A new high-order algorithm for a class of nonlinear evolution equation



2008 J. Phys. A: Math. Theor. 41 015202

(http://iopscience.iop.org/1751-8121/41/1/015202)

IOP PUBLISHING JOURNAL OF PHYSICS A: MATHEMATICAL AND THEORETICAL

J. Phys. A: Math. Theor. **41** (2008) 015202 (17pp) doi:10.1088/1751-8113/41/1/015202

# A new high-order algorithm for a class of nonlinear evolution equation*

**Dingwen Deng and Zhiyue Zhang**

School of Mathematics and Computer Sciences, Nanjing Normal University, Nanjing 210097, People's Republic of China

E-mail: zhangzhiyue@njnu.edu.cn

Received 12 January 2007, in final form 30 October 2007
Published 12 December 2007
Online at stacks.iop.org/JPhysA/41/015202

**Abstract**

We derive a new finite difference scheme which is easily extended to fourth-order accurate in both temporal and spatial dimensions. It is shown through a discrete Fourier analysis that the method is unconditionally stable for a 2D problem. It requires only a regular seven-point difference stencil similar to that used in the standard second-order algorithms, such as the Crank–Nicolson algorithm. Numerical experiments are conducted to test its high accuracy and efficiency of the new algorithm.

PACS numbers: 02.70.Bf, 02.30.Ik

## 1. Introduction

In this paper, we are concerned with the numerical approximation to the nerve conduction equation

$$\begin{cases} u_{tt} - \Delta u_t - \Delta u = f(u)u_t + g(u) + H(x,t), & (x,t) \in \Omega \times [0,T] \\ u_t(x,t) = u(x,t) = 0, & (x,t) \in \partial\Omega \times [0,T] \\ u_t(x,0) = v_0(x), u(x,0) = u_0(x), & x \in \Omega, \end{cases} \tag{1.1}$$

where $\Omega \subset R^2$ is the rectangular domain, $[0,T]$ is the interval and $u_0$ and $v_0$ are given functions of sufficient smoothness.

In the process of nerve conduction, the nerve conduction signal $u$ and its variability with respect to time and space can be characterized by the two-dimensional pseudo-hyperbolic

* This project is partly supported by the National Natural Science Foundation of China (no 10471067), Natural Science Foundation of Jiangsu Provincial Education Department (2005101TSJB156), Jiangsu Provincial Government Scholarship for Overseas Studies and Jiangsu Provincial NSF (no BK2006215).

  

equation [1] in mathematics. Some results about the uniqueness and existence, and asymptotic behavior of solutions for this class equation can be found in [2–4]. There are also some numerical results for equation (1.1) in [5–8]. Since this kind of nonlinear evolution equation can describe much physical phenomena and possess strong physical background, thus it is necessary for us to develop the studies across-the-board and deeply from the theoretical point of view or from the numerical analysis.

Various numerical finite difference schemes have been proposed to solve equation (1.1) problems approximately. Most of these schemes are either first-order or second-order accurate in space, and have poor quality if the mesh is not sufficiently refined. For many application problems it is desirable to use high-order numerical algorithms to compute accurate solutions. However, higher order discretizations are generally associated with large (non-compact) stencils which increase the bandwidth of the resulting matrix and lead to a large number of arithmetic operations, especially for higher dimensional problems.

To obtain satisfactory higher order numerical results with reasonable computational cost, there have been attempts to develop higher order compact (HOC) schemes, which utilize only the grid nodes directly adjacent to the central node. For example, Adam employed a highly accurate compact implicit method for convective–diffusive and Burger's equation in [9]. Noye and Tan proposed a nine-point HOC implicit for unsteady 1D and 2D convection–diffusion equations with constant coefficients in [10, 11], respectively. The scheme is third-order accurate in space and second-order accurate in time. In [12], Karaa and Zhang developed a high order alternating direction implicit (ADI) solution method for solving unsteady 2D convection–diffusion problems, which is second-order accurate in time and fourth-order accurate in space. In [13, 14], an efficient higher order algorithm which is fourth-order accurate in time and fourth-order accurate in space was developed for 2D and 3D reaction–diffusion equations, respectively. For the high-order finite difference method, there are some very good works in [15] and in references cited therein.

In this paper, based on the work of [8, 14, 16], we propose a new high-order implicit solution method for the nonlinear evolution problem (1.1). Numerical simulations of the nonlinear nerve conduction are difficult on a normal computer in a reasonable time because the higher order nonlinear equation imposes severe time step restrictions on the explicit scheme, so the implicit method must be used. This new scheme is based on an approximate factorization of finite difference operators, which only requires solutions of systems of tri-diagonal equations. Meanwhile, there is no need to introduce approximations to the boundary conditions of the second derivatives. Generally speaking, there are much difficulties for mixed derivative terms in both temporal and spatial dimensions to construct a high-performance scheme basing finite difference method. We overcome the difficulty which comes from the mixed derivative term, derive the stability of the new algorithms, and reduce the computational complexity and save computational works. In particular, this new method can obtain that the order of another important physical parameter $u_t$ is the same as the one of parameter $u$. The numerical experiments of the linear and nonlinear equations demonstrate that the new method is efficient and robust with respect to mesh refinement and the time-step size. We remark that this method can be applied to the degenerate PDE such as the Cahn–Hilliard equation and the thin film equation.

The paper is organized as follows. In the following section, we will discuss this new method based on approximate factorization. The stability of this algorithm will be discussed in section 3. The extension to the nonlinear evolution equation will be discussed in section 4. The improvement of the accuracy in the temporal dimension based on the Richardson extrapolation will be presented in section 5, and section 6 presents linear and nonlinear numerical experiments.

## 2. The Fourth-order algorithm based on approximate factorization

In order to simplify to discuss, we will first present the development of an efficient high-order algorithm for the following equation:

$$u_{tt} - \Delta u_t - \Delta u + u_t = H(x,t), \qquad (x,t) \in \Omega \times [0,T]. \tag{2.1}$$

The result will then be generalized to equation (1.1).

In order to obtain the standard Crank–Nicolson scheme for equation (2.1), we set $u_t = v$ and start from the Crank–Nicolson algorithm for equation (2.1) on the rectangular grid $(x_i, y_j)$, $i = 0, 1, 2, \ldots, M$, $j = 0, 1, 2, \ldots, N$:

$$\frac{v_{i,j}^{n+1} - v_{i,j}^{n}}{\Delta t} - \frac{1}{2}\left[(v_{xx})_{i,j}^{n+1} + (v_{xx})_{i,j}^{n}\right] - \frac{1}{2}\left[(v_{yy})_{i,j}^{n+1} + (v_{yy})_{i,j}^{n}\right] - \frac{1}{2}\left[(u_{xx})_{i,j}^{n+1} + (u_{xx})_{i,j}^{n}\right]$$
$$- \frac{1}{2}\left[(u_{yy})_{i,j}^{n+1} + (u_{yy})_{i,j}^{n}\right] + \frac{u_{i,j}^{n+1} - u_{i,j}^{n}}{\Delta t} - \frac{1}{2}\left(H_{i,j}^{n+1} + H_{i,j}^{n}\right) = 0, \tag{2.2}$$

$$(u_t)_{i,j}^{n+\frac{1}{2}} = \frac{1}{2}\left(v_{i,j}^{n+1} + v_{i,j}^{n}\right).$$

The standard discretization is

$$\frac{v_{i,j}^{n+1} - v_{i,j}^{n}}{\Delta t} - \frac{1}{2}\left(\frac{\delta_x^2 v_{i,j}^{n+1}}{h_x^2} + \frac{\delta_x^2 v_{i,j}^{n}}{h_x^2}\right) - \frac{1}{2}\left(\frac{\delta_y^2 v_{i,j}^{n+1}}{h_y^2} + \frac{\delta_y^2 v_{i,j}^{n}}{h_y^2}\right) - \frac{1}{2}\left(\frac{\delta_x^2 u_{i,j}^{n+1}}{h_x^2} + \frac{\delta_x^2 u_{i,j}^{n}}{h_x^2}\right)$$
$$- \frac{1}{2}\left(\frac{\delta_y^2 u_{i,j}^{n+1}}{h_y^2} + \frac{\delta_y^2 u_{i,j}^{n}}{h_y^2}\right) + \frac{u_{i,j}^{n+1} - u_{i,j}^{n}}{\Delta t} = \frac{1}{2}\left(H_{i,j}^{n+1} + H_{i,j}^{n}\right), \tag{2.3}$$

$$\frac{u_{i,j}^{n+1} - u_{i,j}^{n}}{\Delta t} = \frac{1}{2}\left(v_{i,j}^{n+1} + v_{i,j}^{n}\right),$$

which is second-order accurate in both time and space, where $h_x$ and $h_y$ represent the grid spacing in the $x$ and $y$ dimensions, respectively. If we use the fourth-order Pade' approximation [9–14] to replace $u_{xx}$, $u_{yy}$, $v_{xx}$ and $v_{yy}$, then we obtain the following expression:

$$\frac{v_{i,j}^{n+1} - v_{i,j}^{n}}{\Delta t} - \frac{1}{2}\frac{\delta_x^2\left(v_{i,j}^{n+1} + v_{i,j}^{n}\right)}{h_x^2\left(1 + \frac{1}{12}\delta_x^2\right)} - \frac{1}{2}\frac{\delta_y^2\left(v_{i,j}^{n+1} + v_{i,j}^{n}\right)}{h_y^2\left(1 + \frac{1}{12}\delta_y^2\right)} - \frac{1}{2}\frac{\delta_x^2\left(u_{i,j}^{n+1} + u_{i,j}^{n}\right)}{h_x^2\left(1 + \frac{1}{12}\delta_x^2\right)}$$
$$- \frac{1}{2}\frac{\delta_y^2\left(u_{i,j}^{n+1} + u_{i,j}^{n}\right)}{h_y^2\left(1 + \frac{1}{12}\delta_y^2\right)} + \frac{u_{i,j}^{n+1} - u_{i,j}^{n}}{\Delta t} = \frac{1}{2}\left(H_{i,j}^{n+1} + H_{i,j}^{n}\right), \tag{2.4}$$

