algorithms

Article

On the Lagged Diffusivity Method for the Solution of Nonlinear Finite Difference Systems

Francesco Mezzadri [1] and Emanuele Galligani [2,*]

[1] Department of Mechanical Engineering, University of Wisconsin-Madison, Madison WI 53706-1572, USA; mezzadri@wisc.edu

[2] Department of Engineering "Enzo Ferrari", University of Modena and Reggio Emilia, via P. Vivarelli 10/1, building 26, I-41125 Modena, Italy

* Correspondence: emanuele.galligani@unimore.it; Tel.: +39-059-205-6325

Received: 31 May 2017; Accepted: 26 July 2017; Published: 2 August 2017

Abstract: In this paper, we extend the analysis of the Lagged Diffusivity Method for nonlinear, non-steady reaction-convection-diffusion equations. In particular, we describe how the method can be used to solve the systems arising from different discretization schemes, recalling some results on the convergence of the method itself. Moreover, we also analyze the behavior of the method in case of problems presenting boundary layers or blow-up solutions.

Keywords: lagged diffusivity method; diffusion equations; finite-difference discretizations; blow-up solutions

1. Introduction

Let us consider the nonlinear, non-steady reaction-convection-diffusion equation

$$\begin{array}{lll}
\dfrac{\partial u}{\partial t} = \nabla \times (\sigma \nabla u) - \tilde{v} \times \nabla u - \alpha u - g + s & (x,y) \in \Omega,\ t > t_0 & \text{Diff. Eq.} \\
u(x,y,t_0) = u_0(x,y) & (x,y) \in \overline{\Omega},\ t = t_0 \geq 0 & \text{IC} \\
u(x,y,t) = u_1(x,y,t) & (x,y) \in \partial\Omega,\ t > t_0 & \text{Dirichlet BCs,}
\end{array} \tag{1}$$

where t_0 is the initial time, $u = u(x,y,t)$ is the density function, $\sigma = \sigma(x,y,u) > 0$ is the diffusion coefficient, $\tilde{v}$ is the velocity vector (assumed constant), $\alpha = \alpha(x,y) \geq 0$ is the absorption term, $-g = -g(x,y,u)$ is the rate of change of the reaction and $s = s(x,y)$ is the source term. This equation is important in various different fields and it can describe several processes and applications.

If we discretize Equation (1), at each time level the differential problem can be approximated by a nonlinear, algebraic system. However, computing the solution of the discretized system is generally complicated. For instance, if we use a Newton's method, we incur in the problem of computing the Jacobian matrix, which is usually difficult and computationally onerous. Therefore, it is suitable to first linearize the discretized system by setting up an iterative procedure which "lags" the diffusivity. In this way, we obtain the so-called Lagged Diffusivity Method (LDM). The analysis of this kind of strategies can be traced back to [1]. More recently, the LDM applied to steady and non-steady diffusion equations with reaction and/or convection terms has been studied in [2] and following works.

The nonlinear algebraic system is thus replaced by a series of systems of weakly nonlinear algebraic equations, whose Jacobian matrix is much easier to compute. Therefore, these systems can be easily solved by a Newton's method. Lastly, at each Newton iteration we solve a system of linear algebraic equations (which we again do iteratively, for enhanced efficiency).

This paper addresses some topics which have not been considered in earlier works. In particular, we

- compare different discretizations. In particular, we consider what happens when the space is discretized by an upwind method or by central finite differences. We also describe different time discretizations, considering the θ-method and the IMEX schemes;
- study numerically what happens in some particularly critical cases, where some smoothness assumptions are not satisfied. This is the case of boundary layer and blow-up solutions.

Regarding the discretization, this is relevant since the discretization can modify the properties of the discretized systems. This can be exploited to improve the solution procedure: for instance, we notice that the upwind method can significantly reduce the computational times when $\tilde{v}$ is large (at the cost of a worse order of approximation).

The analysis of blow-up solutions is instead interesting for two reasons. First, we evaluate the behavior of the method in critical situations where the LDM is not analytically proven to converge. Indeed, the convergence of the method requires some smoothness assumptions on σ, α, g and s which are not satisfied in proximity of blow-ups and boundary layers. Second, we analyze numerically the problem, while several papers regarding blow-up solutions deal with the analytical problem of determining if and when the blow-up occurs (e.g., if T is finite or not) and they are often referred to 1D differential problems with Neumann boundary conditions. Regarding analyses involving multidimensional domains and/or Dirichlet boundary conditions, it is instead worth citing [3–5], which are concerned with problems similar to the ones we analyze in the following. In particular, criteria for determining whether the blow-up occurs in a finite time or not and for estimating the blow-up time itself are given by using the Hopf's maximum principle (e.g., see [6] (§2.3) or [7]). Other interesting papers on blow-up include [8,9], while [10] is a comprehensive review on blow-up in parabolic equations.

In Section 2 we analyze the discretization. In particular, we describe how the discretized system changes when space is discretized by central finite differences or by an up-wind scheme and when time is integrated by a $\theta-$method or by IMEX strategies.

In Section 3 we then summarize the method and consider how it is affected by the discretization itself. We also report some known results on the monotonicity of the finite-difference (FD) operators and on the convergence of the method.

In Section 4 we provide some details on the implementation, an algorithm and a short summary of the solution method.

Lastly, Section 5 is devoted to the numerical experiments. First, we show numerically the differences which arise when the discretization is changed. Then, we study problems characterized by blow-up and by boundary layer solutions in order to analyze the behavior of the LDM in critical cases, when the smoothness assumptions are not satisfied.

2. Discretization

2.1. Space Discretization

For simplicity, let us assume Ω to be a 2D bounded rectangular domain. Let us superimpose a grid $\overline{\Omega}_h = \Omega_h \cup \partial\Omega_h$ to $\overline{\Omega}$. For simplicity, we use uniform mesh size h along x and y. Thus, Ω_h is made of the mesh points (x_i, y_j), $i = 1, ..., N$, $j = 1, ..., M$ with:

$$x_{i+1} = x_i + h, \quad i = 0, ..., N \qquad\qquad y_{j+1} = y_j + h \quad j = 0, ..., M.$$

Using this grid, $u_{i,j}(t) : \overline{\Omega}_h \to \mathbb{R}$ denotes the grid function which approximates the solution $u(x_i, y_j, t)$ at the mesh points (x_i, y_j) of $\overline{\Omega}_h$, $i = 0, ..., N+1$, $j = 0, ..., M+1$. This grid function satisfies the initial condition for $(x_i, y_j) \in \overline{\Omega}_h$ and $t = t_0$ and the Dirichlet boundary conditions for $(x_i, y_j) \in \partial\Omega_h$, $t > t_0$.

We can now approximate the derivatives of Equation (1) by finite difference methods. In this regard, several choices are possible. In particular, it is interesting to analyze the space discretizations obtained by central finite differences and by the upwind scheme. Indeed, this choice influences the properties of the matrix of the discretized system, as we will see in the following. Regarding the upwind scheme, we consider, for instance, the case of positive $\tilde{\boldsymbol{v}}$. If $\tilde{\boldsymbol{v}} < \mathbf{0}$ the backward difference quotients which approximate the first-order derivatives are simply replaced by forward difference quotients.

Thus, denoting the forward finite differences (in x and in y respectively) by Δ_x and Δ_y, the backward finite differences by ∇_x, ∇_y and the central finite differences by δ_x, δ_y, we have:

- Discretization by central finite differences (error $O(h^2)$)

$$\begin{aligned}\frac{\mathrm{d}u_{i,j}(t)}{\mathrm{d}t} &= \Delta_x[\sigma(x_i,y_j,u_{i,j}(t))\nabla_x u_{i,j}(t)] + \Delta_y[\sigma(x_i,y_j,u_{i,j}(t))\nabla_y u_{i,j}(t)] - \\ &- \tilde{v}_1\delta_x u_{i,j}(t) - \tilde{v}_2\delta_y u_{i,j}(t) - \alpha_{i,j}u_{i,j}(t) - g(x_i,y_j,u_{i,j}(t)) + s(x_i,y_j,t).\end{aligned} \tag{2}$$

- Discretization by upwind scheme (error $O(h)$), case $\tilde{\boldsymbol{v}} > \mathbf{0}$

$$\begin{aligned}\frac{\mathrm{d}u_{i,j}(t)}{\mathrm{d}t} &= \Delta_x[\sigma_{i,j}(u_{i,j}(t))\nabla_x u_{i,j}(t)] + \Delta_y[\sigma_{i,j}(u_{i,j}(t))\nabla_y u_{i,j}(t)] - \\ &- \tilde{v}_1\nabla_x u_{i,j}(t) - \tilde{v}_2\nabla_y u_{i,j}(t) - \alpha_{i,j}u_{i,j}(t) - g_{i,j}(u_{i,j}(t)) + s_{i,j}(t).\end{aligned} \tag{3}$$

In both cases, we can write the discretization in a more convenient way by defining

$$\begin{array}{lll} & \text{Central FD} & \text{Upwind} \\ L_{i,j} = \dfrac{1}{h^2}\sigma(x_i,y_j,u_{i,j}(t)) & \tilde{L}_{i,j} = \dfrac{\tilde{v}_1}{2h} & \tilde{L}_{i,j} = \dfrac{\tilde{v}_1}{h} \\ B_{i,j} = \dfrac{1}{h^2}\sigma(x_i,y_j,u_{i,j}(t)) & \tilde{B}_{i,j} = \dfrac{\tilde{v}_2}{2h} & \tilde{B}_{i,j} = \dfrac{\tilde{v}_2}{h} \\ R_{i,j} = \dfrac{1}{h^2}\sigma(x_{i+1},y_j,u_{i+1,j}(t)) & \tilde{R}_{i,j} = -\tilde{L}_{i,j} & \tilde{R}_{i,j} = 0 \\ T_{i,j} = \dfrac{1}{h^2}\sigma(x_i,y_{j+1},u_{i,j+1}(t)) & \tilde{T}_{i,j} = -\tilde{B}_{i,j} & \tilde{T}_{i,j} = 0 \\ D_{i,j} = L_{i,j} + B_{i,j} + R_{i,j} + T_{i,j} & \tilde{D}_{i,j} = \alpha(x_i,y_j) & \tilde{D}_{i,j} = \alpha(x_i,y_j) + \tilde{L}_{i,j} + \tilde{B}_{i,j} \end{array} \tag{4}$$

Then, setting $\hat{L}_{i,j} = L_{i,j} + \tilde{L}_{i,j}$ (analogously for $\hat{B}_{i,j}$, $\hat{R}_{i,j}$, $\hat{T}_{i,j}$ and $\hat{D}_{i,j}$), the right hand-side of Equations (2) and (3) can be written formally in the same way as

$$\hat{B}_{i,j}u_{i,j-1}(t) + \hat{L}_{i,j}u_{i-1,j}(t) - \hat{D}_{i,j}u_{i,j}(t) + \hat{R}_{i,j}u_{i+1,j}(t) + \hat{T}_{i,j}u_{i,j+1}(t) - g(x_i,y_j,u_{i,j}(t)) + s(x_i,y_j,t). \tag{5}$$

In the following, we denote $\sigma(x_i,y_j,u_{i,j})$ by $\sigma(u_{i,j})$ for compactness of notation.

