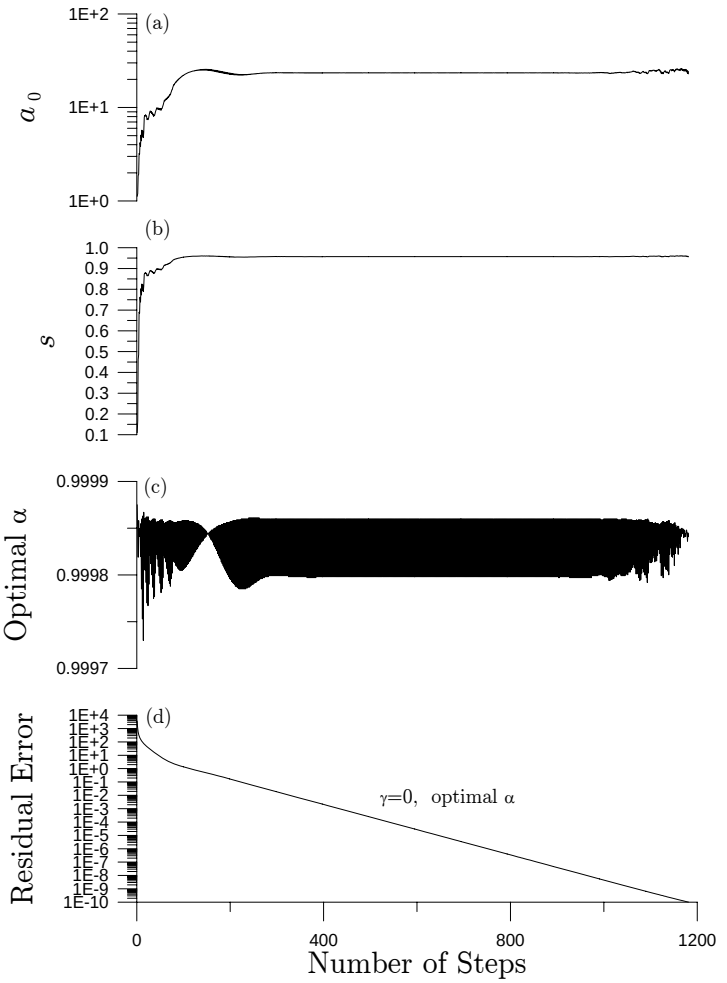

Figure 2: For example 1 using a finite difference approximation, by fixing $\gamma = 0$ showing a_0, s, optimal α and residual error calculated by the present algorithm.

the new algorithm. In Table 1 we compare the numbers of iterations of the above algorithms with different γ and α. It can be seen that the bifurcation parameter γ is first and then the optimization parameter α is second to cause a faster convergence of iteration.

The new algorithms lead to accurate numerical solutions with maximum error being smaller than 2.98×10^{-4} as shown in Fig. 4. In Fig. 5 we compare α for the cases of $\gamma = 0$ and $\gamma = 0.15$. The patterns of these two algorithms are quite different.

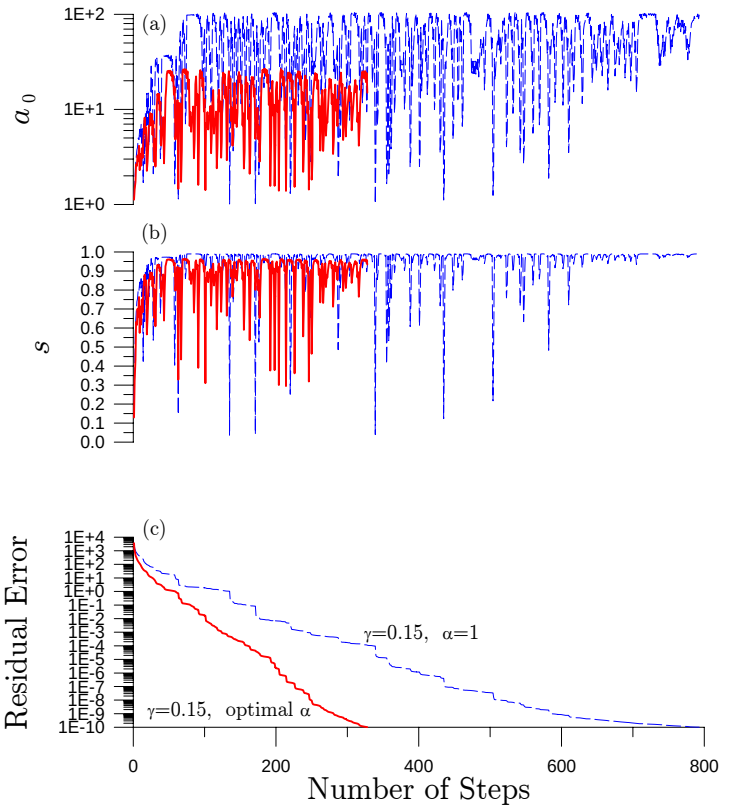

Figure 3: For example 1 using a finite difference approximation, by fixing $\gamma = 0.15$ comparing a_0, s and residual errors calculated by the new algorithm with $\alpha = 1$ and optimal α.

Table 1: Comparison of numbers of iterations for example 1 with finite difference approximation

Present algorithm	$\gamma = 0$, $\alpha = 1$	$\gamma = 0$, optimal α	$\gamma = 0.15$, $\alpha = 1$	$\gamma = 0.15$, optimal α
No. of iterations	over 5000	1182	794	329

From Fig. 5(b) we can observe that the intermittency happens for the case with $\gamma = 0.15$.

Corresponding to the tendency to lead to a constant a_0 with $\gamma = 0$, conversely, for the case $\gamma = 0.15$, a_0 is no longer tending to a constant as shown in Fig. 3(a) by solid line. Because the iterative orbit is not attracted by a constant manifold, the

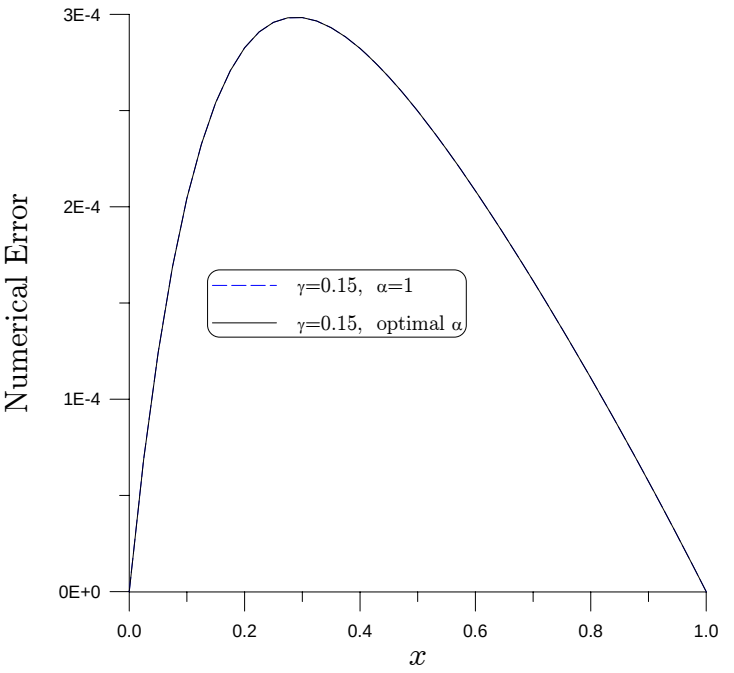

Figure 4: For example 1 using a finite difference approximation, by fixing $\gamma = 0.15$ comparing numerical errors calculated by the new algorithm with $\alpha = 1$ and optimal α.

residual error as shown in Fig. 3(c) by a solid line can be reduced step-by-step very fast, whereas s is frequently leaving the region near to 1 as shown in Fig. 3(b) by a solid line. Thus we can observe that when γ varies from a zero value to a positive value, the iterative dynamics given by Eq. (41) undergoes a Hopf bifurcation (tangent bifurcation), such as the ODEs behavior observed by Liu (2000b, 2007). The original stable manifold existent for $\gamma = 0$ now becomes a ghost manifold for $\gamma = 0.15$, and thus the orbit generated from the case $\gamma = 0.15$ cannot be attracted by that manifold, and instead, leads to an irregularly jumping behavior of a_0 as shown in Fig. 3(a) by a solid line.

