Available at WWW.MATHEMATICSWEB.ORG POWERED BY SCIENCE @ DIRECT•

ELSEVIER

Journal of Computational and Applied Mathematics 155 (2003) 1–17

JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS

www.elsevier.com/locate/cam

An efficient high-order algorithm for solving systems of 3-D reaction–diffusion equations

Yuanxian Gu[a], Wenyuan Liao[b], Jianping Zhu[c,*,1]

[a] *State Key Laboratory of Structural Analysis for Industrial Equipment, Dalian University of Technology, Dalian 116023, People's Republic of China*

[b] *Department of Mathematics & Statistics, Mississippi State University, Mississippi State, MS 39762, USA*

[c] *Department of Theoretical and Applied Mathematics, University of Akron, Akron, OH 44325, USA*

Abstract

We discuss an efficient higher order finite difference algorithm for solving systems of 3-D reaction–diffusion equations with nonlinear reaction terms. The algorithm is fourth-order accurate in both the temporal and spatial dimensions. 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 examples are presented to demonstrate the efficiency and accuracy of the new algorithm.
© 2003 Elsevier Science B.V. All rights reserved.

Keywords: High-order algorithms; Approximate factorization; Reaction–diffusion equations; Finite difference algorithm

1. Introduction

The following system of reaction–diffusion equations is widely used to model important engineering, physical, and biological processes:

$$\mathbf{w}_t = D_1\mathbf{w}_{xx} + D_2\mathbf{w}_{yy} + D_3\mathbf{w}_{zz} + \mathbf{f}(\mathbf{w}, x, y, z, t),$$

$$(x, y, z) \in (0,1)\times(0,1)\times(0,1), \quad t > 0,$$

* Corresponding author.
E-mail address: jzhu@math.uakron.edu (J. Zhu).

[1] This research was supported in part by the Visiting Scholar Foundation of University Key Laboratories in China, and by the United States National Science Foundation under Grant DMS 0075009.

0377-0427/03/$ - see front matter © 2003 Elsevier Science B.V. All rights reserved.
doi:10.1016/S0377-0427(02)00889-0

$$\mathbf{w}(0,y,z,t)=\mathbf{g}_1(y,z,t),\quad \mathbf{w}(1,y,z,t)=\mathbf{g}_2(y,z,t),\quad (y,z)\in[0,1]\times[0,1],\quad t>0,$$

$$\mathbf{w}(x,0,z,t)=\mathbf{h}_1(x,z,t),\quad \mathbf{w}(x,1,z,t)=\mathbf{h}_2(x,z,t),\quad (x,z)\in[0,1]\times[0,1],\quad t>0,$$

$$\mathbf{w}(x,y,0,t)=\mathbf{s}_1(x,y,t),\quad \mathbf{w}(x,y,1,t)=\mathbf{s}_2(x,y,t),\quad (x,y)\in[0,1]\times[0,1],\quad t>0,$$

$$\mathbf{w}(x,y,z,0)=\mathbf{q}(x,y,z),\quad (x,y,z)\in[0,1]\times[0,1]\times[0,1], \tag{1}$$

where D_1, D_2, and D_3 are diagonal matrices of dimensions $p\times p$ with positive coefficients, $\mathbf{f}(\mathbf{w},x,y,z,t)\in R^p$ is a nonlinear vector function, and $\mathbf{w}\in R^p$ is a vector of p dependent variables to be solved. For many application problems it is desirable to use high-order numerical algorithms to compute accurate solutions. To simplify the discussion, we will first present the development of an efficient high-order algorithm using the scalar and linear version of Eq. (1) with constant coefficients

$$u_t=au_{xx}+bu_{yy}+cu_{zz}+f(x,y,z,t),$$

$$(x,y,z)\in(0,1)\times(0,1)\times(0,1),\quad t>0,\quad a,b,c>0. \tag{2}$$

The result will then be generalized to systems of nonlinear equations similar to that given in (1).

It is well known [3] that standard central difference operators δ_x^2, δ_y^2 and δ_z^2 defined by

$$(u_{xx})_{ijk}\approx\frac{1}{h_x^2}\,\delta_x^2u_{i,j,k}\equiv\frac{1}{h_x^2}\,(u_{i-1,j,k}-2u_{i,j,k}+u_{i+1,j,k}), \tag{3}$$

$$(u_{yy})_{ijk}\approx\frac{1}{h_y^2}\,\delta_y^2u_{i,j,k}\equiv\frac{1}{h_y^2}\,(u_{i,j-1,k}-2u_{i,j,k}+u_{i,j+1,k}),$$

$$(u_{zz})_{ijk}\approx\frac{1}{h_z^2}\,\delta_z^2u_{i,j,k}\equiv\frac{1}{h_z^2}\,(u_{i,j,k-1}-2u_{i,j,k}+u_{i,j,k+1})$$

give only second-order approximations to u_{xx}, u_{yy}, and u_{zz}, respectively, where ${}_{ijk}$ represents the x–y–z indices for spatial grid points, and h_x, h_y, and h_z represent the grid spacing in the x, y, and z dimensions, respectively. One way to obtain higher-order approximations [3] is to use

$$(u_{xx})_{ijk}\approx\frac{1}{h_x^2}\left(I-\frac{1}{12}\,\delta_x^2\right)\delta_x^2u_{i,j,k},$$

$$(u_{yy})_{ijk}\approx\frac{1}{h_y^2}\left(I-\frac{1}{12}\,\delta_y^2\right)\delta_y^2u_{i,j,k},$$

$$(u_{zz})_{ijk}\approx\frac{1}{h_z^2}\left(I-\frac{1}{12}\,\delta_z^2\right)\delta_z^2u_{i,j,k}, \tag{4}$$

which are fourth-order accurate. However, the approximations given in (4) require a 13-point stencil, which is much more complex than the seven-point stencil required by the approximations in (3). This will not only significantly increase the computational complexity in solving the final system of algebraic equations, but also cause difficulty in handling boundary conditions since on each side of the computational domain two extra points are needed, while only one boundary condition is given in (1).

To maintain a small finite difference stencil for efficient solution process, we can use compact finite difference algorithm [1,4,5,10] to construct higher-order approximations for the spatial derivatives. For example, the formulas in (4) can be represented by the Padé approximation

$$(u_{xx})_{ijk} = \frac{\delta_x^2}{h_x^2(1+\frac{1}{12}\delta_x^2)}u_{i,j,k},$$

$$(u_{yy})_{ijk} = \frac{\delta_y^2}{h_y^2(1+\frac{1}{12}\delta_y^2)}u_{i,j,k},$$

$$(u_{zz})_{ijk} = \frac{\delta_z^2}{h_z^2(1+\frac{1}{12}\delta_z^2)}u_{i,j,k}. \tag{5}$$

Note that if we expand $1/(1+(1/12)\delta_x^2)$, $1/(1+(1/12)\delta_y^2)$, and $1/(1+(1/12)\delta_z^2)$ into power series in terms of δ_x^2, δ_y^2, and δ_z^2, respectively, the first two terms of $(u_{xx})_{ijk}$, $(u_{yy})_{ijk}$, and $(u_{zz})_{ijk}$ in (5) match the expressions in (4). If we set

$$(u_{xx})_{ijk} = v_{i,j,k}, \quad (u_{yy})_{i,j,k} = w_{i,j,k}, \quad (u_{zz})_{i,j,k} = r_{i,j,k}$$

and apply $1+\frac{1}{12}\delta_x^2$, $1+\frac{1}{12}\delta_y^2$, and $1+\frac{1}{12}\delta_z^2$ to both sides of the three equations in (5), respectively, then the following expressions:

$$\frac{1}{12}v_{i+1,j,k} + \frac{10}{12}v_{i,j,k} + \frac{1}{12}v_{i-1,j,k} = \frac{1}{h_x^2}(u_{i+1,j,k} - 2u_{i,j,k} + u_{i-1,j,k}), \tag{6}$$

$$\frac{1}{12}w_{i,j+1,k} + \frac{10}{12}w_{i,j,k} + \frac{1}{12}w_{i,j-1,k} = \frac{1}{h_y^2}(u_{i,j+1,k} - 2u_{i,j,k} + u_{i,j-1,k}), \tag{7}$$