Finally, we use matrix notation to write the entire system of ordinary differential equations arising from the space discretization. In this regard, we use the row lexicographic ordering for mesh points and define $P_k = (x_i,y_j)$ with $k = (j-1)N + i$, for $i = 1,...,N$, $j = 1,...,M$. We can then write the vector $u(t)$ containing all the $u_{i,j}(t)$ at internal mesh points. Thus, the space discretization leads to

$$\frac{\mathrm{d}\boldsymbol{u}(t)}{\mathrm{d}t} = -A(\boldsymbol{u}(t))\boldsymbol{u}(t) + \boldsymbol{b}(\boldsymbol{u}(t)) - \boldsymbol{G}(\boldsymbol{u}(t)) + \boldsymbol{s}(t). \tag{6}$$

Again, we notice that formally this expression is valid both when the discretization is performed by central FD and when the upwind scheme is used. The following differences are however present:

- when we use the upwind scheme, the order of the approximation is $O(h)$; when we use central FD, the order of the approximation is $O(h^2)$;
- when we use the upwind scheme, $A(\boldsymbol{u}(t))$ is irreducibly diagonally dominant [11] (p. 23) irrespective of the choice of h. Having also positive diagonal elements and non positive off-diagonal elements, it is always a non-singular M-matrix [11] (p. 91). With central FD, $A(\boldsymbol{u}(t))$

is irreducibly diagonally dominant (and a non-singular M-matrix) only if h satisfies a condition on the step-size. In particular, by Equation (4) if is easy to find that this condition corresponds to

$$h < \min\left\{\frac{2\sigma_{\min}}{|\tilde{v}_1|}, \frac{2\sigma_{\min}}{|\tilde{v}_2|}\right\}. \tag{7}$$

The vector $\boldsymbol{b}(\boldsymbol{u}(t))$ in Equation (6) derives from the Dirichlet boundary conditions $u(x_i, y_j, t) = u_1(x_i, y_j, t)$ at points $(x_i, y_j) \in \partial\Omega_h$. Considering the Dirichlet BCs in Equation (1) and using, again, the row-lexicographic order, its components are

$$\begin{aligned}
b_k &= L_{1,j}u_1(x_0, y_j) && \text{with } k = (j-1)N+1,\ j = 1, ..., M \\
b_k &= B_{i,1}u_1(x_i, y_0) && \text{with } k = i,\ i = 1, ..., N \\
b_k &= R_{N,j}u_1(x_{N+1}, y_j) && \text{with } k = jN,\ j = 1, ..., M \\
b_k &= T_{i,M}u_1(x_i, y_{M+1}) && \text{with } k = (M-1)N+i,\ i = 1, ..., N,
\end{aligned}$$

where the coefficients L, B, R, T depend on $\boldsymbol{u}$ at the inner point which is neighbor to the considered point of $\partial\Omega_h$. At corners, where the same inner point is neighbor to two boundary points, we then have two contributions: e.g., $b_1 = L_{1,1}u_1(x_0, y_1) + B_{1,1}u_1(x_1, y_0)$.

Defining $\mu = NM$, $\boldsymbol{G}(\boldsymbol{u}) \in \mathbb{R}^\mu$ is the nonlinear mapping of components $G_k(\boldsymbol{u}) = G_k(u_k) = g(x_i, y_j, u_k)$, with $i = 1, ..., N$, $j = 1, ..., M$ and $k = (j-1)N + i$. It is, then, easy to notice that $\boldsymbol{G}(\boldsymbol{u})$ is a diagonal mapping [12] (p. 11).

Finally, $\boldsymbol{s}(t) \in \mathbb{R}^\mu$ contains the terms arising from the source term: $s_k(t) = s(x_i, y_j, t)$ with $i = 1, ..., N$, $j = 1, ..., M$ and $k = (j-1)N + i$.

2.2. Time Discretization

We now consider time discretization. Let us introduce a time spacing Δt, which defines a series of time levels: the n-th time level t_n is defined by $t_n = n\Delta t + t_0$, $n = 0, 1, ...$

The time derivatives of Equation (6) can be approximated by several different schemes. We could use, for example, the well-known $\theta-$method (e.g., see [13]), which is completely implicit when $\theta \neq 0$. An alternative, as noted in the Introduction, is given by IMEX methods, which have been employed to solve diffusion equations when convection or reaction terms can be treated explicitly.

For instance, let us apply the θ-method. Denoting by $\boldsymbol{u}^n$ the approximation of $\boldsymbol{u}(t_n)$ (with $\boldsymbol{u}(t_n)$ solution of Equation (6) at $t = t_n$) and by $\boldsymbol{s}^n \equiv \boldsymbol{s}(t_n)$, we get:

$$\begin{aligned}
\frac{\boldsymbol{u}^{n+1} - \boldsymbol{u}^n}{\Delta t} =& \theta\left[-A(\boldsymbol{u}^{n+1})\boldsymbol{u}^{n+1} + \boldsymbol{b}(\boldsymbol{u}^{n+1}) - \boldsymbol{G}(\boldsymbol{u}^{n+1}) + \boldsymbol{s}^{(n+1)}\right] \\
&+ (1-\theta)\left[-A(\boldsymbol{u}^n)\boldsymbol{u}^n + \boldsymbol{b}(\boldsymbol{u}^n) - \boldsymbol{G}(\boldsymbol{u}^n) + \boldsymbol{s}^n\right]
\end{aligned} \tag{8}$$

for $n = 0, 1, ...$ and with $0 \leq \theta \leq 1$.

By simple algebraic manipulations, we find that at each time level $n = 0, 1, ...$ the vector $\boldsymbol{u}^{n+1}$ is given by the solution of the nonlinear algebraic system

$$\boldsymbol{F}(\boldsymbol{u}) = [I + \tau A(\boldsymbol{u})]\,\boldsymbol{u} - \tau\,[\boldsymbol{b}(\boldsymbol{u}) - \boldsymbol{G}(\boldsymbol{u})] - \boldsymbol{w} = \boldsymbol{0}, \tag{9}$$

where $\tau = \Delta t\theta$, I is the $\mu \times \mu$ identity matrix and $\boldsymbol{w} = \boldsymbol{w}^n \in \mathbb{R}^\mu$ is a vector containing the known terms, defined as

$$\boldsymbol{w} = [I - \Delta t(1-\theta)A(\boldsymbol{u}^n)]\,\boldsymbol{u}^n + \Delta t(1-\theta)\,[\boldsymbol{b}(\boldsymbol{u}^n) - \boldsymbol{G}(\boldsymbol{u}^n)] + \Delta t\left[\theta\boldsymbol{s}^{n+1} + (1-\theta)\boldsymbol{s}^n\right].$$

If we use an IMEX method, we similarly get a nonlinear algebraic system to solve at each time level. The difference is that some terms are treated explicitly. Thus, if, for instance, the reaction term is treated explicitly, at the time level $n+1$ the mapping $\boldsymbol{G}(\boldsymbol{u})$ is evaluated at time n; $\boldsymbol{G}(\boldsymbol{u})$ then becomes a known term which appears solely in the known vector $\boldsymbol{w}$.

Analogously, if the velocity is not linear, it could be useful to treat explicitly the convection term. In this case, the part of $A(\boldsymbol{u})$ depending on $\tilde{\boldsymbol{v}}$ would appear solely in the known vector $\boldsymbol{w}$.

3. The Lagged Diffusivity Method

We now introduce the lagged diffusivity method to solve the discretized system, referring, for instance, to Equation (9).

Solving Equation (9) by a Newton's method would be onerous, since the Jacobian matrix of $\boldsymbol{F}(\boldsymbol{u})$ is generally hard to compute. The lagged diffusivity method, on the other hand, linearizes the discretized system by iteratively "lagging" the diffusivity term. At each iteration we must thus solve a weakly nonlinear algebraic system, whose Jacobian matrix is significantly easier to compute.

Let us study more in detail the procedure applied to Equation (9). Lagging the diffusivity means that the new iterate $\boldsymbol{u}^{(\nu+1)}$ of the $(\nu+1)$-th lagging iteration is the solution of the weakly nonlinear system

$$\boldsymbol{F}_\nu(\boldsymbol{u}) = \left[I + \tau A(\boldsymbol{u}^{(\nu)})\right]\boldsymbol{u} - \tau\left[\boldsymbol{b}(\boldsymbol{u}^{(\nu)}) - \boldsymbol{G}(\boldsymbol{u})\right] - \boldsymbol{w} = \boldsymbol{0}. \tag{10}$$

When $A(\boldsymbol{u})$ is a nonsingular M-matrix, $I + \tau A(\boldsymbol{u})$ is evidently nonsingular $\forall \boldsymbol{u} \in \mathbb{R}^\mu$.

Thus, we now have to solve Equation (10) at each lagging iteration. We can do this by a Newton's method (or, in general, by an iterative method) which is stopped when the residual

$$\boldsymbol{F}_\nu(\boldsymbol{u}^{(\nu+1)}) = \left[I + \tau A(\boldsymbol{u}^{(\nu)})\right]\boldsymbol{u}^{(\nu+1)} - \tau\left[\boldsymbol{b}(\boldsymbol{u}^{(\nu)}) - \boldsymbol{G}(\boldsymbol{u}^{(\nu+1)})\right] - \boldsymbol{w} \tag{11}$$

satisfies a stopping criterion

$$\left\|\boldsymbol{F}_\nu(\boldsymbol{u}^{(\nu+1)})\right\| \leq \epsilon_{\nu+1} \tag{12}$$

where $\|\cdot\|$ denotes the Euclidean norm and ϵ_ν is a given tolerance such that $\epsilon_\nu \to 0$ for $\nu \to \infty$. When the inner solver converges, we find $\boldsymbol{u}^{(\nu+1)}$, which is used, in turn, to lag the diffusivity at the $(\nu+2)$-th lagged iteration. Proceeding in the same way, we thus find $\boldsymbol{u}^{(\nu+2)}$ and so on for the following iterations.