$$u_{i,j}^{n+1} - \frac{\Delta t}{2} v_{i,j}^{n+1} = \frac{\Delta t}{2} v_{i,j}^{n} + u_{i,j}^{n},$$

which is second-order accurate in time and fourth-order accurate in space. Set $r_x = \frac{\Delta t}{h_x^2}$ and $r_y = \frac{\Delta t}{h_y^2}$, the first formula of (2.4) can be written as

$$\left(1 - \frac{r_x}{2}\frac{\delta_x^2}{1 + \frac{1}{12}\delta_x^2} - \frac{r_y}{2}\frac{\delta_y^2}{1 + \frac{1}{12}\delta_y^2}\right)\left(v_{i,j}^{n+1} + u_{i,j}^{n+1}\right)$$
$$= \left(1 + \frac{r_x}{2}\frac{\delta_x^2}{1 + \frac{1}{12}\delta_x^2} + \frac{r_y}{2}\frac{\delta_y^2}{1 + \frac{1}{12}\delta_y^2}\right)\left(v_{i,j}^{n} + u_{i,j}^{n}\right) + \frac{\Delta t}{2}\left(H_{i,j}^{n+1} + H_{i,j}^{n}\right), \tag{2.5}$$

which can be approximately factorized as [9, 13, 14]

$$\left(1-\frac{r_x}{2}\frac{\delta_x^2}{1+\frac{1}{12}\delta_x^2}\right)\left(1-\frac{r_y}{2}\frac{\delta_y^2}{1+\frac{1}{12}\delta_y^2}\right)\left(v_{i,j}^{n+1}+u_{i,j}^{n+1}\right)$$
$$=\left(1+\frac{r_x}{2}\frac{\delta_x^2}{1+\frac{1}{12}\delta_x^2}\right)\left(1+\frac{r_y}{2}\frac{\delta_x^2}{1+\frac{1}{12}\delta_y^2}\right)\left(v_{i,j}^{n}+u_{i,j}^{n}\right)+\frac{\Delta t}{2}\left(H_{i,j}^{n+1}+H_{i,j}^{n}\right). \tag{2.6}$$

The difference between (2.5) and (2.6) is

$$\begin{aligned}
&\frac{r_xr_y}{4}\frac{\delta_x^2}{1+\frac{1}{12}\delta x^2}\frac{\delta_y^2}{1+\frac{1}{12}\delta_y^2}\left(v_{i,j}^{n+1}+u_{i,j}^{n+1}\right)-\frac{r_xr_y}{4}\frac{\delta_x^2}{1+\frac{1}{12}\delta_x^2}\frac{\delta_y^2}{1+\frac{1}{12}\delta_y^2}\left(v_{i,j}^{n}+u_{i,j}^{n}\right)\\
&\quad=\frac{r_xr_y}{4}\frac{\delta_x^2}{1+\frac{1}{12}\delta_x^2}\frac{\delta_y^2}{1+\frac{1}{12}\delta_y^2}\left(v_{i,j}^{n+1}-v_{i,j}^{n}+u_{i,j}^{n+1}-u_{i,j}^{n}\right)\\
&\quad=\frac{r_xr_y}{4}\frac{\delta_x^2}{1+\frac{1}{12}\delta_x^2}\frac{\delta_y^2}{1+\frac{1}{12}\delta_y^2}\left(\frac{v_{i,j}^{n+1}-v_{i,j}^{n}}{\Delta t}\Delta t+\frac{u_{i,j}^{n+1}-u_{i,j}^{n}}{\Delta t}\Delta t\right)\\
&\quad=\frac{\Delta t^2}{4}\frac{\delta_x^2}{h_x^2\left(1+\frac{1}{12}\delta_x^2\right)}\frac{\delta_y^2}{h_y^2\left(1+\frac{1}{12}\delta_y^2\right)}\left\{\left[(u_{tt})_{i,j}^{n+\frac{1}{2}}+O(\Delta t^2)\right]\Delta t\right.\\
&\qquad\left.+\left[(u_t)_{i,j}^{n+\frac{1}{2}}+O(\Delta t^2)\right]\Delta t\right\}\\
&\quad=\frac{\Delta t^2}{4}\left\{\left[\left(u_{ttxxyy}\right)_{i,j}^{n+\frac{1}{2}}+O\left(h_x^2h_y^2\right)+O(\Delta t^2)\right]\Delta t\right.\\
&\qquad\left.+\left[(u_{txxyy})_{i,j}^{n+\frac{1}{2}}+O\left(h_x^2h_y^2\right)+O(\Delta t^2)\right]\Delta t\right\}\\
&\quad=O(\Delta t^3)+O(\Delta t^5)
\end{aligned} \tag{2.7}$$

as long as all relevant partial derivatives in the error estimate are bounded. This additional error is of the same order as the truncation error in the original algorithm (2.5). Since the operators in (2.6) commute, we can simplify the algorithm by multiplying equation (2.6) by $\left(1+\frac{\delta_x^2}{12}\right)\left(1+\frac{\delta_y^2}{12}\right)$, which gives

$$\begin{aligned}
&\left[1+\left(\frac{1}{12}-\frac{r_x}{2}\right)\delta_x^2\right]\left[1+\left(\frac{1}{12}-\frac{r_y}{2}\right)\delta_y^2\right]\left(v_{i,j}^{n+1}+u_{i,j}^{n+1}\right)\\
&\quad=\left[1+\left(\frac{1}{12}+\frac{r_x}{2}\right)\delta_x^2\right]\left[1+\left(\frac{1}{12}+\frac{r_y}{2}\right)\delta_y^2\right]\left(v_{i,j}^{n}+u_{i,j}^{n}\right)\\
&\qquad+\frac{\Delta t}{2}\left(1+\frac{\delta_x^2}{12}\right)\left(1+\frac{\delta_y^2}{12}\right)\left(H^{n+1}+H_{i,j}^{n}\right).
\end{aligned} \tag{2.8}$$

Set $\psi_{i,j}^{n+1}=v_{i,j}^{n+1}+u_{i,j}^{n+1}$ and $\psi_{i,j}^{n}=v_{i,j}^{n}+u_{i,j}^{n}$, equation (2.8) can be written as

$$\begin{aligned}
&\left[1+\left(\frac{1}{12}-\frac{r_x}{2}\right)\delta_x^2\right]\left[1+\left(\frac{1}{12}-\frac{r_y}{2}\right)\delta_y^2\right]\psi_{i,j}^{n+1}\\
&\quad=\left[1+\left(\frac{1}{12}+\frac{r_x}{2}\right)\delta_x^2\right]\left[1+\left(\frac{1}{12}+\frac{r_y}{2}\right)\delta_y^2\right]\psi_{i,j}^{n}\\
&\qquad+\frac{\Delta t}{2}\left(1+\frac{\delta_x^2}{12}\right)\left(1+\frac{\delta_y^2}{12}\right)\left(H^{n+1}+H_{i,j}^{n}\right).
\end{aligned} \tag{2.9}$$

Hence, we have the numerical approximation algorithm of equation (2.1) as follows. We will first solve equation (2.9) in two steps:

$$\left[1+\left(\frac{1}{12}-\frac{r_x}{2}\right)\delta_x^2\right]\psi_{i,j}^{*}=\left[1+\left(\frac{1}{12}+\frac{r_x}{2}\right)\delta_x^2\right]\left[1+\left(\frac{1}{12}+\frac{r_y}{2}\right)\delta_y^2\right]\psi_{i,j}^{n}+\frac{\Delta t}{2}\left(1+\frac{\delta_x^2}{12}\right)\left(1+\frac{\delta_y^2}{12}\right)\left(H_{i,j}^{n+1}+H_{i,j}^{n}\right), \tag{2.10}$$

$$\left[1+\left(\frac{1}{12}-\frac{r_y}{2}\right)\delta_y^2\right]\psi_{i,j}^{n+1}=\psi_{i,j}^{*}. \tag{2.11}$$

And then combining the equation

$$u_{i,j}^{n+1}-\frac{\Delta t}{2}v_{i,j}^{n+1}=\frac{\Delta t}{2}v_{i,j}^{n}+u_{i,j}^{n},$$

we have the expressions of $u_{i,j}^{n+1}$ and $v_{i,j}^{n+1}$ as

$$\begin{pmatrix}u_{i,j}^{n+1}\\ v_{i,j}^{n+1}\end{pmatrix}=(1+0.5\Delta t)^{-1}\begin{pmatrix}1 & 0.5\Delta t\\ -1 & -0.5\Delta t\end{pmatrix}\begin{pmatrix}u_{i,j}^{n}\\ v_{i,j}^{n}\end{pmatrix}+\begin{pmatrix}0.5\Delta t\\ 1\end{pmatrix}w_{i,j}^{n+1}. \tag{2.12}$$

The solutions to equations (2.10) and (2.11) can be obtained by solving tri-diagonal equations since the left-hand sides of (2.10) and (2.11) include only three-point central difference operators $\delta_x^2$ and $\delta_y^2$ as defined in [9–15]. Even though the right-hand side of (2.10) includes the product of operators $\delta_x^2$ and $\delta_y^2$, it does not complicate the solution process since it is used to the known solution values from the previous time step.

While solving equation (2.10), we need boundary conditions for $\psi_{0,j}^{*}$ and $\psi_{M+1,j}^{*}$, $j=1,2,\ldots,N$. These conditions can be obtained from equation (2.11) and Dirichlet boundary condition in (2.1) by setting $i=0$ and $i=M+1$, respectively:

$$\psi_{0,j}^{*}=\left[1+\left(\frac{1}{12}-\frac{r_y}{2}\right)\delta_y^2\right]\left(u_{0,j}^{n+1}+v_{0,j}^{n+1}\right) \tag{2.13}$$

and

$$\psi_{M+1,j}^{*}=\left[1+\left(\frac{1}{12}-\frac{r_y}{2}\right)\delta_y^2\right]\left(u_{M+1,j}^{n+1}+v_{M+1,j}^{n+1}\right). \tag{2.14}$$

As the spatial discretization used in obtaining (2.10) is fourth-order accurate in space, the boundary conditions given by (2.13) and (2.14) have the same spatial accuracy as (2.10). This method avoids using one-sided difference approximations to the second spatial derivatives at the boundary, as is required by the standard compact difference algorithms [9, 14].