$$\frac{1}{12}r_{i,j,k+1} + \frac{10}{12}r_{i,j,k} + \frac{1}{12}r_{i,j,k-1} = \frac{1}{h_z^2}(u_{i,j,k+1} - 2u_{i,j,k} + u_{i,j,k-1}) \tag{8}$$

provide fourth-order approximations to u_{xx}, u_{yy}, and u_{zz}, respectively. Eqs. (6), (7), and (8) result in systems of tri-diagonal equations along i-lines, j-lines, and k-lines for solving the second derivatives u_{xx}, u_{yy}, and u_{zz}, respectively. Combined with the standard finite difference approximation in the temporal dimension, such as the Crank–Nicolson scheme, of the original PDE

$$\frac{u_{i,j,k}^{n+1} - u_{i,j,k}^{n}}{\Delta t} = \frac{a}{2}(v_{i,j,k}^{n+1} + v_{i,j,k}^{n}) + \frac{b}{2}(w_{i,j,k}^{n+1} + w_{i,j,k}^{n}) + \frac{c}{2}(r_{i,j,k}^{n+1} + r_{i,j,k}^{n}) + \frac{1}{2}(f_{i,j,k}^{n+1} + f_{i,j,k}^{n}) \tag{9}$$

or

$$\frac{u_{i,j,k}^{n+1} - u_{i,j,k}^{n}}{\Delta t} = \frac{a}{2}(v_{i,j,k}^{n+1} + v_{i,j,k}^{n}) + \frac{b}{2}(w_{i,j,k}^{n+1} + w_{i,j,k}^{n}) + \frac{c}{2}(r_{i,j,k}^{n+1} + w_{i,j,k}^{n}) + f_{i,j,k}^{n+1/2} \tag{10}$$

we have the complete system of Eqs. (6)–(9) for calculating solutions with fourth-order accuracy in space using a seven-point stencil.

This approach, while maintaining a seven-point stencil in space, requires the solution of coupled system of Eqs. (6)–(9) at each grid point. With p equations in the original system (1), a total of $4p$

coupled equations need to be solved at each grid point. If an operator-splitting type method is used to turn Eq. (9) into three separate equations, one along each of the x, y, and z dimensions, then a system of $4p$ coupled equations need to be solved at each grid point in each step of the splitting method [9]. Furthermore, the calculation of solutions of Eqs. (6)–(8) requires boundary conditions for u_{xx}, u_{yy}, and u_{zz}, which are usually not known. Therefore, additional one-sided approximations have to be used to approximate the boundary conditions for u_{xx}, u_{yy}, and u_{zz} using lower order derivatives or function values. This could affect the accuracy and stability of the algorithm, as well as the structure of the final coefficient matrix in the equation system.

In [6], an efficient higher order algorithm was developed for solving 2-D reaction–diffusion equations. Here we extend that method to 3-D systems of reaction–diffusion equations. The method is based on approximate factorization of finite difference operators, which only requires solutions of systems of tri-diagonal equations. Furthermore, there is no need to introduce approximations to the boundary conditions of the second derivatives. The approach can be generalized to a system of reaction–diffusion equations with nonlinear reaction terms. In the next section we will discuss this new method based on approximate factorization. The extension to systems of nonlinear equations will be discussed in Section 3. Improvement of accuracy in the temporal dimension based on the Richardson extrapolation will be presented in Section 4, followed by numerical examples in Section 5 and conclusions in Section 6.

2. Fourth-order algorithm based on approximate factorization

We start from the Crank–Nicolson algorithm for Eq. (2) on a rectangular grid (x_i, y_j, z_k), $i = 0,\ldots,M$, $j = 0,\ldots,N$, $k = 0,\ldots,L$:

$$\begin{aligned}\frac{u^{n+1}_{i,j,k} - u^{n}_{i,j,k}}{\Delta t}\\ &= \frac{a}{2}((u_{xx})^{n+1}_{i,j,k} + (u_{xx})^{n}_{i,j,k}) + \frac{b}{2}((u_{yy})^{n+1}_{i,j,k} + (u_{yy})^{n}_{i,j,k})\\ &\quad + \frac{c}{2}((u_{zz})^{n+1}_{i,j,k} + (u_{zz})^{n}_{i,j,k}) + \frac{1}{2}(f^{n+1}_{i,j,k} + f^{n}_{i,j,k}),\\ &i = 0,\ldots,M,\ \ j = 0,\ldots,N,\ \ k = 0,\ldots,L. \end{aligned} \tag{11}$$

The standard discretization is

$$\begin{aligned}\frac{u^{n+1}_{i,j,k} - u^{n}_{i,j,k}}{\Delta t}\\ &= \frac{a}{2h_x^2}\delta_x^2(u^{n+1}_{i,j,k} + u^{n}_{i,j,k}) + \frac{b}{2h_y^2}\delta_y^2(u^{n+1}_{i,j,k} + u^{n}_{i,j,k}) + \frac{c}{2h_z^2}\delta_z^2(u^{n+1}_{i,j,k} + u^{n}_{i,j,k}) + \frac{1}{2}(f^{n+1}_{i,j,k} + f^{n}_{i,j,k}),\end{aligned} \tag{12}$$

which is known to be second-order accurate in both time and space. If the fourth-order Padé approximation (5) is used to replace u_{xx}, u_{yy}, and u_{zz}, then the following algorithm:

$$\begin{aligned} u_{i,j,k}^{n+1} - u_{i,j,k}^{n} &= \frac{r_x}{2}\frac{\delta_x^2}{1+(\delta_x^2/12)}(u_{i,j,k}^{n+1}+u_{i,j,k}^{n}) + \frac{r_y}{2}\frac{\delta_y^2}{1+(\delta_y^2/12)}(u_{i,j,k}^{n+1}+u_{i,j,k}^{n}) \\ &\quad + \frac{r_z}{2}\frac{\delta_z^2}{1+(\delta_z^2/12)}(u_{i,j,k}^{n+1}+u_{i,j,k}^{n}) + \frac{\Delta t}{2}(f_{i,j,k}^{n+1}+f_{i,j,k}^{n}), \end{aligned} \tag{13}$$

where $r_x = a\Delta t/h_x^2$, $r_y = b\Delta t/h_y^2$, and $r_z = c\Delta t/h_z^2$, is second-order accurate in time and fourth-order accurate in space. This algorithm can be written as

$$\begin{aligned} &\left(I - \frac{r_x}{2}\frac{\delta_x^2}{1+(\delta_x^2/12)} - \frac{r_y}{2}\frac{\delta_y^2}{1+(\delta_y^2/12)} - \frac{r_z}{2}\frac{\delta_z^2}{1+(\delta_z^2/12)}\right)u_{i,j,k}^{n+1} \\ &\quad = \left(I + \frac{r_x}{2}\frac{\delta_x^2}{1+(\delta_x^2/12)} + \frac{r_y}{2}\frac{\delta_y^2}{1+(\delta_y^2/12)} + \frac{r_z}{2}\frac{\delta_z^2}{1+(\delta_z^2/12)}\right)u_{i,j}^{n} + \frac{\Delta t}{2}(f_{i,j,k}^{n+1}+f_{i,j,k}^{n}), \end{aligned} \tag{14}$$

which can be approximately factorized as [2,3,7,8]

$$\begin{aligned} &\left(I - \frac{r_x}{2}\frac{\delta_x^2}{1+(\delta_x^2/12)}\right)\left(I - \frac{r_y}{2}\frac{\delta_y^2}{1+(\delta_y^2/12)}\right)\left(I - \frac{r_z}{2}\frac{\delta_z^2}{1+(\delta_z^2/12)}\right)u_{i,j,k}^{n+1} \\ &\quad = \left(I + \frac{r_x}{2}\frac{\delta_x^2}{1+(\delta_x^2/12)}\right)\left(I + \frac{r_y}{2}\frac{\delta_y^2}{1+(\delta_y^2/12)}\right)\left(I + \frac{r_z}{2}\frac{\delta_z^2}{1+(\delta_z^2/12)}\right)u_{i,j,k}^{n} \\ &\qquad + \frac{\Delta t}{2}(f_{i,j,k}^{n+1}+f_{i,j,k}^{n}). \end{aligned} \tag{15}$$