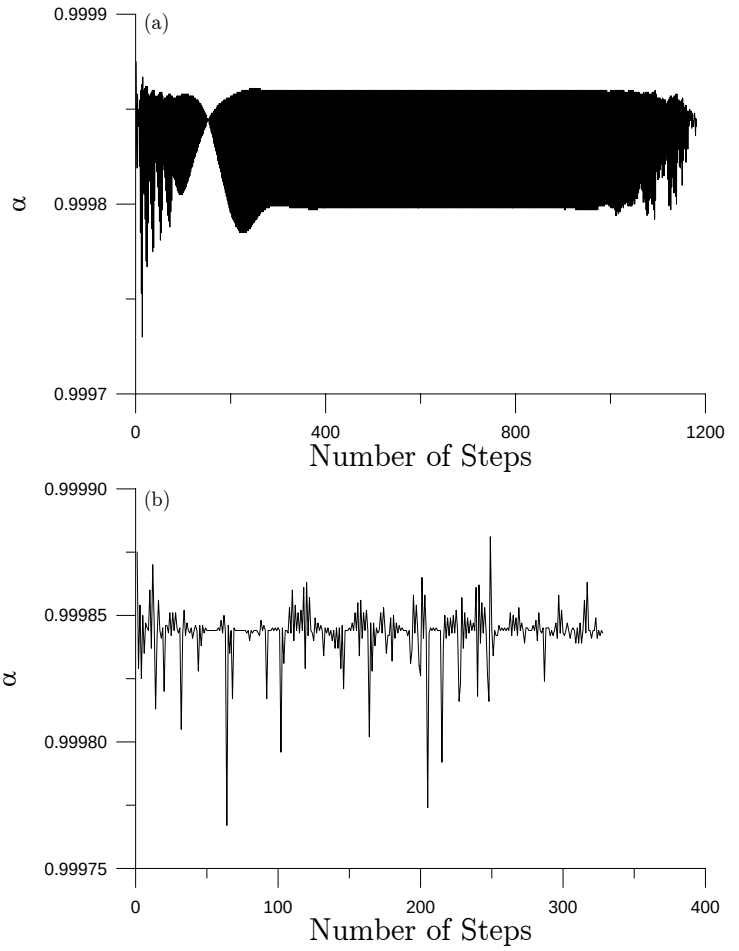

Figure 5: For example 1 using a finite difference approximation, comparing optimal α with respect to number of steps for (a) $\gamma = 0$, and (b) $\gamma = 0.15$.

4.2 Example 1: Polynomial Expansion

As suggested by Liu and Atluri (2009) we use the following modified polynomial expansion to express the solution of Eq. (56):

$$u(x) = \sum_{k=0}^{n} a_k \left(\frac{x}{R_0} \right)^k . \tag{62}$$

Due to the boundary condition $u(0) = 4$ we take $a_0 = 4$, and the other coefficients are solved from a nonlinear algebraic equations system, obtained by inserting

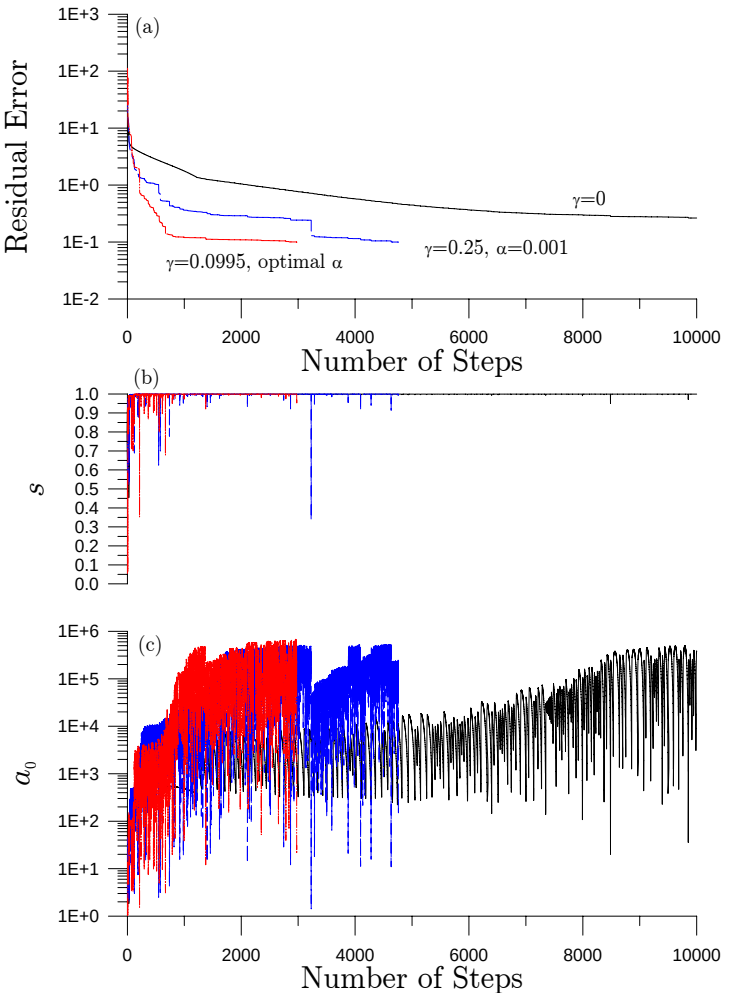

Figure 6: For example 1, using a polynomial expansion, comparing residual errors, s and a_0 under different parameters.

Eq. (62) into Eq. (56) and collocating at some points inside the domain.

With $n = 10$ and $R_0 = 1.5$ we solve this problem by using the new algorithm for three cases (a) $\gamma = 0$ and $\alpha = 0.001$, (b) $\gamma = 0.25$ and $\alpha = 0.001$, and (c) $\gamma = 0.0995$ and α being optimal. Under the convergence criterion $\varepsilon = 0.1$, the algorithm with case (a) does not converge within 10000 steps, but the algorithm with case (b) can converge within 4763 steps. More interestingly, if we use algorithm with case (c), it can converge faster within 2975 steps. In Fig. 6 we compare residual errors, s

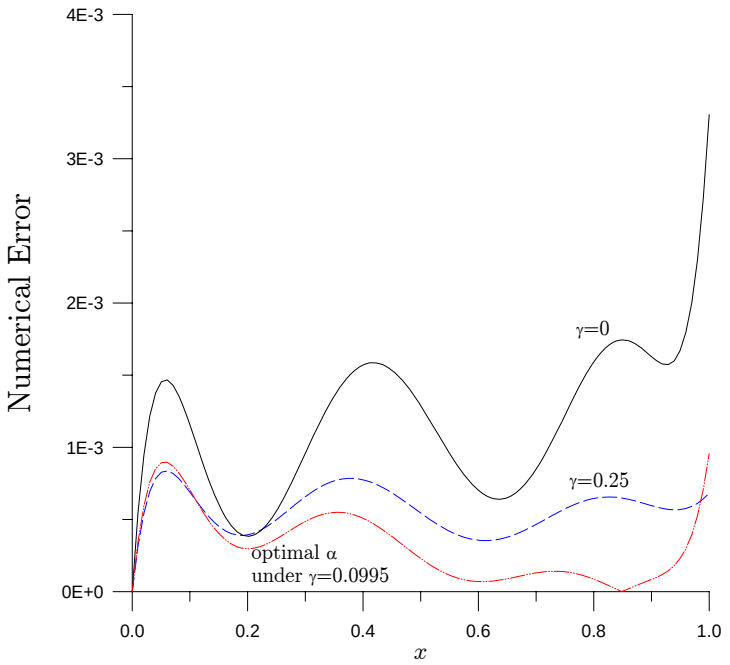

Figure 7: For example 1, using a polynomial expansion, comparing numerical errors under different parameters.

and a_0 for these three algorithms, and the numerical errors are compared in Fig. 7. Obviously, the algorithm with case (c) can provide the most accurate numerical solution. The data of α obtained from the algorithm with case (c) is shown in Fig. 8, which displays a typical intermittent behavior.