## 3. Stability of the algorithm

We discuss the stability of algorithms (2.10)–(2.12). Let $H(x,t)=0$, equation (2.9) can be written as

$$\left[1+\left(\frac{1}{12}-\frac{r_x}{2}\right)\delta_x^2\right]\left[1+\left(\frac{1}{12}-\frac{r_y}{2}\right)\delta_y^2\right]\psi_{i,j}^{n+1}=\left[1+\left(\frac{1}{12}+\frac{r_x}{2}\right)\delta_x^2\right]\left[1+\left(\frac{1}{12}+\frac{r_y}{2}\right)\delta_y^2\right]\psi_{i,j}^{n}. \tag{3.1}$$

In order to study the stability of the new scheme, we use the Von Neumann linear stability analysis. If we let $\psi_{i,j}^{n+1}=\xi_{l_x,l_y}^{n+1}\,\mathrm{e}^{I(\beta_x i h_x+\beta_y j h_y)}$ to be the value of $\psi_{i,j}^{n+1}$ at node $(i,j)$, where $I=\sqrt{-1}$, $\xi_{l_x,l_y}^{n+1}$ is the amplitude at time level $n+1$, $\beta_x=2\pi l_x$ and $\beta_y=2\pi l_y$, $l_x,l_y\in Z$, the amplification factor $G(\beta_x,\beta_y)=\xi_{l_x,l_y}^{n+1}/\xi_{l_x,l_y}^{n}$, for stability, has satisfied the relation

$|G(\beta_x, \beta_y, \Delta t)| \leqslant 1$, for all $rx, ry \in R$. By substituting the expressions of $\psi_{i,j}^{n+1}$ and $\psi_{i,j}^{n}$ in equation (3.1), the amplification factor is found to be

$$G(\beta_x, \beta_y, \Delta t) = \frac{\left[1 - \left(\frac{1}{3} + 2r_x\right)\sin^2\frac{\beta_x h_x}{2}\right]\left[1 - \left(\frac{1}{3} + 2r_y\right)\sin^2\frac{\beta_y h_y}{2}\right]}{\left[1 - \left(\frac{1}{3} - 2r_x\right)\sin^2\frac{\beta_x h_x}{2}\right]\left[1 - \left(\frac{1}{3} - 2r_y\right)\sin^2\frac{\beta_x h_y}{2}\right]}. \tag{3.2}$$

A simple calculation shows that

$$|G(\beta_x, \beta_y, \Delta t)| \leqslant 1, \qquad \text{for all} \quad r_x, r_y \in R.$$

Hence, we conclude that the scheme (3.1) is unconditionally stable for initial value. So there exists a constant $C$ which is independent of $\Delta t$, $h_x$ and $h_y$ in terms of the definition of stability for initial value such that

$$\left\|u^{n+1} + u_t^{n+1}\right\| \leqslant C\left\|u^0 + u_t^0\right\|, \tag{3.3}$$

where $\|\cdot\|$ is $L^2$-norm. If we define

$$P = \begin{pmatrix} \sigma & 0.5\Delta t\sigma \\ -\sigma & -0.5\Delta t\sigma \end{pmatrix}$$

and use inductive inference, then the expression of $P^{m+1}$ is given by

$$P^{m+1} = \begin{pmatrix} \sigma^{m+1}(1 - 0.5\Delta t)^m & 0.5\Delta t\sigma^{m+1}(1 - 0.5\Delta t)^m \\ -\sigma^{m+1}(1 - 0.5\Delta t)^m & -0.5\Delta t\sigma^{m+1}(1 - 0.5\Delta t)^m \end{pmatrix},$$

where $\sigma = (1+0.5\Delta t)^{-1}$. If we set $U^n = \left(u_{i,j}^n, v_{i,j}^n\right)^T$ and $E = (0.5\Delta t, 1)^T$, then the formula (2.12) can be written as

$$U^{n+1} = PU^n + Ew_{i,j}^{n+1}. \tag{3.4}$$

Using (3.4) repeatedly, we get

$$\begin{aligned} U^{n+1} &= PU^n + Ew_{i,j}^{n+1} \\ &= P\left(PU^{n-1} + Ew_{i,j}^n\right) + Ew_{i,j}^{n+1} \\ &= P^2U^{n-1} + PEw_{i,j}^n + Ew_{i,j}^{n+1} \\ &= \cdots \\ &= P^{n+1}U^0 + \sum_{k=1}^{n} P^{n-k+1}Ew_{i,j}^k + Ew_{i,j}^{n+1}. \end{aligned} \tag{3.5}$$

Using the Schwarz inequality, (3.3) and (3.5), we obtain

$$\begin{aligned} \|u^{n+1}\|^2 + \|v^{n+1}\|^2 &= \sum_{i=1}^{M}\sum_{j=1}^{N}\left(u_{i,j}^{n+1}\right)^2 h_x h_y + \sum_{i=1}^{M}\sum_{j=1}^{N}\left(v_{i,j}^{n+1}\right)^2 h_x h_y \\ &\leqslant \frac{2.5(1 - 0.5\Delta t)^{2n}}{(1 + 0.5\Delta t)^{2(n+1)}}\left\{\sum_{i=1}^{M}\sum_{j=1}^{N}\left(u_{i,j}^0\right)^2 h_x h_y + \sum_{i=1}^{M}\sum_{j=1}^{N}\left(v_{i,j}^0\right)^2 h_x h_y\right\} \\ &\quad + \sum_{k=1}^{n}\left\{\frac{2\Delta t^2(1 - 0.5\Delta t)^{2(n-k)}}{(1 + 0.5\Delta t)^{2(n-k+1)}}\sum_{i=1}^{M}\sum_{j=1}^{N}\left(u_{i,j}^k + v_{i,j}^k\right)^2 h_x h_y\right\} \\ &\quad + (1 + 0.25\Delta t^2)\sum_{i=1}^{M}\sum_{j=1}^{N}\left(u_{i,j}^{n+1} + v_{i,j}^{n+1}\right)^2 h_x h_y \\ &\leqslant C(\|u^0\|^2 + \|v^0\|^2), \end{aligned} \tag{3.6}$$

where $C$ is a generic constant, and is not independent of $\Delta t, h_x$ and $h_y$. Therefore, the new finite difference scheme (2.10)–(2.12) is unconditionally stable for initial value with respect to $\|\cdot\|$. By theorem 3.29 in [17], this scheme is also stable for the right-hand term $H(x,t)$. Hence we have the following result.

**Theorem 2.1.** *Let $u^0$ and $v^0$ be the solutions to the scheme (2.10)–(2.12) at 0 time level, $u^{n+1}$ and $v^{n+1}$ to the scheme (2.10)–(2.12) at $n+1$ time level, then there exist positive constants $\Delta x_0$, $\Delta y_0$ and $\Delta t_0$, such that*

$$\|u^{n+1}\|^2 + \left\|u_t^{n+1}\right\|^2 \leqslant C\left(\|u^0\|^2 + \left\|u_t^0\right\|^2 + \Delta t \sum_{k=0}^{n} \|h^k\|^2\right) \tag{3.7}$$

*for $(n+1)\Delta t \leqslant T, 0 \leqslant h_x \leqslant \Delta x_0, 0 \leqslant h_y \leqslant \Delta y_0$ and $0 \leqslant \Delta t \leqslant \Delta t_0$, where $C$ is independent of $\Delta t, h_x$ and $h_y$, and $k$ denote the time level.*

## 4. Numerical approximation for the nonlinear evolution equation

If we add $u_t$ to both sides of equation (1.1), we have

$$u_{tt} - \Delta u_t - \Delta u + u_t = (f(u)+1)u_t + g(u) + H(x,t). \tag{4.1}$$

Set $u_t = v$ and $F(u,v) = (f(u)+1)v + g(u)$, we can write equation (1.1) as

$$v_t - \Delta v - \Delta u + u_t = F(u,v) + H(x,t). \tag{4.2}$$

For the nonlinear evolution equation, algorithm (2.8) will result in the following equation:

$$\begin{aligned}
&\left[1+\left(\frac{1}{12}-\frac{r_x}{2}\right)\delta_x^2\right]\left[1+\left(\frac{1}{12}-\frac{r_y}{2}\right)\delta_y^2\right]\left(v_{i,j}^{n+1}+u_{i,j}^{n+1}\right)\\
&\quad=\left[1+\left(\frac{1}{12}+\frac{r_x}{2}\right)\delta_x^2\right]\left[1+\left(\frac{1}{12}+\frac{r_y}{2}\right)\delta_y^2\right]\left(v_{i,j}^{n}+u_{i,j}^{n}\right)\\
&\qquad+\frac{\Delta t}{2}\left(1+\frac{\delta_x^2}{12}\right)\left(1+\frac{\delta_y^2}{12}\right)\left(H_{i,j}^{n+1}+H_{i,j}^{n}\right)\\
&\qquad+\frac{\Delta t}{2}\left(1+\frac{\delta_x^2}{12}\right)\left(1+\frac{\delta_y^2}{12}\right)\left(F_{i,j}^{n+1}+F_{i,j}^{n}\right).
\end{aligned} \tag{4.3}$$