The difference between (14) and (15) is

$$\begin{aligned} &\left(\frac{r_x r_y}{4}\frac{\delta_x^2}{(1+(\delta_x^2/12))}\frac{\delta_y^2}{(1+(\delta_y^2/12))} + \frac{r_x r_z}{4}\frac{\delta_x^2}{(1+(\delta_x^2/12))}\frac{\delta_z^2}{(1+(\delta_z^2/12))}\right. \\ &\quad \left. + \frac{r_y r_z}{4}\frac{\delta_y^2}{(1+(\delta_y^2/12))}\frac{\delta_z^2}{(1+(\delta_z^2/12))}\right)(u_{i,j,k}^{n+1}-u_{i,j,k}^{n}) \\ &\quad - \frac{r_x r_y r_z}{8}\frac{\delta_x^2}{(1+(\delta_x^2/12))}\frac{\delta_y^2}{(1+(\delta_y^2/12))}\frac{\delta_z^2}{(1+(\delta_z^2/12))}(u_{i,j,k}^{n+1}+u_{i,j,k}^{n}) \\ &\quad \approx \left(\frac{r_x r_y}{4}\frac{\delta_x^2}{(1+(\delta_x^2/12))}\frac{\delta_y^2}{(1+(\delta_y^2/12))} + \frac{r_x r_z}{4}\frac{\delta_x^2}{(1+(\delta_x^2/12))}\frac{\delta_z^2}{(1+(\delta_z^2/12))}\right. \\ &\qquad \left. + \frac{r_y r_z}{4}\frac{\delta_y^2}{(1+(\delta_y^2/12))}\frac{\delta_z^2}{(1+(\delta_z^2/12))}\right)((u_t)_{i,j,k}^{n+1/2} + \mathrm{O}(\Delta t^2))\Delta t \end{aligned}$$

$$
\begin{aligned}
&-2\,\frac{r_x r_y r_z}{8}\,\frac{\delta_x^2}{(1+(\delta_x^2/12))}\,\frac{\delta_y^2}{(1+(\delta_y^2/12))}\,\frac{\delta_z^2}{(1+(\delta_z^2/12))}\,(u_{i,j,k}^{n+1/2}+\mathrm{O}(\Delta t^2))\\
&=\left(\frac{ab}{4}\,\frac{\delta_x^2}{h_x^2(1+(\delta_x^2/12))}\,\frac{\delta_y^2}{h_y^2(1+(\delta_y^2/12))}+\frac{ac}{4}\,\frac{\delta_x^2}{h_x^2(1+(\delta_x^2/12))}\,\frac{\delta_z^2}{h_z^2(1+(\delta_z^2/12))}\right.\\
&\quad\left.+\frac{bc}{4}\,\frac{\delta_y^2}{h_y^2(1+(\delta_y^2/12))}\,\frac{\delta_z^2}{h_z^2(1+(\delta_z^2/12))}\right)((u_t)_{i,j,k}^{n+1/2}+\mathrm{O}(\Delta t^2))\Delta t^3\\
&\quad-\frac{2abc\Delta t^3}{8}\,\frac{\delta_x^2}{h_x^2(1+(\delta_x^2/12))}\,\frac{\delta_y^2}{h_y^2(1+(\delta_y^2/12))}\,\frac{\delta_z^2}{h_z^2(1+(\delta_z^2/12))}(u_{i,j,k}^{n+1/2}+\mathrm{O}(\Delta t^2))\\
&=\frac{ab\Delta t^3}{4}(u_{txxyy}^{n+1/2}+\mathrm{O}(h_x^2h_y^2)+\mathrm{O}(\Delta t^2))+\frac{ac\Delta t^3}{4}(u_{txxzz}^{n+1/2}+\mathrm{O}(h_x^2h_z^2)+\mathrm{O}(\Delta t^2))\\
&\quad+\frac{bc\Delta t^3}{4}(u_{tyyzz}^{n+1/2}+\mathrm{O}(h_y^2h_z^2)+\mathrm{O}(\Delta t^2))-\frac{2abc\Delta t^3}{8}(u_{xxyyzz}^{n+1/2}+\mathrm{O}(h_x^2h_y^2h_z^2)+\mathrm{O}(\Delta t^2))\\
&=\mathrm{O}(\Delta t^3)+\mathrm{O}(\Delta t^5)
\end{aligned}
$$

provided that 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 (14). Since the operators in (15) commute, we can simplify the algorithm by applying $(1+(\delta_x^2/12))(1+(\delta_y^2/12))(1+(\delta_z^2/12))$ to both sides of (15), which leads to

$$
\begin{aligned}
&\left(1+\frac{\delta_x^2}{12}-\frac{r_x}{2}\,\delta_x^2\right)\left(1+\frac{\delta_y^2}{12}-\frac{r_y}{2}\,\delta_y^2\right)\left(1+\frac{\delta_z^2}{12}-\frac{r_z}{2}\,\delta_z^2\right)u_{i,j,k}^{n+1}\\
&\quad=\left(1+\frac{\delta_x^2}{12}+\frac{r_x}{2}\,\delta_x^2\right)\left(1+\frac{\delta_y^2}{12}+\frac{r_y}{2}\,\delta_y^2\right)\left(1+\frac{\delta_z^2}{12}+\frac{r_z}{2}\,\delta_z^2\right)u_{i,j,k}^{n}\\
&\quad+\frac{\Delta t}{2}\left(1+\frac{\delta_x^2}{12}\right)\left(1+\frac{\delta_y^2}{12}\right)\left(1+\frac{\delta_z^2}{12}\right)(f_{i,j,k}^{n+1}+f_{i,j,k}^{n}). \qquad (16)
\end{aligned}
$$

Eq. (16) can be solved in three steps as

$$
\begin{aligned}
\left(1+\frac{\delta_x^2}{12}-\frac{r_x}{2}\,\delta_x^2\right)u_{i,j,k}^{**}&=\left(1+\frac{\delta_x^2}{12}+\frac{r_x}{2}\,\delta_x^2\right)\left(1+\frac{\delta_y^2}{12}+\frac{r_y}{2}\,\delta_y^2\right)\left(1+\frac{\delta_z^2}{12}+\frac{r_z}{2}\,\delta_z^2\right)u_{i,j,k}^{n}\\
&\quad+\frac{\Delta t}{2}\left(1+\frac{\delta_x^2}{12}\right)\left(1+\frac{\delta_y^2}{12}\right)\left(1+\frac{\delta_z^2}{12}\right)(f_{i,j,k}^{n+1}+f_{i,j,k}^{n}), \qquad (17)
\end{aligned}
$$

$$
\left(1+\frac{\delta_y^2}{12}-\frac{r_y}{2}\,\delta_y^2\right)u_{i,j,k}^{*}=u_{i,j,k}^{**}, \qquad (18)
$$

$$\left(1+\frac{\delta_z^2}{12}-\frac{r_z}{2}\,\delta_z^2\right)u_{i,j,k}^{n+1}=u_{i,j,k}^{*}. \tag{19}$$

The solutions to Eqs. (17)–(19) can be computed by solving tri-diagonal equations since the left-hand sides of (17)–(19) involve only three-point central difference operators δ_x^2 or δ_y^2 or δ_z^2 as defined in (3). Although the right-hand side of (17) involves the product of operators δ_x^2, δ_y^2, and δ_y^2, it does not complicate the solution process since it is applied to the known solution values from the previous time step.

