IJMMS 2004:17, 881–899
PII. S0161171204211218
http://ijmms.hindawi.com
© Hindawi Publishing Corp.

AN EFFICIENT APPROACH FOR SOLVING A CLASS OF NONLINEAR 2D PARABOLIC PDEs

DONGJIN KIM and WLODEK PROSKUROWSKI

Received 15 November 2002

We consider a class of nonlinear 2D parabolic equations that allow for an efficient application of an operator splitting technique and a suitable linearization of the discretized problem. We apply our scheme to study the finite extinction phenomenon for the porous-medium equation with strong absorption. A comparison of accuracy and computational efficiency of the resulting algorithms for several test problems is presented.

2000 Mathematics Subject Classification: 35K55, 65M06.

1. Introduction. We study a class of nonlinear 2D parabolic PDEs where the nonlinearity is a power of the solution. We apply a linearization and an operator splitting technique. We use our algorithm for computing to high accuracy the extinction time for the porous-medium equation with strong absorption.

We use a finite-difference method and an implicit scheme of the Crank-Nicolson type in view of its superior stability properties. Then we are led to systems of nonlinear algebraic equations. These can be solved using Newton's type methods, which is costly. Instead, we choose to use a linearization approach that eliminates the need for iterations at each temporal step. Solving the arising large linear system of algebraic equations straightforwardly using direct methods would be extremely inefficient. Thus, the linearization by itself does not solve the main difficulty, the high computational cost, in dealing efficiently with 2D problems. To remedy this, to the linearized equations we apply the operator splitting technique that allows for computationally more efficient solution processes.

The nonlinearity in the considered PDEs has the form of powers of the solution function. This is a quite common situation in many applications, especially, in the porous-medium type equations, see [1, 4], for which various numerical schemes have been introduced, see, for example, [4, 6, 7]. We apply our scheme in the porous-medium equation, and we use it to study the finite-extinction phenomenon for the porous-medium equation with strong absorption.

Throughout the paper, we use the following finite-difference operator notation. A continuous function $u = u(x,y,t)$ is discretized on an equidistant spatio-temporal grid $(x_i, y_j, t_n) = (ih, jh, nk)$, $1 \le i, j \le M$, $0 \le n \le N$, where $h = \Delta x = \Delta y$ and $k = \Delta t$ are step sizes in spatial and temporal directions and $n = 0$ is for an initial function. The discrete function at the point (x_i, y_j, t_n) is denoted by $u_{i,j,n}$. We define U_n as a discrete column vector of order M^2 at the time level n:

$$U_n = \left(u_{1,1,n}, u_{2,1,n}, \dots, u_{M,M,n}\right)^T. \tag{1.1}$$

The forward difference operators of each direction are defined as

$$\begin{aligned} D_{+t} : D_{+t}u_n &= \frac{u_{n+1}-u_n}{k}, \\ D_{+x} : D_{+x}u_i &= \frac{u_{i+1}-u_i}{h}, \\ D_{+y} : D_{+y}u_j &= \frac{u_{j+1}-u_j}{h}. \end{aligned} \tag{1.2}$$

To simplify the notation, indices over variables that do not change under the particular operator (i.e., temporarily frozen) are suppressed. The backward difference operator is defined analogously as

$$D_{-x} : D_{-x}u_i = \frac{u_i-u_{i-1}}{h}. \tag{1.3}$$

These one-sided operators are first-order accurate approximations of the derivatives, that is, $(D_{+x}-D)u_i = \mathbb{O}(h)$, and so forth. Additionally, we use the second-order operator

$$D_{+x}D_{-x}u_i = D_{-x}D_{+x}u_i = \frac{u_{i+1}-2u_i+u_{i-1}}{h^2}, \tag{1.4}$$

which is second-order accurate, that is, $(D_{+x}D_{-x}-D^2)u_i = \mathbb{O}(h^2)$. Note that the result of these operators is a column vector or a matrix depending on contexts when the operators are applied to column vectors.

This paper is arranged as follows. In Section 2, we present our model problem. Section 3 is devoted to the analysis of computational efficiency of the different algorithms for porous-medium equation. Numerical experiments for several problems with various conditions are reported in Section 4, while Section 5 contains the concluding remarks.

2. Model problem: porous-medium equation.

We consider the following initial-boundary value problem:

$$u_t = \Delta(u^m) - cu^p \quad \text{in } \Omega\times \mathrm{I}, \tag{2.1a}$$

$$u|_{\partial\Omega\times \mathrm{I}} = g, \tag{2.1b}$$

$$u(x,y,0) = u_0(x,y), \tag{2.1c}$$

where Δ denotes the Laplace operator, $m \ge 2$, $p > 0$, c is a constant, the bounded spatial region Ω is a rectangle, $[\alpha_1,\beta_1]\times[\alpha_2,\beta_2]$, and $\mathrm{I} = (0,\mathrm{T})$, $\mathrm{T} < \infty$, is a time interval. We let g and u_0 be given smooth data. We will use problem (2.1) as a model apt to describe our method.

If $m \ge 2$ and $c = 0$, (2.1a) is a well-known porous-medium equation which governs the density of an ideal gas [1]. If $m \ge 2$, $c > 0$, and $0 < p < 1$, (2.1a) is called the porous-medium equation with strong absorption. It appears in various applications, most frequently to describe a phenomenon of thermal propagation in an absorptive medium, where the solution u stands for temperature. In other applications, u is a concentration and the process is described as diffusion with absorption. The solution u is required to be nonnegative, and thus we consider nonnegative and bounded initial condition $u_0(x,y)$. Furthermore, in the case of zero boundary conditions, that is, $g(x)=0$, it is known (see [2]) that the solution u vanishes identically as time t goes on and extinction in finite time arises, that is, there exists a time $\mathrm{T}^*>0$ such that $u(\cdot,t)$ is not identically zero for $0<t<\mathrm{T}^*$ but $u(\cdot,\mathrm{T}^*)\equiv 0$. Here, T^* is called an extinction time of a solution u and the behavior is known as finite-extinction phenomenon.

3. Linearization approach. Consider the model porous-medium equation (without absorption)

$$u_t=\Delta(u^m),\quad m\geq 2. \tag{3.1}$$

Applying the standard Crank-Nicolson scheme (which is $\mathbb{O}(k^2+h^2)$-accurate), one obtains

$$D_{+t}U_n=\frac{1}{2}\left(\left(D_{+x}D_{-x}+D_{+y}D_{-y}\right)U_n^m+\left(D_{+x}D_{-x}+D_{+y}D_{-y}\right)U_{n+1}^m\right), \tag{3.2}$$

where the power m to discrete vector U is done element by element. Rearranging the U_n and U_{n+1} terms gives

$$U_{n+1}-\frac{k}{2}\left(D_{+x}D_{-x}+D_{+y}D_{-y}\right)U_{n+1}^m=U_n+\frac{k}{2}\left(D_{+x}D_{-x}+D_{+y}D_{-y}\right)U_n^m. \tag{3.3}$$

This yields a system of nonlinear algebraic equations that can be solved using Newton's iterative method at each temporal step.