Set $\psi_{i,j}^{n+1} = u_{i,j}^{n+1} + v_{i,j}^{n+1}$ and $\psi_{i,j}^{n} = u_{i,j}^{n} + v_{i,j}^{n}$, we have the following algorithm:

$$\begin{aligned}
&\left[1+\left(\frac{1}{12}-\frac{r_x}{2}\right)\delta_x^2\right]\psi_{i,j}^{*}=\left[1+\left(\frac{1}{12}+\frac{r_x}{2}\right)\delta_x^2\right]\left[1+\left(\frac{1}{12}+\frac{r_y}{2}\right)\delta_y^2\right]\psi_{i,j}^{n}\\
&\qquad+\frac{\Delta t}{2}\left(1+\frac{\delta_x^2}{12}\right)\left(1+\frac{\delta_y^2}{12}\right)\left(H_{i,j}^{n+1}+H_{i,j}^{n}\right)\\
&\qquad+\frac{\Delta t}{2}\left(1+\frac{\delta_x^2}{12}\right)\left(1+\frac{\delta_y^2}{12}\right)\left(F_{i,j}^{n+1}+F_{i,j}^{n}\right),
\end{aligned} \tag{4.4}$$

$$\left[1+\left(\frac{1}{12}-\frac{r_y}{2}\right)\delta_y^2\right]\psi_{ij}^{n+1}=\psi_{i,j}^{*}, \tag{4.5}$$

where $r_x = \frac{\Delta t}{h_x^2}$ and $r_y = \frac{\Delta t}{h_y^2}$. Combining with the equation

$$u_{i,j}^{n+1} - \frac{\Delta t}{2}v_{i,j}^{n+1} = \frac{\Delta t}{2}v_{i,j}^{n} + u_{i,j}^{n},$$

we obtain

$$v_{i,j}^{n+1} = \frac{\psi_{i,j}^{n+1} - 0.5\Delta t v_{i,j}^{n} - u_{i,j}^{n}}{1 + 0.5\Delta t} \tag{4.6}$$

and

$$u_{i,j}^{n+1} = \frac{0.5\Delta t \psi_{i,j}^{n+1} + 0.5\Delta t v_{i,j}^{n} + u_{i,j}^{n}}{1 + 0.5\Delta t}. \tag{4.7}$$

Because that equation (4.4) contains solutions $\psi_{i,j}^{n}, \psi_{i,j}^{*}$ and $\psi_{i,j}^{n+1}$ (implicitly in $F_{i,j}^{n+1}$), equation (4.4) cannot be linearized by simply using Newton's method, or its variations to expand $F_{i,j}^{n+1}$ at $\left(u_{i,j}^{n}, v_{i,j}^{n}\right)$.

In [14], an efficient way was used to overcome this difficulty. Here we will introduce this way to deal with the nonlinear term. Note that the algorithm (4.3) can be written as

$$\begin{aligned}
&\left[1 + \left(\frac{1}{12} - \frac{r_x}{2}\right)\delta_x^2\right]\left[1 + \left(\frac{1}{12} - \frac{r_y}{2}\right)\delta_y^2\right]\psi_{i,j}^{n+1} \\
&\quad = \left[1 + \left(\frac{1}{12} + \frac{r_x}{2}\right)\delta_x^2\right]\left[1 + \left(\frac{1}{12} + \frac{r_y}{2}\right)\delta_y^2\right]\psi_{i,j}^{n} \\
&\qquad + \frac{\Delta t}{2}\left(1 + \frac{\delta_x^2}{12}\right)\left(1 + \frac{\delta_y^2}{12}\right)\left(H_{i,j}^{n+1} + H_{i,j}^{n}\right) \\
&\qquad + \frac{\Delta t}{2}\left[1 + \left(\frac{1}{12} + \frac{r_x}{2}\right)\delta_x^2\right]\left(1 + \frac{\delta_y^2}{12}\right)F_{i,j}^{n} \\
&\qquad + \frac{\Delta t}{2}\left[1 + \left(\frac{1}{12} - \frac{r_x}{2}\right)\delta_x^2\right]\left(1 + \frac{\delta_y^2}{12}\right)F_{i,j}^{n+1}.
\end{aligned} \tag{4.8}$$

The difference between (4.3) and (4.8) is

$$\left(1 + \frac{\delta_y^2}{12}\right)\frac{\Delta t}{2}\frac{r_x}{2}\delta_x^2\left(F_{i,j}^{n} - F_{i,j}^{n+1}\right) = O(\Delta t^3) + O(\Delta t^5), \tag{4.9}$$

provided that all relative derivatives in the error estimate are bounded. This is of the same order as the original truncation error in the algorithm (2.5). Therefore, we obtain the following numerical approximation algorithm of equation (1.1):

$$\begin{aligned}
&\left[1 + \left(\frac{1}{12} - \frac{r_x}{2}\right)\delta_x^2\right]\psi_{i,j}^{*} = \left[1 + \left(\frac{1}{12} + \frac{r_x}{2}\right)\delta_x^2\right]\left[1 + \left(\frac{1}{12} + \frac{r_y}{2}\right)\delta_y^2\right]\psi_{i,j}^{n} \\
&\qquad + \frac{\Delta t}{2}\left(1 + \frac{\delta_x^2}{12}\right)\left(1 + \frac{\delta_y^2}{12}\right)\left(H_{i,j}^{n+1} + H_{i,j}^{n}\right) \\
&\qquad + \frac{\Delta t}{2}\left[1 + \left(\frac{1}{12} + \frac{r_x}{2}\right)\delta_x^2\right]\left(1 + \frac{\delta_y^2}{12}\right)F_{i,j}^{n},
\end{aligned} \tag{4.10}$$

$$\left[1 + \left(\frac{1}{12} - \frac{r_y}{2}\right)\delta_y^2\right]\psi_{i,j}^{n+1} = \psi_{i,j}^{*} + \frac{\Delta t}{2}\left(1 + \frac{\delta_y^2}{12}\right)F_{i,j}^{n+1}, \tag{4.11}$$

$$u_{i,j}^{n+1} - \frac{\Delta t}{2}v_{i,j}^{n+1} = \frac{\Delta t}{2}v_{i,j}^{n} + u_{i,j}^{n}. \tag{4.12}$$

With this new formulation, equation (4.10) is linear and can be solved in a straightforward manner. Equations (4.11) and (4.12) are nonlinear. To achieve high accuracy for strongly

nonlinear problems, Newton's iterations can be applied to solve equations (4.11) and (4.12). The initial approximations for Newton's iterations are taken to be $u^n$ and $v^n$ when the solutions to equation (4.11) and (4.12) are calculated at $n+1$ time level, namely, $\left(u_{i,j}^{n+1}\right)^0 = u_{i,j}^n$ and $\left(v_{i,j}^{n+1}\right)^0 = v_{i,j}^n$.

## 5. High-order accuracy in the temporal dimension

The algorithms given in (2.10)–(2.12) and (4.10)–(4.12) are fourth-order accurate in space, but only second-order accurate in time. Because of the special formulae which led to the fourth-order accuracy in space on a seven-point stencil, it is difficult to combine these algorithms with available high-order ODE solution algorithms to obtain better accuracy in the temporal dimension. Following the derivation from (2.2) to (2.8), we can see that the temporal discretization is included in the very beginning of this algorithm development. Consequently, it is difficult to use some of the well-established methods, such as the method of lines (MOL), first to discretize the space derivatives, and then use the high-order ODE time integration methods to get high temporal accuracy.

In [14], the Richardson extrapolation was applied to improve the accuracy in the temporal dimension. Here we also applied the Richardson extrapolation on the computed solution to eliminate the lower order term in the truncation error. As the Crank–Nicolson algorithm has a temporal truncation error in the form of $O(\Delta t^2) + O(\Delta t^3) + O(\Delta t^4)$, we apply

$$u = \frac{4u^{h/2} - u^h}{3} \tag{5.1}$$

and

$$u_t = \frac{4u_t^{h/2} - u_t^h}{3} \tag{5.2}$$

to eliminate the term $O(\Delta t^2)$, where $u^{h/2}$ and $u^h$ are the solutions at the final time level computed using $\Delta t = h$ and $h/2$, respectively. Likewise, $u_t^h$ and $u_t^{h/2}$ are also the solutions at the final time level computed using $\Delta t = h/2$ and $h$, respectively. This makes the final solution fourth-order accurate in both the temporal and spatial dimensions. Even though the extrapolation needs three times as much computation as the original algorithm, the resulting high-order accuracy permits the use of much larger time steps in the computation. For the stability of this case, we can easily obtain to extend the above stability procedure.

## 6. Numerical results

We will give four examples to confirm the theoretical analysis presented in the previous section. These examples with exact solutions against which we can compare the numerical solution prove the efficiency and order of accuracy of the new algorithm in both the spatial and temporal dimensions.

**Example 1.** The equations to be solved are

$$\begin{aligned} &u_{tt} - \Delta u_t - \Delta u + u_t = h(x,t), && (x,t) \in \Omega \times [0,T], \\ &u_t(x,t) = u(x,t) = 0, && (x,t) \in \partial\Omega \times [0,T], \\ &u_t(x,0) = -0.5\sin(\pi x_1)\sin(\pi x_2), && u(x,0) = \sin(\pi x_1)\sin(\pi x_2), \quad x \in \Omega, \end{aligned} \tag{6.1}$$

where $\Omega = [0,1] \times [0,1]$ and $h(x,t) = (\pi^2 - 0.25)\,\mathrm{e}^{-0.5t}\sin(\pi x_1)\sin(\pi x_2)$. The exact solutions of this test problem are $u = \mathrm{e}^{-0.5t}\sin(\pi x_1)\sin(\pi x_2)$ and $u_t = -0.5\,\mathrm{e}^{-0.5t}$

**Table 1.** The maximum absolute error, $l^2$-norm error and maximum relative error at $t = 1.0$, $\Delta t = \Delta x_1 = \Delta x_2 = h$.

