An Efficient High-Order Algorithm for Solving Systems of Reaction-Diffusion Equations

Wenyuan Liao,[1] **Jianping Zhu,**[1] **and Abdul Q. M. Khaliq**[2]
[1] *Department of Mathematics and Statistics*
Mississippi State University
Mississippi State, MS 39762

[2] *Department of Mathematics*
Western Illinois University
Macomb, IL 61455

Received 29 May 2001; accepted 27 September 2001

An efficient higher-order finite difference algorithm is presented in this article for solving systems of two-dimensional reaction-diffusion equations with nonlinear reaction terms. The method is fourth-order accurate in both the temporal and spatial dimensions. It requires only a regular five-point difference stencil similar to that used in the standard second-order algorithm, such as the Crank-Nicolson algorithm. The Padé approximation and Richardson extrapolation are used to achieve high-order accuracy in the spatial and temporal dimensions, respectively. Numerical examples are presented to demonstrate the efficiency and accuracy of the new algorithm. © 2002 Wiley Periodicals, Inc. Numer Methods Partial Differential Eq 18: 340–354, 2002; Published online in Wiley InterScience (www.interscience.wiley.com). DOI 10.1002/num.10012

Keywords: high order algorithms; reaction-diffusion equations; extrapolation and interpolations

I. INTRODUCTION

We consider the following equation

$$\mathbf{w}_t = D_1\mathbf{w}_{xx} + D_2\mathbf{w}_{yy} + \mathbf{f}(\mathbf{w}, x, y, t), \qquad (x, y) \in (0, 1) \times (0, 1), t > 0,$$

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

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

$$\mathbf{w}(x, y, 0) = \mathbf{q}(x, y), \qquad (x, y) \in [0, 1] \times [0, 1], \tag{1.1}$$

Correspondence to: Abdul Q. M. Khaliq, Department of Mathematics, Western Illinois University, Morgan 476, 1 University Circle, Macomb, IL 61455-1390 (e-mail: A-Khaliq@wiu.edu)

where D_1 and D_2 are diagonal matrices of dimensions $p \times p$ with positive coefficients, $\mathbf{f}(\mathbf{w}, x, y, t) \in R^p$ 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 in science and engineering, it is desirable to use high-order numerical algorithms to compute accurate solutions. To simplify the discussion, we will first start the development of an efficient high-order algorithm using the scalar and linear version of Eq. (1.1) with constant coefficients

$$u_t = au_{xx} + bu_{yy} + f(x, y, t), (x, y) \in (0, 1) \times (0, 1), \qquad t > 0, \quad a, b > 0. \tag{1.2}$$

The result will then be generalized to systems of nonlinear equations similar to that given in (1.1).

It is well known [1] that the standard central difference operators δ_x^2 and δ_y^2 defined by

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

$$(u_{yy})_{ij} \approx \frac{1}{h_y^2}\delta_y^2 u_{i,j} \equiv \frac{1}{h_y^2}\,(u_{i,j-1} - 2u_{i,j} + u_{i,j+1}), \tag{1.3}$$

give only second-order approximations to u_{xx} and u_{yy}, respectively, where $_{ij}$ represents the $x-y$ indices for spatial grid points, h_x and h_y represent the grid spacing in the x and y dimensions, respectively. One way to obtain higher-order approximations is to use [1]

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

$$(u_{yy})_{ij} \approx \frac{1}{h_y^2}\,\left(I - \frac{1}{12}\delta_y^2\right)\delta_y^2 u_{ij}, \tag{1.4}$$

which are fourth-order accurate. However, the approximations given in (1.4) require a 9-point stencil, which is much more complex than the 5-point stencil required by the approximations in (1.3). 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, whereas only one boundary condition is given in (1.1).

To maintain a small finite difference stencil for efficient solution process, we can use compact finite difference algorithm [2–5] to construct higher-order approximations for the spatial derivatives. For example, the formulas in (1.4) can be represented by the Padé approximation

$$(u_{xx})_{ij} = \frac{\delta_x^2}{h_x^2\left(1 + \frac{1}{12}\delta_x^2\right)}u_{ij}, \qquad (u_{yy})_{ij} = \frac{\delta_y^2}{h_y^2\left(1 + \frac{1}{12}\delta_y^2\right)}u_{ij}. \tag{1.5}$$

Note that if we expand $1/(1 + \frac{1}{12}\delta_x^2)$ and $1/(1 + \frac{1}{12}\delta_y^2)$ into a power series in terms of δ_x^2 and δ_y^2, respectively, the first two terms of $(u_{xx})_{ij}$ and $(u_{yy})_{ij}$ in (1.5) match the expressions in (1.4). If we set

$$(u_{xx})_{ij} = v_{ij}, \qquad (u_{yy})_{ij} = w_{ij},$$

and apply $1 + \frac{1}{12}\delta_x^2$ and $1 + \frac{1}{12}\delta_y^2$ to both sides of the two equations in (1.5), respectively, then the following expressions

$$\frac{1}{12}v_{i+1,j} + \frac{10}{12}v_{i,j} + \frac{1}{12}v_{i-1,j} = \frac{1}{h_x^2}\,(u_{i+1,j} - 2u_{i,j} + u_{i-1,j}), \tag{1.6}$$

$$\frac{1}{12}w_{i,j+1} + \frac{10}{12}w_{i,j} + \frac{1}{12}w_{i,j-1} = \frac{1}{h_y^2}\,(u_{i,j+1} - 2u_{i,j} + u_{i,j-1}). \tag{1.7}$$

provide fourth-order approximations to u_{xx} and u_{yy}, respectively. Both Eq. (1.6) and (1.7) result in systems of tridiagonal equations along j-lines and i-lines for solving the second derivatives u_{xx} and u_{yy}, respectively. Combined with the standard finite difference approximation in the temporal dimension, such as the Crank-Nicolson scheme, of the original PDE

$$\frac{u_{ij}^{n+1} - u_{ij}^{n}}{\Delta t} = \frac{a}{2}\,(v_{ij}^{n+1} + v_{ij}^{n}) + \frac{b}{2}\,(w_{ij}^{n+1} + w_{ij}^{n}) + \frac{1}{2}\,(f_{ij}^{n+1} + f_{ij}^{n}) \tag{1.8}$$

or

$$\frac{u_{ij}^{n+1} - u_{ij}^{n}}{\Delta t} = \frac{a}{2}\,(v_{ij}^{n+1} + v_{ij}^{n}) + \frac{b}{2}\,(w_{ij}^{n+1} + w_{ij}^{n}) + f_{ij}^{n+\frac{1}{2}}, \tag{1.9}$$

we have the complete system of Eqs. (1.6)–(1.8) for calculating solutions with fourth-order accuracy in space using a five-point stencil.