At $(n+1)$th temporal step, this system of nonlinear equations is

$$F(U)=U-\frac{k}{2}\left(D_{+x}D_{-x}+D_{+y}D_{-y}\right)U^m-b_n=0, \tag{3.4}$$

where $U=U_{n+1}$, $b_n=U_n+(k/2)(D_{+x}D_{-x}+D_{+y}D_{-y})U_n^m$, and U_n is the computed solution vector at the previous time level, a known vector. Then Newton's iterative method is given by

$$J\left(U^{(l)}\right)W^{(l+1)}=-F\left(U^{(l)}\right),\quad l\geq 0, \tag{3.5}$$

where (l) is an iterative index, $W^{(l+1)}=U^{(l+1)}-U^{(l)}$, $J(U^{(l)})$ is the $M^2\times M^2$ Jacobi matrix given by

$$J\left(U^{(l)}\right)=I-\frac{mk}{2}\left(D_{+x}D_{-x}+D_{+y}D_{-y}\right)\left(U^{(l)}\right)^{m-1}, \tag{3.6}$$

and I is the $M^2\times M^2$ identity matrix.

The feature of the Crank-Nicolson scheme is its unconditional stability, see the appendix. For all values of $\lambda = k/h^2$, in practice, however, one uses moderate values of λ to avoid slowly decaying oscillations. The main problem one needs to deal with in solving (3.3) is its computational cost. Although one might improve the efficiency of the procedure by exploiting sparsity (see [10]), the Jacobian matrices involved here typically are very large, thus the computational cost is large.

3.1. Standard linearization method. Here, we follow the idea of Richtmyer and Mortan [8] who applied it to 1D problems $u_t = (u^m)_{xx}$ (some call it Richtmyer's linearization method, see [9]). Recently, this idea was also applied to Korteweg-de Vries equation, see [5].

For $m \geq 2$,

$$U_{n+1}^m = U_n^m + mU_n^{m-1}(U_{n+1} - U_n) + \mathbb{O}(k^2) \tag{3.7}$$

or

$$U_{n+1}^m = U_n^m + mU_n^{m-1}\omega_{n+1}, \tag{3.8}$$

where $\omega_{n+1} \approx U_{n+1} - U_n$.

Substituting (3.8) into (3.3) and rearranging, we obtain

$$\left(I - \frac{mk}{2}(D_{+x}D_{-x} + D_{+y}D_{-y})U_n^{m-1}\right)\omega_{n+1} = k(D_{+x}D_{-x} + D_{+y}D_{-y})U_n^m. \tag{3.9}$$

Equation (3.9) needs to be solved at each time level $n = 1, 2, \ldots, N$. In matrix form, it can be written as a linear system in the unknown ω_{n+1}:

$$A\omega_{n+1} = b, \tag{3.10}$$

where A is $M^2 \times M^2$ block tridiagonal matrix:

$$A = \begin{pmatrix} T_1 & D_2 & & & \\ D_1 & T_2 & D_3 & & \\ & \ddots & \ddots & \ddots & \\ & & D_{M-2} & T_{M-1} & D_M \\ & & & D_{M-1} & T_M \end{pmatrix}, \tag{3.11}$$

each of the blocks T_j's and D_j's are $M \times M$ tridiagonal and diagonal matrices, respectively. They are not constant diagonal (Toeplitz) but depend on the solution u at the previous time level n:

$$T_j = I + \frac{m\lambda}{2}\hat{T}_j, \tag{3.12}$$

where

$$\hat{T}_j = \begin{pmatrix} 4u^{m-1}_{1,j,n} & -u^{m-1}_{2,j,n} & & & \\ -u^{m-1}_{1,j,n} & 4u^{m-1}_{2,j,n} & -u^{m-1}_{3,j,n} & & \\ & \ddots & \ddots & \ddots & \\ & & -u^{m-1}_{M-2,j,n} & 4u^{m-1}_{M-1,j,n} & -u^{m-1}_{M,j,n} \\ & & & -u^{m-1}_{M-1,j,n} & 4u^{m-1}_{M,j,n} \end{pmatrix}, \tag{3.13}$$

$$D_j = -\frac{m\lambda}{2} \begin{pmatrix} u^{m-1}_{1,j,n} & & & & \\ & u^{m-1}_{2,j,n} & & & \\ & & \ddots & & \\ & & & u^{m-1}_{M-1,j,n} & \\ & & & & u^{m-1}_{M,j,n} \end{pmatrix},$$

for $1 \le j \le M$ and $\lambda = k/h^2$.

The M^2 vector b with the contribution from the boundary in (3.10) has the form

$$b = BU^m_n - \frac{\lambda}{2}U^m_n\Big|_{\partial\Omega} - \frac{\lambda}{2}U^m_{n+1}\Big|_{\partial\Omega}. \tag{3.14}$$

Here, $U^m_l|_{\partial\Omega}$ is an M^2 vector for $l = n, n+1$. Since Ω is the rectangle $[\alpha_1, \beta_1] \times [\alpha_2, \beta_2]$, we can separate the boundary of Ω, $\partial\Omega$, into two parts as follows:

$$\begin{aligned} \partial\Omega &= \partial\Omega_x \cup \partial\Omega_y, \\ \partial\Omega_x &= \{(x,y) \in \Omega \mid \alpha_1 < x < \beta_1,\ y = \alpha_2, \beta_2\}, \\ \partial\Omega_y &= \{(x,y) \in \Omega \mid x = \alpha_1, \beta_1,\ \alpha_2 < y < \beta_2\}. \end{aligned} \tag{3.15}$$