| $h$ | 1/4 | 1/5 | 1/8 | 1/10 | 1/16 | 1/20 |
|---|---|---|---|---|---|---|
| $erroru_1$ | $8.816E-003$ | $5.097E-003$ | $2.192E-003$ | $1.401E-003$ | $5.464E-004$ | $3.496E-004$ |
| $erroru_1/h^2$ | 0.141 | 0.127 | 0.140 | 0.140 | 0.139 | 0.140 |
| $Eu_1$ | $4.408E-003$ | $2.818E-003$ | $1.096E-003$ | $7.005E-004$ | $2.732E-004$ | $1.748E-004$ |
| $errorv_1$ | $4.152E-003$ | $2.201E-003$ | $8.641E-004$ | $5.410E-004$ | $2.063E-004$ | $1.314E-004$ |
| $errorv_1/h^2$ | $6.643E-002$ | $5.502E-002$ | $5.530E-002$ | $5.410E-002$ | $5.283E-002$ | $5.255E-002$ |
| $Ev_1$ | $2.076E-003$ | $1.217E-003$ | $4.321E-004$ | $2.705E-004$ | $1.032E-004$ | $6.568E-005$ |
| $Rerroru_1$ | $1.453E-002$ | $9.291E-003$ | $3.614E-003$ | $2.310E-003$ | $9.010E-004$ | $5.763E-004$ |
| $Rerrorv_1$ | $1.369E-002$ | $8.023E-003$ | $2.849E-003$ | $1.784E-003$ | $6.806E-004$ | $4.332E-004$ |
| $erroru_2$ | $4.543E-004$ | $1.642E-004$ | $2.700E-005$ | $1.100E-005$ | $1.661E-006$ | $6.755E-007$ |
| $erroru_2/h^4$ | 0.116 | 0.103 | 0.111 | 0.109 | 0.102 | 0.108 |
| $Eu_2$ | $2.272E-004$ | $9.077E-005$ | $1.350E-005$ | $5.496E-006$ | $8.306E-007$ | $3.378E-007$ |
| $errorv_2$ | $6.869E-005$ | $2.227E-005$ | $3.223E-006$ | $1.271E-006$ | $1.834E-007$ | $7.247E-008$ |
| $errorv_2/h^4$ | $1.758E-002$ | $1.271E-002$ | $1.392E-002$ | $1.320E-002$ | $1.202E-002$ | $1.159E-002$ |
| $Ev_2$ | $3.435E-005$ | $1.231E-005$ | $1.611E-006$ | $6.353E-007$ | $9.171E-008$ | $3.623E-008$ |
| $Rerroru_2$ | $7.491E-004$ | $2.993E-004$ | $4.451E-005$ | $1.812E-005$ | $2.739E-006$ | $1.114E-006$ |
| $Rerrorv_2$ | $2.265E-004$ | $8.120E-005$ | $1.063E-005$ | $4.200E-006$ | $6.048E-007$ | $2.390E-007$ |

**Table 2.** The maximum absolute error, $l^2$-norm error and maximum relative error at $t = 1.0$, $\Delta t = (\Delta x_1)^2 = (\Delta x_2)^2 = h^2$.

| $h$ | 1/4 | 1/5 | 1/8 | 1/10 | 1/16 | 1/20 |
|---|---|---|---|---|---|---|
| $\Delta t$ | 1/16 | 1/25 | 1/64 | 1/100 | 1/256 | 1/400 |
| $erroru_3$ | $9.250E-004$ | $3.420E-004$ | $5.751E-005$ | $2.353E-005$ | $3.582E-006$ | $1.463E-006$ |
| $erroru_3/h^4$ | 0.214 | 0.237 | 0.236 | 0.235 | 0.235 | 0.234 |
| $Eu_3$ | $4.625E-004$ | $1.890E-004$ | $2.875E-005$ | $1.177E-005$ | $1.791E-006$ | $7.314E-007$ |
| $errorv_3$ | $3.316E-004$ | $1.215E-004$ | $2.034E-005$ | $8.321E-006$ | $1.271E-006$ | $5.224E-007$ |
| $Ev_3$ | $1.658E-004$ | $6.717E-005$ | $1.017E-005$ | $4.161E-006$ | $6.355E-007$ | $2.612E-007$ |
| $errorv_3/h^4$ | $8.488-002$ | $7.594E-002$ | $8.330E-002$ | $8.321E-002$ | $8.330E-002$ | $8.359E-002$ |
| $Rerroru_3$ | $1.093E-003$ | $4.430E-004$ | $6.706E-005$ | $3.880E-005$ | $4.191E-006$ | $2.720E-007$ |
| $Rerrorv_3$ | $1.525E-003$ | $6.233E-004$ | $9.481E-005$ | $2.744E-005$ | $5.905E-006$ | $3.585E-007$ |

$\sin(\pi x_1)\sin(\pi x_2)$. The data in table 1 show the maximum absolute error, $l^2$-norm error and maximum relative error between the calculated solution and the exact solution at $t = 1.0$. The difference scheme given by (2.10)–(2.12) was used. We denote the maximum absolute error of solutions $u$ and $v$ from the algorithm which is second-order accurate in time and fourth-order accurate in space by $erroru_1$ and $errorv_1$, respectively. We use $Eu_1$ and $Ev_1$ to denote the $l^2$-norm error of solutions $u$ and $v$ from the algorithm which is second-order accurate in time and fourth-order accurate in space, respectively. The notations $Rerroru_1$ and $Rerrorv_1$ represent the maximum relative error of solutions $u$ and $v$ from the algorithm that is second-order accurate in time and fourth-order accurate in space, respectively. Likewise, we denote the maximum absolute error of solutions $u$ and $v$ from the algorithm which is fourth-order accurate in both the temporal and spatial dimensions by $erroru_2$ and $errorv_2$, respectively. The notations $Eu_2$ and $Ev_2$ represent the $l^2$-norm error of solutions $u$ and $v$ from the algorithm which is fourth-order accurate in both time and space, respectively. The notations $Rerroru_2$ and $Rerrorv_2$ represent the maximum relative error of solutions $u$ and $v$ from the algorithm which is fourth-order accurate in both time and space, respectively. The data in table 2 show the maximum absolute error, the $l^2$-norm error and the maximum relative

error between the calculated solution and the exact solution at $t = 1.0$. The algorithm given by (2.10)–(2.12) was used. The notations $erroru_3$ and $errorv_3$ represent the maximum absolute error of solutions $u$ and $v$ from the algorithm which is second-order accurate in time and fourth-order accurate in space, respectively. We denote $l^2$-norm error of solutions $u$ and $v$ from the algorithm which is second-order accurate in the temporal dimension and fourth-order accurate in spatial dimension by $Eu_3$ and $Ev_3$, respectively. The notations $Rerroru_3$ and $Rerrorv_3$ represent the maximum relative error of $u$ and $v$ from the algorithm which is second-order accurate in time and fourth-order accurate in space, respectively.

Obviously, we can see from table 1 that the error denoted by $erroru_1$ and $errorv_1$ shows a second-order decrease, while that denoted by $erroru_2$ and $errorv_2$ shows a fourth-order decrease. This is illustrated by the fact that the ratios of $erroru_1/h^2$, $errorv_1/h^2$, $erroru_2/h^4$ and $errorv_2/h^4$ keep roughly a constant as the computational grid is being refined. Each time when the computation grid is refined by having $\Delta t$, $\Delta x_1$ and $\Delta x_2$, $erroru_1$ and $errorv_1$ are reduced by a factor of 4 while $erroru_2$ and $errorv_2$ are reduced by a factor of 16. It is also noted that for the same value of $h$, the $l^2$-norm errors denoted by $Eu_2$ and $Ev_2$ and the maximum relative errors represented by $Rerroru_2$ and $Rerrorv_2$ are much smaller than those denoted by $Eu_1$, $Ev_1$, $Rerroru_1$ and $Rerrorv_1$, respectively. Table 2 shows similar results as those represented by $erroru_1$ and $errorv_1$. Setting $\Delta t = (\Delta x_1)^2$ and $\Delta x_1 = \Delta x_2 = h$, we can see from table 2 that $erroru_3$ and $errorv_3$ are being reduced by a factor of 16 when $h$ is decayed by a factor of 2 each time, the ratios of $erroru_3/h^4$ and $errorv_3/h^4$ remain roughly a constant as the computational grid is refined, indicating fourth-order convergence. However, the accuracy is still not as good as those represented by $erroru_2$ and $errorv_2$. It is also observed that for the same value of $h$, the $l^2$-norm errors denoted by $Eu_3$ and $Ev_3$ are also slightly larger than those represented by $Eu_2$ and $Ev_2$, respectively.

**Example 2.** This is a simple example of a two-dimensional linear equation. The equation to be solved is

$$\begin{aligned} &u_{tt} - \Delta u_t - \Delta u = u + h(x,t), && (x,t) \in \Omega \times [0,T], \\ &u_t(x,t) = u(x,t) = 0, && (x,t) \in \partial\Omega \times [0,T], \\ &u_t(x,0) = -\sin(\pi x_1)\sin(\pi x_2), && u(x,0) = \sin(\pi x_1)\sin(\pi x_2), \quad x \in \Omega, \end{aligned} \tag{6.2}$$

where $\Omega = [0,1] \times [0,1]$ and $h(x,t) = (\pi^2 - 0.75)\,\mathrm{e}^{-0.5t}\sin\pi x_1 \sin(\pi x_2)$. The exact solutions of this test problem are $u = \mathrm{e}^{-0.5t}\sin(\pi x_1)\sin(\pi x_2)$ and $u_t = -0.5\,\mathrm{e}^{-0.5t}\sin(\pi x_1)\sin(\pi x_2)$.

The linear case is the special nonlinear case, in order to test the accuracy of scheme (4.10)–(4.12), we use the linear case here. This can make us to avoid to use the Newton method, otherwise it will produce extra errors. It seems so hard to see the accuracy of the scheme. In the following example, we will give a nonlinear case.