While solving Eq. (18), we need boundary conditions for $u_{i,0,k}^{*}$ and $u_{i,N+1,k}^{*}$, $i=1,\ldots,M$, $k=1,\ldots,L$. These conditions can be obtained from Eq. (19) and the Dirichlet boundary condition in (1) by setting $j=0$ and $j=N+1$, respectively:

$$u_{i,0,k}^{*}=\left(1+\frac{\delta_z^2}{12}-\frac{r_z}{2}\,\delta_z^2\right)u_{i,0,k}^{n+1},$$

$$u_{i,N+1,k}^{*}=\left(1+\frac{\delta_z^2}{12}-\frac{r_z}{2}\,\delta_z^2\right)u_{i,N+1,k}^{n+1}. \tag{20}$$

Similarly, while solving Eq. (17), we need boundary conditions for $u_{0,j,k}^{**}$ and $u_{M+1,j,k}^{**}$, $j=1,\ldots,N$, $k=1,\ldots,L$. These conditions can be obtained from Eqs. (18) and (19) by setting $i=0$ and $i=M+1$, respectively:

$$u_{0,j,k}^{**}=\left(1+\frac{\delta_y^2}{12}-\frac{r_y}{2}\,\delta_y^2\right)\left(1+\frac{\delta_z^2}{12}-\frac{r_z}{2}\,\delta_z^2\right)u_{0,j,k}^{n+1},$$

$$u_{M+1,j,k}^{**}=\left(1+\frac{\delta_y^2}{12}-\frac{r_y}{2}\,\delta_y^2\right)\left(1+\frac{\delta_z^2}{12}-\frac{r_z}{2}\,\delta_z^2\right)u_{M+1,j,k}^{n+1}. \tag{21}$$

Since the spatial discretization used in obtaining (17) is fourth-order accurate in space, the boundary conditions given by (20) and (21) have the same spatial accuracy as (17). This approach avoids using one-sided difference approximations to the second spatial derivatives at the boundary, as is required by the standard compact difference algorithms [1,4,5,10].

3. Systems of equations with nonlinear reaction terms

For a system of equations with linear diffusion and nonlinear reaction as given in (1), algorithms (17)–(19) will result in the following system of equations:

$$\begin{aligned}\left(I+\frac{\delta_x^2}{12}-\frac{\mathbf{r}_x}{2}\,\delta_x^2\right)\mathbf{w}_{i,j,k}^{**}\\ &=\left(I+\frac{\delta_x^2}{12}+\frac{\mathbf{r}_x}{2}\,\delta_x^2\right)\left(I+\frac{\delta_y^2}{12}+\frac{\mathbf{r}_y}{2}\,\delta_y^2\right)\left(I+\frac{\delta_z^2}{12}+\frac{\mathbf{r}_z}{2}\,\delta_z^2\right)\mathbf{w}_{i,j,k}^{n}\\ &\quad+\frac{\Delta t}{2}\left(I+\frac{\delta_x^2}{12}\right)\left(I+\frac{\delta_y^2}{12}\right)\left(I+\frac{\delta_z^2}{12}\right)(\mathbf{f}_{i,j,k}^{n+1}+\mathbf{f}_{i,j,k}^{n}),\end{aligned} \tag{22}$$

$$\left(I+\frac{\delta_y^2}{12}-\frac{\mathbf{r}_y}{2}\,\delta_y^2\right)\mathbf{w}^{*}_{i,j,k}=\mathbf{w}^{**}_{i,j,k}, \tag{23}$$

$$\left(I+\frac{\delta_z^2}{12}-\frac{\mathbf{r}_z}{2}\,\delta_z^2\right)\mathbf{w}^{n+1}_{i,j,k}=\mathbf{w}^{*}_{i,j,k}, \tag{24}$$

where I is the identify matrix and $\mathbf{r}_x$, $\mathbf{r}_y$, and $\mathbf{r}_z$ are defined by

$$\mathbf{r}_x=\frac{\Delta t}{h_x^2}D_1,\quad \mathbf{r}_y=\frac{\Delta t}{h_y^2}D_2,\quad \mathbf{r}_y=\frac{\Delta t}{h_z^2}D_3.$$

Note that Eq. (22) contains solutions $\mathbf{w}^{n}_{i,j,k}$, $\mathbf{w}^{**}_{i,j,k}$, and $\mathbf{w}^{n+1}_{i,j,k}$ (implicitly in $\mathbf{f}^{n+1}_{i,j,k}$). Since both $\mathbf{w}^{**}_{i,j,k}$ and $\mathbf{w}^{n+1}_{i,j,k}$ are unknown, Eq. (22) cannot be linearized by simply using Newton's method or its variations to expand $\mathbf{f}^{n+1}_{i,j,k}$ at $\mathbf{w}^{n}_{i,j,k}$. In [9], a predictor–corrector type algorithm was used to overcome this difficulty. The algorithm begins by using the expansion

$$\mathbf{f}^{n+1}_{i,j,k}=\mathbf{f}^{n}_{i,j,k}+\mathbf{J}^{n}_{i,j,k}(\mathbf{w}^{n+1}_{i,j,k}-\mathbf{w}^{n}_{i,j,k})+\Delta t(\mathbf{f}_t)^{n}_{i,j,k}, \tag{25}$$

where $\mathbf{J}^{n}_{i,j,k}=(\partial\mathbf{f}/\partial\mathbf{w})^{n}_{i,j,k}$ is the local Jacobian matrix at grid point (i,j,k). The algorithm in (22)–(24) can then be written as

$$\begin{aligned}\left(I+\frac{\delta_x^2}{12}-\frac{\mathbf{r}_x}{2}\,\delta_x^2\right)&\mathbf{w}^{**}_{i,j,k}\\ &=\left(I+\frac{\delta_x^2}{12}+\frac{\mathbf{r}_x}{2}\,\delta_x^2\right)\left(I+\frac{\delta_y^2}{12}+\frac{\mathbf{r}_y}{2}\,\delta_y^2\right)\left(I+\frac{\delta_z^2}{12}+\frac{\mathbf{r}_z}{2}\,\delta_z^2\right)\mathbf{w}^{n}_{i,j,k}\\ &\quad+\frac{\Delta t}{2}\left(I+\frac{\delta_x^2}{12}\right)\left(I+\frac{\delta_y^2}{12}\right)\left(I+\frac{\delta_z^2}{12}\right)(2\mathbf{f}^{n}_{i,j,k}+\Delta t(\mathbf{f}_t)^{n}_{i,j,k})+\mathbf{e}^{n+1}_{i,j,k},\end{aligned} \tag{26}$$

$$\left(I+\frac{\delta_y^2}{12}-\frac{\mathbf{r}_y}{2}\,\delta_y^2\right)\mathbf{w}^{*}_{i,j,k}=\mathbf{w}^{**}_{i,j,k}, \tag{27}$$

$$\left(I+\frac{\delta_z^2}{12}-\frac{\mathbf{r}_z}{2}\,\delta_z^2\right)\mathbf{w}^{n+1}_{i,j,k}=\mathbf{w}^{*}_{i,j,k}, \tag{28}$$

where $\mathbf{e}^{n+1}_{i,j,k}=\mathbf{J}^{n}_{i,j,k}(\mathbf{w}^{n+1}_{i,j,k}-\mathbf{w}^{n}_{i,j,k})$. An intermediate solution $\mathbf{w}^{(n+1)^P}_{ijk}$ is calculated by first using the predictor

$$\begin{aligned}\left(I+\frac{\delta_x^2}{12}-\frac{\mathbf{r}_x}{2}\,\delta_x^2\right)&\mathbf{w}^{**^P}_{i,j,k}\\ &=\left(I+\frac{\delta_x^2}{12}+\frac{\mathbf{r}_x}{2}\,\delta_x^2\right)\left(I+\frac{\delta_y^2}{12}+\frac{\mathbf{r}_y}{2}\,\delta_y^2\right)\left(I+\frac{\delta_z^2}{12}+\frac{\mathbf{r}_z}{2}\,\delta_z^2\right)\mathbf{w}^{n}_{i,j,k}\\ &\quad+\frac{\Delta t}{2}\left(I+\frac{\delta_x^2}{12}\right)\left(I+\frac{\delta_y^2}{12}\right)\left(I+\frac{\delta_z^2}{12}\right)(2\mathbf{f}^{n}_{i,j,k}+\Delta t(\mathbf{f}_t)^{n}_{i,j,k}),\end{aligned} \tag{29}$$

$$\left(I+\frac{\delta_y^2}{12}-\frac{\mathbf{r}_y}{2}\,\delta_y^2\right)\mathbf{w}^{*^P}_{i,j,k}=\mathbf{w}^{**^P}_{i,j,k}, \tag{30}$$

$$\left(I+\frac{\delta_z^2}{12}-\frac{\mathbf{r}_z}{2}\,\delta_z^2\right)\mathbf{w}^{(n+1)^P}_{i,j,k}=\mathbf{w}^{*^P}_{i,j,k}. \tag{31}$$