This approach, while maintaining a five-point stencil in space, requires the solution of coupled system of Eqs. (1.6)–(1.8) at each grid point. With p equations in the original system (1.1), a total of $3p$ coupled equations need to be solved at each grid point. If an operator-splitting type method is used to turn Eq. (1.8) into two separate equations, one along each of the x and y dimensions, then a system of $2p$ coupled equations need to be solved at each grid point in each step of the splitting method [6]. Furthermore, the formulation of Eqs. (1.6) and (1.7) requires boundary conditions for u_{xx} and u_{yy}, which are usually not known. Therefore, additional one-sided approximations have to be used to approximate the boundary conditions for u_{xx} and u_{yy} 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.

To simplify the computation, two methods for eliminating the second derivatives in (1.8) were discussed in [2]. One is the explicit elimination in which the second derivatives are represented in terms of the first derivatives and the function values. This approach works for the equations that involve both first and second derivative, such as the convection-diffusion equations. For reaction-diffusion equations that do not have the first-order derivatives, this approach will not improve computational efficiency because it eliminates the second derivatives by introducing the first derivatives into the equation.

The other method is the implicit elimination of the second derivative using the relations (1.6) and (1.7). It works well in one-dimensional cases. For example, for the equation

$$u_t = u_{xx},$$

we have

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = (u_{xx})_i^{n+1} = v_i^{n+1}, \tag{1.10}$$

where n and i represent the time step and spatial grid points, respectively. By combining (1.10) at points $i-1, i$, and $i+1$, and using relation (1.6), we can eliminate the second derivative and obtain

$$\frac{1}{12}\left(\frac{u_{i+1}^{n+1} - u_{i+1}^{n}}{\Delta t}\right) + \frac{10}{12}\left(\frac{u_i^{n+1} - u_i^{n}}{\Delta t}\right) + \frac{1}{12}\left(\frac{u_{i-1}^{n+1} - u_{i-1}^{n}}{\Delta t}\right)$$
$$= \frac{1}{12}v_{i+1}^{n+1} + \frac{10}{12}v_i^{n+1} + \frac{1}{12}v_{i-1}^{n+1} = \frac{1}{h_x^2}\,(u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}),$$

which is fourth-order accurate in space using a three-point stencil.

This approach does not apply directly to the general form of the two-dimensional Eq. (1.2) since the elimination of both u_{xx} and u_{yy} will create a nine-point stencil. If the ADI [7, 8] algorithm is used, then the solution procedure is

$$\frac{u_{i,j}^{n+\frac{1}{2}} - u_{i,j}^{n}}{\frac{\Delta t}{2}} = a(u_{xx})_{i,j}^{n+\frac{1}{2}} + b(u_{yy})_{i,j}^{n} + f_{i,j}^{n+\frac{1}{2}} \tag{1.11}$$

$$\frac{u_{i,j}^{n+1} - u_{i,j}^{n+\frac{1}{2}}}{\frac{\Delta t}{2}} = a(u_{xx})_{i,j}^{n+\frac{1}{2}} + b(u_{yy})_{i,j}^{n+1} + f_{i,j}^{n+\frac{1}{2}}, \tag{1.12}$$

where the term $(u_{xx})_{i,j}^{n+\frac{1}{2}}$ in (1.11) can be eliminated by combining three equations at points $(i-1,j)$, (i,j), and $(i+1,j)$, and the term $(u_{yy})_{i,j}^{n+\frac{1}{2}}$ in (1.12) can be eliminated by combining three equations at points $(i,j-1)$, (i,j), and $(i,j+1)$. The new systems of equations are

$$\begin{aligned}
&\frac{1}{12}\left(\frac{u_{i+1,j}^{n+\frac{1}{2}} - u_{i+1,j}^{n}}{\frac{\Delta t}{2}}\right) + \frac{10}{12}\left(\frac{u_{i,j}^{n+\frac{1}{2}} - u_{i,j}^{n}}{\frac{\Delta t}{2}}\right) + \frac{1}{12}\left(\frac{u_{i-1,j}^{n+\frac{1}{2}} - u_{i-1,j}^{n}}{\frac{\Delta t}{2}}\right) \\
&= \frac{a}{h_x^2}\left(u_{i+1,j}^{n+\frac{1}{2}} - 2u_{i,j}^{n+\frac{1}{2}} + u_{i-1,j}^{n+\frac{1}{2}}\right) + \frac{1}{12}b(u_{yy})_{i+1,j}^{n} + \frac{10}{12}b(u_{yy})_{i,j}^{n} + \frac{1}{12}b(u_{yy})_{i-1,j}^{n} \\
&+ \frac{1}{12}f_{i+1,j}^{n+\frac{1}{2}} + \frac{10}{12}f_{i,j}^{n+\frac{1}{2}} + \frac{1}{12}f_{i-1,j}^{n+\frac{1}{2}},
\end{aligned} \tag{1.13}$$

$$\begin{aligned}
&\frac{1}{12}\left(\frac{u_{i,j+1}^{n+1} - u_{i,j+1}^{n+\frac{1}{2}}}{\frac{\Delta t}{2}}\right) + \frac{10}{12}\left(\frac{u_{i,j}^{n+1} - u_{i,j}^{n+\frac{1}{2}}}{\frac{\Delta t}{2}}\right) + \frac{1}{12}\left(\frac{u_{i,j-1}^{n+1} - u_{i,j-1}^{n+\frac{1}{2}}}{\frac{\Delta t}{2}}\right) \\
&= \frac{1}{12}a(u_{xx})_{i,j+1}^{n+\frac{1}{2}} + \frac{10}{12}a(u_{xx})_{i,j}^{n+\frac{1}{2}} + \frac{1}{12}a(u_{xx})_{i,j-1}^{n+\frac{1}{2}} + \frac{b}{h_y^2}\left(u_{i,j+1}^{n+1} - 2u_{i,j}^{n+1} + u_{i,j-1}^{n+1}\right) \\
&+ \frac{1}{12}f_{i,j+1}^{n+\frac{1}{2}} + \frac{10}{12}f_{i,j}^{n+\frac{1}{2}} + \frac{1}{12}f_{i,j-1}^{n+\frac{1}{2}}.
\end{aligned} \tag{1.14}$$