The data in table 3 show the maximum absolute error, $L^2$-norm error and the maximum relative error between the numerical solution and the analytical solution at $t = 1.0$. The algorithm given by (4.9)–(4.11) was applied. The notations $erroru_4$ and $errorv_4$ represent the maximum absolute error of solutions $u$ and $v$ from the algorithm which is second-order accurate in time and fourth-order accurate in space, respectively. The notations $erroru_4$ and $errorv_4$ represent the maximum absolute error of solutions $u$ and $v$ from the algorithm which is second-order accurate in time and fourth-order accurate in space, respectively. $Eu_4$ and $Ev_4$ represent the $l^2$-norm error of solutions $u$ and $v$ from the algorithm which is second-order accurate in time and fourth-order accurate in space, respectively. We denote the maximum relative error of solutions $u$ and $v$ from the algorithm which is second-order accurate in time and

**Table 3.** The maximum absolute error and maximum relative error at $t = 1.0$, $\Delta t = \Delta x_1 = \Delta x_2 = h$.

| $h$ | 1/4 | 1/5 | 1/8 | 1/10 | 1/16 | 1/20 |
|---|---|---|---|---|---|---|
| $erroru_4$ | $8.374E-003$ | $4.829E-003$ | $2.072E-003$ | $1.323E-003$ | $5.159E-004$ | $3.299E-004$ |
| $erroru_4/h^2$ | 0.134 | 0.121 | 0.133 | 0.132 | 0.132 | 0.132 |
| $Eu_4$ | $4.187E-003$ | $2.670E-003$ | $1.036E-003$ | $6.617E-004$ | $2.579E-004$ | $1.650E-004$ |
| $errorv_4$ | $3.946E-003$ | $2.099E-003$ | $8.270E-004$ | $5.182E-004$ | $1.979E-004$ | $1.260E-004$ |
| $errorv_4/h^2$ | $6.314E-002$ | $5.247E-002$ | $5.293E-002$ | $5.128E-002$ | $5.065E-002$ | $5.039E-002$ |
| $Ev_4$ | $1.973E-003$ | $1.160E-003$ | $4.135E-004$ | $2.591E-004$ | $9.893E-005$ | $6.292E-005$ |
| $Rerroru_4$ | $1.381E-002$ | $8.803E-003$ | $3.416E-003$ | $2.182E-003$ | $8.504E-004$ | $5.440E-004$ |
| $Rerrorv_4$ | $1.301E-002$ | $7.651E-003$ | $2.727E-003$ | $1.709E-003$ | $6.524E-004$ | $4.154E-004$ |
| $erroru_5$ | $4.661E-004$ | $1.689E-004$ | $2.785E-005$ | $1.135E-005$ | $1.716E-006$ | $6.981E-007$ |
| $erroru_5/h^4$ | 0.119 | 0.105 | 0.114 | 0.113 | 0.112 | 0.106 |
| $Eu_5$ | $2.331E-004$ | $9.338E-005$ | $1.393E-005$ | $5.673E-006$ | $8.581E-007$ | $3.491E-007$ |
| $errorv_5$ | $4.000E-005$ | $1.223E-005$ | $1.625E-006$ | $6.248E-007$ | $8.625E-008$ | $3.250E-008$ |
| $errorv_5/h^4$ | $1.024E-002$ | $7.643E-003$ | $6.657E-003$ | $6.248E-003$ | $5.653E-003$ | $5.248E-003$ |
| $Ev_5$ | $2.000E-005$ | $6.760E-006$ | $8.126E-007$ | $3.124E-007$ | $4.313E-008$ | $1.640E-008$ |
| $Rerroru_5$ | $7.685E-004$ | $3.079E-004$ | $4.592E-005$ | $1.871E-005$ | $2.830E-006$ | $1.151E-006$ |
| $Rerrorv_5$ | $1.319E-004$ | $4.458E-005$ | $5.359E-006$ | $2.060E-006$ | $2.844E-007$ | $1.082E-007$ |

**Table 4.** The maximum absolute error and maximum relative error at $t = 1.0$, $\Delta t = (\Delta x_1)^2 = (\Delta x_2)^2 = h^2$.

| $h$ | 1/4 | 1/5 | 1/8 | 1/10 | 1/16 | 1/20 |
|---|---|---|---|---|---|---|
| $\Delta t$ | 1/16 | 1/25 | 1/64 | 1/100 | 1/256 | 1/400 |
| $erroru_6$ | $9.143E-004$ | $3.379E-004$ | $5.681E-005$ | $2.324E-005$ | $3.538E-006$ | $1.445E-006$ |
| $erroru_6/h^4$ | 0.234 | 0.211 | 0.233 | 0.232 | 0.232 | 0.231 |
| $Eu_6$ | $4.572E-004$ | $1.868E-004$ | $2.840E-005$ | $1.162E-005$ | $1.769E-006$ | $7.223E-007$ |
| $errorv_6$ | $3.308E-004$ | $1.213E-004$ | $2.030E-005$ | $8.303E-006$ | $1.268E-006$ | $5.213E-007$ |
| $errorv_6/h^4$ | $8.468-002$ | $7.579E-002$ | $8.313E-002$ | $8.303E-002$ | $8.312E-002$ | $8.341E-002$ |
| $Ev_6$ | $1.654E-004$ | $6.703E-004$ | $1.015E-005$ | $4.152E-006$ | $6.342E-007$ | $2.607E-007$ |
| $Rerroru_6$ | $1.507E-003$ | $6.159E-004$ | $9.366E-005$ | $3.832E-005$ | $5.832E-006$ | $2.382E-006$ |
| $Rerrorv_6$ | $1.091E-003$ | $4.421E-004$ | $6.692E-005$ | $2.738E-005$ | $4.182E-006$ | $1.719E-006$ |

fourth-order accurate in space by $Rerroru_4$ and $Rerrorv_4$, respectively. The notations $erroru_5$ and $errorv_5$ represent the maximum absolute error of solutions $u$ and $v$ from the algorithm which is fourth-order accurate in both time and space, respectively. The notations $Eu_5$ and $Ev_5$ denote the $l^2$-norm error of solutions $u$ and $v$ from the algorithm which is fourth-order accurate in both time and space, respectively. We denote the maximum relative error of $u$ and $v$ from the algorithm which is fourth-order accurate in both time and space by $Rerroru_5$ and $Rerrorv_5$, respectively. The data in table 4 show the maximum absolute error, the $l^2$-norm error and the maximum relative error between the calculated solution and the exact solution at $t = 1.0$. The notations $erroru_6$ and $errorv_6$ represent the maximum absolute error of $u$ and $v$ from the algorithm which is second-order accurate in time and fourth-order accurate in space, respectively. We use $Eu_6$ and $Ev_6$ to denote the $l^2$-norm error of $u$ and $v$ from the algorithm which is second-order accurate in time and fourth-order accurate in space, respectively. The notations $Rerroru_6$ and $Rerrorv_6$ represent the maximum relative error of solutions $u$ and $v$ from the algorithm which is second-order accurate in time and fourth-order accurate in space, respectively.

It is clear from table 3 that the error denoted by $erroru_4$ and $errorv_4$ shows a second-order decrease, while that denoted by $erroru_5$ and $errorv_5$ shows a fourth-order decrease. This is

**Table 5.** The maximum absolute error, $L^2$-norm error and maximum relative error at $t = 1.0$, CPU times in seconds, $\Delta t = \Delta x_1 = \Delta x_2 = h$.

| $h$ | 1/4 | 1/5 | 1/8 | 1/10 | 1/16 | 1/20 |
|---|---|---|---|---|---|---|
| $erroru_7$ | $1.120E-003$ | $6.574E-004$ | $2.876E-004$ | $1.844E-004$ | $7.198E-005$ | $1.591E-005$ |
| $Eu_7$ | $5.573E-004$ | $3.624E-004$ | $1.431E-004$ | $9.176E-005$ | $3.582E-005$ | $2.285E-005$ |
| $errorv_7$ | $2.308E-003$ | $1.276E-003$ | $5.243E-004$ | $3.319E-004$ | $1.281E-004$ | $8.179E-005$ |
| $Ev_7$ | $1.158E-003$ | $7.061E-004$ | $2.625E-004$ | $1.662E-004$ | $6.414E-005$ | $4.094E-005$ |
| $Rerroru_7$ | $3.044E-003$ | $1.976E-003$ | $7.818E-004$ | $5.012E-004$ | $1.957E-004$ | $1.248E-004$ |
| $Rerrorv_7$ | $4.311E-003$ | $3.851E-003$ | $1.431E-003$ | $9.059E-004$ | $3.497E-004$ | $2.232E-004$ |
| $time_7$ | 0.000 | 0.000 | 0.000 | 0.010 | 0.060 | 0.17 |
| $erroru_8$ | $1.057E-005$ | $2.916E-006$ | $1.037E-007$ | $2.446E-007$ | $8.604E-007$ | $1.320E-006$ |
| $Eu_8$ | $5.273E-006$ | $1.589E-006$ | $4.374E-008$ | $1.260E-007$ | $4.298E-007$ | $6.590E-007$ |
| $errorv_8$ | $7.046E-005$ | $2.476E-005$ | $3.899E-006$ | $1.574E-006$ | $2.450E-007$ | $1.409E-007$ |
| $Ev_8$ | $3.591E-005$ | $1.384E-005$ | $1.975E-006$ | $7.969E-007$ | $1.243E-007$ | $5.819E-008$ |
| $Rerroru_8$ | $2.879E-005$ | $8.765E-006$ | $2.818E-007$ | $7.525E-007$ | $2.342E-006$ | $3.589E-006$ |
| $Rerrorv_8$ | $1.992E-004$ | $7.747E-005$ | $1.124E-005$ | $4.553E-006$ | $7.164E-007$ | $3.374E-007$ |
| $time_8$ | 0.000 | 0.000 | 0.010 | 0.030 | 0.220 | 0.580 |

proved by the fact that the ratios of $erroru_4/h^2$, $errorv_4/h^2$, $erroru_5/h^4$ and $errorv_5/h^4$ keep roughly a constant as computational grid is being refined. Each time when the computational grid is refined by having $\Delta t$, $\Delta x_1$ and $\Delta x_2$, $erroru_4$ and $errorv_4$ are reduced by a factor of 4, while $erroru_5$ and $errorv_5$ are reduced by a factor of 16. It is also noted that for the same value of $h$, the $l^2$-norm errors denoted by $Eu_5$ and $Ev_5$ and the maximum relative errors represented by $Rerroru_5$ and $Rerroru_5$ are much smaller than those denoted by $Eu_4$, $Ev_4$, $Rerroru_4$ and $Rerroru_4$, respectively. Table 4 shows analogous results as those denoted by $erroru_4$ and $errorv_4$ in table 3. Setting $\Delta t = (\Delta x_1)^2$ and $\Delta x_1 = \Delta x_2 = h$, we can see from table 4 that $erroru_6$ and $errorv_6$ are now being reduced by a factor of 16 when $h$ is reduced by a factor of 2 each time, and the ratios of $erroru_6/h^4$ and $errorv_6/h^4$ keep approximately a constant as the computational grid is refined, implying fourth-order convergence. Meanwhile, we find that accuracy is not better than those presented by $erroru_5$, $errorv_5$, respectively. We also note that the $l^2$-norm errors denoted by $Eu_6$ and $Ev_6$ are also not smaller than those represented by $Eu_5$ and $Ev_5$, respectively.