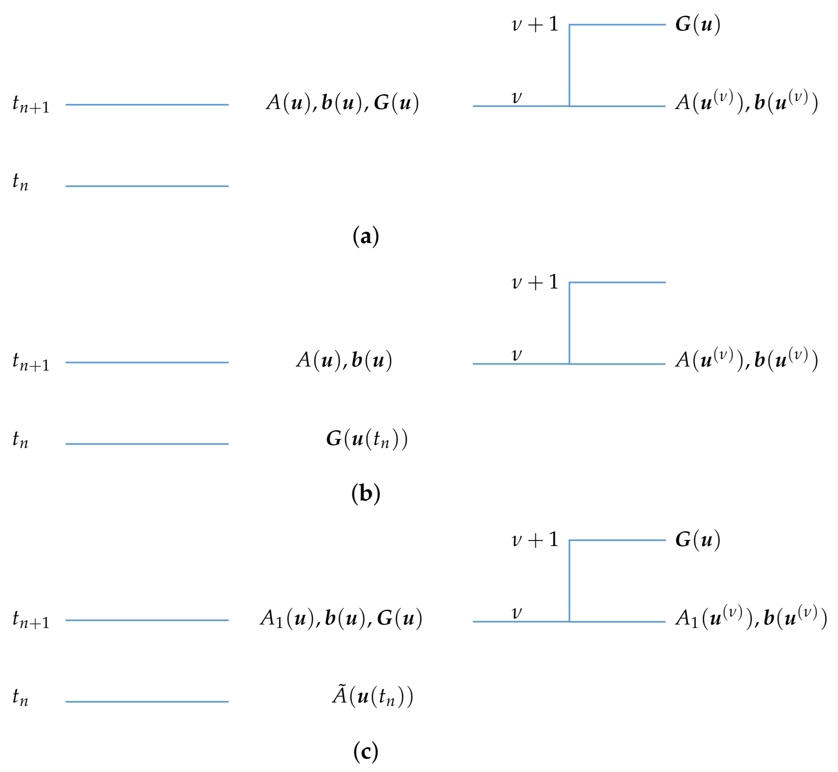
3.1. Effect of the Discretization

In the previous paragraphs, we applied the lagged diffusivity method to Equation (9), which was obtained by discretizing time using a $\theta-$method. The time discretization, however, affects some properties of the lagging iteration. For instance, if we can use an IMEX method and treat $\boldsymbol{G}(\boldsymbol{u})$ explicitly, the lagged diffusivity method completely linearizes the discretized system.

Analogously, an IMEX scheme can simplify the analysis of cases with non-constant velocity terms: if we can treat non-constant terms $\tilde{\boldsymbol{v}}(\boldsymbol{u})$ explicitly, all the convection terms are evaluated at the previous time level and are, thus, known. Both monotonicity and convergence analyses can then be traced back to the results obtained for constant $\tilde{\boldsymbol{v}}$.

Figure 1 summarizes all these cases, representing which functions are unknown at each time level and at each lagging iteration.

Finally, although formally the system does not change, also the space discretization has an effect on the solution procedure. Indeed, we noticed earlier that $A(\boldsymbol{u})$ is always an M-matrix when the upwind discretization is used, while we have a condition on the step-size h when central finite differences are used. If $A(\boldsymbol{u})$ is an M-matrix, also the coefficient matrices of the linear systems arising from applying a Newton's method to solve Equation (10) are M-matrices. This is important since some linear solvers converge only when the coefficient matrix is an M-matrix. We can thus always use these solvers when we use the upwind discretization, while we can do so only when condition Equation (7) is satisfied when we use central FD. This is of great importance especially when $\tilde{\boldsymbol{v}}$ is large: indeed, in this case the coefficient matrix of the linear systems is highly non-symmetric and we would like to use splitting methods such as the Arithmetic Mean (AM) [14], which are particularly efficient in this case but require that the coefficient matrix is an M-matrix. Thus, also considering that Equation (7) implies that h must be really small when $\tilde{\boldsymbol{v}}$ is large, we may want to use an upwind discretization in order to use these linear solvers. This is further studied in the numerical experiments.



$\nu+1$
$\boldsymbol{G}(\boldsymbol{u})$
t_{n+1}
$A(\boldsymbol{u}), \boldsymbol{b}(\boldsymbol{u}), \boldsymbol{G}(\boldsymbol{u})$
ν
$A(\boldsymbol{u}^{(\nu)}), \boldsymbol{b}(\boldsymbol{u}^{(\nu)})$
t_n
(**a**)
$\nu+1$
t_{n+1}
$A(\boldsymbol{u}), \boldsymbol{b}(\boldsymbol{u})$
ν
$A(\boldsymbol{u}^{(\nu)}), \boldsymbol{b}(\boldsymbol{u}^{(\nu)})$
t_n
$\boldsymbol{G}(\boldsymbol{u}(t_n))$
(**b**)
$\nu+1$
$\boldsymbol{G}(\boldsymbol{u})$
t_{n+1}
$A_1(\boldsymbol{u}), \boldsymbol{b}(\boldsymbol{u}), \boldsymbol{G}(\boldsymbol{u})$
ν
$A_1(\boldsymbol{u}^{(\nu)}), \boldsymbol{b}(\boldsymbol{u}^{(\nu)})$
t_n
$\tilde{A}(\boldsymbol{u}(t_n))$
(**c**)

Figure 1. Summary of the nonlinear terms present at each time level and at each lagging iteration for different time discretizations. (**a**) $\theta-$method; (**b**) IMEX scheme with explicit treatment of $\boldsymbol{G}(\boldsymbol{u})$; (**c**) IMEX scheme with non-constant velocity term $\tilde{\boldsymbol{v}}(\boldsymbol{u})$ treated explicitly. Here $A_1(\boldsymbol{u})$ represents the part of A dependent on $\sigma(\boldsymbol{u})$ and $\tilde{A}(\boldsymbol{u})$ represents the part of A dependent on $\tilde{\boldsymbol{v}}(\boldsymbol{u})$.

3.2. Monotonicity of the Finite-Difference Operator and Convergence Analysis

Let us assume that the following smoothness properties hold true:

1. the functions σ and g are continuous in u and continuously differentiable in (x, y). Moreover, the functions α and s are continuous in their variables;
2. σ is uniformly bounded in $(x, y) \in \Omega$ and $u \in L^\infty(\Omega)$: there exist two positive constants $\sigma_{\min}$ and $\sigma_{\max}$ such that $0 < \sigma_{\min} \leq \sigma(x, y, u) \leq \sigma_{\max}$. Moreover, we also assume $\alpha(x, y) \geq \alpha_{\min} \geq 0$ and that $\tilde{\boldsymbol{v}} = (\tilde{v}_1, \tilde{v}_2)^T$ is constant;
3. for fixed $(x, y) \in \Omega$, the function σ is Lipschitz continuous in u (uniformly in x, y) with constant $\Lambda > 0$;
4. for fixed $(x, y) \in \Omega$, the function g is uniformly monotone in u (uniformly in x, y) with constant $c > 0$ [15] (p. 114); moreover, it is continuously differentiable in u.

Under these smoothness assumptions, the monotonicity of the finite different operator and the convergence of the algorithm are assured for grid functions $u_{i,j}$ belonging to a set $\mathcal{B}_{\rho,\beta}$, where ρ is a bound on the discrete $l_2(\Omega_h)$ norm $\|\boldsymbol{u}\|_h := (h^2 \sum_{i=1}^N \sum_{j=1}^M u_{i,j}^2)^{\frac{1}{2}}$ and β is a bound over the absolute value of the backward difference quotients of $u_{i,j}$.

Thus, considering for instance the discretization by central finite differences and time integration by the $\theta-$method, the following theorems hold true

Theorem 1. *Let $\boldsymbol{u}^* \in \mathcal{B}_{\rho,\beta}$ be a solution of the nonlinear system $\boldsymbol{F}(\boldsymbol{u}) = \boldsymbol{0}$. If*

$$\alpha_{\min} + \frac{1}{\tau} + c > \frac{\beta^2 \Lambda^2}{2\sigma_{\min}} \tag{13}$$

then $\boldsymbol{F}(\boldsymbol{u})$ is uniformly monotone in $\mathcal{B}_{\rho,\beta}$ and the solution $\boldsymbol{u}^*$ of Equation (9) is unique.

Theorem 2. *Let $\boldsymbol{u}^* \in \mathcal{B}_{\rho,\beta}$ be the solution of the nonlinear system $\boldsymbol{F}(\boldsymbol{u}) = \boldsymbol{0}$ defined in Equation (9) with $A(\boldsymbol{u})$ non-singular M-matrix and $\boldsymbol{G}(\boldsymbol{u})$ diagonal mapping. We assume the smoothness Properties 1–4 to be satisfied and that condition Equation (13) on the monotonicity of $\boldsymbol{F}(\boldsymbol{u})$ is satisfied.*

Starting from an arbitrary starting vector $\boldsymbol{u}^{(0)}$, let then $\boldsymbol{u}^{(\nu+1)}$ be the solution of system Equation (10) with residual $\boldsymbol{r}^{(\nu+1)}$ satisfying Equation (12), with $\epsilon_\nu \to 0$ for $\nu \to \infty$. Finally, $\boldsymbol{u}^{(\nu)}$ belongs to $\mathcal{B}_{\rho_\nu,\beta_\nu}$, $\nu = 0, 1, \ldots$. Then, the sequence $\{\boldsymbol{u}^{(\nu)}\}$ converges to $\boldsymbol{u}^*$.

The proofs of these theorems and more information on $\mathcal{B}_{\rho,\beta}$ can be found in [16]. In the literature it is also possible to find proofs of the convergence of the procedure for less general cases (e.g., see [2] for steady state reaction diffusion problems). Proceeding analogously, similar results can be easily found also for the other discretizations.

4. Solution Procedure

In this section, we present some insights on the implementation that is used in the numerical experiments. This implementation follows the description in [2], with the difference that the inner systems are here solved by an inexact, simplified Newton iteration and that we use the correction on the initialization of tolerances presented in [16] and here summarized in Section 4.2. Then, we also briefly summarize the systems arising from the discretization and from the LDM and provide an algorithm which describes the entire procedure.

4.1. Solution of the Inner Systems

We solve the lagged system of Equation (10) by a simplified Newton method [12] (p. 182). At each Newton's iteration we then have to solve a system of linear equations. For achieving the maximum efficiency, we do this approximately by an iterative linear solver, whose tolerance is chosen so that the solution procedure of Equation (10) makes up an inexact (simplified) Newton method [17].