4.3 Example 1: Differential Quadrature

According to the concept of Differential Quadrature (DQ), the first order derivative of a differentiable function $f(x)$ with respect to x at a point x_i, is approximately expressed as

$$f'(x_i) = \sum_{j=1}^{n} a_{ij} f(x_j), \tag{63}$$

where a_{ij} is the weighting coefficient contributed from the j-th grid point to the first order derivative at the i-th grid point. Similarly, in DQ the second order derivative

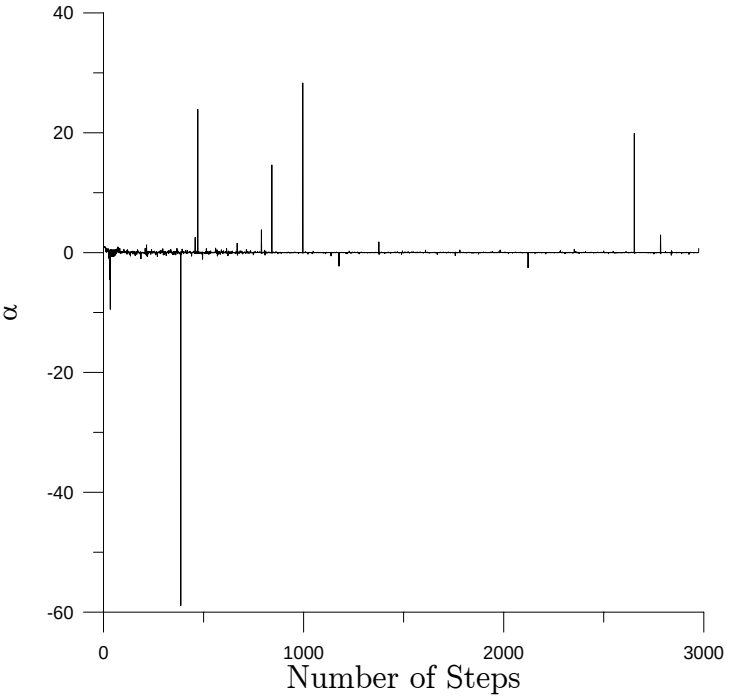

Figure 8: For example 1, using a polynomial expansion, showing α with respect to number of steps, with a typical intermittency behavior.

at the i-th grid point is given by

$$f''(x_i) = \sum_{j=1}^{n} b_{ij} f(x_j), \tag{64}$$

where $b_{ij} = \sum_{k=1}^{n} a_{ik} a_{kj}$ is the weighting coefficient of the second order derivative. In order to determine the weighting coefficients, the test functions were proposed by Bellman, Kashef and Casti (1972). They used the polynomials as test functions with orders from zero to $n-1$ when the number of grid points is n, that is,

$$g(x_i) = x_i^{k-1},\ i = 1,\ldots,n,\ \ k = 1,\ldots,n. \tag{65}$$

Then inserting the test functions $g(x) = x^{k-1}$ for $f(x)$ into Eq. (63) leads to a set of linear algebraic equations with the Vandermonde matrix as a coefficient matrix. The ill-posedness of Vandermonde system is stronger when the number of grid points is larger. Usually, the solutions were not accurate when the number of grid points was larger than 13 [Shu (2000)].

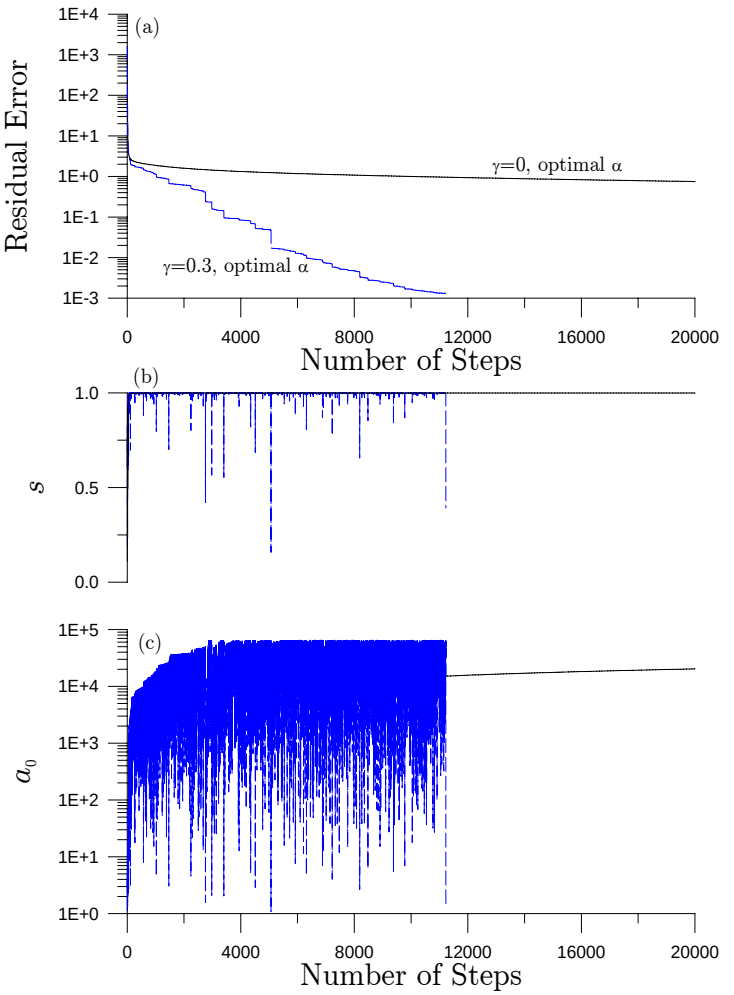

Figure 9: For example 1, using differential quadrature, comparing residual errors, s and a_0 under different parameters.

We can employ the following test functions as an improvement:

$$g(x_i) = \left(\frac{x_i}{R_0}\right)^{k-1}, \; i = 1,\ldots,n, \;\; k = 1,\ldots,n. \tag{66}$$

With a suitable choice of the constant R_0, the resultant system is well-posed [Liu and Atluri (2009)].

As an application we apply the DQ and the new algorithm to solve the problem in

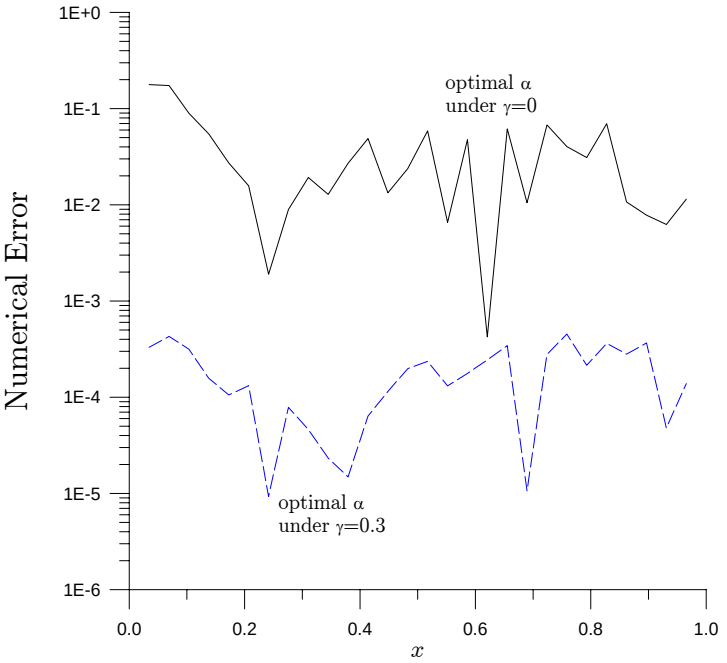

Figure 10: For example 1, using differential quadrature, comparing numerical errors under different parameters.