**Example 3.** This is an example of a two-dimensional nonlinear equation. The equation to be solved is

$$\begin{aligned} &u_{tt} - \Delta u_t - \Delta u = uu_t + u^2 + h(x,t), && (x,t) \in \Omega \times [0,T], \\ &u_t(x,t) = u(x,t) = 0, && (x,t) \in \partial\Omega \times [0,T], \\ &u_t(x,0) = -\sin(\pi x_1)\sin(\pi x_2), && u(x,0) = \sin(\pi x_1)\sin(\pi x_2), \quad x \in \Omega, \end{aligned} \tag{6.3}$$

where $\Omega = [0,1] \times [0,1]$ and $h(x,t) = \mathrm{e}^{-t}\sin(\pi x_1)\sin(\pi x_2)$. The exact solutions of this test problem are $u = \mathrm{e}^{-t}\sin(\pi x_1)\sin(\pi x_2)$ and $u_t = -\mathrm{e}^{-t}\sin(\pi x_1)\sin(\pi x_2)$.

The algorithm given by (4.9)–(4.11) and quasi-Newton iterative method was used. The initial approximations for the quasi-Newton iterative method were taken to be $u^n$ and $v^n$ when the solutions to equations (4.9)–(4.11) are calculated at time level $n+1$, namely, $\left(u_{i,j}^{n+1}\right)^0 = u_{i,j}^n$ and $\left(v_{i,j}^{n+1}\right)^0 = v_{i,j}^n$. The iterations were terminated when the residual in maximum norm was reduced by a factor of $10^{10}$.

The data in table 5 show the maximum absolute error, the $l^2$-norm error and the maximum relative error between the calculated solution and the exact solution at $t = 1.0$ and CPU times in seconds. We denote the maximum absolute error of solutions $u$ and $v$ from the algorithm which

**Table 6.** The maximum absolute error, $L^2$-norm error and maximum relative error at $t = 1.0$, CPU times in seconds, $\Delta t = (\Delta x_1)^2$, $\Delta x_1 = \Delta x_2 = h$.

| $h$ | 1/4 | 1/5 | 1/8 | 1/10 | 1/16 | 1/20 |
|---|---|---|---|---|---|---|
| $erroru_9$ | $7.203E-005$ | $2.628E-005$ | $1.819E-006$ | $4.458E-006$ | $3.917E-006$ | $1.591E-005$ |
| $Eu_9$ | $3.584E-005$ | $1.448E-005$ | $9.017E-007$ | $2.227E-006$ | $1.055E-006$ | $7.750E-006$ |
| $errorv_9$ | $1.281E-004$ | $4.728E-005$ | $7.907E-006$ | $3.133E-006$ | $1.281E-007$ | $1.860E-006$ |
| $Ev_9$ | $6.414E-005$ | $2.616E-005$ | $3.957E-006$ | $1.566E-006$ | $1.867E-007$ | $9.679E-007$ |
| $Rerroru_9$ | $1.958E-004$ | $7.898E-005$ | $4.945E-006$ | $1.212E-005$ | $1.957E-005$ | $2.587E-004$ |
| $Rerrorv_9$ | $3.491E-004$ | $1.424E-004$ | $2.154E-005$ | $8.518E-006$ | $1.210E-006$ | $5.888E-006$ |
| $time_7$ | 0.000 | 0.000 | 0.030 | 0.080 | 0.93 | 3.14 |

is second-order accurate in time and fourth-order accurate in space by $erroru_7$ and $errorv_7$, respectively. $Eu_7$ and $Ev_7$ denote the $l^2$-norm error of solutions $u$ and $v$ from the algorithm which is second-order accurate in time and fourth-order accurate in space, respectively. The notations $Rerroru_7$ and $Rerrorv_7$ represent the maximum relative error of solutions $u$ and $v$ from the algorithm which is second-order accurate in time and fourth-order accurate in space, respectively. The notation $time_7$ denote the total elapsed time (CPU) in seconds delivered from the algorithm which is second-order accurate in time and fourth-order accurate in space. Likewise, we denote the maximum absolute error of solutions $u$ and $v$ from the algorithm which is fourth-order accurate in both time and space by $erroru_8$ and $errorv_8$, respectively. $Eu_8$ and $Ev_8$ denote the $l^2$-norm error of solutions $u$ and $v$ from the algorithm which is fourth-order accurate in both time and space, respectively. The notations $Rerroru_8$ and $Rerrorv_8$ represent, respectively, the maximum relative error of solutions $u$ and $v$ from the algorithm which is fourth-order accurate in both time and space. We use $time_8$ to denote the total elapsed time (CPU) in seconds delivered from the algorithm which is fourth-order accurate in both time and space. The data in table 6 show the maximum absolute error, the $l^2$-norm error and the maximum relative error between the calculated solution and the exact solution at $t = 1.0$ and CPU times in seconds. We denote the maximum absolute error of solutions $u$ and $v$ from the algorithm which is second-order accurate in time and fourth-order accurate in space by $erroru_9$ and $errorv_9$, respectively. We denote the $l^2$-norm error of solutions $u$ and $v$ from the algorithm which is second-order accurate in time and fourth-order accurate in space by $Eu_9$ and $Ev_9$, respectively. The notations $Rerroru_9$ and $Rerrorv_9$ represent the maximum relative error of solutions $u$ and $v$ from the algorithm which is second-order accurate in time and fourth-order accurate in space, respectively. The notation $time_9$ denote the total elapsed time (CPU) in seconds delivered from the algorithm which is second-order accurate in time and fourth-order accurate in space.

We can see from table 5 that for the same value of $h$, the maximum absolute errors denoted by $erroru_8$ and $errorv_8$ are much smaller than those denoted by $erroru_7$ and $errorv_7$, respectively, and the $l^2$-norm errors represented by $Eu_8$ and $Ev_8$ are much smaller than those denoted by $Eu_7$ and $Ev_7$, respectively, although time consumption represented by $time_8$ is a little more than the one denoted by $time_7$. It is also noted from tables 5 and 6 that for the same value of $h$, the accuracy denoted by $erroru_9$, $errorv_9$, $Eu_9$ and $Ev_9$ is not as good as the one represented by $erroru_8$, $errorv_8$, $Eu_8$ and $Ev_8$, while time consumption represented by $time_9$ is more than the one denoted by $time_8$, especially as the meshsize $h$ is very small, for example $h = 0.05$; and the grid size reduces the maximum relative error of the solution decreases as well. As above, the numerical method combining the difference scheme (4.9)–(4.11) with the Richardson extrapolation technique is highly efficient for solving this kind of nonlinear evolution equation.

**Table 7.** The maximum absolute error and $l^2$-norm error at $t = 1.0$, $\Delta t = \Delta x_1 = \Delta x_2 = h$.

| $h$ | 1/4 | 1/5 | 1/8 | 1/10 | 1/16 | 1/20 |
|---|---|---|---|---|---|---|
| $erroru_{10}$ | $7.091E-002$ | $4.685E-002$ | $1.945E-002$ | $1.237E-002$ | $4.797E-003$ | $3.064E-003$ |
| $Eu_{10}$ | $4.252E-002$ | $2.662E-002$ | $1.012E-002$ | $6.437E-003$ | $2.496E-003$ | $1.594E-003$ |
| $errorv_{10}$ | $9.251E-002$ | $7.024E-002$ | $2.838E-002$ | $1.794E-002$ | $6.917E-003$ | $4.413E-003$ |
| $Ev_{10}$ | $6.553E-002$ | $3.994E-002$ | $1.479E-002$ | $9.350E-003$ | $3.603E-003$ | $2.298E-003$ |
| $erroru_{11}$ | $3.253E-003$ | $1.154E-003$ | $1.876E-004$ | $7.514E-005$ | $8.267E-006$ | $4.179E-007$ |
| $Eu_{11}$ | $1.580E-003$ | $6.240E-004$ | $9.135E-005$ | $3.660E-005$ | $3.991E-006$ | $2.509E-007$ |
| $errorv_{11}$ | $2.848E-003$ | $1.102E-003$ | $1.717E-004$ | 6.859-005 | $5.799E-006$ | $3.096E-006$ |
| $Ev_{11}$ | $1.296E-003$ | $5.283E-004$ | $7.980E-005$ | 3.191-005 | $2.569E-006$ | $1.736E-006$ |

**Table 8.** The maximum absolute error and at $t = 1.0$, $\Delta t = \Delta x_1 = \Delta x_2 = h$.

| $h$ | 1/4 | 1/8 | 1/10 | 1/16 | 1/20 | 1/32 |
|---|---|---|---|---|---|---|
| $erroru_{10}$ | 1.106 | 1.257 | 1.264 | 1.263 | 1.260 | 1.253 |
| $errorv_{10}$ | 1.336 | 1.263 | 1.236 | 1.189 | 1.181 | 1.226 |
| $erroru_{11}$ | 1.056 | 1.238 | 1.251 | 1.258 | 1.257 | 1.252 |
| $errorv_{11}$ | 1.111 | 1.229 | 1.218 | 1.183 | 1.179 | 1.225 |

**Example 4.** In this example, first we test our new scheme for continuous case of the following nonlinear equation, then for discontinuous case of the nonlinear equation

$$\begin{aligned} &u_{tt} - \Delta u_t - \Delta u = 0.5u_t + 3\pi^2 u - u^2 + h(x,t), \qquad (x,t) \in \Omega \times [0,T], \\ &u_t(x,t) = u(x,t) = 0, \qquad (x,t) \in \partial\Omega \times [0,T], \\ &u_t(x,0) = 0.5\sin(\pi x_1)\sin(\pi x_2), \qquad u(x,0) = \sin(\pi x_1)\sin(\pi x_2), \quad x \in \Omega, \end{aligned} \tag{6.4}$$

where $\Omega = [0,1] \times [0,1]$ and $h(x,t) = \mathrm{e}^t(\sin(\pi x_1)\sin(\pi x_2)^2$. The exact solution of this problem is $u = \mathrm{e}^{0.5t}\sin(\pi x_1)\sin(\pi x_2)$. In tables 7 and 8, we denote the maximum absolute error of solutions $u$ and $v$ from the method which we apply directly to the formulae (4.9)–(4.11) to solve this problem by $erroru_{10}$ and $errorv_{10}$, respectively. The notations $Eu_{10}$ and $Ev_{10}$ denote the $l^2$-norm error of solutions $u$ and $v$ from the algorithm which we use directly to the formulae (4.9)–(4.11). The notations $erroru_{11}$ and $errorv_{11}$ represent the maximum absolute error of solutions $u$ and $v$ from the algorithm which we use to the formulae (4.9)–(4.11) and the Richardson extrapolation technique to solve this problem. $Eu_{11}$ and $Ev_{11}$ represent the $l^2$-norm error of solutions $u$ and $v$ from the algorithm which we use to the formulae (4.9)–(4.11) and the Richardson extrapolation technique to solve this problem, respectively. In table 9, we denote the absolute error of solutions $u$ and $v$ at the mesh point $(i, j)$ from the method which we apply directly to the formulae (4.9)–(4.11) to solve the discontinuous problem by $erroru_{12}$ and $errorv_{12}$, respectively. The notations $erroru_{13}$ and $erroru_{13}$ represent the absolute error of solutions $u$ and $v$ at the mesh point $(i, j)$ from the algorithm which we use to the formulae (4.9)–(4.11) and the Richardson extrapolation technique to solve this discontinuous problem. Similarly with example 3, we obtain the good calculation results for this continuous test problem in table 7.