In particular, denoting the Newton's iteration by a second superscript $k = 0, 1, \ldots$, the simplified Newton iteration at the $(\nu + 1)$-th lagged iteration consists in computing the solution $\Delta \boldsymbol{u}^{(k+1)} = \boldsymbol{u}^{(\nu+1,k+1)} - \boldsymbol{u}^{(\nu+1,k)}$ of the linear system

$$F'_\nu(\boldsymbol{u}^{(\nu+1,0)})\Delta \boldsymbol{u} = -\boldsymbol{F}_\nu(\boldsymbol{u}^{(\nu+1,k)}), \tag{14}$$

where $F'_\nu(\boldsymbol{u}^{(\nu+1,0)})$ is the Jacobian matrix of $\boldsymbol{F}_\nu(\boldsymbol{u})$ evaluated at $\boldsymbol{u}^{(\nu+1,0)}$, which is

$$F'_\nu(\boldsymbol{u}^{(\nu+1,0)}) = I + \tau A(\boldsymbol{u}^{(\nu)}) + \tau G'(\boldsymbol{u}^{(\nu+1,0)}), \tag{15}$$

where $G'(\boldsymbol{u}^{(\nu+1,0)})$ is the Jacobian matrix of $\boldsymbol{G}(\boldsymbol{u})$ evaluated at $\boldsymbol{u}^{(\nu+1,0)}$. From Equation (15), we can appreciate the advantage of the lagging iteration. Indeed, since the LDM linearizes $A(\boldsymbol{u})$, it is easy to compute the Jacobian matrix of $F'_\nu(\boldsymbol{u})$, since the only nonlinear term is the diagonal mapping $\boldsymbol{G}(\boldsymbol{u})$.

Finally, the linear system arising at each Newton iteration is solved by an iterative solver. We henceforth refer to the inner linear iteration by a third superscript j.

4.2. Starting Vectors and Stopping Criteria

Starting vectors and tolerances of the iterative procedures are chosen as in the diagram in Figure 2, where $\hat{\eta} \in (0, 1)$ and ϵ_ν is initialized by $\epsilon_1 = \tilde{\epsilon}_0 \|\boldsymbol{F}(\boldsymbol{u}^{(0)})\|$, with $\tilde{\epsilon}$ positive constant smaller than 1.

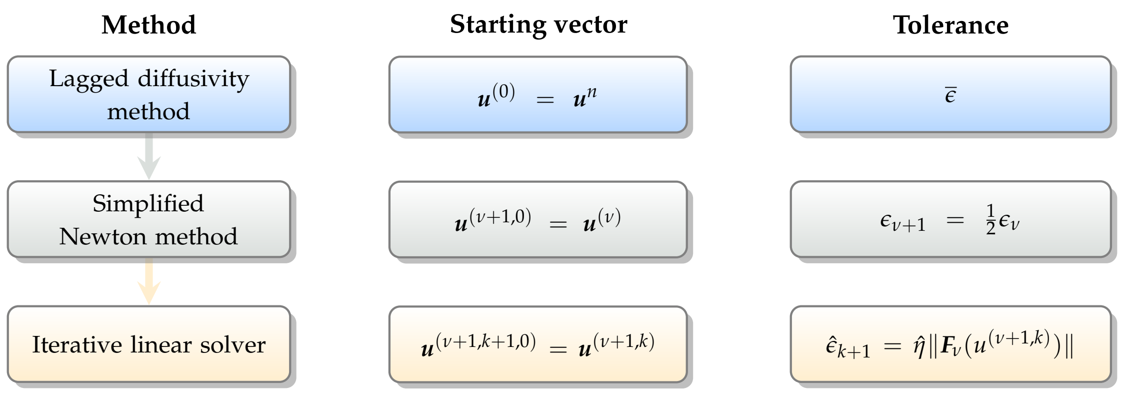

Method
Starting vector
Tolerance
Lagged diffusivity method
$\boldsymbol{u}^{(0)} = \boldsymbol{u}^n$
$\bar{\epsilon}$
Simplified Newton method
$\boldsymbol{u}^{(\nu+1,0)} = \boldsymbol{u}^{(\nu)}$
$\epsilon_{\nu+1} = \frac{1}{2}\epsilon_\nu$
Iterative linear solver
$\boldsymbol{u}^{(\nu+1,k+1,0)} = \boldsymbol{u}^{(\nu+1,k)}$
$\hat{\epsilon}_{k+1} = \hat{\eta}\|\boldsymbol{F}_\nu(u^{(\nu+1,k)})\|$

Figure 2. Starting vectors and tolerances of the iterative methods.

The choice of the starting vectors is made so to start as close as possible to the solution of the system. Moreover, it also allows to write conveniently the linear system arising at each Newton iteration (e.g., see [16]). The stopping criteria, on the other hand, descend directly by the description of the LDM and by the choice of an inexact Newton iteration to solve the inner systems.

The starting vectors determine also the initialization of the tolerances: for example, at the first Newton iteration of the $(\nu+1)$-th lagged iteration, the tolerance of the linear solver is $\hat{\epsilon}_1 = \hat{\eta}\|\boldsymbol{F}_\nu(\boldsymbol{u}^{(\nu+1,0)})\| = \hat{\eta}\|\boldsymbol{F}_\nu(\boldsymbol{u}^{(\nu)})\|$. We can also easily notice that all the tolerances are initialized at $\hat{\eta}\|\boldsymbol{F}_0(\boldsymbol{u}^{(0)})\| = \hat{\eta}\|\boldsymbol{F}(\boldsymbol{u}^{(0)})\| = \epsilon_1$ at the first lagged iteration.

However, if we do not apply any correction, the initial tolerance of the linear solver, $\hat{\epsilon}_1$, may become smaller than the minimum possible tolerance of the simplified Newton method, $\hat{\eta}\epsilon_{\nu+1}$. Thus, $\hat{\epsilon}_1$ would be smaller than what could be required by an inexact Newton method, leading to an increase in computational cost with no foreseeable improvement in accuracy.

To avoid this situation, we simply impose $\hat{\eta}\epsilon_{\nu+1}$ as the minimum acceptable tolerance of the linear solver:

$$\hat{\epsilon}_1 = \max\left(\hat{\eta}\|\boldsymbol{F}_\nu(\boldsymbol{u}^{(\nu)})\|, \hat{\eta}\epsilon_{\nu+1}\right). \tag{16}$$

This does not lead to a worsening of the accuracy of the lagged procedure (indeed, $\hat{\eta}\epsilon_{\nu+1}$ is the tolerance of the simplified Newton method), while significantly reducing the computational cost.

4.3. Summary of the Solution Method

The diagram in Figure 3 summarizes the equations and the systems that we used, illustrating how the partial differential equation is first transformed in a system of ordinary differential equations and then in a series of nonlinear algebraic systems, which are afterwards progressively linearized by the iterative methods.

$$\frac{\partial u}{\partial t} = \nabla \times (\sigma \nabla u) - \tilde{v} \times \nabla u - \alpha u - g + s(x,y) \in \Omega,\, t > t_0$$

$$\frac{\mathrm{d}\boldsymbol{u}(t)}{\mathrm{d}t} = -A(\boldsymbol{u}(t))\boldsymbol{u}(t) + \boldsymbol{b}(\boldsymbol{u}(t)) - \boldsymbol{G}(\boldsymbol{u}(t)) + \boldsymbol{s}(t)$$

$$\boldsymbol{F}(\boldsymbol{u}) = [I + \tau A(\boldsymbol{u})]\,\boldsymbol{u} - \tau\,[\boldsymbol{b}(\boldsymbol{u}) - \boldsymbol{G}(\boldsymbol{u})] - \boldsymbol{w} = \boldsymbol{0}$$

$$\boldsymbol{F}_\nu(\boldsymbol{u}) = \left[I + \tau A(\boldsymbol{u}^{(\nu)})\right]\boldsymbol{u} - \tau\left[\boldsymbol{b}(\boldsymbol{u}^{(\nu)}) - \boldsymbol{G}(\boldsymbol{u})\right] - \boldsymbol{w} = \boldsymbol{0}$$

$$\boldsymbol{F}'_\nu(\boldsymbol{u}^{(\nu)})\boldsymbol{u} = \tau G'(\boldsymbol{u}^{(\nu)})\boldsymbol{u}^{(\nu+1,k)} - \tau\boldsymbol{G}(\boldsymbol{u}^{(\nu+1,k)}) + \tau\boldsymbol{b}(\boldsymbol{u}^{(\nu)}) + \boldsymbol{w}$$

Figure 3. Systems arising from the discretization of the initial PDE and from the used iterative methods.

Finally, we can write the algorithm summarizing the entire procedure as follows.

Algorithm 1 Lagged diffusivity procedure

```
Require: initial condition $\boldsymbol{u}|_0$ at $t = 0$; a tolerance $\bar{\epsilon}$
 1: for $n = 1, 2, \ldots$ do                                                    Time level
 2:   Initialize solution vector for lagged iteration: $\boldsymbol{u}^{(0)} = \boldsymbol{u}|_{n-1}$
 3:   Initialize lagged tol.: $\epsilon_1 = \tilde{\epsilon}_0\|\boldsymbol{F}(\boldsymbol{u}^{(0)})\|$
 4:   for $\nu = 0, 1, \ldots$ do                                                Lagged iteration
 5:     Initialize linear solver tol.: $\hat{\epsilon}_1 = \max(\hat{\eta}\|\boldsymbol{F}_\nu(\boldsymbol{u}^{(\nu)})\|, \hat{\eta}\epsilon_{\nu+1})$
 6:     for $k = 0, 1, \ldots$ do                                                Simpl. Newton iteration
 7:       for $j = 0, 1, \ldots$ do                                              Linear solver iteration
 8:         Compute $(j+1)$-th iterate $\boldsymbol{u}^{(\nu+1,k+1,j+1)}$
 9:         Compute residual $\boldsymbol{r}_{j+1}$ of the linear system
10:         if $\|\boldsymbol{r}_{j+1}\| \leq$ then break
11:         j = j+1
12:       end for
13:       Compute Newton residual $\boldsymbol{F}_\nu(\boldsymbol{u}^{(\nu+1,k+1)})$
14:       if $\|\boldsymbol{F}_\nu(\boldsymbol{u}^{(\nu+1,k+1)})\| \leq \epsilon_{\nu+1}$ then break
15:       Update linear solver tol.: $\hat{\epsilon}_{k+1} = \hat{\eta}\|\boldsymbol{F}_\nu(\boldsymbol{u}^{(\nu+1,k+1)})\|$
16:       k = k+1
17:     end for
18:     Update vectors and matrices: find $\boldsymbol{F}_{\nu+1}(\boldsymbol{u}^{(\nu+1)})$
19:     $\nu = \nu + 1$
20:     $\epsilon_{\nu+1} = 0.5\epsilon_\nu$
21:     if $\epsilon_{\nu+1} \leq \bar{\epsilon}$ then return
22:   end for
23:   $n = n + 1$
24: end for
```

Here, **break** means that we exit the current loop and go back to the outer one. On the other hand, **return** means that we exit from the entire procedure and the algorithm returns the final result. If the smoothness assumptions in Section 3.2 are satisfied (and if the inner linear solver converges), then the convergence of Algorithm 1 is ensured by Theorem 2 or by analogous theorems, depending on the chosen discretization.