The solution $\mathbf{w}^{n+1}_{i,j,k}$ is then calculated as the converged results of the following iterative correction step:

$$\begin{aligned}\left(I+\frac{\delta_x^2}{12}-\frac{\mathbf{r}_x}{2}\,\delta_x^2\right)\mathbf{w}^{**^k}_{i,j,k}\\ &=\left(I+\frac{\delta_x^2}{12}+\frac{\mathbf{r}_x}{2}\,\delta_x^2\right)\left(I+\frac{\delta_y^2}{12}+\frac{\mathbf{r}_y}{2}\,\delta_y^2\right)\left(I+\frac{\delta_z^2}{12}+\frac{\mathbf{r}_z}{2}\,\delta_z^2\right)\mathbf{w}^n_{i,j,k}\\ &\quad+\frac{\Delta t}{2}\left(I+\frac{\delta_x^2}{12}\right)\left(I+\frac{\delta_y^2}{12}\right)\left(I+\frac{\delta_z^2}{12}\right)(2\mathbf{f}^n_{i,j,k}+\Delta t(\mathbf{f}_t)^n_{i,j,k})+\mathbf{e}^{(n+1)^{(k-1)}}_{i,j,k},\end{aligned} \tag{32}$$

$$\left(I+\frac{\delta_y^2}{12}-\frac{\mathbf{r}_y}{2}\,\delta_y^2\right)\mathbf{w}^{*^k}_{i,j,k}=\mathbf{w}^{**^k}_{i,j,k}, \tag{33}$$

$$\left(I+\frac{\delta_z^2}{12}-\frac{\mathbf{r}_z}{2}\,\delta_z^2\right)\mathbf{w}^{(n+1)^k}_{i,j,k}=\mathbf{w}^{*^k}_{i,j,k},\quad k=1,2,\ldots, \tag{34}$$

where k represents the number of iterations in the correction step. For $k=1$, the solution values from the predictor step are used in the computation. This method requires iterations over all three Eqs. (32)–(34) to consider the nonlinear effect of the reaction term.

Here, we introduce a more efficient way to deal with the nonlinear reaction term. Note that if we rewrite algorithms (22)–(24) as

$$\begin{aligned}\left(I+\frac{\delta_x^2}{12}-\frac{\mathbf{r}_x}{2}\,\delta_x^2\right)\mathbf{w}^{**}_{i,j,k}\\ &=\left(I+\frac{\delta_x^2}{12}+\frac{\mathbf{r}_x}{2}\,\delta_x^2\right)\left(I+\frac{\delta_y^2}{12}+\frac{\mathbf{r}_y}{2}\,\delta_y^2\right)\left(I+\frac{\delta_z^2}{12}+\frac{\mathbf{r}_z}{2}\,\delta_z^2\right)\mathbf{w}^n_{i,j,k}\\ &\quad+\frac{\Delta t}{2}\left(I+\frac{\delta_x^2}{12}+\frac{\mathbf{r}_x}{2}\,\delta_x^2\right)\left(I+\frac{\delta_y^2}{12}+\frac{\mathbf{r}_y}{2}\,\delta_y^2\right)\left(I+\frac{\delta_z^2}{12}\right)\mathbf{f}^n_{i,j,k},\end{aligned} \tag{35}$$

$$\left(I+\frac{\delta_y^2}{12}-\frac{\mathbf{r}_y}{2}\,\delta_y^2\right)\mathbf{w}^{*}_{i,j,k}=\mathbf{w}^{**}_{i,j,k}, \tag{36}$$

$$\left(I+\frac{\delta_z^2}{12}-\frac{\mathbf{r}_z}{2}\,\delta_z^2\right)\mathbf{w}^{n+1}_{i,j,k}=\mathbf{w}^{*}_{i,j,k}+\frac{\Delta t}{2}\left(1+\frac{\delta_z^2}{12}\right)\mathbf{f}^{n+1}_{i,j,k}, \tag{37}$$

where $\mathbf{f}_{i,j,k}^{n+1}$ can be expanded using (25), the difference between the algorithm in (22)–(24) and that in (35)–(37) is

$$\frac{\Delta t}{2}\frac{\mathbf{r}_x}{2}\delta_x^2\frac{\mathbf{r}_y}{2}\delta_y^2\left(I+\frac{\delta_z^2}{12}\right)(\mathbf{f}_{i,j,k}^n+\mathbf{f}_{i,j,k}^{n+1})$$

$$+\frac{\Delta t}{2}\left(\left(\left(I+\frac{\delta_x^2}{12}\right)\frac{\mathbf{r}_y}{2}\delta_y^2+\frac{\mathbf{r}_x}{2}\delta_x^2\left(I+\frac{\delta_y^2}{12}\right)\right)\left(I+\frac{\delta_z^2}{12}\right)\right)(\mathbf{f}_{i,j,k}^n-\mathbf{f}_{i,j,k}^{n+1})$$

for which we have the estimate $O(\Delta t^3)+O(\Delta t^5)$. This is of the same order as that of the original truncation error in algorithm (14). With this new formulation, Eqs. (35) and (36) are linear and can be solved in a straightforward manner. Eq. (37) can be linearized by Newton's method, or its variations, such as that given in (25). The new algorithm can then be written as

$$\left(I+\frac{\delta_x^2}{12}-\frac{\mathbf{r}_x}{2}\delta_x^2\right)\mathbf{w}_{i,j,k}^{**}$$

$$=\left(I+\frac{\delta_x^2}{12}+\frac{\mathbf{r}_x}{2}\delta_x^2\right)\left(I+\frac{\delta_y^2}{12}+\frac{\mathbf{r}_y}{2}\delta_y^2\right)\left(I+\frac{\delta_z^2}{12}+\frac{\mathbf{r}_z}{2}\delta_z^2\right)\mathbf{w}_{i,j,k}^n$$

$$+\frac{\Delta t}{2}\left(I+\frac{\delta_x^2}{12}+\frac{\mathbf{r}_x}{2}\delta_x^2\right)\left(I+\frac{\delta_y^2}{12}+\frac{\mathbf{r}_y}{2}\delta_y^2\right)\left(I+\frac{\delta_z^2}{12}\right)\mathbf{f}_{i,j,k}^n, \tag{38}$$

$$\left(I+\frac{\delta_y^2}{12}-\frac{\mathbf{r}_y}{2}\delta_y^2\right)\mathbf{w}_{i,j,k}^{*}=\mathbf{w}_{i,j,k}^{**}, \tag{39}$$

$$\left(I+\frac{\delta_z^2}{12}-\frac{\mathbf{r}_z}{2}\delta_z^2-\frac{\Delta t}{2}\left(I+\frac{\delta_z^2}{12}\right)\mathbf{J}_{i,j,k}^n\right)\mathbf{w}_{i,j,k}^{n+1}$$

$$=\mathbf{w}_{i,j,k}^{*}+\frac{\Delta t}{2}\left(I+\frac{\delta_z^2}{12}\right)(\mathbf{f}_{i,j,k}^n-\mathbf{J}_{i,j,k}^n\mathbf{w}_{i,j,k}^n+\Delta t(\mathbf{f}_t)_{i,j,k}^n). \tag{40}$$

To achieve high accuracy for strongly nonlinear problems, Newton's iterations can be used to solve (40), which leads to

$$\left(I+\frac{\delta_z^2}{12}-\frac{\mathbf{r}_z}{2}\delta_z^2-\frac{\Delta t}{2}\left(I+\frac{\delta_z^2}{12}\right)\mathbf{J}_{i,j,k}^{(n+1)^{m-1}}\right)\mathbf{w}_{i,j,k}^{(n+1)^m}$$

$$=\mathbf{w}_{i,j,k}^{*}+\frac{\Delta t}{2}\left(I+\frac{\delta_z^2}{12}\right)(\mathbf{f}_{i,j,k}^{(n+1)^{m-1}}-\mathbf{J}_{i,j,k}^{(n+1)^{m-1}}\mathbf{w}_{i,j,k}^{(n+1)^{m-1}}+\Delta t(\mathbf{f}_t)_{i,j,k}^n),\quad m=1,2,\ldots,$$

where $\mathbf{w}_{i,j,k}^{(n+1)^0}=\mathbf{w}_{i,j,k}^n$.