Our numerical schemes assume that the solution is smooth. Otherwise the accuracy will deteriorate. If we choose discontinuous initial conditions,

$$\begin{aligned} &u(x,0) = -\sin(\pi x_1)\sin(\pi x_2), \qquad u_t(x,0) = -0.5\sin(\pi x_1)\sin(\pi x_2), \\ &\qquad (x_1,x_2) \in [0,0.5] \times [0.1] \end{aligned} \tag{6.5}$$

$$\begin{aligned} &u(x,0) = \sin(\pi x_1)\sin(\pi x_2), \qquad u_t(x,0) = 0.5\sin(\pi x_1)\sin(\pi x_2) \\ &\qquad (x_1,x_2) \in (0.5,1] \times [0,1] \end{aligned} \tag{6.6}$$

**Table 9.** The absolute error at the mesh point $(i, j)$ at $t = 1.0$, $\Delta t = \Delta x_1 = \Delta x_2 = 0.0625$.

| $(x_1, x_2)$ | $erroru_{12}$ | $errorv_{12}$ | $erroru_{13}$ | $errorv_{13}$ | $(x_1, x_2)$ | $erroru_{12}$ | $errorv_{12}$ | $erroru_{13}$ | $errorv_{13}$ |
|---|---|---|---|---|---|---|---|---|---|
| $(\frac{1}{16}, \frac{1}{2})$ | 1.133 | 0.179 | 0.133 | 0.179 | $(\frac{9}{16}, \frac{1}{2})$ | 1.168 | 1.099 | 1.163 | 1.094 |
| $(\frac{1}{8}, \frac{1}{2})$ | 0.166 | 0.355 | 0.267 | 0.354 | $(\frac{5}{8}, \frac{1}{2})$ | 1.001 | 1.011 | 0.999 | 1.005 |
| $(\frac{3}{16}, \frac{1}{2})$ | 0.399 | 0.522 | 0.401 | 0.521 | $(\frac{11}{16}, \frac{1}{2})$ | 0.885 | 0.889 | 0.833 | 0.885 |
| $(\frac{1}{4}, \frac{1}{2})$ | 0.533 | 0.677 | 0.534 | 0.675 | $(\frac{3}{4}, \frac{1}{2})$ | 0.666 | 0.742 | 0.666 | 0.738 |
| $(\frac{5}{16}, \frac{1}{2})$ | 0.666 | 0.815 | 0.667 | 0.812 | $(\frac{13}{16}, \frac{1}{2})$ | 0.499 | 0.574 | 0.499 | 0.606 |
| $(\frac{3}{8}, \frac{1}{2})$ | 0.798 | 0.928 | 0.798 | 0.926 | $(\frac{7}{8}, \frac{1}{2})$ | 0.333 | 0.391 | 0.333 | 0.413 |
| $(\frac{7}{16}, \frac{1}{2})$ | 0.927 | 1.013 | 0.925 | 1.010 | $(\frac{15}{16}, \frac{1}{2})$ | 0.177 | 0.198 | 0.166 | 0.197 |
| $(\frac{1}{2}, \frac{1}{2})$ | 1.052 | 1.063 | 1.048 | 1.059 | $(1, \frac{1}{2})$ | 0.000 | 0.000 | 0.000 | 0.000 |

and obtain the true solution of the test problem:

$$u(x, 0) = -e^{0.5t} \sin(\pi x_1) \sin(\pi x_2), \qquad (x_1, x_2) \in [0, 0.5] \times [0.1] \tag{6.7}$$

$$u(x, 0) = e^{0.5t} \sin(\pi x_1) \sin(\pi x_2), \qquad (x_1, x_2) \in (0.5, 1] \times [0, 1]. \tag{6.8}$$

We will obtain the following terrible numerical results in tables 8 and 9.

## 7. Summary

An efficient numerical method based on the Pade' approximation and Richardson extrapolation for solving the 2D generalized nerve conduction equation is discussed in this paper. It is fourth-order accurate in both time and space, uses a compact five-point finite difference stencil similar to that applied in the standard second-order algorithm, such as the Crank–Nicolson, and allows a considerable saving in computing time. The method is easily extendible to multi-dimensional problems. Our test problems involving both the linear term and the nonlinear term are conducted to demonstrate the high-order accuracy in both temporal and spatial dimensions. For the continuous case, our theoretical analysis and numerical results indicate the new method has good numerical stability and high efficiency. In examples 1–3, for two cases of $\Delta t = h$ and $\Delta t = h^2$, the results of $l^2$- and $l_\infty$-norms are shown in tables 1–6. For every case in examples 1–3, with the spacial step $h$ decreasing, the errors in $l^2$- and $l_\infty$-norms are decreasing. Convergence orders coincide with the ones of the theoretical analysis. On the other hand, although the computational grid size is refined (i.e. for the case of $\Delta t = h^2$), the accuracy of the algorithm in which the Richardson extrapolation is not used is not as good as the one of the algorithms in which the Richardson extrapolation is used; we also find that the computational cost in the former case is larger than that in the latter case. In addition, we test the discontinuous case in example 4, the results show that the new scheme is not suitable for this case, because our numerical schemes assume that the solution is smooth.

In a word, the difference scheme given by (2.10)–(2.12) and (4.9)–(4.12) are both second-order accurate in time and fourth-order accurate in space, respectively. Numerical results which we have provided indicate the accuracy in the temporal dimension is indeed increased to fourth-order accurate by using the Richardson extrapolation, and these methods have good numerical stability and their time cost is cheap, especially our new numerical method is very efficient for solving this kind of nonlinear continuous problems. However, the numerical results from example 4 demonstrate the new numerical algorithm is not suitable for solving

discontinuous problem. Since many models in science and engineering have discontinuous solutions, we will work on extending this new method to the problems and find better numerical method to solve this kind of discontinuous problems.

## Acknowledgments

The authors are grateful to Professor Zhilin Li for his many help. The second author also would like to thank Professor Charlie Elliott and Professor Kewei Zhang for their kindness to provide wonderful work facilities when he visits the University of Sussex.

## References

[1] Pao C V 1985 An mixed initial boundary value problem arising in neurophysiology *J. Math. Anal. Appl.* **52** 105–19
[2] Ponce G 1985 Global existence of small solutions to a class of nonlinear evolution equations *Nonlinear Anal.* **9** 399–418
[3] Liu Y C and Yu T 1995 Blow-up of the nerve conduction equation *Acta Math. Appl. Sin.* **18** 264–71
[4] Wan W W and Liu C Y 1999 Long-time behavior of initial boundary problem for equation of nerve conduction *Acta Math. Appl. Sin.* **22** 311–4
[5] Gao B X *et al* 2000 The finite difference method for the initial-boundary-value problem of the equation $u_{tt} = u_{xxt} + f(u_x)_x$[J] *Math. Numer. Sin.* **22** 166–76 (in Chinese)
[6] Zhang Z Y 2002 ASE-I parallel numerical method for a class of nonlinear evolution equation *Chin. J. Comput. Mech.* **19** 154–8
[7] Zhang Z Y and Wei H 2003 A multistep characteristic finite difference method for a class of nerve conduction equations *N. P. Sci. Comput.* **11** 315–23
[8] Zhang Z Y 2005 The finite element numerical analysis for a class of nonlinear evolution equations *Appl. Math. Comput.* **166** 489–500
[9] Adam Y 1977 Highly accurate compact implicit methods and boundary conditions *J. Comput. Phys.* **24** 10–22
[10] Noye B J and Tan H H 1988 A third-order semi-implicit finite difference method for solving the one-dimensional convection–diffusion equation *Int. J. Numer. Methods Eng.* **26** 1615–29
[11] Noye B J and Tan H H 1988 Finite difference methods for solving the two-dimensional advection–diffusion equation *Int. J. Numer Methods Fluids* **26** 1615–29
[12] Karaa S and Zhang J 2004 High order ADI method for solving unsteady convection–diffusion problems *J. Comput. Phys.* **198** 1–9
[13] Liao W, Zhu J and Khaliq A Q M 2002 An efficient high order algorithm for solving systems of reaction–diffusion equations *J. Numer. Methods Partial Diff. Eqns* **18** 340–54
[14] Gu Y, Liao W and Zhu J 2003 An efficient high-order algorithm for solving systems of 3-d reaction–diffusion equations *J. Comput. Appl. Math.* **155** 1–17
[15] Singer I and Turquel E 1998 High-order finite difference method for the Helmholtz equation *Comput. Methods Appl. Mech. Eng.* **163** 343–58
[16] Zhang Z Y 2004 An economical difference scheme for heat transport equation at the microscale *J. Numer. Methods Partial Diff. Eqns* **20** 855–63
[17] Hu W J and Tang M H 2003 The numerical method for differential equations, Science Press *J. Comput. Appl. Math.* **155** 1–17