However, the terms $(u_{yy})_{i-1,j}^{n}$, $(u_{yy})_{i,j}^{n}$, and $(u_{yy})_{i+1,j}^{n}$ in (1.13) and $(u_{xx})_{i,j-1}^{n+\frac{1}{2}}$, $(u_{xx})_{i,j}^{n+\frac{1}{2}}$, and $(u_{xx})_{i,j+1}^{n+\frac{1}{2}}$ in (1.14) still need to be calculated (though no longer coupled with the calculation of u_{ij}^{n} and $u_{ij}^{n+\frac{1}{2}}$), which requires solving systems of tri-diagonal equations given by (1.6) and (1.7) using the calculated solutions $u_{i,j}^{n}$ and $u_{i,j}^{n+\frac{1}{2}}$, respectively. Furthermore, one can not avoid dealing with the approximations of boundary conditions for $(u_{yy})_{i,j}^{n}$ and $(u_{xx})_{i,j}^{n+\frac{1}{2}}$. Finally, the generalization of this approach to three-dimensional problems is not obvious, since the straightforward fraction-time step (using $\frac{1}{3}\Delta t$) formulation

$$\frac{u_{i,j,k}^{n+\frac{1}{3}} - u_{i,j,k}^{n}}{\frac{\Delta t}{3}} = (u_{xx})_{i,j,k}^{n+\frac{1}{3}} + (u_{yy})_{i,j,k}^{n} + (u_{zz})_{i,j,k}^{n},$$

$$\frac{u_{i,j,k}^{n+\frac{2}{3}} - u_{i,j,k}^{n+\frac{1}{3}}}{\frac{\Delta t}{3}} = (u_{xx})_{i,j,k}^{n+\frac{1}{3}} + (u_{yy})_{i,j,k}^{n+\frac{2}{3}} + (u_{zz})_{i,j,k}^{n+\frac{1}{3}},$$

$$\frac{u_{i,j,k}^{n+1}-u_{i,j,k}^{n+\frac{2}{3}}}{\frac{\Delta t}{3}}=(u_{xx})_{i,j,k}^{n+\frac{2}{3}}+(u_{yy})_{i,j,k}^{n+\frac{2}{3}}+(u_{zz})_{i,j,k}^{n+1}$$

for three-dimensional problems is only conditionally stable and first-order accurate in time.

In this article, we will use a different approach to eliminate the second-order derivatives while maintaining a compact five-point stencil. It is based on the approximate factorization of the finite difference operators, which only requires solutions of systems of tri-diagonal equations. Furthermore, there is no need to introduce approximations for the boundary conditions of the second derivatives. The approach can be generalized to system of reaction-diffusion equations with nonlinear reaction terms. It also allows discontinuity in the initial and boundary condition as discussed in [9]. The new approach based on the approximate factorization will be discussed in the next section. The application to system 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.

II. EFFICIENT FOURTH-ORDER ALGORITHM BASED ON APPROXIMATE FACTORIZATION

We start from the Crank-Nicolson algorithm for Eq. (1.2) on a rectangular grid $(x_i, y_j), i = 0, \ldots, M, j = 0, \ldots, N$,:

$$\frac{u_{i,j}^{n+1}-u_{i,j}^{n}}{\Delta t}=\frac{a}{2}\left((u_{xx})_{i,j}^{n+1}+(u_{xx})_{i,j}^{n}\right)+\frac{b}{2}\left((u_{yy})_{i,j}^{n+1}+(u_{yy})_{i,j}^{n}\right)+\frac{1}{2}\left(f_{i,j}^{n+1}+f_{i,j}^{n}\right),$$
$$i=0,\ldots,M, j=0,\ldots,N. \quad (2.1)$$

The standard discretization is

$$\frac{u_{i,j}^{n+1}-u_{i,j}^{n}}{\Delta t}=\frac{a}{2h_x^2}\delta_x^2(u_{i,j}^{n+1}+u_{i,j}^{n})+\frac{b}{2h_y^2}\delta_y^2(u_{i,j}^{n+1}+u_{i,j}^{n})+\frac{1}{2}\left(f_{i,j}^{n+1}+f_{i,j}^{n}\right), \quad (2.2)$$

which is known to be second-order accurate in both time and space. If the fourth-order Padé approximation (1.5) is used to replace u_{xx} and u_{yy}, then the following algorithm

$$u_{i,j}^{n+1}-u_{i,j}^{n}=\frac{r_x}{2}\,\frac{\delta_x^2}{1+\frac{\delta_x^2}{12}}\,(u_{i,j}^{n+1}+u_{i,j}^{n})$$
$$+\frac{r_y}{2}\,\frac{\delta_y^2}{1+\frac{\delta_y^2}{12}}\,(u_{i,j}^{n+1}+u_{i,j}^{n})+\frac{\Delta t}{2}\left(f_{i,j}^{n+1}+f_{i,j}^{n}\right), \quad (2.3)$$

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

$$\left(I-\frac{r_x}{2}\,\frac{\delta_x^2}{1+\frac{\delta_x^2}{12}}-\frac{r_y}{2}\,\frac{\delta_y^2}{1+\frac{\delta_y^2}{12}}\right)u_{i,j}^{n+1}$$
$$=\left(I+\frac{r_x}{2}\,\frac{\delta_x^2}{1+\frac{\delta_x^2}{12}}+\frac{r_y}{2}\,\frac{\delta_y^2}{1+\frac{\delta_y^2}{12}}\right)u_{i,j}^{n}+\frac{\Delta t}{2}\left(f_{i,j}^{n+1}+f_{i,j}^{n}\right), \quad (2.4)$$

which can be approximately factorized as

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

The difference between (2.4) and (2.5) is

$$\frac{r_x r_y}{4}\,\frac{\delta_x^2}{\left(1+\frac{\delta_x^2}{12}\right)}\,\frac{\delta_y^2}{\left(1+\frac{\delta_y^2}{12}\right)}\,(u_{i,j}^{n+1}-u_{i,j}^{n})\approx\frac{r_x r_y}{4}\,\frac{\delta_x^2}{\left(1+\frac{\delta_x^2}{12}\right)}\,\frac{\delta_y^2}{\left(1+\frac{\delta_y^2}{12}\right)}\,(u_t)_{i,j}^{n+\frac{1}{2}}\Delta t$$
$$=\frac{ab\Delta t^3}{4}\,\frac{\delta_x^2}{h_x^2\left(1+\frac{\delta_x^2}{12}\right)}\,\frac{\delta_y^2}{h_y^2\left(1+\frac{\delta_y^2}{12}\right)}\,(u_t)_{i,j}^{n+\frac{1}{2}}=\frac{ab\Delta t^3}{4}\,(u_{txxyy}+O(h_x^4)+O(h_y^4))=O(\Delta t^3),$$

provided u_{txxyy} is bounded. This additional error is of the same order as the truncation error in the original algorithm (2.4). Since the operators in (2.5) commute, we can simplify the algorithm by applying $(1+\frac{\delta_x^2}{12})(1+\frac{\delta_y^2}{12})$ to both sides of (2.5), which leads to

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

Equation (2.6) can be solved in two steps as

$$\left(1+\frac{\delta_x^2}{12}-\frac{r_x}{2}\delta_x^2\right)u_{i,j}^{*}=\left(1+\frac{\delta_x^2}{12}+\frac{r_x}{2}\delta_x^2\right)\left(1+\frac{\delta_y^2}{12}+\frac{r_y}{2}\delta_y^2\right)u_{i,j}^{n}$$
$$+\frac{\Delta t}{2}\,\left(1+\frac{\delta_x^2}{12}\right)\left(1+\frac{\delta_y^2}{12}\right)(f_{i,j}^{n+1}+f_{i,j}^{n}), \quad (2.7)$$

$$\left(1+\frac{\delta_y^2}{12}-\frac{r_y}{2}\delta_y^2\right)u_{i,j}^{n+1}=u_{i,j}^{*}. \quad (2.8)$$

The solutions to Eqs. (2.7) and (2.8) can be computed by solving only tridiagonal equations since the left hand sides of (2.7) and (2.8) involve only the three-point central difference operators δ_x^2 and δ_y^2 as defined in (1.3). Although the right-hand side of (2.7) involves the product of operators $\delta_x^2\delta_y^2$, it does not complicate the solution process since it is applied to the known solution values from the previous time step.