5. Numerical Experiments

We consider a square domain $\Omega = [0,1] \times [0,1]$, which we discretize by a uniform grid of $N \times N$ inner points. We build test problems from known solutions u^* (specified in the following) by computing an appropriate source term. When not otherwise specified, time discretization is performed by a θ−method with $\Delta t = 0.1$ and $\theta = 0.5$ and we compute the solution from a starting time $t_0 = 0$ to a final time $t_f = 1$.

In the test problems we choose $\alpha = 0$, $\sigma = 0.4 + 0.5u$ and $g = 100e^{0.5u}$, which has the form of the radiation model. We instead consider several different choices of u^* and of $\tilde{v}$.

The linear systems arising at each Newton iteration are solved considering Krylov and splitting methods. In particular, we use the Preconditioned Conjugate Gradient (PCG) method [18] when the coefficient matrix of the linear system is symmetric and positive definite (i.e. $\tilde{\boldsymbol{v}} = \mathbf{0}$). Here, the used preconditioner is the diagonal matrix M whose elements $m_{i,i}$ are given by the l_2-norm of the i-th row of the coefficient matrix of the linear system. We instead compare the AM method [14] and the BiConjugate Gradient-stabilized method with parameter l (BiCGstab(l)) [19] when $\tilde{\boldsymbol{v}} \neq \mathbf{0}$.

We choose $\bar{\epsilon} = 10^{-4}$; the other stopping criteria are determined as in Section 4. The maximum number of simplified inexact Newton iterations is $k_{\max} = 500$ and the maximum number of linear iterations is $j_{\max} = 10,000$. In the following subsections, ν^*, k^* and j^* denote, respectively, the total number of lagged, Newton and linear solver iterations required for computing the solution at the last time level. By res_0 and res we instead denote, respectively, the initial residual and the last residual of the linear system at the last time level, while err_h and rel. err_2 denote the global error (in $l_2(\Omega_h)$ and in relative Euclidean norm respectively).

5.1. Effect of the Discretization Scheme

Let us analyze the effect of the space discretization. In Table 1 we report the results obtained for different choices of $\tilde{\boldsymbol{v}}$ with $u^* = u_1^* := (1 + x - y)^3 t$. We set $N = 250$; with this choice, Equation (7) is satisfied for $\tilde{v}_1, \tilde{v}_2 < 200$.

Table 1. Comparison between discretization by central finite differences and by upwind scheme.

$\tilde{\boldsymbol{v}}$	**Discretization**	**Lin. Solver**	res_0	res	err_h	err_2 **rel.**	j^*	t_{tot}
$(0,0)^T$	Central FD	AM	91,812	1.37×10^{-5}	4.16×10^{-5}	1.97×10^{-5}	7581	341.9
		BiCG(1)	91,812	1.36×10^{-5}	4.16×10^{-5}	1.97×10^{-5}	1199	57.5
	Upwind	AM	91,812	1.37×10^{-5}	4.16×10^{-5}	1.97×10^{-5}	7581	356.5
		BiCG(1)	91,812	1.36×10^{-5}	4.16×10^{-5}	1.97×10^{-5}	1199	64.0
$(10,10)^T$	Central FD	AM	91,814	1.37×10^{-5}	3.63×10^{-5}	1.71×10^{-5}	6608	311.0
		BiCG(1)	91,814	1.33×10^{-5}	3.61×10^{-5}	1.71×10^{-5}	979	63.6
	Upwind	AM	92,397	1.38×10^{-5}	2.25×10^{-3}	1.07×10^{-3}	6710	407.9
		BiCG(1)	92,397	1.33×10^{-5}	2.25×10^{-3}	1.07×10^{-3}	967	72.5
$(300,300)^T$	Central FD	AM	93,532	1.38×10^{-5}	1.21×10^{-5}	5.71×10^{-6}	594	37.5
		BiCG(1)	-	-	-	-	-	-
	Upwind	AM	110,834	1.62×10^{-5}	9.07×10^{-3}	4.30×10^{-3}	748	41.7
		BiCG(1)	110,834	1.57×10^{-5}	9.08×10^{-3}	4.30×10^{-3}	1534	79.7
$(500,500)^T$	Central FD	AM	-	-	-	-	-	-
		BiCG(1)	-	-	-	-	-	-
	Upwind	AM	124,785	1.71×10^{-5}	9.41×10^{-3}	4.46×10^{-3}	552	42.2
		BiCG(1)	124,785	1.47×10^{-5}	9.41×10^{-3}	4.46×10^{-3}	1398	94.0

In Table 1, we see that the two discretization schemes are identical for $\tilde{\boldsymbol{v}} = \mathbf{0}$, also in terms of global errors and number of iterations. Indeed, in this case, the first-order derivative in the differential equation is multiplied by a null term. Thus, $A(\boldsymbol{u})$ is symmetric positive definite and the two discretizations are equivalent.

In the other cases, both methods always work properly when $\tilde{\boldsymbol{v}}$ respects the spacing condition, although global errors are larger when we use the upwind discretization. As mentioned above, this is to be attributed to the discretization of the first-order derivative by forward or backward finite difference quotients, leading to a larger discretization error.

Lastly, when the spacing condition is not respected anymore, we can have problems in the solution of the linear system. Indeed, the convergence of the AM method requires that the coefficient matrix is an M-matrix. Although the AM still works for values of $\tilde{v}$ for which this is not satisfied (e.g., $\tilde{v} = (300, 300)^T$), we soon have problems for larger values of the velocity term. For example, $\tilde{v} = (500, 500)^T$ (and any other larger velocity term) leads the AM to stall. Also the BiCGstab(1) presents some problems and it already stalls for $\tilde{v} = (300, 300)^T$. Contrary to the AM, however, the BiCGstab(1) does not explicitly require that the coefficient matrix is an M-matrix. Its stalling is, however, not surprising. Indeed, the BiCGstab(1) (which is equivalent to the BiCG-STAB [20]) is known for stalling especially when the coefficient matrix is real but its eigenvalues are complex with large imaginary part. This could certainly be the case, since, irrespective of the chosen discretization, the coefficient matrices are non-symmetric when $\tilde{v} \neq \mathbf{0}$. Complex eigenvalues are, thus, possible.

Then, it is interesting to see what happens increasing l in the BiCGstab(l) method. Indeed, l has been introduced exactly to overcome these difficulties. Let us then consider yet larger values of $\tilde{v}$ and analyze what happens using BiCGstab(2) and BiCGstab(4) as well. The results are reported in Table 2.

Table 2. Comparison between discretization by finite differences and by upwind scheme with high $\tilde{v}$.

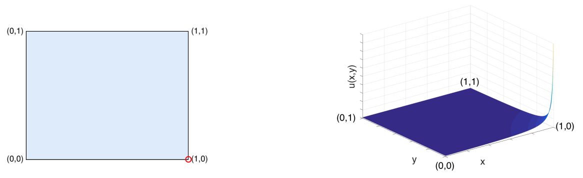
$\tilde{v}$	**Discretization**	**Lin. Solver**	res_0	res	err_h	err_2 **rel.**	j^*	t_{tot}
$(500, 500)^T$	Central FD	AM	-	-	-	-	-	-
		BiCG(1)	-	-	-	-	-	-
		BiCG(2)	96,513	1.14×10^{-5}	9.45×10^{-6}	4.47×10^{-6}	807	99.1
		BiCG(4)	96,513	1.37×10^{-5}	9.45×10^{-6}	4.47×10^{-6}	382	96.0
	Upwind	AM	124,785	1.71×10^{-5}	9.41×10^{-3}	4.46×10^{-3}	552	42.2
		BiCG(1)	124,785	1.47×10^{-5}	9.41×10^{-3}	4.46×10^{-3}	1398	94.0
		BiCG(2)	124,785	1.46×10^{-5}	9.41×10^{-3}	4.46×10^{-3}	673	72.7
		BiCG(4)	124,785	5.36×10^{-6}	9.41×10^{-3}	4.46×10^{-3}	299	96.8
$(2000, 2000)^T$	Central FD	AM	-	-	-	-	-	-
		BiCG(1)	-	-	-	-	-	-
		BiCG(2)	150,303	8.03×10^{-6}	4.77×10^{-6}	2.26×10^{-6}	629	69.6
		BiCG(4)	150,303	1.04×10^{-5}	4.77×10^{-6}	2.26×10^{-6}	296	89.7
	Upwind	AM	241,926	1.49×10^{-5}	9.81×10^{-3}	4.65×10^{-3}	370	36.7
		BiCG(1)	-	-	-	-	-	-
		BiCG(2)	241,926	1.37×10^{-5}	9.81×10^{-3}	4.65×10^{-3}	514	73.2
		BiCG(4)	241,926	1.74×10^{-5}	9.81×10^{-3}	4.65×10^{-3}	247	86.0
$(50000, 50000)^T$	Central FD	AM	-	-	-	-	-	-
		BiCG(1)	-	-	-	-	-	-
		BiCG(2)	2,976,435	1.36×10^{-5}	1.01×10^{-6}	4.79×10^{-7}	1077	162.4
		BiCG(4)	2,976,435	1.35×10^{-5}	1.01×10^{-6}	4.79×10^{-7}	507	197.7
	Upwind	AM	4,300,370	9.19×10^{-6}	9.96×10^{-3}	4.71×10^{-3}	322	33.5
		BiCG(1)	-	-	-	-	-	-
		BiCG(2)	4,300,370	2.39×10^{-6}	9.96×10^{-3}	4.71×10^{-3}	476	69.0
		BiCG(4)	4,300,370	8.79×10^{-6}	9.95×10^{-3}	4.71×10^{-3}	233	86.6

As we expected, the AM and the BiCGstab(1) are not able to solve the linear systems arising when $\tilde{v}$ is large and central FD are used. Indeed, they stall and the maximum number of allowed linear iterations is reached. On the other hand, if the discretization is performed by an upwind scheme, the AM method always converges without presenting any problem (on the contrary, the algorithm gets faster with larger values of $\tilde{v}$). This was to be expected, since the coefficient matrix is now an M-matrix, as required by the convergence of the AM method. The BiCGstab(1), instead, keeps stalling for large values of $\tilde{v}$, as we could expect from what we said in the previous paragraph. However, we can see that this can be overcome by increasing the parameter l of the BiCGstab(l): the algorithms using BiCGstab(2) or BiCGstab(4), indeed, converge in all cases. We can also see the big difference between the global errors obtained when discretizing by central finite differences or by an upwind scheme. Indeed, they are in the order of 10^{-6} for central finite differences and in the order of 10^{-3} for the upwind scheme. However, it is also to be noted that the upwind scheme is much faster, especially when we use the AM as inner solver. Thus, we may opt for central finite differences with, for instance, BiCGstab(2) as linear solver if we require smaller errors, while we may choose an upwind method with AM if we require lower computational cost.