Eqs. (56) and (57). By introducing the DQ of u' and u'' at the grid points we can obtain

$$u_i' = \sum_{j=1}^{n} a_{ij} u_j, \quad u_i'' = \sum_{j=1}^{n} b_{ij} u_j, \tag{67}$$

where $u_i' = u'(x_i)$, $u_i'' = u''(x_i)$, $x_i = (i-1)\Delta x$ with $\Delta x = 1/(n-1)$ the grid length, and $b_{ij} = a_{ik} a_{kj}$. Thus we have the following nonlinear algebraic equations for u_i:

$$F_i = \sum_{j=1}^{n} b_{ij} u_j - \frac{3}{2} u_i^2 = 0, \tag{68}$$

$$u_1 = 4, \quad u_n = 1. \tag{69}$$

Using the following parameters $R_0 = 1.5$ and $n = 30$, we solve this problem by using the new algorithm for two cases (a) $\gamma = 0$ and α optimized, (b) $\gamma = 0.3$ and α optimized. Under the convergence criterion $\varepsilon = 10^{-3}$, the algorithm with case

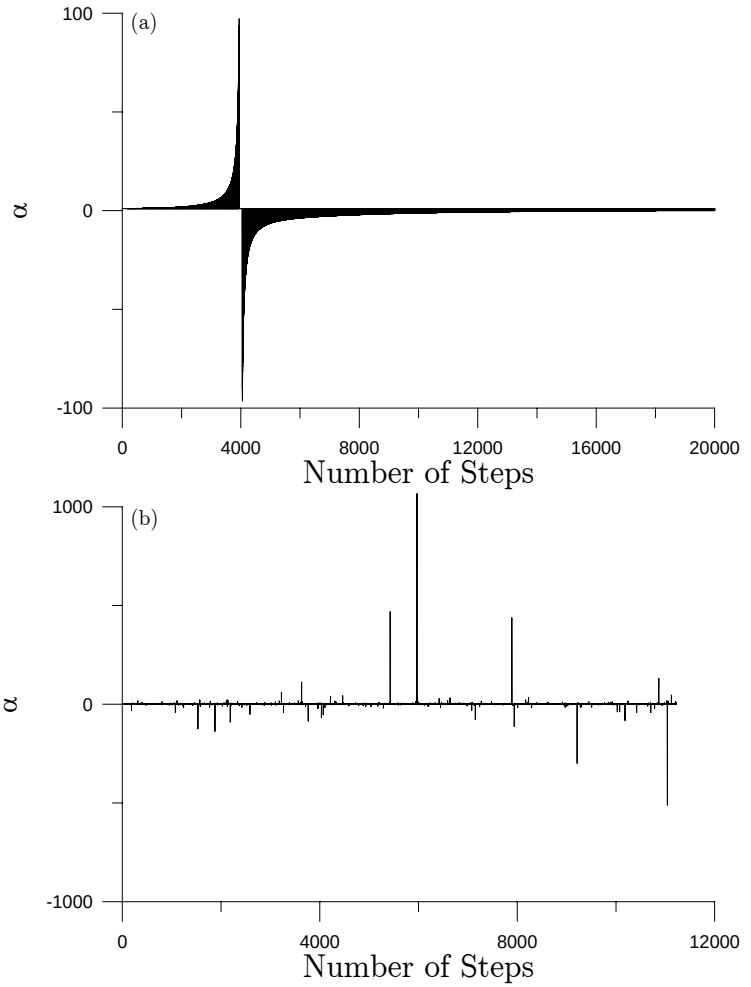

Figure 11: For example 1, using differential quadratue, comparing optimized α under different parameters.

(a) does not converge within 20000 steps, but the algorithm with case (b) can converge within 11228 steps. In Fig. 9 we compare residual errors, s and a_0 for these two algorithms, and the numerical errors are compared in Fig. 10. When the algorithm with case (b) can provide a very accurate solution with a maximum error 4.5×10^{-4}, the algorithm with case (a) is poor. From Fig. 9(c) it can be seen that the iterative orbit generated from the algorithm with $\gamma = 0$ will approach to a constant manifold with its a_0 being a constant, which causes the slowness of convergence

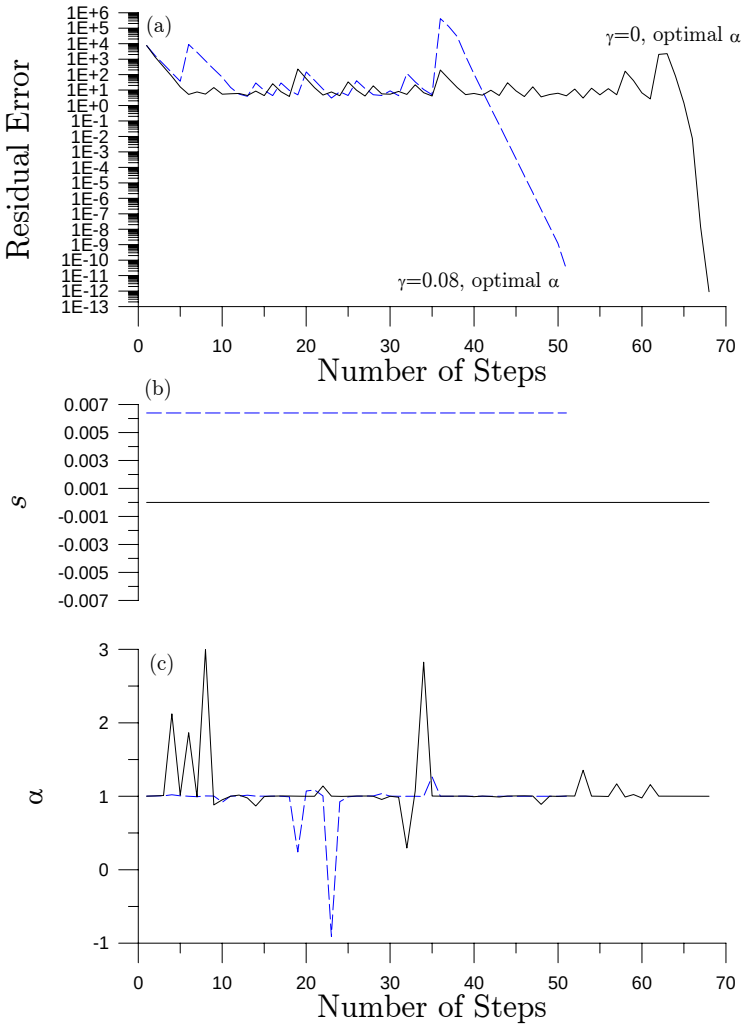

Figure 12: For example 2, with optimized α and different γ, comparing residual errors, s and α.

of this algorithm with $\gamma = 0$. Thus when we choose a suitable $\gamma > 0$, the situation is quite different that the convergence becomes fast. The data of α obtained from these two algorithms are shown in Fig. 11. For the later case it displays a typical intermittent behavior. Shen and Liu (2011) have calculated this problem by using the scaling DQ with $R_0 = 1.5$ together with the Fictitious Time Integration Method proposed by Liu and Atluri (2008). However, in order to obtain the same accurate

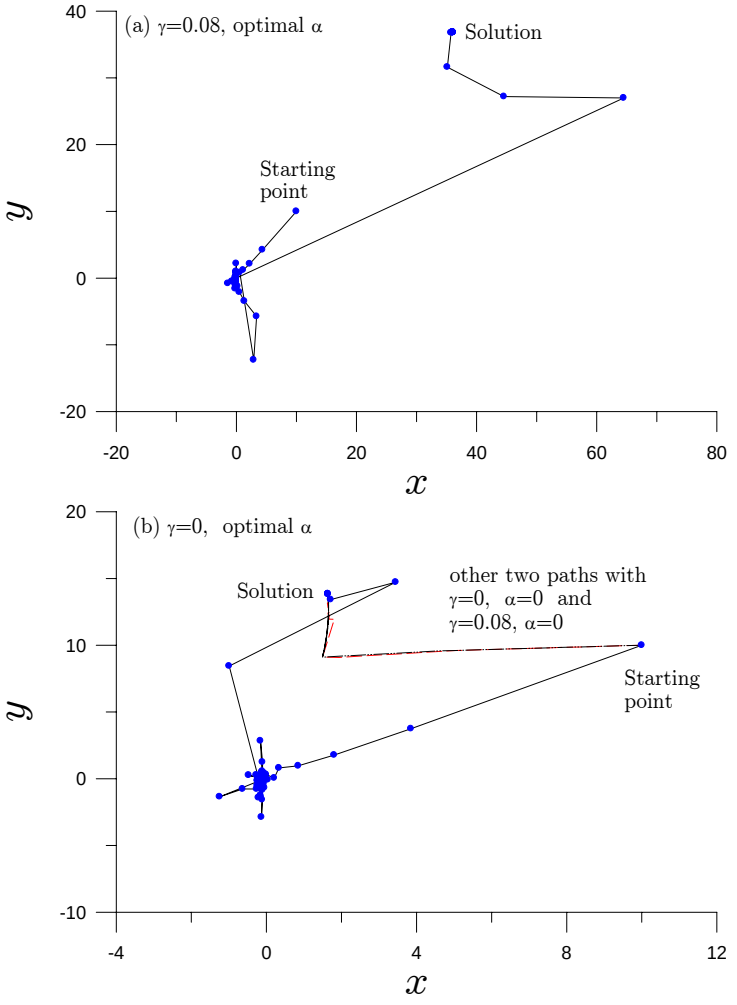

Figure 13: For example 2, with optimized α and different γ, comparing the iterative paths with that of $\alpha = 0$.