While solving Eq. (2.7), we need boundary conditions for $u_{0,j}^{*}$ and $u_{M+1,j}^{*}, j=1,\ldots,N$. These conditions can be obtained from Eq. (2.8) by setting $i=0$ and $i=M+1$, respectively:

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

$$u_{M+1,j}^{*}=\left(1+\frac{\delta_y^2}{12}-\frac{r_y}{2}\delta_y^2\right)u_{M+1,j}^{n+1}. \quad (2.9)$$

Since the spatial discretization used in obtaining (2.8) is fourth-order accurate in space, the boundary conditions given by (2.9) have the same spatial accuracy as (2.8). 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 [2–5].

III. EQUATIONS WITH NONLINEAR REACTION TERMS

For a system of equations with linear diffusion and nonlinear reaction as given in (1.1), algorithm (2.7) and (2.8) will result in the following system of equations:

$$\left(I+\frac{\delta_x^2}{12}-\frac{\mathbf{r}_x}{2}\delta_x^2\right)\mathbf{w}_{i,j}^{*}=\left(I+\frac{\delta_x^2}{12}+\frac{\mathbf{r}_x}{2}\delta_x^2\right)\left(I+\frac{\delta_y^2}{12}+\frac{\mathbf{r}_y}{2}\delta_y^2\right)\mathbf{w}_{i,j}^{n} + \frac{\Delta t}{2}\left(I+\frac{\delta_x^2}{12}\right)\left(I+\frac{\delta_y^2}{12}\right)(\mathbf{f}_{i,j}^{n+1}+\mathbf{f}_{i,j}^{n}), \quad (3.1)$$

$$\left(I+\frac{\delta_y^2}{12}-\frac{\mathbf{r}_y}{2}\delta_y^2\right)\mathbf{w}_{i,j}^{n+1}=\mathbf{w}_{i,j}^{*}, \quad (3.2)$$

where

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

Note that Eq. (3.1) contains solutions $\mathbf{w}_{i,j}^{n}$, $\mathbf{w}_{i,j}^{*}$, and $\mathbf{w}_{i,j}^{n+1}$ (implicitly in $\mathbf{f}_{i,j}^{n+1}$). Since both $\mathbf{w}_{i,j}^{*}$ and $\mathbf{w}_{i,j}^{n+1}$ are unknown, Eq. (3.1) can not be linearized by simply using Newton's method or its variations to expand $\mathbf{f}_{i,j}^{n+1}$ at $\mathbf{w}_{i,j}^{n}$. In [6], a predictor-corrector type algorithm was used to overcome this difficulty. The algorithm begins by using the expansion

$$\mathbf{f}_{i,j}^{n+1}=\mathbf{f}_{i,j}^{n}+\mathbf{J}_{i,j}^{n}(\mathbf{w}_{i,j}^{n+1}-\mathbf{w}_{i,j}^{n})+\Delta t(\mathbf{f}_t)_{i,j}^{n}, \quad (3.3)$$

where $\mathbf{J}_{i,j}^{n}=(\partial\mathbf{f}/\partial\mathbf{w})_{i,j}^{n}$ is the local Jacobian matrix. The algorithm in (3.1) and (3.2) can then be written as

$$\left(I+\frac{\delta_x^2}{12}-\frac{\mathbf{r}_x}{2}\delta_x^2\right)\mathbf{w}_{i,j}^{*}=\left(I+\frac{\delta_x^2}{12}+\frac{\mathbf{r}_x}{2}\delta_x^2\right)\left(I+\frac{\delta_y^2}{12}+\frac{\mathbf{r}_y}{2}\delta_y^2\right)\mathbf{w}_{i,j}^{n} + \frac{\Delta t}{2}\left(I+\frac{\delta_x^2}{12}\right)\left(I+\frac{\delta_y^2}{12}\right)(2\mathbf{f}_{i,j}^{n}+\Delta t(\mathbf{f}_t)_{ij}^{n}+\mathbf{e}_{ij}^{n+1}), \quad (3.4)$$

$$\left(I+\frac{\delta_y^2}{12}-\frac{\mathbf{r}_y}{2}\delta_y^2\right)\mathbf{w}_{i,j}^{n+1}=\mathbf{w}_{i,j}^{*}, \quad (3.5)$$

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

$$\left(I + \frac{\delta_x^2}{12} - \frac{\mathbf{r}_x}{2}\delta_x^2\right)\mathbf{w}_{i,j}^{*P} = \left(I + \frac{\delta_x^2}{12} + \frac{\mathbf{r}_x}{2}\delta_x^2\right)\left(I + \frac{\delta_y^2}{12} + \frac{\mathbf{r}_y}{2}\delta_y^2\right)\mathbf{w}_{i,j}^{n} + \frac{\Delta t}{2}\left(I + \frac{\delta_x^2}{12}\right)\left(I + \frac{\delta_y^2}{12}\right)(2\mathbf{f}_{i,j}^{n} + \Delta t(\mathbf{f}_t)_{ij}^{n}), \quad (3.6)$$

$$\left(I + \frac{\delta_y^2}{12} - \frac{\mathbf{r}_y}{2}\delta_y^2\right)\mathbf{w}_{i,j}^{(n+1)^P} = \mathbf{w}_{i,j}^{*P}. \quad (3.7)$$

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

$$\left(I + \frac{\delta_x^2}{12} - \frac{\mathbf{r}_x}{2}\delta_x^2\right)\mathbf{w}_{i,j}^{*k} = \left(I + \frac{\delta_x^2}{12} + \frac{\mathbf{r}_x}{2}\delta_x^2\right)\left(I + \frac{\delta_y^2}{12} + \frac{\mathbf{r}_y}{2}\delta_y^2\right)\mathbf{w}_{i,j}^{n} + \frac{\Delta t}{2}\left(I + \frac{\delta_x^2}{12}\right)\left(I + \frac{\delta_y^2}{12}\right)(2\mathbf{f}_{i,j}^{n} + \Delta t(\mathbf{f}_t)_{ij}^{n}) + \mathbf{e}_{ij}^{(n+1)^{(k-1)}}, \quad (3.8)$$

$$\left(I + \frac{\delta_y^2}{12} - \frac{r_y}{2}\delta_y^2\right)\mathbf{w}_{i,j}^{(n+1)^k} = \mathbf{w}_{i,j}^{*k}, \qquad k = 1, 2, \ldots, \quad (3.9)$$

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.