So we can write $U^m_l|_{\partial\Omega}$, for $l = n, n+1$, the contribution from the boundary:

$$U^m_l\big|_{\partial\Omega} = U^m_l\big|_{\partial\Omega_x} + U^m_l\big|_{\partial\Omega_y} = \left(\begin{array}{c} u^m_{1,0,l} \\ u^m_{2,0,l} \\ \vdots \\ u^m_{M-1,0,l} \\ u^m_{M,0,l} \\ \hline 0 \\ 0 \\ \vdots \\ \vdots \\ 0 \\ 0 \\ \hline u^m_{1,M+1,l} \\ u^m_{2,M+1,l} \\ \vdots \\ u^m_{M-1,M+1,l} \\ u^m_{M,M+1,l} \end{array}\right) + \left(\begin{array}{c} u^m_{0,1,l} \\ 0 \\ \vdots \\ 0 \\ u^m_{M+1,1,l} \\ \hline u^m_{0,2,l} \\ 0 \\ \vdots \\ \vdots \\ 0 \\ u^m_{M+1,M-1,l} \\ \hline u^m_{0,M,l} \\ 0 \\ \vdots \\ 0 \\ u^m_{M+1,M,l} \end{array}\right), \tag{3.16}$$

where indices 0 and $M+1$ denote the boundary values. The matrix B is an $M^2 \times M^2$ block tridiagonal matrix, I is the $M \times M$ identity matrix, and T is an $M \times M$ tridiagonal matrix:

$$B = \lambda \begin{pmatrix} T & I & & & \\ I & T & I & & \\ & \ddots & \ddots & \ddots & \\ & & I & T & I \\ & & & I & T \end{pmatrix}, \quad T = \begin{pmatrix} -4 & 1 & & & \\ 1 & -4 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -4 & 1 \\ & & & 1 & -4 \end{pmatrix}. \tag{3.17}$$

It should be noted that the linearized equation (3.9) is identical to those obtained by using only one iteration of Newton's method with the initial guess $U_{n+1}^{(0)} = U_n$, the computed solution vector at the previous time level. This can be verified simply by inspection. The cited papers failed to make this observation.

Again, the straightforward attempt to solve system (3.10) is unacceptably costly. The half bandwidth of A is M, and thus applying a band matrix solver would cost $\mathbb{O}(M^2)$ flops *per unknown*, which is far from optimal.

3.2. Operator splitting method. In order to solve system (3.10), we propose an operator splitting technique.

We add a benign (because it does not affect the $\mathbb{O}(k^2+h^2)$-accuracy of the Crank-Nicolson scheme) term with a truncation error $\mathbb{O}(k^2)$:

$$\frac{m^2k^2}{4}(D_{+x}D_{-x}D_{+y}D_{-y})(U_n^{m-1}\omega_{n+1}), \tag{3.18}$$

to the only left-hand side of (3.9). As a result we can factorize the left-hand side of (3.9) as follows:

$$\left(I - \frac{mk}{2}D_{+x}D_{-x}U_n^{m-1}\right)\left(I - \frac{mk}{2}D_{+y}D_{-y}U_n^{m-1}\right)\omega_{n+1} = \tilde{b}, \tag{3.19}$$

where $\tilde{b}$ is $k(D_{+x}D_{-x}+D_{+y}D_{-y})U_n^m$ with the suitable boundary contribution. By splitting (3.19) into two simpler systems, we have

$$\left(I - \frac{mk}{2}D_{+x}D_{-x}U_n^{m-1}\right)\omega_{n+1/2} = \tilde{b}, \tag{3.20}$$

$$\left(I - \frac{mk}{2}D_{+y}D_{-y}U_n^{m-1}\right)\omega_{n+1} = \omega_{n+1/2}, \tag{3.21}$$

where $U_{n+1} = U_n + \omega_{n+1}$ and $\omega_{n+1/2}$ is an intermediate value. Since $\Omega = [\alpha_1,\beta_1] \times [\alpha_2,\beta_2]$, the intermediate boundary values can be given from (3.21) by

$$\omega_{n+1/2}\big|_{\partial\Omega_y} = \left(I - \frac{mk}{2}D_{+y}D_{-y}G_n^{m-1}\right)(G_{n+1}-G_n), \tag{3.22}$$

where

$$\begin{gathered}\partial\Omega_y = \{(x,y)\in\Omega \mid x=\alpha_1,\beta_1,\ \alpha_2<y<\beta_2\},\\ u(x,y,t)=g(x,y,t)\quad \text{for } (x,y,t)\in\partial\Omega\times\mathrm{I},\\ G_l=\left(g(x_1,y_1,t_l),g(x_2,y_1,t_l),\dots,g(x_M,y_M,t_l)\right)^T,\quad l=n,n+1.\end{gathered}\tag{3.23}$$

The structures of the matrices in (3.20) and (3.21) are identical. They differ in the ordering of variables, horizontal (along x-axis) in (3.20) and vertical (along y-axis) in (3.21). The matrices in (3.20) and (3.21) are block diagonal with tridiagonal blocks: for (3.20),

$$A=\begin{pmatrix} T_1 & & & & \\ & T_2 & & & \\ & & \ddots & & \\ & & & T_{M-1} & \\ & & & & T_M\end{pmatrix},\qquad T_j=I+\frac{m\lambda}{2}\hat{T}_j,\tag{3.24}$$

where

$$\hat{T}_j=\begin{pmatrix} 2u^{m-1}_{1,j,n} & -u^{m-1}_{2,j,n} & & & \\ -u^{m-1}_{1,j,n} & 2u^{m-1}_{2,j,n} & -u^{m-1}_{3,j,n} & & \\ & \ddots & \ddots & \ddots & \\ & & -u^{m-1}_{M-2,j,n} & 2u^{m-1}_{M-1,j,n} & -u^{m-1}_{M,j,n} \\ & & & -u^{m-1}_{M-1,j,n} & 2u^{m-1}_{M,j,n}\end{pmatrix},\tag{3.25}$$

for $1\le j\le M$. For (3.21) the indices in T_j are transposed: $u^{m-1}_{j,i,n}$ instead of $u^{m-1}_{i,j,n}$, for $1\le i,\ j\le M$. Since the entries depend on the solution u, the matrices in (3.20) and (3.21) are identical only if the solution is symmetric in x and y.

Each tridiagonal block can be solved independently of other blocks at the cost proportional to its size, that is, $\mathbb{O}(M)$. Since there are $2M$ blocks to be solved, the total cost is $\mathbb{O}(M^2)$ or $\mathbb{O}(1)$ *flops per unknown*, which is optimal.

Now we study the finite-extinction phenomenon for (2.1a) with $m\ge 2$, $0<p<1$, and $c>0$. Applying the operator splitting technique, we obtain a system of two equations which is similar to the system in (3.20) and (3.21) except $\tilde{b}$ in (3.20):

$$\tilde{b}=k\left(D_{+x}D_{-x}+D_{+y}D_{-y}\right)U^m_n-\frac{kc}{2}\left(U^p_n+U^p_{n+1}\right).\tag{3.26}$$

Here, however, in the presence of the additional nonlinear zero-order term cu^p, (3.20) becomes nonlinear (because of the term U^p_{n+1} in $\tilde{b}$). To solve it, we use Newton's method, again with the Jacobian matrix that is block diagonal, and thus each tridiagonal block can be solved independently. As a consequence, the computational cost can significantly be reduced as in the case of the porous-medium equation (without absorption).