5.2. Blow-Up

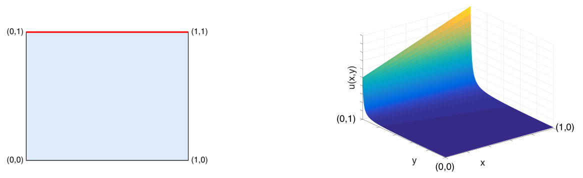
We now apply the LDM to problems with blow-up and boundary layer solutions. In particular, we consider the solutions in Figure 4, where T denotes the blow-up time.

$$u_2^* = \frac{1}{x(y+T-t)+1-x} \tag{17}$$



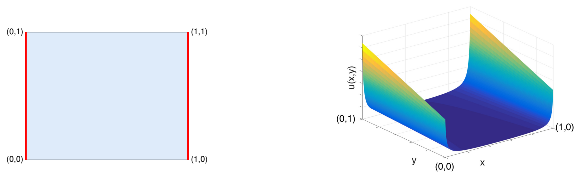
Blow-up zone Plot of u_2^* at $t \simeq T$

$$u_3^* = \frac{x+1}{1-x+T-t} \tag{18}$$



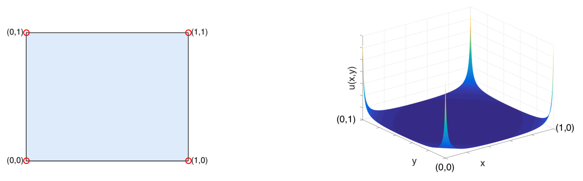
Blow-up zone Plot of u_3^* at $t \simeq T$

$$u_4^* = \frac{y+1}{\sin((T-t)\pi)+\sin(\pi x)}, \quad T < 1$$



Blow-up zones Plot of u_4^* at $t \simeq T$

$$u_5^* = \frac{1}{\sin((T-t)\pi)+\sin(\pi x)+\sin(\pi y)}, \quad T < 1$$



Blow-up zones Plot of u_5^* at $t \simeq T$

Figure 4. Blow-up solutions and representation of where and how the blow-up occurs.

We set $T = 1$ for $u^* = u_2^*$ and $u^* = u_3^*$; instead, u_4^* and u_5^* require $T < 1$ by the periodicity of the trigonometric functions. In these two cases we therefore set $T = 0.8$.

With no loss of generality, the velocity vector $\tilde{v}$ is here set equal to zero and the system matrix $A(\boldsymbol{u})$ is a Stieltjes matrix. We thus use the PCG solver for solving the linear systems arising at each Newton iteration and discretize space by central finite differences.

In the following results, $\max \boldsymbol{u}$ denotes the maximum of the computed solution vector at the last computed time level, while $\max u^*$ denotes the maximum of the analytical solution u^* at the last computed time level in the point where $\max \boldsymbol{u}$ occurs. Analogously, we define $\min \boldsymbol{u}$ and $\min u^*$.

5.2.1. Preliminary Results and Analysis

Let us first analyze how the computed solution evolves as the blow-up time approaches. To this end, let us start from $t_0 = 0$ and see what happens at different times, until reaching T; at first, we here choose $\Delta t = 0.1$ and $N = 100$. In Figure 5 we report what we get considering e.g., u_2^* (example of solutions with blow-up on one corner), u_4^* (example of solutions with blow-up on one or more edges) and u_5^* (example of solutions with blow-up on more corners). In Table 3 we report some information on errors and data of the iterative procedure.

Table 3. Error and data of the lagged diffusivity procedure for u_2^*, u_4^* and u_5^* as t approaches T.

t	res_0	res	err_h	err_2 rel.	max u	min u	k_t	j_t
$u^* = u_2^*$, $T = 1.0$								
0.3	369	1.35×10^{-5}	1.74×10^{-5}	1.86×10^{-5}	1.403	0.594	19	185
0.6	1209	8.66×10^{-6}	6.89×10^{-5}	6.27×10^{-5}	2.405	0.721	21	202
0.9	15,619	1.57×10^{-5}	1.04×10^{-3}	6.88×10^{-4}	8.402	0.918	25	254
$u^* = u_4^*$, $T = 0.8$								
0.3	592	1.03×10^{-5}	1.84×10^{-3}	1.88×10^{-3}	1.930	0.505	20	186
0.5	1828	1.68×10^{-5}	1.76×10^{-3}	1.56×10^{-3}	2.369	0.558	21	208
0.7	22,484	1.08×10^{-5}	1.54×10^{-2}	7.66×10^{-3}	5.851	0.772	25	263
$u^* = u_5^*$, $T = 0.8$								
0.3	112	1.53×10^{-5}	3.51×10^{-4}	7.59×10^{-4}	0.941	0.333	17	120
0.5	346	1.17×10^{-5}	3.29×10^{-4}	6.42×10^{-4}	1.148	0.356	19	138
0.7	3077	1.45×10^{-5}	1.36×10^{-3}	1.85×10^{-3}	2.694	0.433	22	175

Of course, in all these cases, the iterative procedure stops when we are still quite far from the blow-up time: indeed, using $\Delta t = 0.1$, the last computable time step is at $t = 0.9$ when $T = 1$ and $t = 0.7$ when $T = 0.8$. Indeed, the blow-up occurs at $t = 0.9 + \Delta t$ and at $t = 0.7 + \Delta t$ respectively. We therefore need a smaller time step in order to get closer to the singularity. However, we notice that the computed solutions behave as the analytical ones (as proven by the errors in Table 3) and that, as the blow-up time approaches, they get more and more similar to the ones in Figure 4.

It is also interesting to notice that the residual computed at the beginning of the lagged iteration, res_0, gets larger as we get closer to T. This can be viewed as an indicator of incoming blow-up: indeed, by the initialization $\boldsymbol{u}^{(0)} = \boldsymbol{u}^n$, at the time step $n + 1$ we have $res_0 = \|\boldsymbol{F}(\boldsymbol{u}^n)\|$. Therefore, res_0 is the residual of $\boldsymbol{F}(\boldsymbol{u}) = 0$ as in (9) for $\boldsymbol{u}^n$ at time $n + 1$. So, if the solution is smooth in the time variable (and Δt is not excessively large), we do not expect res_0 to be large: in these hypotheses, the solution $\boldsymbol{u}^{n+1}$ will not be dramatically different from the solution $\boldsymbol{u}^n$. However, if we have a singularity at some time T, the solution vector will change faster and faster as we approach T, leading to progressively larger initial residuals res_0. In this light, the analysis of res_0 gives information similar to the analysis of the derivative of $\boldsymbol{u}^*$ with respect to t.

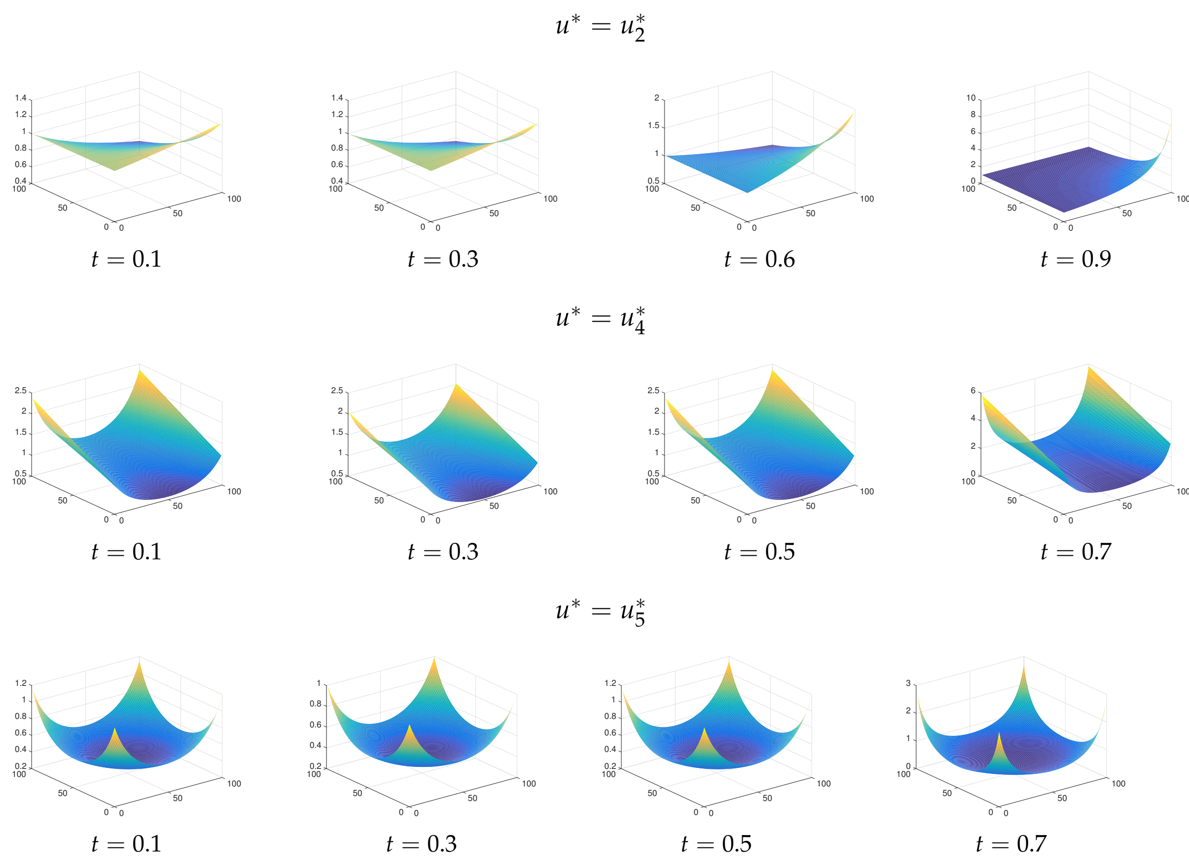


Figure 5. Evolution of the computed solution as t approaches T in the cases $u^* = u_2^*$, $u^* = u_4^*$ and $u^* = u_5^*$.