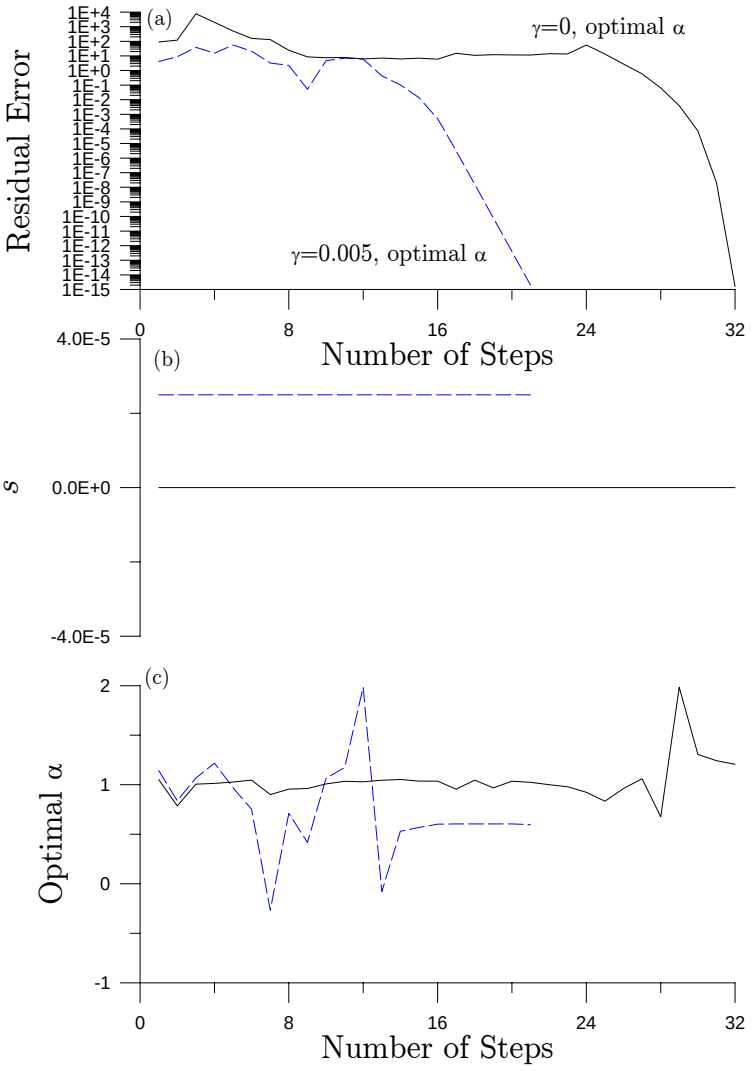

Figure 14: For example 3, with optimized α and different γ, comparing residual errors, a_0 and α.

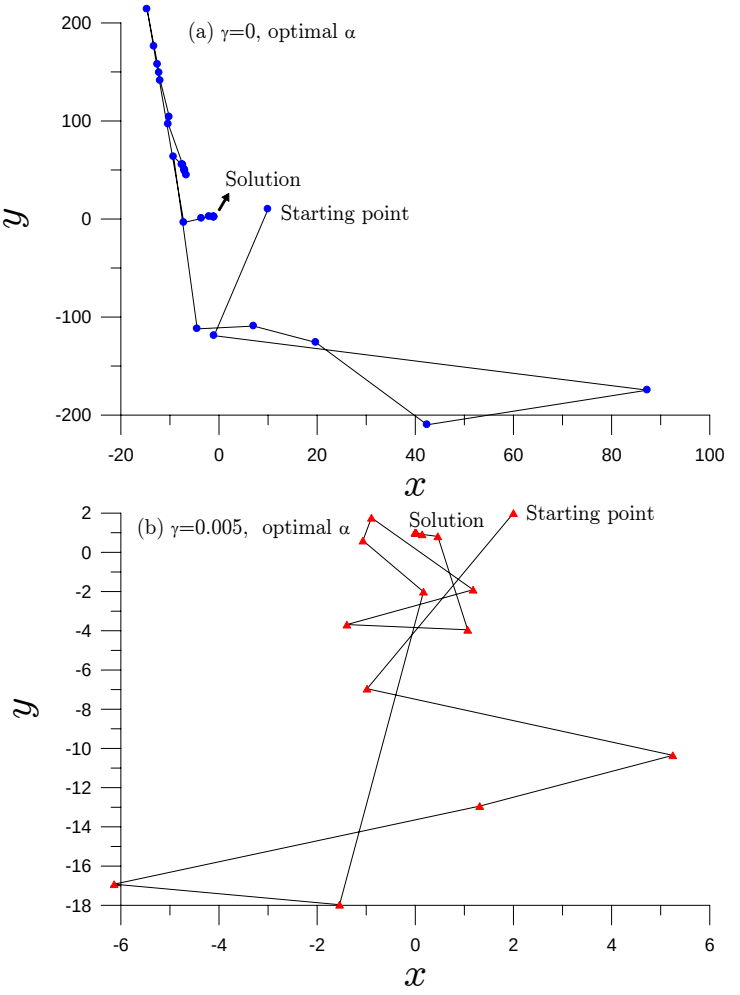

Figure 15: For example 3, with optimized α and different γ, displaying the iterative paths for (a) the first solution, and (b) the second solution. a_0 and α.

solution, that algorithm requires 154225 steps.

4.4 Example 2

We revist the following two-variable nonlinear equation [Hirsch and Smale (1979)]:

$$F_1(x,y) = x^3 - 3xy^2 + a_1(2x^2 + xy) + b_1y^2 + c_1x + a_2y = 0, \tag{70}$$

$$F_2(x,y) = 3x^2y - y^3 - a_1(4xy - y^2) + b_2x^2 + c_2 = 0, \tag{71}$$

where $a_1 = 25$, $b_1 = 1$, $c_1 = 2$, $a_2 = 3$, $b_2 = 4$ and $c_2 = 5$.

This equation has been studied by Liu and Atluri (2008) by using the fictitious time integration method (FTIM), and then by Liu, Yeih and Atluri (2010) by using the multiple-solution fictitious time integration method (MSFTIM). Liu and Atluri (2008) found three solutions by guessing three different initial values, and Liu, Yeih and Atluri (2010) found four solutions.

Starting from an initial value of $(x_0, y_0) = (10, 10)$ we solved this problem under two cases with (a) $\gamma = 0.08$ and optimal α, and (b) $\gamma = 0$ and optimal α under a convergence criterion $\varepsilon = 10^{-10}$. For the case (a) we find one root $(x,y) = (36.045402, 36.807508)$ through 51 iterations with residual errors of $(F_1, F_2) = (-1.78 \times 10^{-11}, -2.18 \times 10^{-11})$. Previously, Liu and Atluri (2008) have found this root by using the fictitious time integration method (FTIM) through 1474 steps. For the case (b) we found the fifth root $(x,y) = (1.635972, 13.847665)$ through 68 iterations with residual errors of $(F_1, F_2) = (-2.13 \times 10^{-14}, 9.09 \times 10^{-13})$. It is indeed very near to an exact solution. If we fix $\alpha = 0$ and $\gamma = 0$ we can also find this root, but it spends 2466 steps. Conversely, when $\alpha = 0$ and $\gamma = 0.08$ we can find this root only through 262 steps. It can be seen that the present algorithm with optimal α is very powerful to search the solution of nonlinear equations. For comparison purpose, we compare the iterative paths for the fifth root obtained by these algorithms.

For these two cases we compare the residual errors, s and α in Fig. 12. The iterative paths are shown in Fig. 13. It is interesting that for the algorithm with optimal α, the value of a_0 attains its minmum $a_0 = 1$ for all iterative steps, and thus $s = 0.08^2$ for case (a) and $s = 0$ for case (b). From the residual errors as shown in Fig. 12(a) and the iterative paths as shown in Fig. 13 we can observe that the mechanism to search solution has three stages: a mild convergence stage, an orientation adjusting stage where residual error appearing to be a plateau, and then following a fast convergence stage.