Here we introduce a more efficient way to eliminate $\mathbf{w}_{ij}^{n+1}$ from Eq. (3.1) by rewriting algorithm (3.1) and (3.2) as

$$\left(I + \frac{\delta_x^2}{12} - \frac{\mathbf{r}_x}{2}\delta_x^2\right)\mathbf{w}_{i,j}^{*} = \left(I + \frac{\delta_x^2}{12} + \frac{\mathbf{r}_x}{2}\delta_x^2\right)\left(I + \frac{\delta_y^2}{12} + \frac{\mathbf{r}_y}{2}\delta_y^2\right)\mathbf{w}_{i,j}^{n} + \frac{\Delta t}{2}\left(I + \frac{\delta_x^2}{12} + \frac{\mathbf{r}_x}{2}\delta_x^2\right)\left(I + \frac{\delta_y^2}{12}\right)\mathbf{f}_{i,j}^{n}, \quad (3.10)$$

$$\left(I + \frac{\delta_y^2}{12} - \frac{\mathbf{r}_y}{2}\delta_y^2\right)\mathbf{w}_{i,j}^{n+1} = \mathbf{w}_{i,j}^{*} + \frac{\Delta t}{2}\left(I + \frac{\delta_y^2}{12}\right)\mathbf{f}_{i,j}^{n+1}. \quad (3.11)$$

The difference between the algorithms in (3.1) and (3.2) and that in (3.10) and (3.11) is

$$\frac{\Delta t}{4}\mathbf{r}_x\delta_x^2\left(I + \frac{\delta_y^2}{12}\right)(\mathbf{f}_{i,j}^{n+1} - \mathbf{f}_{i,j}^{n}),$$

for which we have the estimate

$$\left\|\frac{\Delta t}{4}\mathbf{r}_x\delta_x^2\left(I + \frac{\delta_y^2}{12}\right)(\mathbf{f}_{i,j}^{n+1} - \mathbf{f}_{i,j}^{n})\right\| \approx O(\Delta t^3) + O(\Delta t^5).$$

This is in the same order as that of the original truncation error in algorithm (2.4). With this new formulation, Eq. (3.10) is linear and can be solved in a straightforward manner. Equation (3.11) can be linearized by Newton's method, or its variations, such as that given by (3.3). The new algorithm can then be written as

$$\left(I+\frac{\delta_x^2}{12}-\frac{\mathbf{r}_x}{2}\delta_x^2\right)\mathbf{w}_{i,j}^*=\left(I+\frac{\delta_x^2}{12}+\frac{\mathbf{r}_x}{2}\delta_x^2\right)\left(I+\frac{\delta_y^2}{12}+\frac{\mathbf{r}_y}{2}\delta_y^2\right)\mathbf{w}_{i,j}^n + \frac{\Delta t}{2}\left(I+\frac{\delta_x^2}{12}+\frac{\mathbf{r}_x}{2}\delta_x^2\right)\left(I+\frac{\delta_y^2}{12}\right)\mathbf{f}_{i,j}^n, \quad (3.12)$$

$$\left(I+\frac{\delta_y^2}{12}-\frac{\mathbf{r}_y}{2}\delta_y^2-\frac{\Delta t}{2}\left(I+\frac{\delta_y^2}{12}\right)\mathbf{J}_{i,j}^n\right)\mathbf{w}_{i,j}^{n+1} = \mathbf{w}_{i,j}^*+\frac{\Delta t}{2}\left(I+\frac{\delta_y^2}{12}\right)(\mathbf{f}_{i,j}^n-\mathbf{J}_{i,j}^n\mathbf{w}_{i,j}^n+\Delta t(\mathbf{f}_t)_{i,j}^n). \quad (3.13)$$

To achieve high accuracy for strongly nonlinear problems, Newton's iterations can be used to solve (3.13), which leads to

$$\left(I+\frac{\delta_x^2}{12}-\frac{\mathbf{r}_x}{2}\delta_x^2\right)\mathbf{w}_{i,j}^*=\left(I+\frac{\delta_x^2}{12}+\frac{\mathbf{r}_x}{2}\delta_x^2\right)\left(I+\frac{\delta_y^2}{12}+\frac{\mathbf{r}_y}{2}\delta_y^2\right)\mathbf{w}_{i,j}^n + \frac{\Delta t}{2}\left(I+\frac{\delta_x^2}{12}+\frac{\mathbf{r}_x}{2}\delta_x^2\right)\left(I+\frac{\delta_y^2}{12}\right)\mathbf{f}_{i,j}^n, \quad (3.14)$$

$$\left(I+\frac{\delta_y^2}{12}-\frac{\mathbf{r}_y}{2}\delta_y^2-\frac{\Delta t}{2}\left(I+\frac{\delta_y^2}{12}\right)\mathbf{J}_{i,j}^{(n+1)^{k-1}}\right)\mathbf{w}_{i,j}^{(n+1)^k} = \mathbf{w}_{i,j}^*+\frac{\Delta t}{2}\left(I+\frac{\delta_y^2}{12}\right)(\mathbf{f}_{i,j}^{(n+1)^{k-1}} - \mathbf{J}_{i,j}^{(n+1)^{k-1}}\mathbf{w}_{i,j}^{(n+1)^{k-1}}+\Delta t(\mathbf{f}_t)_{i,j}^n), \quad k=1,2,\ldots, \quad (3.15)$$

where $\mathbf{w}_{i,j}^{(n+1)^0}=\mathbf{w}_{ij}^n$. The boundary conditions for $\mathbf{w}_{0,j}^*$ and $\mathbf{w}_{M+1,j}^*$ can be handled in the same way as in the original algorithm using Eq. (3.11):

$$\mathbf{w}_{0,j}^*=\left(1+\frac{\delta_y^2}{12}-\frac{r_y}{2}\delta_y^2\right)\mathbf{w}_{0,j}^{n+1}-\frac{\Delta t}{2}\left(I+\frac{\delta_y^2}{12}\right)\mathbf{f}_{0,j}^{n+1},$$

$$\mathbf{w}_{M+1,j}^*=\left(1+\frac{\delta_y^2}{12}-\frac{r_y}{2}\delta_y^2\right)\mathbf{w}_{M+1,j}^{n+1}-\frac{\Delta t}{2}\left(I+\frac{\delta_y^2}{12}\right)\mathbf{f}_{M+1,j}^{n+1}. \quad (3.16)$$

Note that the system of equations in (3.12) are decoupled, which makes parallel implementation of this part of the algorithm easy. The equations in (3.13), however, are coupled due to the Jacobian matrix $\mathbf{J}_{i,j}^n$. Several methods have been proposed to further improve the computational efficiency. One way to decouple the solution process for (3.13) is to use only part of the local Jacobian matrix $\mathbf{J}_{i,j}^n$ [10, 11]. Let

$$\mathbf{J}_{i,j}^n=E_{i,j}^n+L_{i,j}^n+U_{i,j}^n,$$

where $E^n_{i,j}$, $L^n_{i,j}$, and $U^n_{i,j}$ represent the diagonal, the lower triangular, and the upper triangular part of the Jacobian matrix, respectively. The approximate local Jacobian matrix $\tilde{\mathbf{J}}^n_{i,j}$ is obtained by using one of the following:

- $\tilde{\mathbf{J}}^n_{i,j} = E^n_{i,j}$,
- $\tilde{\mathbf{J}}^n_{i,j} = E^n_{i,j} + L^n_{i,j}$,
- $\tilde{\mathbf{J}}^n_{i,j} = E^n_{i,j} + U^n_{i,j}$.