5.2.2. Effect of Refining the Time Grid

Taking into account the previous results, let us set $\Delta t = 10^{-3}$. We now start our analysis from $t_0 = T - 0.1$, so that we do not need an extremely large number of iterations in order to get close to the blow-up zone.

In Figure 6 we show how the computed solution looks like at the last computable time step before the blow-up; we call t_f this final time. In Table 4 we report some data on the lagged iterative procedure at $t = t_f$.

Table 4. Error and data of the lagged diffusivity procedure for the last computable solution for the problems in Figure 4 for $\Delta t = 10^{-3}$.

u^*	t_f	res_0	res	err_h	err_2 **rel.**	**max** u	**min** u	**max** u^*	**min** u^*
u_2^*	0.993	3.48×10^6	1.04×10^{-5}	4.74×10^{-3}	2.35×10^{-3}	37.545	1.000	37.545	1.000
u_3^*	0.972	1.28×10^{10}	1.03×10^{-5}	2.90×10^{-2}	3.54×10^{-3}	52.508	0.992	52.508	0.992
u_4^*	0.789	353,703	1.29×10^{-5}	3.54×10^{-1}	6.65×10^{-2}	30.314	0.976	30.314	0.976
u_5^*	0.806 [1]	8301	1.89×10^{-5}	9.77×10^{-2}	7.38×10^{-2}	24.049	0.505	23.067	0.505

[1] In this case we slightly exceed the limit $T = 0.8$. This is due to the discretization: also when we have $t > T$, the singularity is still sufficiently far from the grid points and the algorithm keeps working for a while. However, after a few other time steps, the algorithm stalls as expected. The role of the discretization can be proven by increasing the number of elements (e.g., see Table 5, where the algorithms stalls at $t = 0.795$).

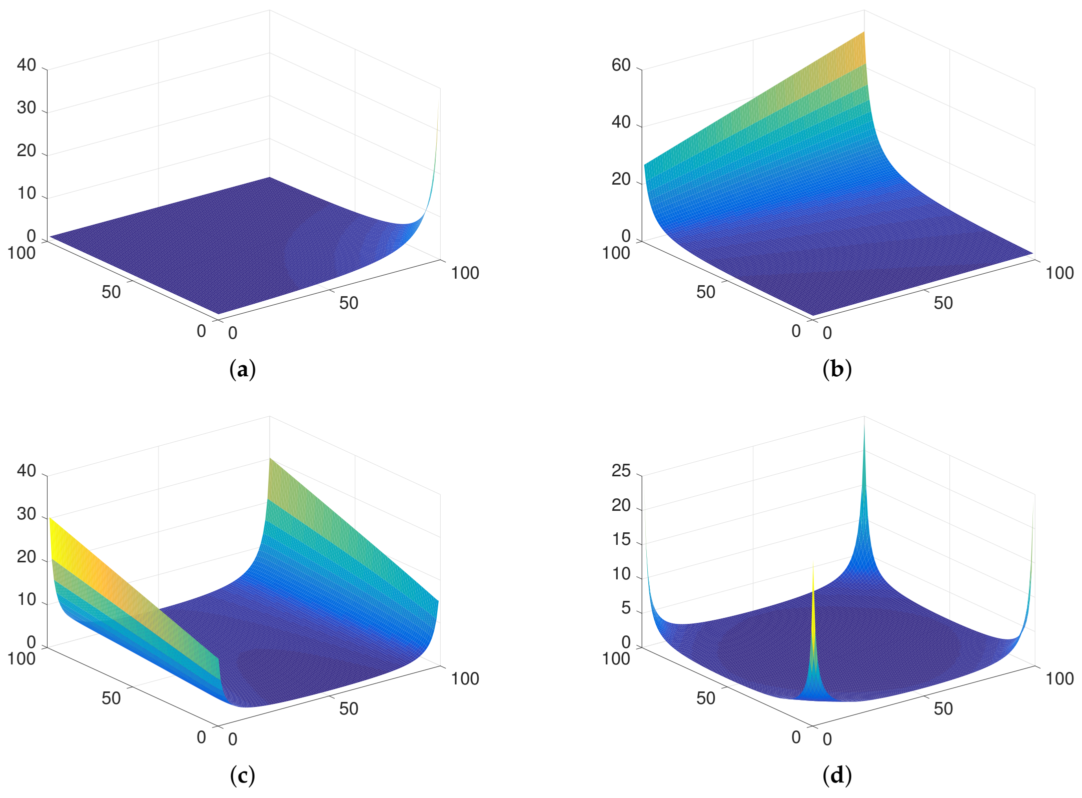


Figure 6. Last computed solution for the problems in Figure 4 for $\Delta t = 10^{-3}$. (**a**) u_2, $t_f = 0.993$; (**b**) u_3, $t_f = 0.972$; (**c**) u_4, $t_f = 0.789$; (**d**) u_5, $t_f = 0.806$.

The first thing we notice is the extremely large values of res_0, which are even more remarkable considering that Δt has been reduced. This can be explained as in the previous paragraph by considering that the solution changes more and more rapidly as we approach T. A gradual increase of res_0 with t is indeed what can be observed by considering the value of res_0 at different time levels.

The analysis of the plots reveals that the lagged diffusivity procedure is able to reproduce the behavior of all the solutions near their respective blow-up zones. On the other hand, the global errors gradually increase, but they generally remain in the order of $10^{-3} \div 10^{-2}$. Indeed, only for u_4^* they exceed 10^{-1} near T.

Notwithstanding this, maximum and minimum values of the solution vector $\boldsymbol{u}$ correspond to the maximum and minimum values of u^* (computed in the inner grid points) to at least the second decimal digit in all the cases. It is to be noticed that the maximum of $\boldsymbol{u}$ is the point where the solution is nearest to the blow-up one; a good precision in evaluating the solution in this point is thus important for accurately reproducing the blow-up behavior.

Finally, we also notice that these maxima and minima of $\boldsymbol{u}$ are not so large after all: we are near to the blow-up time, but the maximum of the solution never exceeds values ~ 30, nonetheless. This cannot be explained by numerical errors in the solution procedure, since we said that the maximum of the computed $\boldsymbol{u}$ is really similar to the value of $\boldsymbol{u}^*$ in the grid point where the maximum occurs. Different causes concur to this:

- the blow-up occurs on the boundary, where the solution is given by the Dirichlet boundary conditions. Thus, at inner points, where we compute the solution vector, the maximum of $\boldsymbol{u}$ is always smaller than the maximum of $\boldsymbol{u}$ at boundary points when $t \sim T$;
- as a consequence of the previous point, the smaller the number of grid points, the smaller $\max(\boldsymbol{u})$. Indeed, we get farther from the boundary and, thus, from the singularity. This is also shown in

the next subsection, where we observe that $\max(\boldsymbol{u})$ increases when we increase the number of points of the space grid;
- the singularity occurs abruptly when $t = T$ is reached, so the solution is still quite small also at times quite close to T. Next, we show that by further reducing Δt, we are able to get nearer to T, leading to an increase of $\max(\boldsymbol{u})$.

All these facts can be better viewed by considering a short example. Let us consider $u^* = u_2^*$ and $N = 100$. We remind that the blow-up here occurs at $T = 1.00$ in the point $(1, 0)$. The solution at $t = t_f = 0.993$ at the inner point closest to $(1, 0)$, which is $(x_n, y_1) = (100/101, 1/101)$ is:

$$\left.\frac{1}{x(y + T - t) + 1 - x}\right|_{(x_n,y_1)} = \frac{1}{x_n(y_1 + T - 0.993) + 1 - x_n} \sim 37.545,$$

as computed.

The algorithm, then, does not stall due to large values of $\boldsymbol{u}$, but because its derivatives become large. Indeed, considering the largest first-order derivative in x, we have

$$u_x|_{(x_n,y_1)} \simeq -\frac{y_1 + T - 0.993 - 1}{(x_n(y_1 + T - 0.993) + 1 - x_n)^2} \sim 10^3$$

This makes so that the bound β on backward difference quotients increases as the blow-up approaches. Then, it is harder to satisfy the monotonicity condition of Theorem 1, which, in turn, affects the uniqueness of the solution and the convergence of the LDM. We can try to counterbalance the larger β by reducing Δt (and, thus, τ, which appears in the condition of Theorem 1), as we verify in the next subsection. However, further decreasing Δt means increasing the number of time levels and, thus, the computational time. Nonetheless, we can compensate this effect by starting our computations nearer to the blow-up time.

5.2.3. Effect of Refining the Space Discretization

Let us then analyze what happens when the space grid is refined; in this regard, the results obtained with $N = 250$ and $\Delta t = 10^{-3}$ are reported in Table 5.

We notice that t_f slightly decreases, while the maximum and the minimum of $\boldsymbol{u}$ in t_f are about the same as previously computed for $N = 100$. Indeed, being the grid finer, the point nearest to the blow-up point is closer (in space) to the blow-up point than in the case $N = 100$. Thus, the solution gets larger earlier.

Table 5. Error and data of the lagged diffusivity procedure for the last computable solution for the problems in Figure 4 for $\Delta t = 10^{-3}$ and $N = 250$.

u^*	t_f	res_0	res	err_h	err_2 rel.	max u	min u	max u^*	min u^*
u_2^*	0.981	2.93×10^6	1.11×10^{-5}	9.68×10^{-4}	4.97×10^{-4}	37.207	0.985	37.207	0.985
u_3^*	0.965	9.86×10^9	1.95×10^{-5}	1.04×10^{-2}	1.34×10^{-3}	51.201	0.974	51.201	0.974
u_4^*	0.783	544,443	8.42×10^{-6}	1.53×10^{-1}	3.09×10^{-2}	30.287	0.953	30.290	0.953
u_5^*	0.795	14,438	1.20×10^{-5}	1.82×10^{-2}	1.42×10^{-2}	24.430	0.496	24.546	0.486

To better see this, we can again consider the short example proposed earlier and analyze what happens for $N = 250$. Let us again consider $u^* = u_2^*$ with $T = 1.00$ in the point $(1, 0)$. The solution at $t = t_f = 0.981$ at the inner point closest to $(1, 0)$, which is $(x_n, y_1) = (250/251, 1/251)$ is:

$$\left.\frac{1}{x(y + T - t) + 1 - x}\right|_{(x_n,y_1)} = \frac{1}{x_n(y_1 + T - 0.981) + 1 - x_n} \sim 37.207,$$

as computed. Analogous results can be easily verified with regard to the derivatives. We also notice that global errors decrease due to the finer space grid, while res_0 is of the same order of magnitude as above.