4.5 Example 3

We consider a singular case of **B** obtained from the following two nonlinear algebraic equations [Boggs (1971)]:

$$F_1 = x_1^2 - x_2 + 1, \tag{72}$$

$$F_2 = x_1 - \cos\left(\frac{\pi}{2}x_2\right), \tag{73}$$

$$\mathbf{B} = \begin{bmatrix} 2x_1 & -1 \\ 1 & \frac{\pi}{2}\sin\left(\frac{\pi}{2}x_2\right) \end{bmatrix}. \tag{74}$$

Obviously, on the curve of $\pi x_1 \sin(\pi x_2/2) + 1 = 0$, **B** is singular, i.e., $\det(\mathbf{B}) = 0$. They have closed-form solutions $(-1,2)$ and $(0,1)$.

As demonstrated by Boggs (1971), the Newton method does not converge to $(0,1)$, but rather it crosses the singular curve and converges to $(-\sqrt{2}/2, 3/2)$. Even under a very stringent convergence criterion $\varepsilon = 10^{-14}$, starting from the initial condition $(10,10)$ we can apply the present algorithm with $\gamma = 0$ to solve this problem within 32 iterations, and the results are shown in Fig. 14 for residual error, $s = 0$ due to $a_0 = 1$, and optimal α by solid lines. In the termination of iterative process we found that the residual errors are $F_1 = 4.44 \times 10^{-16}$ and $F_2 = -1.55 \times 10^{-15}$. It approaches to the true solution $(-1,2)$ very accurately, whose iterative path is shown in Fig. 15(a).

Starting from the initial condition $(2,2)$ we can apply the present algorithm with $\gamma = 0.005$ to solve this problem within 21 iterations, and the results are shown in Fig. 14 for residual error, $s = 0.005^2$ due to $a_0 = 1$, and optimal α by dashed lines. In the termination of iterative process we found that the residual errors are $F_1 = 1.998 \times 10^{-15}$ and $F_2 = 4.09 \times 10^{-17}$. It approaches to the true solution $(0,1)$ very accurately, and the iterative path is shown in Fig. 15(b).

The accuracy and efficiency obtained in the present algorithms are much better than those obtained by Boggs (1971), and Han and Han (2010).

4.6 Example 4

The following nonlinear diffusion reaction equation is considered:

$$\Delta u = 4u^3(x^2 + y^2 + a^2). \tag{75}$$

The amobea-like domain is given by

$$\rho = \exp(\sin\theta)\sin^2(2\theta) + \exp(\cos\theta)\cos^2(2\theta). \tag{76}$$

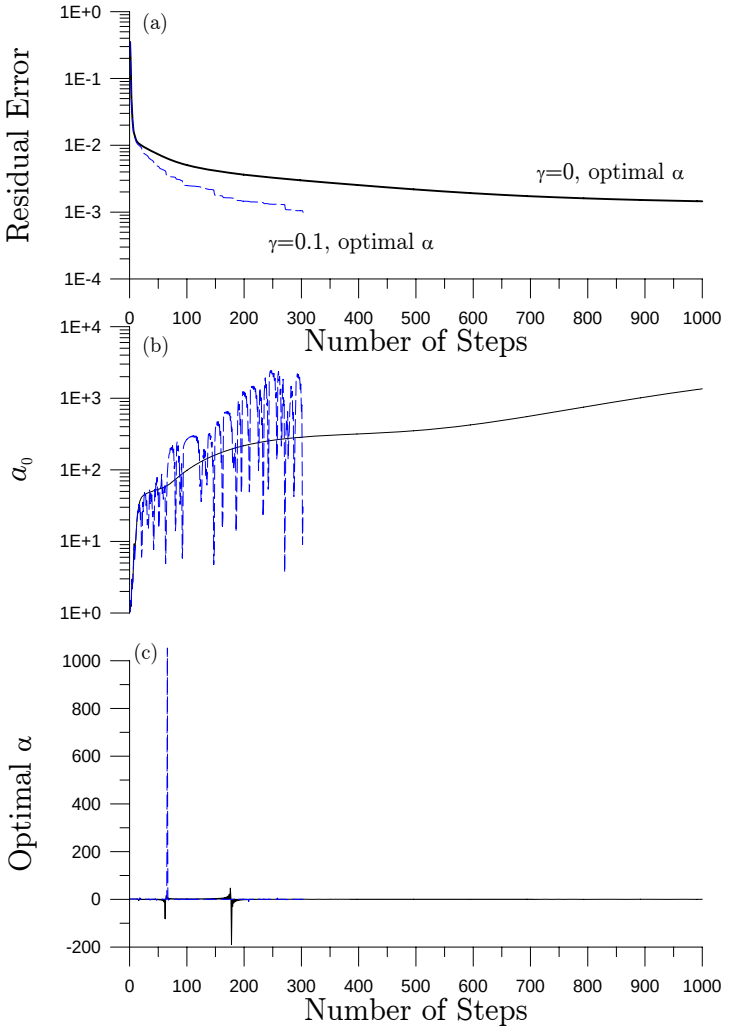

Figure 16: For example 4, with optimized α and different γ, comparing residual errors, a_0 and α.

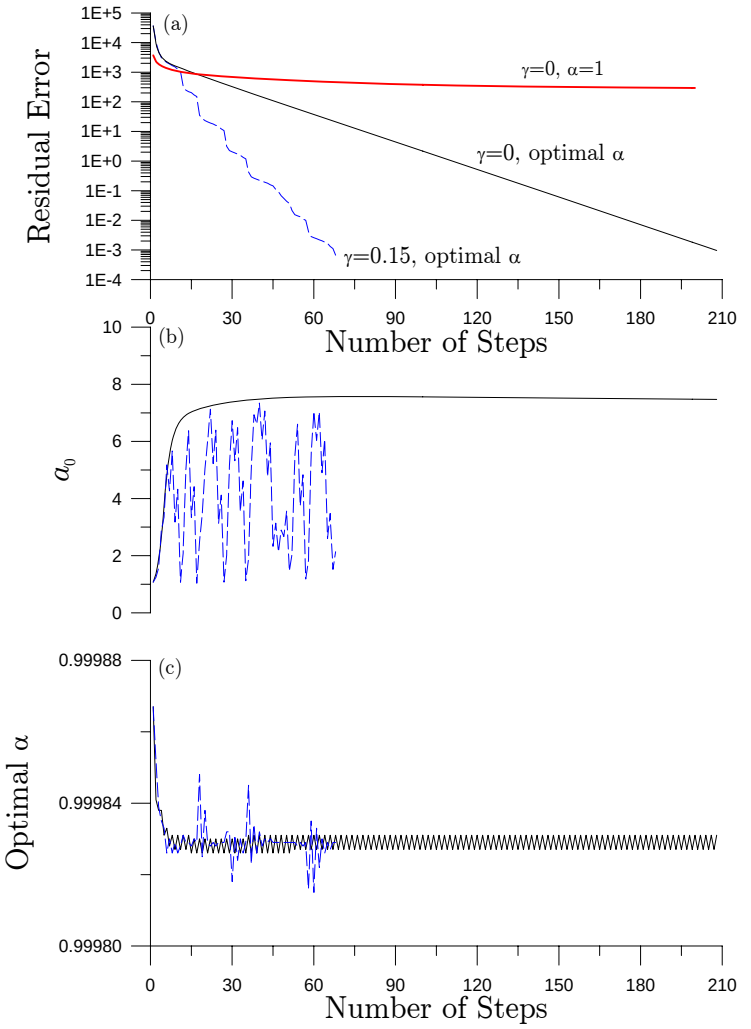

Figure 17: For example 5, with different α and γ, comparing residual errors, a_0 and α.