4. Higher-order accuracy in the temporal dimension

The algorithm given in (38)–(40) is fourth-order accurate in space, but only second-order accurate in time. Because of the special formulation that led to the fourth-order accuracy in space on a seven-point stencil, it is difficult to combine this algorithm with available high-order ODE solution algorithms to achieve better accuracy in the temporal dimension. Following the derivation from (12) to (19), we can see that the temporal discretization is involved in the very beginning of this algorithm development. As a result, it is difficult to use some of the well established methods, such as method of lines (MOL), to first discretize the space derivatives, and then use high-order ODE time integration methods to achieve high temporal accuracy.

We used Richardson extrapolation on the computed solution to eliminate the lower order term in the truncation error. Since the Crank–Nicolson algorithm has a temporal truncation error in the form of $O(\Delta t^2) + O(\Delta t^4)$, we use

$$\mathbf{w} = \frac{4\mathbf{w}^{h/2} - \mathbf{w}^h}{3} \tag{41}$$

to eliminate the term $O(\Delta t^2)$, where $\mathbf{w}^{h/2}$ and $\mathbf{w}^h$ are the solutions at the final time level calculated using $\Delta t = h$ and $\Delta t = h/2$, respectively. This makes the final solution fourth-order accurate in both the temporal and spatial dimensions. Although the extrapolation requires three times as much computation as the original algorithm, the resulting high-order accuracy allows the use of much larger time steps in the computation.

5. Numerical experiment

We discuss three numerical examples here: two with analytic solutions against which we can compare the numerical solution to demonstrate the efficiency and order of accuracy of the new algorithm in both the spatial and temporal dimensions; and the other with unknown exact solution for which we plot the numerical results to demonstrate the time evolution of the solutions.

Example 1. The equations to be solved are

$$u_t = u_{xx} + u_{yy} + u_{zz} + u(v-1) + f(x,y,z,t),$$

$$v_t = v_{xx} + v_{yy} + v_{zz} + v(u-1) + g(x,y,z,t),$$

$$0 < x,y,z < 1, \quad t > 0,$$

where $f(x,y,z,t)$, $g(x,y,z,t)$, and the boundary and initial conditions have been selected to accommodate the exact solutions of $u = e^{-t/3}\sin(\frac{x}{3})\sin(\frac{y}{3})\sin(\frac{z}{3})$ and $v = e^{-3t}\sin(x)\sin(y)\sin(z)$. The data in Table 1 shows the maximum error between the calculated solution and the exact solution at $T = 1$. The discretization grid is $\Delta x = \Delta y = \Delta z = \Delta t = h$, and the algorithm given by (38)–(40) was used with a full analytic Jacobian matrix and Newton's iterations. The notation e_1 represents the error from the algorithm that is second-order accurate in time and fourth-order accurate in space, and e_2 represents the error from the algorithm that is fourth-order accurate in both time and space.

It is clear from Table 1 that the error represented by e_1 shows a second-order decrease, while that represented by e_2 shows a fourth-order decrease. This is demonstrated by the fact that the ratios of e_1/h^2 and e_2/h^4 remain roughly a constant as the computational grid is being refined. Each time when the computation grid is refined by halving Δt, Δx, Δy, and Δz, e_1 is reduced by a factor of 4,

Table 1
Maximum error between the calculated solution and the exact solution at $T = 1.0$. e_1 is the maximum error from the algorithm that is second-order accurate in time and fourth-order accurate in space. e_2 is the maximum error from the algorithm that is fourth-order accurate in both time and space. $\Delta t = \Delta x = \Delta y = \Delta z = h$

h	0.25	0.20	0.125	0.10	0.0625	0.05
e_1	1.322e − 6	8.088e − 07	2.841e − 07	1.736e − 07	6.265e − 08	3.893e − 08
e_1/h^2	2.115e − 05	2.022e − 05	1.818e − 05	1.736e − 05	1.604e − 05	1.557e − 05
e_2	6.171e − 08	2.387e − 08	3.422e − 9	1.389e − 9	2.098e − 10	8.626e − 11
e_2/h^4	1.579e − 05	1.492e − 05	1.402e − 05	1.389e − 05	1.375e − 05	1.380e − 05

Table 2
Maximum error between the calculated solution and the exact solution at $T = 1.0$. e_3 is the maximum error from the algorithm that is second-order accurate in time and fourth-order accurate in space. The initial grid is $\Delta t = \Delta x = \Delta y = \Delta z = 0.2$. In each refinement step, Δt is reduced by a factor of 4 and $\Delta x = \Delta y = \Delta z = h$ is reduced by a factor of 2

h	0.2	0.1	0.05	0.025
Δt	0.2	0.05	0.0125	0.003125
e_3	8.088e − 07	4.443e − 08	2.450e − 09	1.516e − 10
e_3/h^4	5.055e − 04	4.443e − 04	3.919e − 04	3.882e − 04

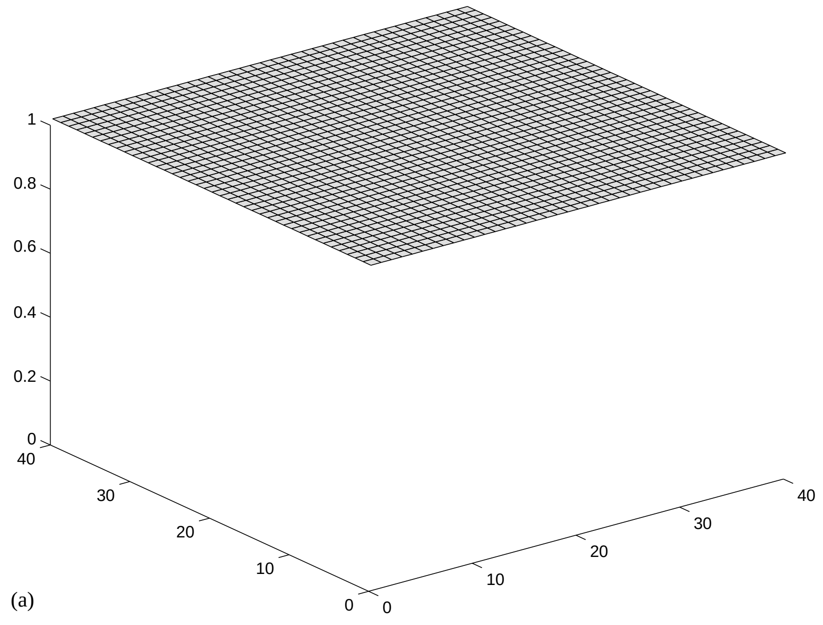
Table 3
CPU times in seconds for achieving the same accuracy using the standard ADI method and the new fourth-order method at $t = 1.0$. t_2: Time using standard ADI method. t_4: Time using the new fourth-order method

Error	8.0e − 4	2.0e − 4	5.0e − 5	8.0e − 6	2.0e − 6	5.0e − 7
t_2	0.022	0.162	1.334	21.335	198.792	2189.390
t_4	0.002	0.013	0.016	0.062	0.520	1.037

while e_2 is reduced by a factor of 16. Table 2 shows similar results as those represented by e_1 in Table 1, except Δt is refined by a factor of 4 each time, while Δx, Δy, and Δz is refined by a factor of 2 each time. It is clear that the error e_3 is now being reduced by a factor of 16 with each grid refinement, and the ratio e_3/h^4 remains 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 e_2 in Table 1.