This approach, while decoupling the equations in system (3.13), will make it only first-order accurate in time. The original algorithm (3.13) is second-order accurate in time. Another approach was discussed in [12] that can maintain the original order of accuracy. It uses the upper triangular part of the Jacobian for one of the two steps and the lower triangular part for the other step. This makes the algorithm less amenable to parallel computations since the equations will have to be solved following a particular order, either backward or forward. A different strategy was discussed in [13] to completely decouple the equations in (3.13) while still maintaining the order of accuracy in time. The basic idea is to use proper extrapolations to represent $\mathbf{f}^{n+1}_{i,j}$ using solution values $\mathbf{w}^n_{i,j}$ and $\mathbf{w}^{n-1}_{i,j}$:

$$\begin{aligned}2\mathbf{f}^n_{i,j} - \mathbf{f}^{n-1}_{i,j} &= 2[\mathbf{f}^{n+1}_{i,j} + \mathbf{J}^{n+1}_{i,j}(\mathbf{w}^n_{i,j} - \mathbf{w}^{n+1}_{i,j}) - (\mathbf{f}_t)^{n+1}_{i,j}\Delta t + O(\Delta t^2)] \\ &\quad -[\mathbf{f}^{n+1}_{i,j} + \mathbf{J}^{n+1}_{i,j}(\mathbf{w}^{n-1}_{i,j} - \mathbf{w}^{n+1}_{i,j}) - 2(\mathbf{f}_t)^{n+1}_{i,j}\Delta t + O(\Delta t^2)] = \mathbf{f}^{n+1}_{i,j} + O(\Delta t^2), \quad (3.17)\end{aligned}$$

where we have used the relations

$$\mathbf{w}^n_{i,j} - \mathbf{w}^{n+1}_{i,j} = -(\mathbf{w}_t)^{n+1}_{i,j}\Delta t + O(\Delta t^2),$$

$$\mathbf{w}^{n-1}_{i,j} - \mathbf{w}^{n+1}_{i,j} = -2(\mathbf{w}_t)^{n+1}_{i,j}\Delta t + O(\Delta t^2).$$

This leads to

$$\begin{aligned}\frac{\mathbf{w}^{n+1}_{i,j} - \mathbf{w}^n_{i,j}}{\Delta t} &= \frac{1}{2}D_1((\mathbf{w}_{xx})^{n+1}_{i,j} + (\mathbf{w}_{xx})^n_{i,j}) + \frac{1}{2}D_2((\mathbf{w}_{yy})^{n+1}_{i,j} \\ &\quad + (\mathbf{w}_{yy})^n_{i,j}) + 2\mathbf{f}^n_{i,j} - \mathbf{f}^{n-1}_{i,j}. \quad (3.18)\end{aligned}$$

The p equations in (3.18) are now decoupled since D_1 and D_2 are diagonal matrices and the nonlinear term $\mathbf{f}$ is evaluated at the previous time levels. It is shown in [13] that the use of this extrapolation maintains the order of accuracy of the original algorithm (3.2). It also avoids the need of linearization for the nonlinear reaction term. The disadvantage is the stability concern caused by the explicit treatment of the reaction term. Following similar approach that led to (2.7) and (2.8) from (2.2), we obtain the following algorithm that is second-order accurate in time and fourth-order accurate in space:

$$\begin{aligned}\left(I + \frac{\delta_x^2}{12} - \frac{r_x}{2}D_1\delta_x^2\right)\mathbf{w}^*_{i,j} &= \left(I + \frac{\delta_x^2}{12} + \frac{r_x}{2}D_1\delta_x^2\right)\left(1 + \frac{\delta_y^2}{12} + \frac{r_y}{2}D_2\delta_y^2\right)\mathbf{w}^n_{i,j} \\ &\quad + \Delta t\left(I + \frac{\delta_x^2}{12}\right)\left(I + \frac{\delta_y^2}{12}\right)(2\mathbf{f}^n_{i,j} - \mathbf{f}^{n-1}_{i,j}), \quad (3.19)\end{aligned}$$

$$\left(I + \frac{\delta_y^2}{12} - \frac{r_y}{2}D_2\delta_y^2\right)\mathbf{w}^{n+1}_{i,j} = \mathbf{w}^*_{i,j}. \quad (3.20)$$

Because both D_1 and D_2 are diagonal, the equations in (3.19) and (3.20) are decoupled and can be written as

$$\left(I+\frac{\delta_x^2}{12}-\frac{r_x d_l'}{2}\delta_x^2\right)(w_l)_{i,j}^{*}=\left(I+\frac{\delta_x^2}{12}+\frac{r_x d_l'}{2}\delta_x^2\right)\left(1+\frac{\delta_y^2}{12}+\frac{r_y d_l''}{2}\delta_y^2\right)(w_l)_{i,j}^{n}$$
$$+\Delta t\left(I+\frac{\delta_x^2}{12}\right)\left(I+\frac{\delta_y^2}{12}\right)\frac{1}{2}\left(3(f_l)_{i,j}^{n}-(f_l)_{i,j}^{n-1}\right),$$

$$\left(I+\frac{\delta_y^2}{12}-\frac{r_y d_l''}{2}\delta_y^2\right)(w_l)_{i,j}^{n+1}=(w_l)_{i,j}^{*},\qquad l=1,\ldots,d,$$

where w_l and f_l are the lth components of the vectors $\mathbf{w}$ and $\mathbf{f}$, respectively, and d_l' and d_l'' are the diagonal elements of matrices D_1 and D_2, respectively.

IV. HIGHER-ORDER ACCURACY IN THE TEMPORAL DIMENSION

The algorithm given in (3.12) and (3.13) 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 five-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 (2.2) to (2.7) and (2.8), 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. Because the Crank-Nicolson algorithm has a temporal truncation error in the form of $O(\Delta t^2)+O(\Delta t^4)$, we use

$$\mathbf{w}=\frac{4\mathbf{w}^{\frac{h}{2}}-\mathbf{w}^{h}}{3} \tag{4.1}$$

to eliminate the term $O(\Delta t^2)$, where $\mathbf{w}^{\frac{h}{2}}$ 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.