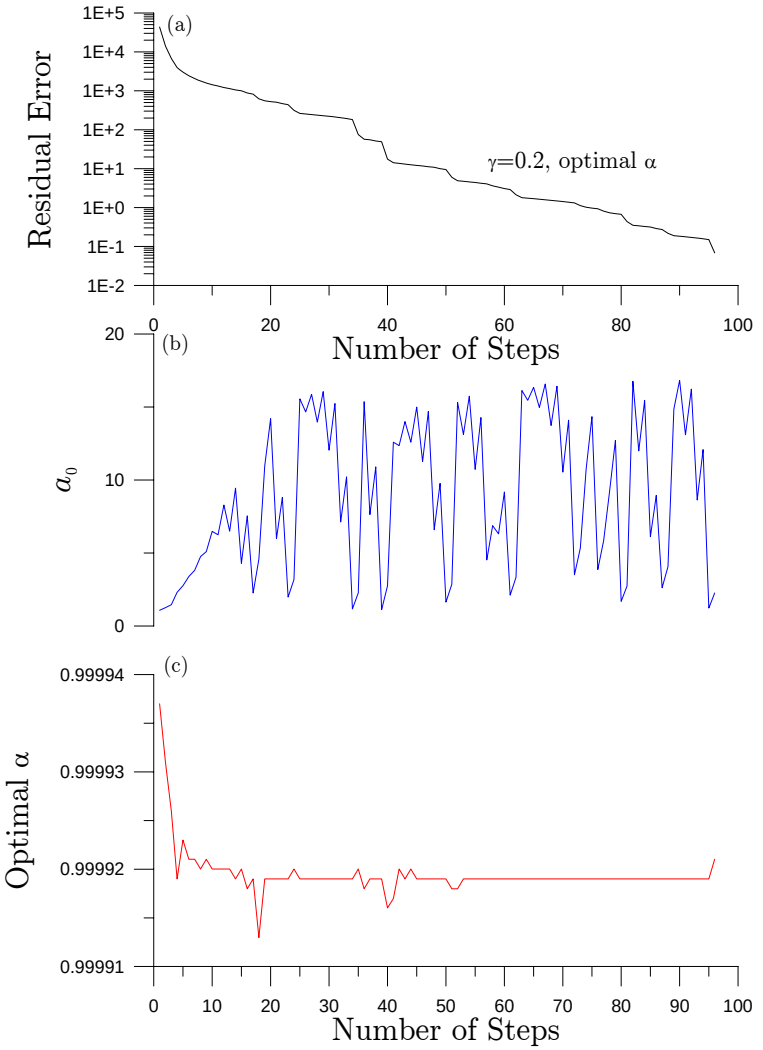

Figure 18: By applying the present algorithm to example 6 with a nonlinear backward heat conduction problem, showing (a) the residual error, (b) a_0, and (c) optimal α.

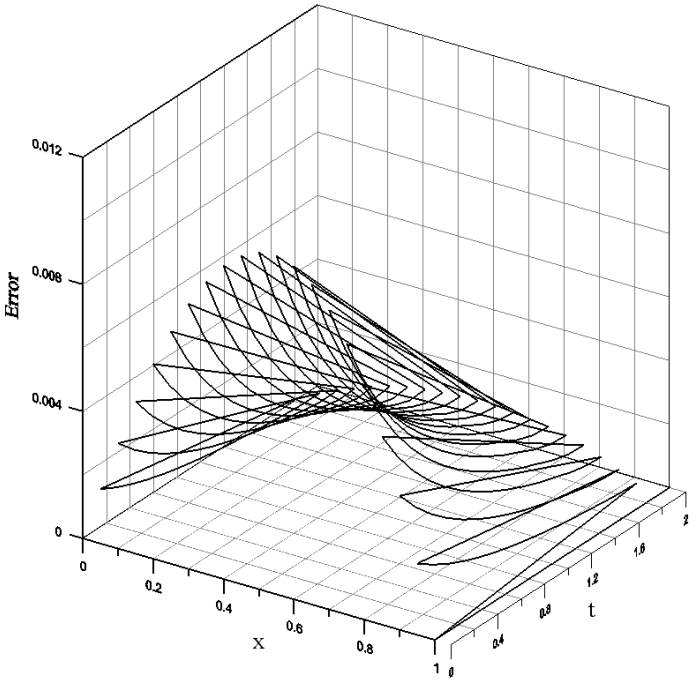

Figure 19: By applying the present algorithm to example 6 with a nonlinear backward heat conduction problem, showing the relative error of u over the plane of (x,t).

The analytic solution

$$u(x,y)=\frac{-1}{x^2+y^2-a^2} \tag{77}$$

is singular on the circle with a radius a.

Accoring to the suggestion by Liu and Atluri (2009) we can employ the following modified polynomial expansion method to express the solution:

$$u(x,y)=\sum_{j=1}^{m+2-i}\sum_{i=1}^{m+1}c_{ij}\left(\frac{x}{R_x}\right)^{i-1}\left(\frac{y}{R_y}\right)^{j-1}, \tag{78}$$

where the coefficients c_{ij} are to be determined, whose number is $n=(m+2)(m+1)/2$. The highest order of the polynomials is m. Here we use a modified Pascal triangle to expand the solution. $R_x>0$ and $R_y>0$ are characteristic lengths of the plane domain we consider.

We use the optimal vector driven method to solve this problem, where the initial values of c_{ij} are all set zeros. Let $m = 10$, $R_x = 3$ and $R_y = 2$, in Fig. 16 we compare the curves of residual error, a_0 and optimal α for two cases with $\gamma = 0$ and $\gamma = 0.1$. Under the convergence criterion $\varepsilon = 10^{-3}$, the algorithm with $\gamma = 0$ does not converge over 1000 steps; conversely, the algorithm with $\gamma = 0.1$ converges within 303 steps. Both cases give rather accurate numerical results with the maximum error 1.02×10^{-3} for $\gamma = 0$, and 7.74×10^{-4} for $\gamma = 0.1$. It can be seen that the algorithm with $\gamma = 0.1$ is convergent much fast and accurate than the algorithm with $\gamma = 0$.

4.7 Example 5

We consider a nonlinear heat conduction equation:

$$u_t = k(x)u_{xx} + k'(x)u_x + u^2 + H(x,t), \tag{79}$$

$$k(x) = (x-3)^2, \;\; H(x,t) = -7(x-3)^2 e^{-t} - (x-3)^4 e^{-2t}, \tag{80}$$

with a closed-form solution being $u(x,t) = (x-3)^2 e^{-t}$.

By applying the new algorithms to solve the above equation in the domain of $0 \leq x \leq 1$ and $0 \leq t \leq 1$ we fix $n_1 = n_2 = 15$, which are numbers of nodal points in a standard finite difference approximation of Eq. (79). Because a_0 defined in Eq. (26) is a very important factor of our new algorithms we show it in Fig. 17(b) for the present algorithm with $\gamma = 0$, while the residual error is shown in Fig. 17(a), and α is shown in Fig. 17(c) by the solid lines. Under a convergence criterion $\varepsilon = 10^{-3}$ the present algorithm with $\gamma = 0$ can also converge with 208 steps, and attains an accurate solution with the maximum error 4.67×10^{-3}. The optimal α varies in a narrow band with the range from 0.9998 to 0.99984, and a_0 approaches to a constant, which reveals an attracting set for the iterative orbit. However, due to the optimization of α, the value of a_0 does not tend to a large value. For the purpose of comparison we also plot the residual error curve obtained from $\gamma = 0$ and $\alpha = 1$ in Fig. 17(a), whose corresponding a_0 is much large than the a_0 obtained from the present algorithm and causes the very slow convergence of the algorithm without considering the optimization of α. Indeed, it does not converge within 20000 steps. Under the same convergence criterion, the present algorithm with $\gamma = 0.15$ converges much fast with only 68 steps. The residual error, a_0 and α are shown in Fig. 17 by the dashed lines. By employing $\gamma = 0.15$ the value of a_0 does not tend to a constant, and its value is smaller than the a_0 obtained from the present algorithm with $\gamma = 0$ and optimal α, which is the main reason to cause the fast convergence of the present algorithm with $\gamma = 0.15$. Very interestingly, the optimal α as shown in Fig. 17(c) by the dashed line is sometimes leaving from the narrow

band formed by the algorithm with $\gamma = 0$ and optimal α. The numbers of iterations are compared in Table 2. Amazingly, a large improvement can be obatined by using the *bifurcation parameter* and *optimization parameter*, whose convergence speed is faster about 300 times than the algorithm with $\gamma = 0$ and $\alpha = 1$.

Table 2: Comparison of numbers of iterations for example 5

Present algorithm	$\gamma = 0$, $\alpha = 1$	$\gamma = 0$, optimal α	$\gamma = 0.15$, optimal α
No. of iterations	over 20000	208	68

4.8 Example 6: A nonlinear ill-posed problem

Now, we turn our attention to a nonlinear backward heat conduction problem of Eq. (79), which is known to be a highly ill-posed nonlinear problem. In order to test the stability of present algorithm we also add a relative noise in the final time data at $t_f = 2$ with intensity $\sigma = 0.01$. The boundary conditions and a final time condition are available from the above solution of $u(x,t) = (x-3)^2 e^{-t}$.