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

$$u_t = u_{xx} + u_{yy} + f(x, y, t), \quad (x, y) \in (0, 1) \times (0, 1), \; t > 0, \tag{42}$$



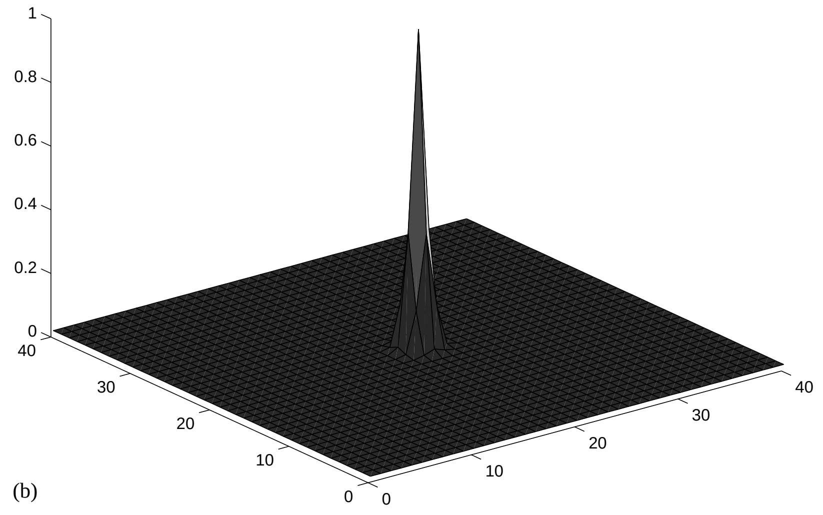


Fig. 1. Initial conditions for Eq. (43): (a) u at $t=0$ and $x=0.5$; (b) v at $t=0$ and $x=0.5$.

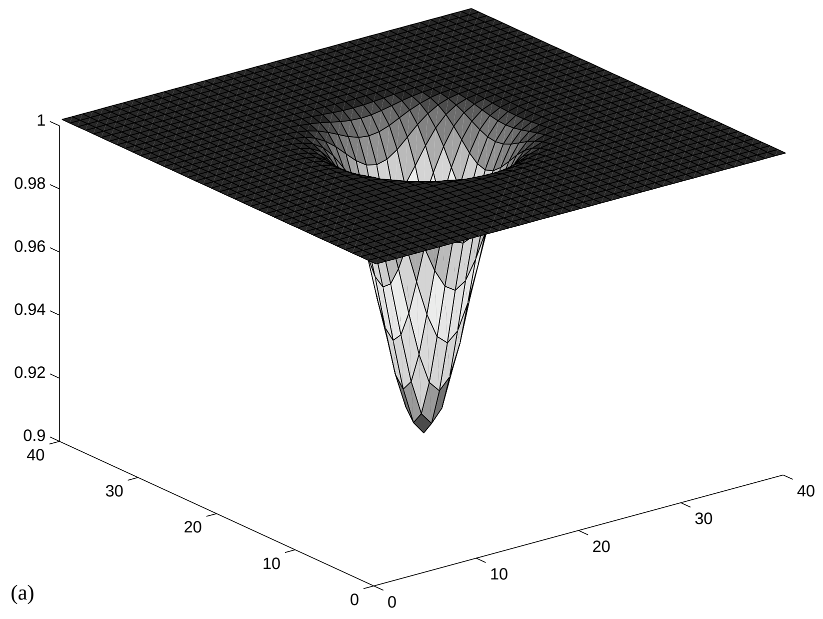
$$f(x,y,t)=7.5e^{-0.5t}\sin(2x)\sin(2y),$$

$$u(0,y,t)=0,\quad u(1,y,t)=e^{-0.5t}\sin(2.0)\sin(2y),$$

$$u(x,0,t)=0,\quad u(x,1,t)=e^{-0.5t}\sin(2x)\sin(2.0),$$

$$u(x,y,0)=\sin(2x)\sin(2y)$$

with an exact solution $u(x,y,t)=e^{-0.5t}\sin(2x)\sin(2y)$.



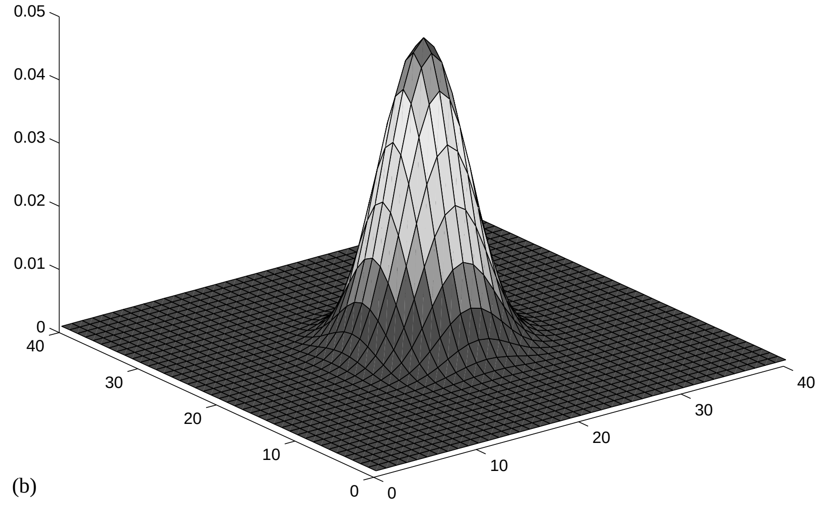
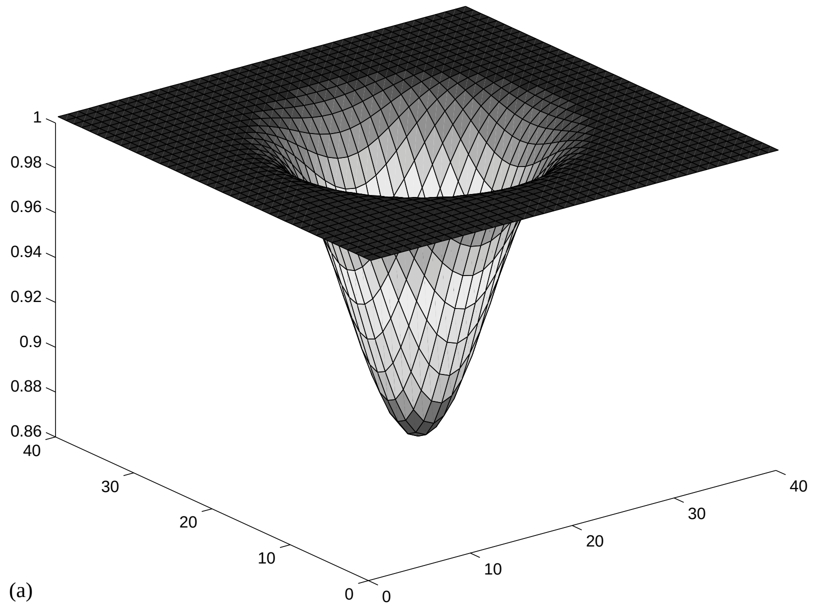


Fig. 2. Solutions to Eq. (43): (a) u at $t = 4$ and $x = 0.5$; (b) v at $t = 4$ and $x = 0.5$.

The data in Table 3 shows that the fourth-order algorithm is much more efficient than the standard second-order ADI algorithm. In the case of reducing the error to 5.0×10^{-7}, the fourth-order algorithm is more than two thousand times faster than the second-order algorithm.

Example 3. The equations to be solved are

$$\begin{aligned}&u_t = u_{xx} + u_{yy} + u_{zz} - u^4 v,\\&v_t = v_{xx} + v_{yy} + + v_{zz} + u^4 v - 0.5v,\\&0 < x, y, z < 1, \quad t > 0\end{aligned} \tag{43}$$



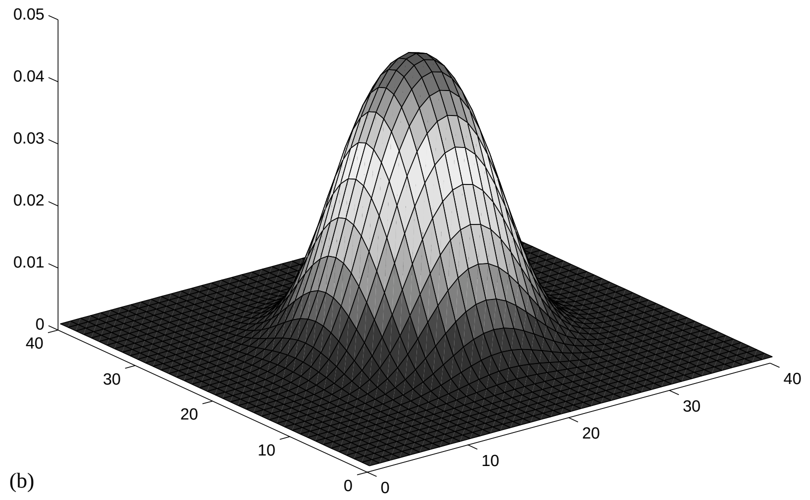


Fig. 3. Solutions to Eq. (43): (a) u at $t = 8$ and $x = 0.5$; (b) v at $t = 8$ and $x = 0.5$.