4. Numerical experiments.

In this section, we consider two different numerical experiments. First, we investigate the efficiency of our linearization with splitting method

TABLE 4.1. Comparison of the accuracy.

M	Example 4.1						Example 4.2		
	$\lambda = 1$			$\lambda = 10$			$\lambda = 1$		
	N	L	S	N	L	S	N	L	S
20	2.3e-8	2.1e-7	2.1e-8	2.0e-7	5.9e-4	1.5e-7	5.0e-6	5.1e-6	4.9e-6
40	5.9e-9	1.9e-8	5.8e-9	7.4e-9	1.3e-6	6.3e-9	1.8e-6	1.8e-6	1.8e-6
80	1.5e-9	2.3e-9	1.5e-9	1.6e-9	8.8e-8	7.0e-10	6.5e-7	6.5e-7	6.5e-7

for solving 2D problems. We compare three different methods: two iterations of Newton's method (denoted by N in tables and figures), and the linearization method, first in its standard form (denoted by L), then combined with the operator splitting (denoted by S), in terms of the computational efficiency (cost). We also comment on the accuracy of the schemes. Second, we study the finite-extinction phenomenon for the porous-medium equation with strong absorption employing our scheme. We present the experimental results of the numerical extinction time values for various spatial and temporal step sizes. All our programs, written in Matlab, are implemented on a PC with Pentium III processor at 933 MHz.

4.1. Efficiency of the splitting method. We consider problems defined in a unit square spatial domain divided equidistantly into M grid points in the x- and y-directions. Thus, the spatial step size of the uniform grid becomes $h = 1/(M+1)$. To study the computational efficiency (cost) of the schemes (*per temporal step*), we compare the CPU time and the number of Mflops for $M = 20, 40$, and 80 with a fixed value of $\lambda = k/h^2 = 1$, where k is the temporal step size. We report on the cost reduction factor as well as the power, p, of the cost's growth model: cost $= \mathcal{O}(M^{\mathrm{p}})$ (computed as the slope of the least-squares linear polynomial of logarithmically scaled points). The total cost is then increasing linearly with the number of temporal steps, that is, proportional to the value of $1/\lambda$. It should be noted that the Crank-Nicolson scheme allows for the use of higher values of λ than of the order of 1, see Table 4.1.

The normalized L^2-norm of the error is defined as

$$\|\text{error}\| = \frac{1}{M}\sqrt{\sum_{i,j=1}^{M}\left(u_{i,j,n} - u(ih, jh, T)\right)^2}, \tag{4.1}$$

where $u_{i,j,n}$ is the numerical solution and $u(ih, jh, T)$ is the analytical solution at the time $t = nk = T$, and h and k are the spatial and temporal step sizes, respectively.

We investigate two numerical examples which are different initial-boundary value problems of the same porous-medium equation $u_t = \Delta(u^5)$.

EXAMPLE 4.1. We choose initial and boundary conditions corresponding to the exact solution of $u_t = \Delta(u^5)$ which is defined as (see [10])

$$u(x, y, t) = \sqrt[4]{\frac{4}{5}(2t + x + y)}, \tag{4.2}$$

for $0 \le x$, $y \le 1$, and $0 \le t \le 1$.

Example 4.2. Similarly, we consider the initial-boundary value problem corresponding to the exact solution

$$u(x,y,t) = \frac{1}{\sqrt{5}} \sqrt[4]{\frac{x^2+y^2}{1-t}}, \tag{4.3}$$

for $0 \le x,\ y \le 1$, and $0 \le t \le 0.5$.

The exact solution of Example 4.2 (given analytically by Barenblatt, and thus called the Barenblatt solution) is a radially symmetric self-solution $u \ge 0$, defined on $\{(\mathbf{x},t) \mid \mathbf{x} \in \mathbb{R}^N,\ 0 < t < T\}$ (see [1, 3]). It is of the general form

$$u(\mathbf{x},t) = (T-t)^{-K/(m-1)} \left(AT^K + \frac{BT\|\mathbf{x}\|^2}{(T-t)^{1-K}} \right)^{1/(m-1)}, \tag{4.4}$$

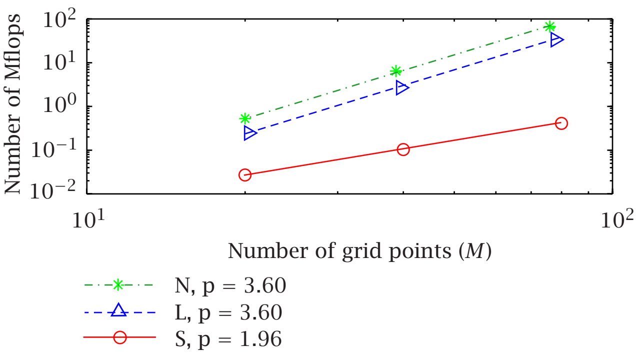
where $K = N(m-1)/(N(m-1)+2)$, $C = K/2mN$, $T = C/B$, and $A \ge 0$. For Example 4.2, we choose $N = 2$, $T = 1$, $m = 5$, and $A = 0$, thus obtaining (4.3).

The computational cost per temporal step of solving the linear system represented by system (3.10) is the critical component of the total computational expenses. A brute force band Gaussian elimination of the $M^2 \times M^2$ system with the half-band width M would cost $\mathbb{O}(M^4)$ flops, a prohibitively high expense.

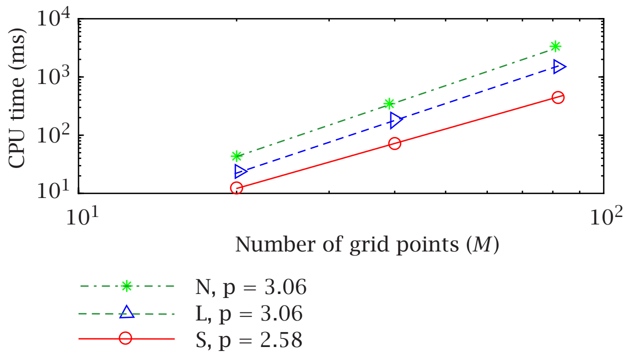
The pentadiagonal (or block-tridiagonal) matrices (3.11) have the same structure as the discrete Laplacian, although they are not constant diagonal (Toeplitz) and thus do not allow for the use of FFT-based fast solvers. One possibility of reducing the cost is to apply nearly optimal reordering before the elimination. Such a tool is provided in Matlab (as a default option) using the backslash $(\backslash)$ operation in the sparse mode; it employs the minimum degree ordering algorithm. Because of this, the flop count for solving (3.10) decreases to about $O(M^{3.6})$, see Figure 4.1. At the same time, for the splitting method, see Figure 4.1, the computational cost for solving (3.20) and (3.21) is of about $\mathbb{O}(M^2)$ flops, or $O(1)$ per unknown, which is optimal. Thus, our analysis from Section 3.2 is confirmed by the experiments.