By applying the new algorithm to solve the above equation in the domain of $0 \le x \le 1$ and $0 \le t \le 2$ we fix $\Delta x = 1/n_1$ and $\Delta t = 2/n_2$, where $n_1 = 14$ and $n_2 = 10$ are numbers of nodal points in a standard finite difference approximation of Eq. (79):

$$k(x_i)\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{(\Delta x)^2} + k'(x_i)\frac{u_{i+1,j} - u_{i-1,j}}{2\Delta x} + u_{i,j}^2 + H(x_i,t_j) - \frac{u_{i,j+1} - u_{i,j}}{\Delta t} = 0. \tag{81}$$

Because a_0 defined in Eq. (26) is a very important factor of our new algorithm we show it in Fig. 18(b) for the present algorithm with $\gamma = 0.2$, while the residual error is shown in Fig. 18(a), and α is shown in Fig. 18(c). The present algorithm is convergent with 96 steps under the convergence criterion $\varepsilon = 0.1$, which attains an accurate solution with the maximum error 1.07×10^{-2} as shown in Fig. 19.

5 Conclusions

A residual-norm based and optimization based algorithm, namely an Optimal Vector Driven Algorithm (OVDA), where $\dot{\mathbf{x}} = \lambda \mathbf{u}(\alpha)$ with $\mathbf{u} = \alpha \mathbf{F} + (1-\alpha)\mathbf{B}^{\mathrm{T}}\mathbf{F}$ the driving vector involving an optimal value of α and $B_{ij} = \partial F_i / \partial x_j$, was established in this paper to solve $\mathbf{F} = \mathbf{0}$. Although we were starting from a continuous invariant manifold based on the residual-norm and specifying a vector-driven ODEs to govern the evolution of unknown variables, we were able to derive a final novel

algorithm of purely iterative type without resorting to the fictitious time and its stepsize:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \eta \frac{\mathbf{F}_k^{\mathrm{T}} \mathbf{B}_k \mathbf{u}_k}{\|\mathbf{B}_k \mathbf{u}_k\|^2} \mathbf{u}_k, \tag{82}$$

where

$$\eta = 1 - \gamma, \;\; 0 \le \gamma < 1, \tag{83}$$

and

$$\mathbf{u}_k = \alpha_k \mathbf{F}_k + (1 - \alpha_k) \mathbf{B}_k^{\mathrm{T}} \mathbf{F}_k \tag{84}$$

is an optimal vector involving an optimal parameter α_k. The parameter γ is a very important factor, which is a bifurcation parameter, enabling us to switch the slow convergence to a new situation that the residual-error is quickly decreased. The optimal parameter α_k was derived exactly in terms of a Jordan algebra, and thus it is very time saving to implement the optimization technique into the numerical program. We have proved that the present algorithm is convergent automatically, and it is easy to implement, and without calculating the inversions of the Jacobian matrices. It can solve a large system of nonlinear algebraic equations very quickly. Several numerical examples of nonlinear equations, nonlinear ODEs, nonlinear PDEs as well as a nonlinear ill-posed problem were tested to validate the efficiency and accuracy of the present OVDA. Two mechanisms for improving the convergence speed of the present algorithm were found. For some problems only the use of the bifurcation parameter $\gamma > 0$, or only the use of the optimization parameter α is already enough to accelerate the convergence speed. However, when both the effects of *bifurcation* and *optimization* were used in all the tested problems, very high efficiencies and high accuracies which were never seen before, were achieved by the present algorithm.

Acknowledgement: Taiwan's National Science Council project NSC-99-2221-E-002-074-MY3 granted to the first author is highly appreciated. The works of the second author is supported by the Army Research Labs, Vehicle Technology Division, with Drs. A. Ghoshal and D. Lee as the program officials. The research was also supported by the World Class University (WCU) program through the National Research Foundation of Korea, funded by the Ministry of Education, Science & Technology (Grant no: R33-10049).

References

Atluri, S. N.; Liu, C.-S.; Kuo, C. L. (2009): A modified Newton method for solving non-linear algebraic equations. *J. Marine Sciences & Tech.*, vol. 17, pp. 238-247.

Bellman, R. E.; Kashef, B. G.; Casti, J. (1972): Differential quadrature: a technique for the rapid solution of nonlinear partial differential equations. *J. Comp. Phys.*, vol. 10, pp. 40-52.

Boggs, P. T. (1971): The solution of nonlinear systems of equations by A-stable integration technique. *SIAM J. Numer. Anal.*, vol. 8, pp. 767-785.

Davidenko, D. (1953): On a new method of numerically integrating a system of nonlinear equations. *Doklady Akad. Nauk SSSR*, vol. 88, pp. 601-604.

Deuflhard, P. (2004): Newton Methods for Nonlinear Problems: Affine Invariance and Adaptive Algorithms. Springer, New York.

Han, T.; Han Y. (2010): Solving large scale nonlinear equations by a new ODE numerical integration method. *Appl. Math.*, vol. 1, pp. 222-229.

Hirsch, M.; Smale, S. (1979): On algorithms for solving $f(x) = 0$. *Commun. Pure Appl. Math.*, vol. 32, pp. 281-312.

Ku, C. Y.; Yeih, W.; Liu, C.-S. (2010): Solving non-linear algebraic equations by a scalar Newton-homotopy continuation method. *Int. J. Non-Linear Sci. Num. Simul.*, vol. 11, pp. 435-450.

Liu, C.-S. (2000a): A Jordan algebra and dynamic system with associator as vector field. *Int. J. Non-Linear Mech.*, vol. 35, pp. 421-429.

Liu, C.-S. (2000b): Intermittent transition to quasiperiodicity demonstrated via a circular differential equation. *Int. J. Non-Linear Mech.*, vol. 35, pp. 931-946.

Liu, C.-S. (2001): Cone of non-linear dynamical system and group preserving schemes. *Int. J. Non-Linear Mech.*, vol. 36, pp. 1047-1068.

Liu, C.-S. (2007): A study of type I intermittency of a circular differential equation under a discontinuous right-hand side. *J. Math. Anal. Appl.*, vol. 331, pp. 547-566.

Liu, C.-S. (2008): A time-marching algorithm for solving non-linear obstacle problems with the aid of an NCP-function. *CMC: Computers, Materials & Continua*, vol. 8, pp. 53-65.

Liu, C.-S. (2009a): A fictitious time integration method for a quasilinear elliptic boundary value problem, defined in an arbitrary plane domain. *CMC: Computers, Materials & Continua*, vol. 11, pp. 15-32.

Liu, C.-S. (2009b): A fictitious time integration method for the Burgers equation. *CMC: Computers, Materials & Continua*, vol. 9, pp. 229-252.

Liu, C.-S. (2009c): A fictitious time integration method for solving delay ordinary differential equations. *CMC: Computers, Materials & Continua*, vol. 10, pp. 97-116.

Liu, C.-S.; Atluri, S. N. (2008): A novel time integration method for solving a large system of non-linear algebraic equations. *CMES: Computer Modeling in Engineering & Sciences*, vol. 31, pp. 71-83.

Liu, C.-S.; Atluri, S. N. (2009): A highly accurate technique for interpolations using very high-order polynomials, and its applications to some ill-posed linear problems. *CMES: Computer Modeling in Engineering & Sciences*, vol. 43, pp. 253-276.

Liu, C.-S.; Atluri, S. N. (2011): Simple "residual-norm" based algorithms, for the solution of a large system of non-linear algebraic equations, which converge faster than the Newton's method. *CMES: Computer Modeling in Engineering & Sciences*, vol. 71, pp. 279-304.

Liu, C.-S.; Chang, C. W. (2009): Novel methods for solving severely ill-posed linear equations system. *J. Marine Sciences & Tech.*, vol. 17, pp. 216-227.

Ramm, A. G. (2007): Dynamical System Methods for Solving Operator Equations. Elsevier, Amsterdam, Netherlands.

Shen, Y. H.; Liu, C.-S. (2011): A new insight into the differential quadrature method in solving 2-D elliptic PDEs. *CMES: Computer Modeling in Engineering & Sciences*, vol. 71, pp. 157-178.

Shu, C. (2000): Differential Quadrature and its Applications in Engineering. Springer, Berlin.