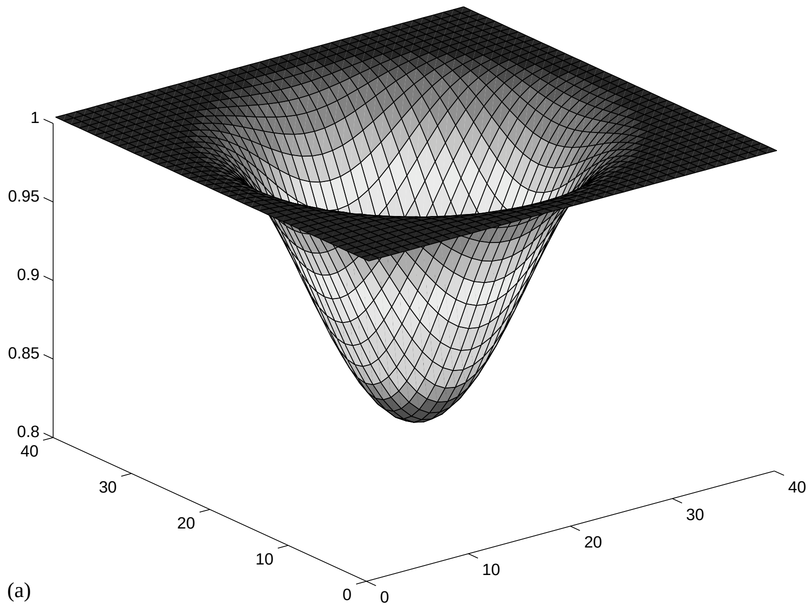
with the following boundary conditions:

$$u(0, y, z, t) = u(1, y, z, t) = u(x, 0, z, t) = u(x, 1, z, t) = u(x, y, 0, t) = u(x, y, 1, t) = 1.0,$$

$$v(0, y, z, t) = v(1, y, z, t) = v(x, 0, z, t) = v(x, 1, z, t) = v(x, y, 0, t) = v(x, y, 1, t) = 0.0.$$

The initial conditions for u and v are

$$u(x, y, z, 0) = 1.0, \quad v(x, y, z, 0) = \mathrm{e}^{-1600((x-0.5)^2+(y-0.5)^2+(z-0.5)^2)}.$$



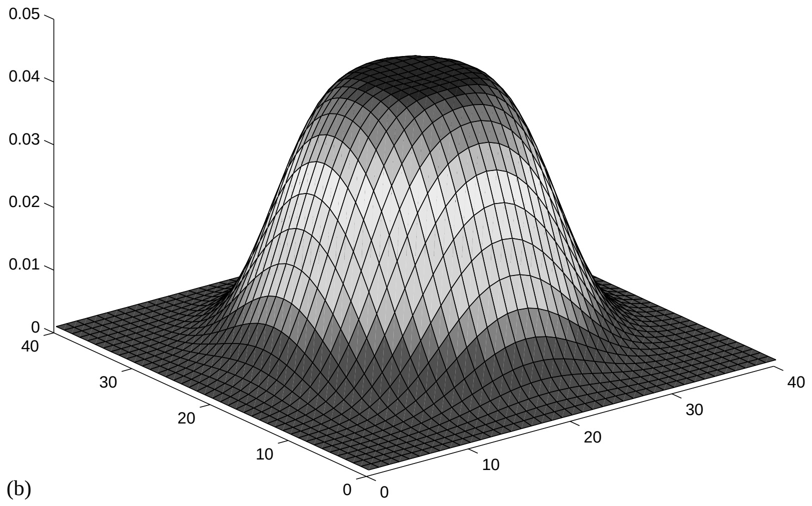


Fig. 4. Solutions to Eq. (43): (a) u at $t = 12$ and $x = 0.5$; (b) v at $t = 12$ and $x = 0.5$.

Figs. 1(a) and (b) show the initial conditions u and v, respectively, at $x = 0.5$ as functions of y and z.

Figs. 2–4 show the numerical solutions u and v to Eq. (43) at time $T = 4, 8, 12$, respectively. They are plotted at $x = 0.5$ as functions of y and x. The computations were carried out using the algorithm given in (38)–(41) on an 81×81 grid with a time step size of $\Delta t = 0.001$. The time evolution of the solution is clearly demonstrated in these figures. Due to the diffusion and chemical reaction (negative source term), u gradually decreases. The rate of decrease is much higher in the middle of the domain due to the large initial value of v there (hence large value of $u^4 v$), thus forming a deep valley that expands radially. Similarly, the peak of v in the middle of the domain also decreases in amplitude and spreads radially due to diffusion and reaction. Note, however, Fig. 4(b) shows a crater-like peak for solution v at $T = 12$. This can be explained by the fact that the rapid decrease of u at the center of the domain makes the reaction term $u^4v - 0.5v$ in the second equation much more negative there than in other part of the domain, hence causing v to decrease much more rapidly at the center of the domain.

6. Conclusion

An efficient implicit high-order method for solving systems of 3-D reaction–diffusion equations with linear diffusion and nonlinear reaction is discussed in this paper. It is fourth-order accurate in time and space, and uses a compact seven-point finite difference stencil for three-dimensional problems. Numerical results have demonstrated the high-order accuracy in both temporal and spatial dimensions. The algorithm can be used to solve various application problems involving reaction and diffusion. Since many models in science and engineering require Neumann boundary conditions, the authors are current working on extending this algorithm to problems involving Neumann bounday conditions.

Acknowledgements

The authors are grateful to the two anonymous referees for their helpful comments on the revision of the manuscript.

References

[1] Y. Adam, Highly accurate compact implicit methods and boundary conditions, J. Comput. Phys. 24 (1977) 10–22.
[2] J. Douglas Jr., On the numerical integration of $\partial^2u/\partial x^2 + \partial^2u/\partial y^2 = \partial u/\partial t$ by implicit methods, J. Soc. Ind. Appl. Math. 3 (1955) 42–65.
[3] B. Gustafson, H. Kreiss, J. Oliger, Time Dependent Problems and Difference Methods, Wiley, New York, 1995.
[4] R.S. Hirsch, Higher order accurate difference solutions of fluid mechanics problems by a compact differencing technique, J. Comput. Phys. 19 (1975) 90–109.
[5] S.K. Lele, Compact finite difference schemes with spectral-like resolution, J. Comput. Phys. 103 (1992) 16–42.
[6] W. Liao, J. Zhu, A.Q.M. Khaliq, An efficient high order algorithm for solving systems of reaction–diffusion equations, J. Numer. Methods Partial Differential Equations 18 (2002) 340–354.
[7] A.R. Mitchell, D.F. Griffths, The Finite Difference Method in Partial Differential Equations, Wiley, New York, 1980.
[8] D.W. Peaceman, H.H. Rachford Jr., The numerical solution of parabolic and elliptic differential equations, J. Soc. Ind. Appl. Math. 3 (1955) 28–41.
[9] J.I. Ramos, Implicit, compact, linearized θ-methods with factorization for multidimensional reaction–diffusion equations, Appl. Math. Comput. 94 (1998) 17–43.
[10] R.V. Wilson, A.O. Demuren, M. Carpenter, Higher-order compact schemes for numerical simulation of incompressible flows, ICASE Report, No. 98-13, 1998.