One should point out that the CPU growth factors are significantly closer, of about $\mathbb{O}(M^3)$ and $\mathbb{O}(M^{2.5})$, respectively. This measure of efficiency is much more computer- and language-dependent. In this implementation, the difference in CPU time between the two methods is less pronounced than the flop count. Nevertheless, for $M = 80$, the actual cost (in CPU) reduction ratio from Newton's method to our operator splitting method is about 8 times (see Table 4.2). The computational cost for both Examples 4.1 and 4.2 is almost identical, therefore we provide the data only for one of them.

We also provide the comparison of the accuracy of the considered methods. One should remark that the accuracy is heavily problem-dependent as the discretization error depends not only on the numerical scheme and the parameters of discretization but also on the smoothness of the solution. Because of this, we provide the two examples. The computed solution in Example 4.1, Figure 4.2(a), is much more accurate, in the range of 10^{-7} to 10^{-8} (except for the standard linearization method), than in Example 4.2, Figure 4.2(b) (10^{-5} to 10^{-6}) for the given range of grid sizes, M. Here, p is



(a)



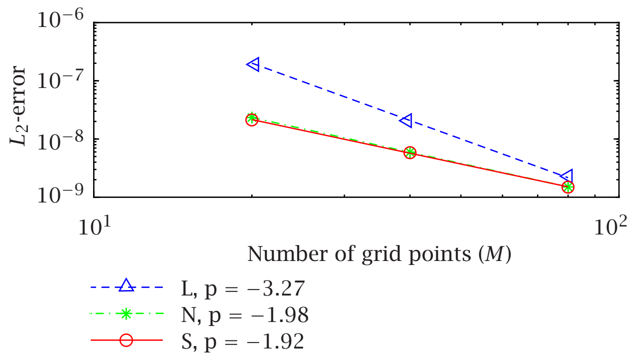
(b)

FIGURE 4.1. Comparison of the efficiency (measured in Mflops and CPU time (ms)) for the Newton (N) method (two iterations), standard linearization (L) method, and our splitting (S) method as a function of the number of grid points, M, for Example 4.1. Here, p is the power of the cost's growth model: cost $= \mathbb{O}(M^{p})$.

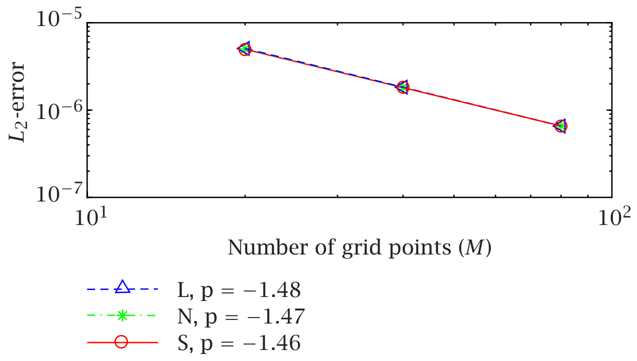
TABLE 4.2. Comparison of the efficiency (average cost per temporal step) for Example 4.1.

Grid size		Number of Mflops			CPU time (ms)		
M	h	N	L	S	N	L	S
20	0.0476	0.53	0.26	0.03	48	24	12
40	0.0244	5.13	2.56	0.10	281	138	65
80	0.0123	78.13	39.06	0.41	3 362	1 669	442

the power of the discretization error model: error $= \mathbb{O}(M^{p})$ with the theoretical value for smooth enough functions of $p = -2$. The experiments indicate that the accuracy of our method is almost the same as the one after two iterations of Newton's method, see



(a)



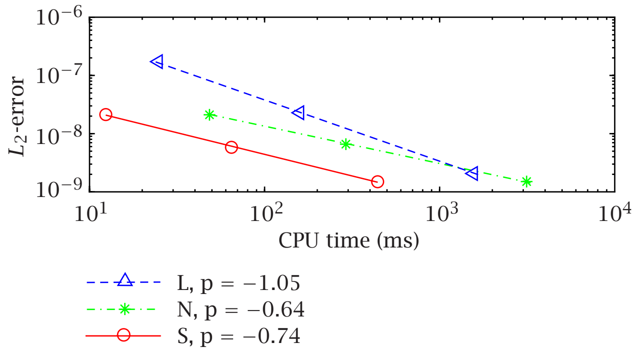
(b)

FIGURE 4.2. Comparison of the accuracy for the Newton (N) method (two iterations), standard linearization (L) method, and our splitting (S) method as a function of the number of grid points, M, for $\lambda = 1$. Here, p is the power of the error's growth model: error $= \mathbb{O}(M^{\mathrm{p}})$; (a) shows the computed solution in Example 4.1 while (b) shows that in Example 4.2.

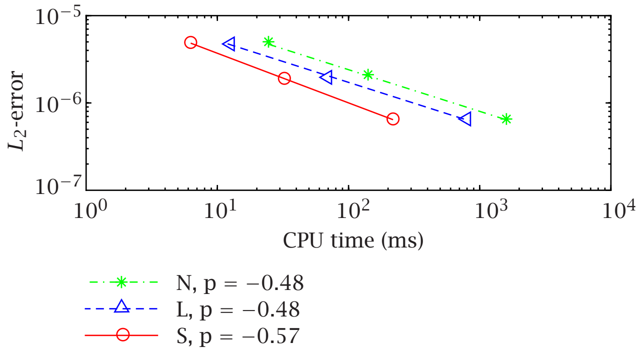
Table 4.1. We also verified the influence of larger values of λ on accuracy, see Table 4.1. Only for the coarse grids, the slowly decaying oscillations affect the error.

To bring yet another perspective, we plot the comparison of the accuracy versus efficiency of the considered algorithms, see Figure 4.3. It clearly shows the superiority of the splitting method. As a consequence, the results presented in the next section were obtained only by the latter.

4.2. Finite-extinction phenomenon. To study the finite-extinction phenomenon numerically, we consider the initial-boundary value problem (2.1) with $m = 2$, $p = 0.5$, $c = 5$, and zero boundary condition ($g = 0$). The presented results were obtained by the splitting method.