Lastly, we verify that further reducing Δt allows us to better approach the blow-up time, as remarked at the end of the previous subsection. Thus, in Table 6 we report the results obtained by setting $\Delta t = 10^{-5}$ and by using as t_0 the last computed time step t_f of Table 5.

Table 6. Error and data of the lagged diffusivity procedure for the last computable solution for the problems in Figure 4 for $\Delta t = 10^{-5}$, $N = 250$.

u^*	t_0	t_f	res_0	res	err_h	err_2 rel.	max u	min u	max u^*	min u^*
u_2^*	0.981	0.99193	3.67×10^8	7.42×10^{-7}	6.02×10^{-4}	2.87×10^{-4}	62.539	0.999	62.539	0.999
u_3^*	0.965	0.97141	2.75×10^8	7.48×10^{-5}	2.07×10^{-3}	2.41×10^{-4}	61.276	0.980	61.276	0.980
u_4^*	0.783	0.79343	6.87×10^8	1.05×10^{-6}	4.33×10^{-2}	5.89×10^{-3}	60.203	0.984	60.203	0.984
u_5^*	0.795	0.80277 [1]	1.13×10^9	2.15×10^{-7}	1.38×10^{-2}	9.34×10^{-3}	61.238	0.502	61.238	0.502

[1] Again, we slightly exceed the blow-up time; this is again due to the discretization.

In all cases we are able to get closer to the blow-up time, as expected. Moreover, global errors decrease.

5.3. Boundary Layer

Lastly, for completeness, let us consider a stationary case with non-constant $\tilde{v}$ which presents a boundary layer solution.

Let us consider the problem given by

$$-\sigma_1 \frac{\partial^2 u}{\partial x^2} - \sigma_2 \frac{\partial^2 u}{\partial y^2} + v \nabla u = 0 \tag{19}$$

and

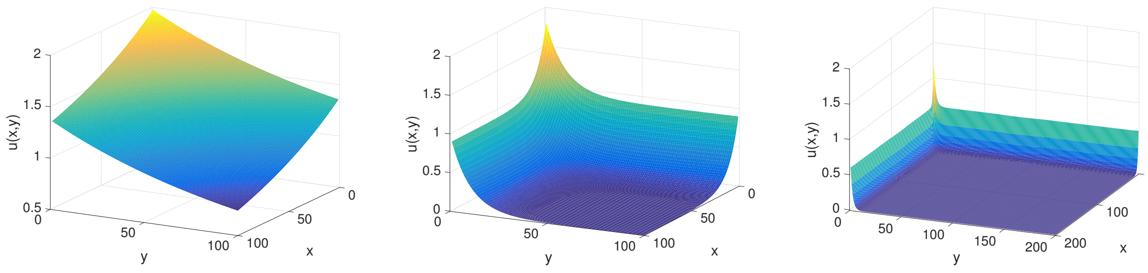
$$\sigma_1 = u^2 + 1 \qquad \sigma_2 = u^2 \qquad v_1 = \lambda u^2 + \lambda \qquad v_2 = \lambda u^2. \tag{20}$$

with $\lambda < 0$ real parameter and boundary condition $u = e^{\lambda x} + e^{\lambda y}$ on $\partial\Omega$. The solution of this test problem is evidently

$$u^* = e^{\lambda x} + e^{\lambda y}, \tag{21}$$

which presents a boundary layer as the absolute value of λ increases. Since we do not have any nonlinear reaction term, at each iteration of the LDM we just have to solve a linear system, which we do by the BiCGstab(l) algorithm.

The results as $|\lambda|$ increases are reported in Table 7 and the computed solutions are in Figure 7.



(**a**) $\lambda = -1$ (**b**) $\lambda = -10$ (**c**) $\lambda = -100$

Figure 7. Solution of the boundary layer test problem for $N = 250$; inner solver: BiCGstab(4).

Table 7. Analysis of a boundary layer example as $|\lambda|$ increases (boundary layer occurs for large $|\lambda|$).

$\tilde{v}$	**Lin. Solver**	res_0	res	err_h	err_2 **rel.**	v^*	j^*
$\lambda = -1$	BiCG(1)	1.6×10^6	1.48×10^{-4}	7.50×10^{-8}	5.84×10^{-8}	31	1925
	BiCG(2)	1.6×10^6	1.25×10^{-4}	7.26×10^{-8}	5.65×10^{-8}	31	977
	BiCG(4)	1.6×10^6	1.32×10^{-4}	7.09×10^{-8}	5.53×10^{-8}	31	460
$\lambda = -10$	BiCG(1)	1.7×10^6	1.41×10^{-4}	2.18×10^{-5}	6.43×10^{-5}	31	1389
	BiCG(2)	1.7×10^6	1.14×10^{-4}	2.18×10^{-5}	6.43×10^{-5}	31	708
	BiCG(4)	1.7×10^6	1.53×10^{-4}	2.18×10^{-5}	6.43×10^{-5}	31	366
$\lambda = -100$	BiCG(1)	1.8×10^6	1.36×10^{-4}	7.63×10^{-4}	9.36×10^{-3}	31	638
	BiCG(2)	1.8×10^6	1.41×10^{-4}	7.63×10^{-4}	9.36×10^{-3}	31	156
	BiCG(4)	1.8×10^6	9.10×10^{-5}	7.63×10^{-4}	9.36×10^{-3}	31	314

We notice that the number of iterations of the linear solver decreases as $|\lambda|$ increases. The global errors, instead, increase, but they never get larger than 10^{-3}, confirming the good approximation of the solution. In particular, Figure 7c, correctly reproduces the profile of a boundary layer solution.

6. Conclusions

We analyzed the effect of the discretization on the LDM for the solution of nonlinear, non-steady convection-reaction-diffusion equations. Some insights on an efficient implementation of the algorithm are provided and are followed by numerical experiments, which highlight the differences arising when the space discretization is performed by central finite differences or by the upwind method.

The paper is completed by a numerical study on the application of the LDM to problems characterized by blow-up and boundary layer solutions. The numerical experiments show that the LDM is able to correctly reproduce the analytical solution also in proximity of the blow-up time or in presence of a boundary layer.

Author Contributions: The two authors contributed equally to both the theoretical and experimental parts of the work presented in this paper. They elaborated and discussed the results together at all stages of the preparation of the manuscript, which they wrote and revised jointly.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Meyer, G. The numerical solution of quasilinear elliptic equations. In *Numerical Solution of Systems of Nonlinear Algebraic Equations*; Byrne, G., Hall, C., Eds.; Academic Press: New York, NY, USA, 1973.
2. Galligani, E. Lagged diffusivity fixed point iteration for solving steady-state reaction diffusion problems. *Int. J. Comp. Math.* **2012**, *89*, 998–1016.
3. Ding, J. Blow-up solutions for a class of nonlinear parabolic equations with Dirichlet boundary conditions. *Nonlinear Anal. Theor.* **2003**, *52*, 1645–1654.
4. Ding, J.; Wang, M. Blow-up solutions, global existence and exponential decay estimates for second order parabolic problems. *Bound Value Probl.* **2015**, *2015*, 160.
5. Ding, J.; Hu, H. Blow-up and global solutions for a class of nonlinear reaction diffusion equations under Dirichlet boundary conditions. *J. Math. Anal. Appl.* **2016**, *433*, 1718–1735.
6. Protter, M.H.; Weinberger, H.F. *Maximum Principles in Differential Equations*; Springer: New York, NY, USA, 1984.
7. Pucci, P.; Serrin, J. *The Maximum Principle*; Birkhauser Verlag AG: Basel, Switzerland; Boston, MA, USA; Berlin, Germany, 2007.
8. Wang, Y.; Mu, C. Blow-up and scattering of solution for a generalized Boussinesq equation. *Appl. Math. Comp.* **2007**, *188*, 1131–1141.
9. Abia, L.; Lopez-Marcos, J.; Martinez, J. The Euler method in the numerical integration of reaction-diffusion problems with blow-up. *Appl. Numer. Math.* **2001**, *38*, 287–313.

10. Galaktionov, V.A.; Vazquez, J.L. The problem of blow-up in nonlinear parabolic equations. *Discret. Contin. Dyn. Syst.* **2002**, *8*, 399–433.
11. Varga, R. *Matrix Iterative Analysis*, 2nd ed.; Springer: Berlin, Germany, 2000.
12. Ortega, J.; Rheinboldt, W. *Iterative Solution of Nonlinear Equations in Several Variables*; Classics in Applied Mathematics; SIAM: Philadelphia, PA, USA, 2000.
13. Isaacson, E.; Keller, H.B. *Analysis of Numerical Methods*; John Wiley & and Sons: New York, NY, USA, 1966.
14. Ruggiero, V.; Galligani, E. An iterative method for large sparse linear systems on a vector computer. *Comput. Math. Appl.* **1990**, *20*, 25–28.
15. Rheinboldt, W. *Methods for Solving Systems of Nonlinear Equations*, 2nd ed.; CBMS-NSF Regional Conference Series in Applied Mathematics; SIAM: Philadelphia, PA, USA, 1998.
16. Mezzadri, F.; Galligani, E. A Lagged Diffusivity Method for Reaction-Convection-Diffusion Equations with Dirichlet Boundary Conditions. Available online: http://www.nau.unimore.it/ANM-2017.pdf (accessed on 1 Auguest 2017).
17. Dembo, R.S.; Eisenstat, S.C.; Steihaug, T. Inexact Newton Methods. *SIAM J. Numer. Anal.* **1982**, *19*, 400–408.
18. Hestenes, M.R.; Steifel, E. Methods of conjugate gradient for solving linear systems. *J. Res. Natl. Bur. Stand.* **1952**, *49*, 409–436.
19. Sleijpen, G.L.G.; Fokkema, D.R. Bicgstab(l) for linear equations involving unsymmetric matrices with complex spectrum. *Electron. Trans. Numer. Anal.* **1993**, *1*, 11–32.
20. van der Vorst, H. Bi-CGSTAB: A fast and smoothly convergent variant to the Bi-CG for the solution of nonlinear systems. *SIAM J. Sci. Stat. Comp.* **1992**, *13*, 631–644.