Note that if explicit extrapolation is used to decouple the equations, the order of extrapolation in (3.17) needs to be increased to ensure that the temporal error in the final solution is of order $O(\Delta t^2)+O(\Delta t^4)$.

V. NUMERICAL EXPERIMENT

We discuss two numerical examples here: One with an analytic solution against which we can compare the numerical solution to demonstrate the order of accuracy 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.

TABLE I. Maximum error between the calculated solution and the exact solution at $T = 1.0$.

h	0.1	0.05	0.025	0.0125	0.00625
e_1	3.634e−05	8.869e−06	1.998e−06	4.628e−07	1.141e−07
e_1/h^2	3.634e−03	3.548e−03	3.197e−03	2.962e−03	2.921e−03
e_2	4.283e−08	2.624e−09	1.624e−10	1.011e−11	6.292e−13
e_2/h^4	4.283e−04	4.198e−04	4.157e−04	4.141e−04	4.124e−04

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. $\Delta t = \Delta x = \Delta y = h$.

Example 1. The equations to be solved are

$$u_t = u_{xx} + u_{yy} + u^2(1 - v^2) + f(x, y, t),$$

$$v_t = v_{xx} + v_{yy} + v^2(1 - u^2) + g(x, y, t), \qquad 0 < x, y < 1, t > 0,$$

where $f(x, y, t), g(x, y, t)$, and the boundary and initial conditions have been selected to accommodate the exact solutions of $u = e^{-t}\sin(x)\sin(y)$ and $v = e^{-2t}\sin(2x)\sin(2y)$. The data in Table I show the maximum error between the calculated solution and the exact solution at $T = 1$. The discretization grid is $\Delta x = \Delta y = \Delta t = h$, and the algorithm given by (3.14) and (3.15) was used with a full analytic Jacobian matrix and Newton's iterations. Because the effect of using various approximate Jacobian matrices has been extensively studied, for example in [6, 10, 11], we will not repeat those results here. 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 I that the error represented by e_1 shows a second-order decrease, whereas 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 $\Delta t, \Delta x$, and $\Delta y, e_1$ is reduced only by a factor of 4, whereas e_2 is reduced by a factor of 16.

Table II shows similar results as those represented by e_1 in Table I, except Δt is refined by a factor of 4 each time, whereas Δx and Δy 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 I. For example, if the error represented by e_3 is to be reduced to the same level as that of e_2 at $\Delta x = \Delta y = 0.00625$, we must use $\Delta t = 0.000015$, more than 400 times smaller than that used for achieving e_2. This demonstrates the effectiveness of the new algorithm with fourth-order accuracy in both space and time.

TABLE II. Maximum error between the calculated solution and the exact solution at $T = 1.0$.

h	0.1	0.05	0.025	0.0125	0.00625
Δt	0.1	0.025	0.00625	0.0015625	0.000390625
e_3	3.634e−05	1.995e−06	1.140e−07	7.082e−09	4.403e−10
e_3/h^4	3.634e−01	3.192e−01	2.918e−01	2.901e−01	2.886e−01

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 $\Delta t = \Delta x = \Delta y = 0.1$. In each refinement step, Δt is reduced by a factor of 4 and $\Delta t = \Delta y = h$ is reduced by a factor of 2.

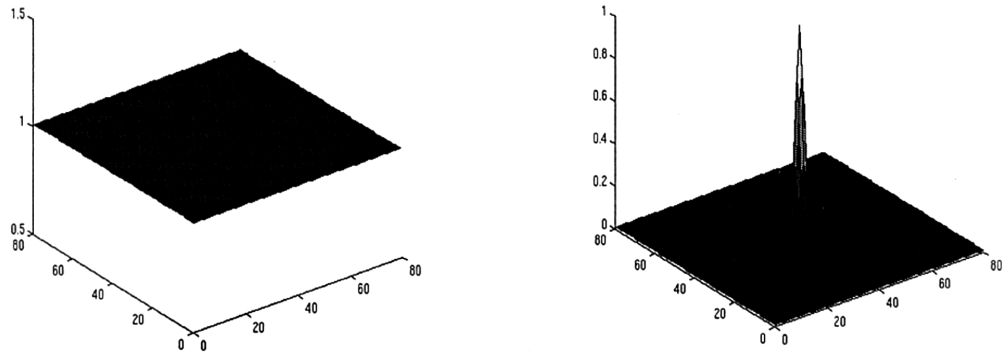

FIG. 1. Initial conditions for Eq. (5.1). (a) u at $t = 0$; (b) v at $t = 0$.

Example 2. The equations to be solved are

$$u_t = u_{xx} + u_{yy} - u^4 v,$$

$$v_t = v_{xx} + v_{yy} + u^4 v - 0.5v, \qquad 0 < x, y < 1, t > 0, \tag{5.1}$$

with the following boundary conditions:

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

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

The initial conditions for u and v are

$$u(x, y, 0) = 1.0, \qquad v(x, y, 0) = e^{-1600(x^2+y^2)},$$

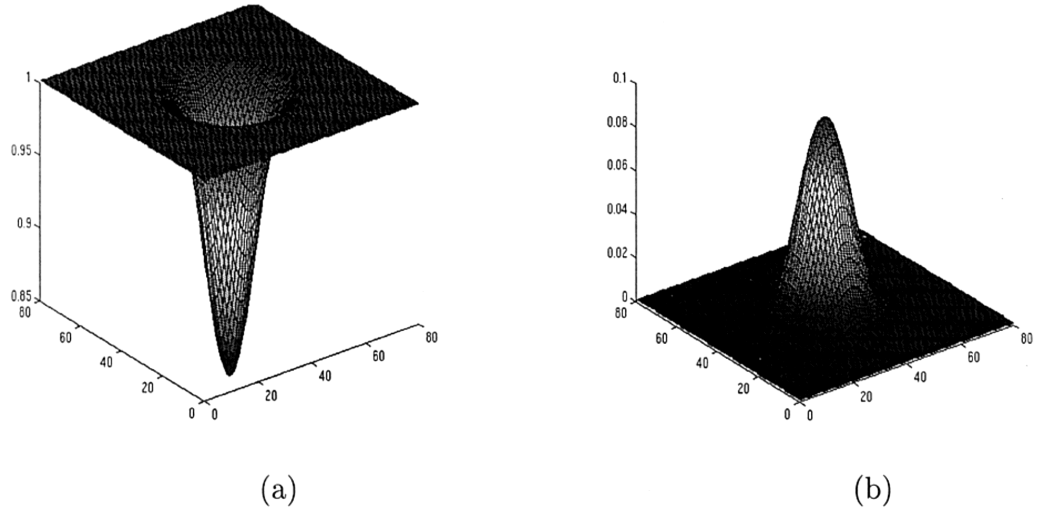

FIG. 2. Solutions to Eq. (5.1). (a) u at $t = 4$; (b) v at $t = 4$.