(a)



(b)

FIGURE 4.3. Comparison of the accuracy (L_2-error), Example 4.1, versus efficiency (CPU time), Example 4.2, for the Newton (N) method (two iterations), standard linearization (L) method, and our splitting (S) method.

EXAMPLE 4.3. We choose the initial function $u_0(x,y)$ given as (see [7])

$$u_0(x,y) = \begin{cases} 1 & \text{if } (x,y) = (0,0), \\ \left(1 - \dfrac{(x^2+y^2)^2}{\sqrt{x^6+y^6}}\right)_+ & \text{if } (x,y) \neq (0,0), \end{cases} \tag{4.5}$$

where $\Omega = [-1.2, 1.2] \times [-1.2, 1.2]$.

We denote M to be the number of spatial steps in the x- and y-directions, that is, $(M-1)$ is the number of grid points in each direction. Thus the step size of the uniform grid is $h = 2.4/M$.

The second equation of our system, (3.21), is linear. We solve it just as described in Section 4.1. On the other hand, the first equation, (3.20), becomes nonlinear. To solve it we use Newton's iterative method: at the $(n+1)$th temporal step, the system of $(M-1)$

decoupled nonlinear equations in the x-direction is

$$F(U) = U - kD_{+x}D_{-x}U_nU + 2.5k\sqrt{U} - \tilde{b} = 0, \tag{4.6}$$

where for $i = 1,2,\dots,M-1$,

$$\begin{aligned} U &= \left(u_{i,1,n+1}, u_{i,2,n+1}, \dots, u_{i,M-1,n+1}\right)^T, \\ U_n &= \left(u_{i,1,n}, u_{i,2,n}, \dots, u_{i,M-1,n}\right)^T, \\ \tilde{b} &= U_n + kD_{+y}D_{-y}U_n^2 - 2.5k\sqrt{U_n}, \end{aligned} \tag{4.7}$$

with the known vector of U_n and element-by-element operations.

The Jacobian matrix $J(U)$ is

$$J(U) = I - kD_{+x}D_{-x}U_n + \frac{1.25k}{\sqrt{U}}, \tag{4.8}$$

where I is the $(M-1)\times(M-1)$ identity matrix. The Newton iterations are performed until the stopping criterion $\|U_{n+1}^{(l+1)} - U_{n+1}^{(l)}\| < \tau$, $l = 0,1,\dots$, is satisfied. Here, the initial guess is $U_{n+1}^{(0)} = U_n$ for $n = 0,1,\dots,$ and τ is the given tolerance. To prevent the singularity of the Jacobian matrix, we use regularization, that is, $J_{\text{new}} = J + \epsilon I$, using the machine precision $\epsilon \simeq 2.2e{-}16$. Moreover, we force its solution to remain nonnegative at each temporal step.

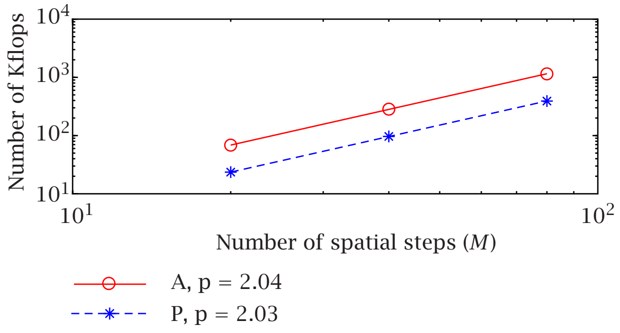
On the average, the algorithm uses only about two Newton's iterations to solve the nonlinear equation (3.20). As a consequence, the computational cost per temporal step is only about twice that of the cost discussed in Section 4.1 (without the absorption term), see Figure 4.4 and Table 4.3. Here, we denote by P the porous-medium equation, Example 4.2, and by A the porous-medium equation with absorption, Example 4.3. Note that in order to make the comparison, we modify Example 4.2 and run it with the same number of the spatial steps, $M = 20, 40, 80$, and equal exponents, $m = 2$.

In Figure 4.5 we plot the time evolution of the maximum height, H, of the computed solution, u, of (4.5) in standard and log-log scales. Note that the log of the solution is decreasing rapidly in the vicinity of the extinction time, T^*. Additionally, in Figure 4.6 we plot traces of the computed solution, u, on the xy-plane at four different points in its time evolution.

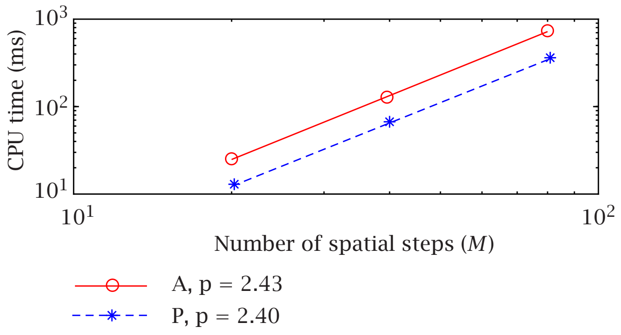
The goal here is to accurately compute the extinction time, T^*, the time at which the solution becomes identically zero, $u(x,y,\mathrm{T}^*) = 0$. Thus, one must accurately compute both the solution, u, and the time, T^*. The latter task imposes additional restriction: the temporal step k must be smaller or equal to the required precision in determining the value of T^*. This point is well illustrated by Table 4.4.

It is clear that the number of significant digits in the computed value of T^* is limited by the $-\log k$. It does not mean though that it is sufficient to take small values of k (the temporal step size) as of course the *spatial* discretization error also plays an important role, see Table 4.5.

In these experiments with $k = 1e-5$, we limited the refining of the spatial grid because of large computational costs. To remedy this, we designed a modified algorithm. From the point of view of accuracy, one could conduct the experiments with much



(a)



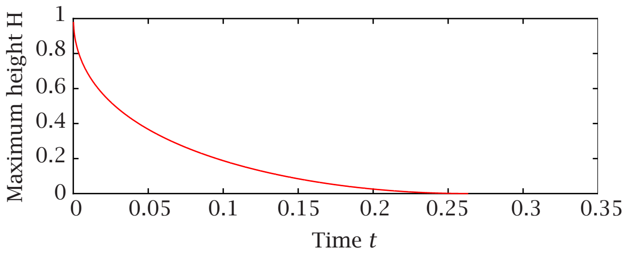
(b)

FIGURE 4.4. Comparison between the porous-medium equation with strong absorption (A) and the porous-medium equation (P) in terms of (a) the number of Kflops and (b) CPU time (bottom), as a function of the number of spatial steps, M, for $\lambda = 1$. Here, p is the power of the cost's growth model: error $= \mathbb{O}(M^{\mathrm{p}})$.

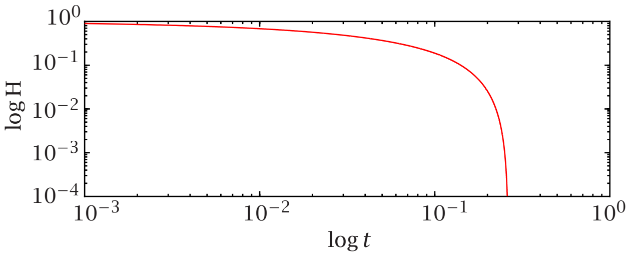
TABLE 4.3. Comparison of computational costs (per temporal step) for modified Examples 4.2 and 4.3.

M	Number of Kflops		CPU time (ms)	
	P	A	P	A
20	23.5	68.3	12.9	25.4
40	96.6	284.5	58.3	128.0
80	391.5	1 147.8	362.1	735.1

larger values of λ (see Table 4.1) than those in Table 4.5 if it was not for the resolution requirement that k be sufficiently small *at the extinction time* T^*. The time evolution (in the log-log scale) of the maximum height H of the computed solution u, see Figure 4.5,



(a)



(b)

FIGURE 4.5. Time evolution of the maximum height, H, of the numerical solution, u, with the spatial step size $h = 0.04$ and temporal step size $k = 0.0001$. Plot (b) is the log-log plot of (a).

TABLE 4.4. The extinction time T^* for fixed $h = 0.04$.

k	0.01	0.001	0.0001	0.00001
λ	6.25	0.625	0.0625	0.00625
T^*	0.26	0.263	0.2634	0.26343

TABLE 4.5. T^* for fixed $k = 1e\text{-}5$.

h	0.08	0.04
λ	0.0015625	0.00625
T^*	0.26360	0.26343

shows an abrupt downward turn at some point T^o in the vicinity of the extinction time T^*. The modified algorithm employs a much larger time step, k^o for $t \le T^o$, and a small k, corresponding to the required resolution, afterwards.

In the experiments reported below, see Table 4.6, we have chosen an ad hoc value $k^o = 100k$, and the program was adaptively changing the temporal step size at a point when the slope, p, of the maximum height of u becomes $p \le p^o$. Again, we have, ad hoc, chosen $p^o = -20$. The reported λ in Table 4.6 is that of $\lambda^o = k^o/h^2$, that is, that for $t \le T^o$.

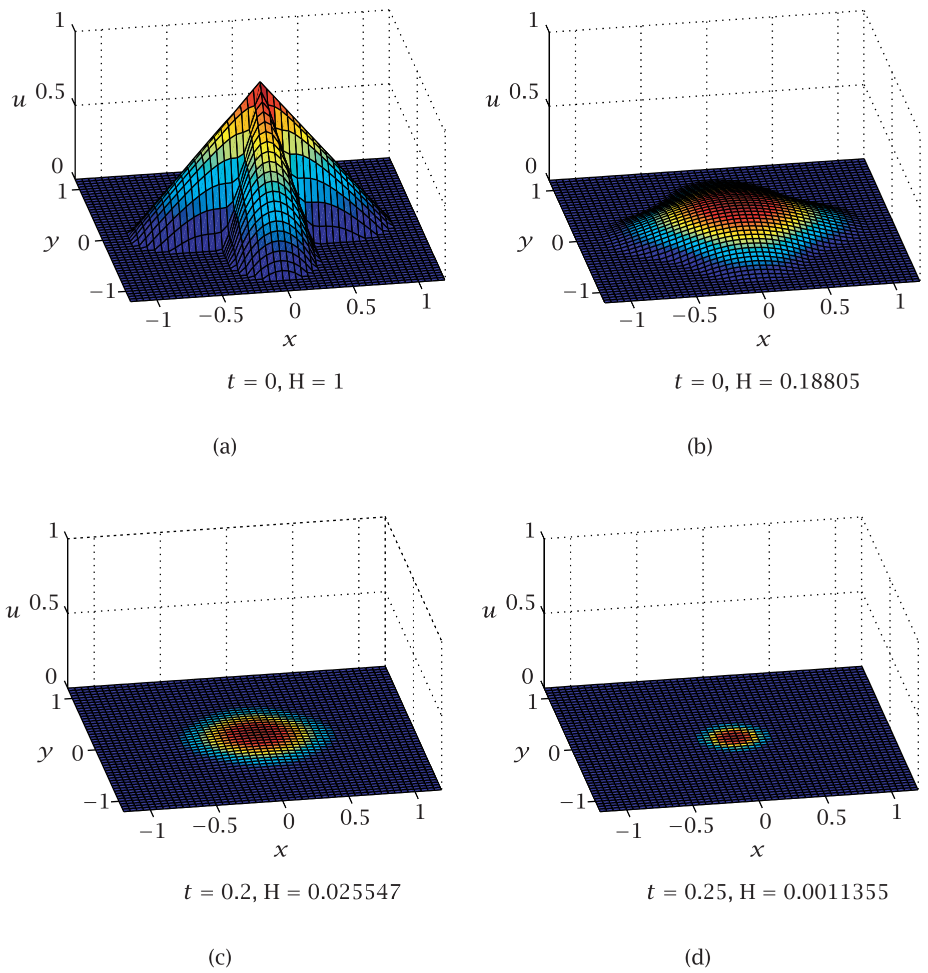


FIGURE 4.6. Finite-extinction phenomenon of the numerical solution for Example 4.3 with the spatial step size $h = 0.04$ and temporal step size $k = 0.0001$. Here, H is the maximum height of the numerical solution at the time t.

TABLE 4.6. Modified algorithm: T* for fixed $k = 1e\text{-}5$ after T^o.

M	30	60	120	240
h	0.08	0.04	0.02	0.01
λ^o	0.15625	0.625	2.5	10
T*	0.26360	0.26343	0.26337	0.26336

For a fixed spatial step size, h, the total computational cost is proportional to the number of temporal steps, T^*/k for the original algorithm and $\mathrm{T}^o/k^o + (\mathrm{T}^* - \mathrm{T}^o)/k$ for the modified one. For example, for $h = 0.04$, the computed value of T^o turned out to be $T^o = 0.241$. In this case, the reduction of the number of temporal steps was from 26 343 to 2 484, or more than tenfold. We have not made an attempt to optimize the speedup.