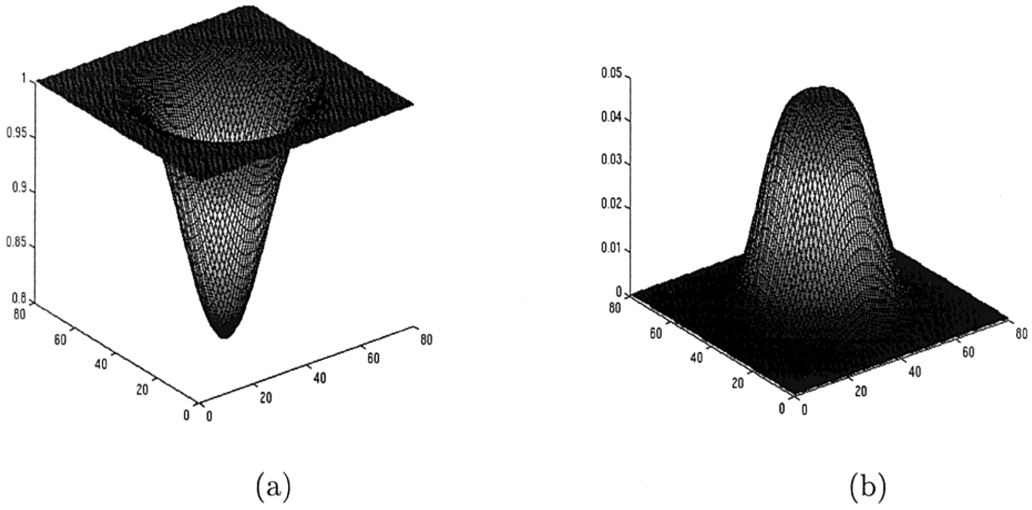

FIG. 3. Solutions to Eq. (5.1). (a) u at $t = 8$; (b) v at $t = 8$.

which are plotted in Fig. 1(a) and 1(b), respectively.

Figures 2, 3, and 4 show the numerical solutions of Eq. (5.1) at time $T = 4, 8$, and 12, respectively. The computations were carried using the algorithm given in (3.14) and (3.15) 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. Because of 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 because of diffusion and reaction. Note, however, Fig. 4b shows a crater-like

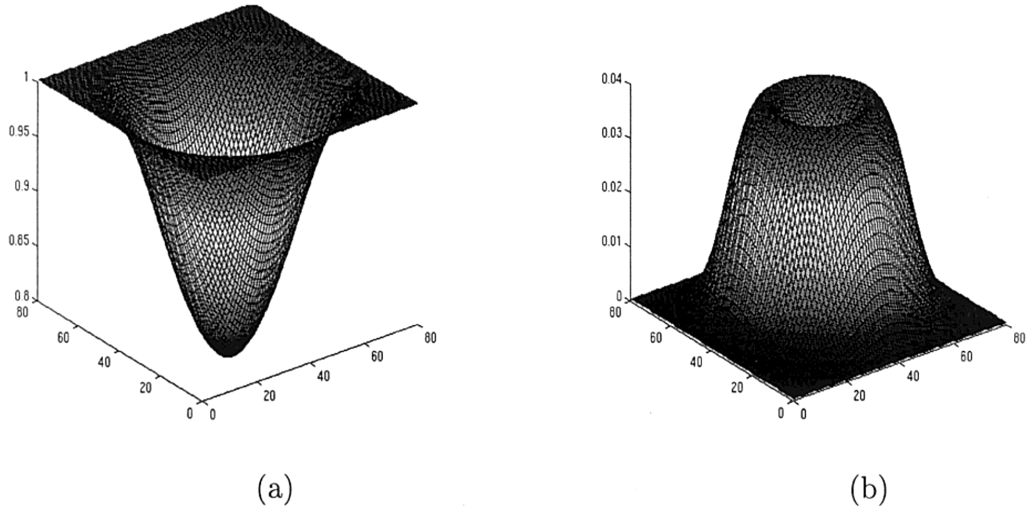

FIG. 4. Solutions to Eq. (5.1). (a) u at $t = 12$; (b) v at $t = 12$.

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.

VI. CONCLUSION

An efficient high-order algorithm for solving systems of reaction-diffusion equations with linear diffusion and nonlinear reaction is discussed in this article. It is fourth-order accurate in time and space and uses a compact five-point finite difference stencil for two-dimensional problems. Numerical results have demonstrated the high-order accuracy in both temporal and spatial dimensions. The authors plan to generalize the new method to three-dimensional problems.

We thank the anonymous referees for their constructive comments on the revision of the manuscript.

References

1. B. Gustafson, H. Kreiss, and J. Oliger, Time Dependent Problems and Difference Methods, John Wiley & Sons, New York, 1995.
2. Yves Adam, Highly accurate compact implicit methods and boundary conditions, Journal of Computational Physics 24, 10–22 (1977).
3. Richard S. Hirsch, Higher order accurate difference solutions of fluid mechanics problems by a compact differencing technique, Journal Of Computational Physics 19, 90–109 (1975).
4. S. K. Lele, Compact finite difference schemes with spectral-like resolution, J. Computational Physics 103, 16–42 (1992).
5. R. V. Wilson, A. O. Demuren, and M. Carpenter, Higher-order compact schemes for numerical simulation of incompressible flows, ICASE report, No. 98-13, (1998).
6. J. I. Ramos, Implicit, compact, linearized θ-methods with factorization for multidimensional reaction-diffusion equations, Applied Mathematics and Computation 94, 17–43 (1998).
7. J. Douglas Jr., On the numerical integration of $\partial^2 u/\partial x^2 + \partial^2 u/\partial y^2 = \partial u/\partial t$ by implicit methods, J. of Society for Industrial and Applied Mathematics 3, 42–65 (1955).
8. D. W. Peaceman and H. H. Rachford Jr., The numerical solution of parabolic and elliptic differential equations, J. Society of Industrial and Applied Mathematics 3, 28–41 (1955).
9. J. D. Lawson and J. L. Morris, The extrapolation of first-order methods for parabolic partial differential equations I, SIAM J. Numerical Analysis 15, 1212–1124 (1978).
10. J. I. Ramos, Linearization methods for reaction-diffusion equations: 1-D problems, Applied Mathematics and Computation 88, 199–224 (1997).
11. J. I. Ramos, Linearization methods for reaction-diffusion equations: multi-dimensional problems, Applied Mathematics and Computation 88, 225–254 (1997).
12. E. Hairer and G. Wanner, Solving ordinary differential equations, II, stiff and algebraic problems, 2nd Ed., Springer-Verlag, Berlin, 1996.
13. L. Cao and J. Zhu, Efficient and accurate local time stepping algorithms for multi-rate problems, Proceedings of the Eighth International Colloquium on Differential Equations, VSP International Science Publishes, 97–104 (1998).