Although we do not know the exact value of the extinction time, we can obtain a very rough estimate of the power, p, of the asymptotic error model using the Richardson extrapolation method on the values in Table 4.6. We obtain $2^p = (0.26360-0.26343)/(0.26343-0.26337) = 17/6$. We can thus conclude that the accuracy of this computation is of about $\mathbb{O}(h^{1.5})$, which is consistent with the experiments in Section 4.1.

5. Concluding remarks.

For a class of nonlinear parabolic PDEs in 2D rectangular domains, we have constructed an operator splitting algorithm of optimal efficiency. We have verified experimentally for the porous-medium equation that the computational cost of our scheme is $O(1)$ flops per unknown per temporal step while the accuracy remains the same as for two Newton's iterations.

In the presence of an additional zero-order term (strong absorption term), the asymptotic efficiency remains unchanged, $O(1)$, with the leading constant only twice larger. We have modified the algorithm to adaptively change the temporal step size. This allows computing the extinction times extremely accurately and with significant computational savings.

Appendix

Linear stability analysis. Strictly speaking, the linear Fourier analysis applied to nonlinear equations to show the stability of the scheme cannot be rigorously justified. Nevertheless, it has been found to be effective in practice. For an example of such analysis applied to Korteweg-de Vries equation, see [5].

We assume that the solution function u is bounded in the given spatio-temporal region and so let

$$v = \max\left|U^{m-1}\right|. \tag{A.1}$$

Substituting it in (3.3), the corresponding linear equation to which we apply the von Neumann analysis becomes

$$\left(I-\frac{mk}{2}v\left(D_{+x}D_{-x}+D_{+y}D_{-y}\right)\right)\omega_{n+1} = kv\left(D_{+x}D_{-x}+D_{+y}D_{-y}\right)U_n, \tag{A.2}$$

that is,

$$\begin{aligned} w_{p,q,n+1}-\frac{m\lambda}{2}v\left(w_{p-1,q,n+1}+w_{p+1,q,n+1}+w_{p,q-1,n+1}+w_{p,q+1,n+1}-4w_{p,q,n+1}\right)\\ =\lambda v\left(u_{p-1,q,n+1}+u_{p+1,q,n+1}+u_{p,q-1,n+1}+u_{p,q+1,n+1}-4u_{p,q,n+1}\right). \end{aligned} \tag{A.3}$$

Let

$$u_{p,q,n} = \xi^n e^{i\beta p}e^{i\gamma q}, \tag{A.4}$$

where $\beta = \kappa_x h$, $\gamma = \kappa_y h$, and $\beta,\gamma \in [-\pi,\pi]$.

Substituting (A.4) into (A.3) and then dividing by $\xi^n e^{i\beta p} e^{i\gamma q}$, we obtain

$$\begin{aligned}(\xi-1)-\frac{m\lambda}{2}v\left(e^{-i\beta}+e^{i\beta}+e^{-i\gamma}+e^{i\gamma}-4\right)(\xi-1)\\ =\lambda v\left(e^{-i\beta}+e^{i\beta}+e^{-i\gamma}+e^{i\gamma}-4\right),\end{aligned}\tag{A.5}$$

which can be written as

$$\left(1-\frac{m\lambda}{2}v(2\cos\beta+2\cos\gamma-4)\right)(\xi-1)=\lambda v(2\cos\beta+2\cos\gamma-4).\tag{A.6}$$

Hence,

$$\xi-1=\frac{-4\lambda v\left(\sin^2(\beta/2)+\sin^2(\gamma/2)\right)}{1+2m\lambda v\left(\sin^2(\beta/2)+\sin^2(\gamma/2)\right)}.\tag{A.7}$$

Therefore, the amplification factor ξ is

$$\xi=\frac{1+2(m-2)\lambda v\left(\sin^2(\beta/2)+\sin^2(\gamma/2)\right)}{1+2m\lambda v\left(\sin^2(\beta/2)+\sin^2(\gamma/2)\right)}.\tag{A.8}$$

Since $m \geq 2$, it is clear that $0 < \xi \leq 1$ for all positive λ and all β, γ. Thus the Crank-Nicolson method for the porous-medium equations is unconditionally linearly stable.

References

[1] D. G. Aronson, *The porous medium equation*, Nonlinear Diffusion Problems (A. Fasano and M. Primicerio, eds.), Lecture Notes in Mathematics, vol. 1224, Springer, Berlin, 1986.

[2] C. Bandle, T. Nanbu, and I. Stakgold, *Porous medium equation with absorption*, SIAM J. Math. Anal. **29** (1998), no. 5, 1268–1278.

[3] P. Bénilan, M. Chipot, L. C. Evans, and M. Pierre (eds.), *Recent Advances in Nonlinear Elliptic and Parabolic Problems*, Pitman Research Notes in Mathematics Series, vol. 208, Longman Scientific & Technical, Harlow, 1989.

[4] S. I. Betelú, D. G. Aronson, and S. B. Angenent, *Renormalization study of two-dimensional convergent solutions of the porous medium equation*, Phys. D **138** (2000), no. 3-4, 344–359.

[5] B.-F. Feng and T. Mitsui, *A finite difference method for the Korteweg-de Vries and the Kadomtsev-Petviashvili equations*, J. Comput. Appl. Math. **90** (1998), no. 1, 95–116.

[6] W. Jäger and J. Kačur, *Solution of porous medium type systems by linear approximation schemes*, Numer. Math. **60** (1991), no. 3, 407–427.

[7] K. Mikula, *Numerical solution of nonlinear diffusion with finite extinction phenomenon*, Acta Math. Univ. Comenian. (N.S.) **64** (1995), no. 2, 173–184.

[8] R. D. Richtmyer and K. W. Morton, *Difference Methods for Initial-Value Problems*, Interscience Tracts in Pure and Applied Mathematics, no. 4, Interscience Publishers, New York, 1967.

[9] G. D. Smith, *Numerical Solution of Partial Differential Equations. Finite Difference Methods*, 3rd ed., Oxford Applied Mathematics and Computing Science Series, The Clarendon Press, Oxford University Press, New York, 1985.

[10] B. P. Sommeijer and P. J. van der Houwen, *Algorithm* 621*. Software with low storage requirements for two-dimensional nonlinear, parabolic differential equations*, ACM Trans. Math. Software **10** (1984), no. 4, 378–396.

Dongjin Kim: Department of Mathematics, University of Southern California, Los Angeles, CA 90089-1113, USA
Current address: Department of Mathematics, University of Wyoming, Laramie, WY 82071, USA
E-mail address: dongkim@uwyo.edu

Wlodek Proskurowski: Department of Mathematics, University of Southern California, Los Angeles, CA 90089-1113, USA
E-mail address: proskuro@math.usc.edu