Weierstraß-Institut
für Angewandte Analysis und Stochastik
Leibniz-Institut im Forschungsverbund Berlin e. V.

Preprint ISSN 2198-5855

An assessment of solvers for algebraically stabilized discretizations of convection-diffusion-reaction equations

Abhinav Jha[1], Ondřej Pártl[2], Naveed Ahmed[3], Dmitri Kuzmin[4]

submitted: November 4, 2021

[1] RWTH Aachen University
Applied and Computational Mathematics
Schinkelstr. 2
52062 Aachen
Germany
E-Mail: jha@acom.rwth-aachen.de

[2] Weierstrass Institute
Mohrenstr. 39
10117 Berlin
Germany
E-Mail: ondrej.partl@wias-berlin.de

[3] Gulf University for Science & Technology
Block 5, Building 1
Mubarak Al-Abdullah Area
West Mishref
Kuwait
E-Mail: ahmed.n@gust.edu.kw

[4] TU Dortmund University
Institute of Applied Mathematics (LS III)
Vogelpothsweg 87
44227 Dortmund
Germany
E-Mail: kuzmin@math.uni-dortmund.de

No. 2889
Berlin 2021

2020 *Mathematics Subject Classification.* 65M12, 65M15, 65M60.

Key words and phrases. Finite element methods, discrete maximum principles, algebraic flux correction, flux-corrected transport, monolithic convex limiting, iterative solvers.

Edited by
Weierstraß-Institut für Angewandte Analysis und Stochastik (WIAS)
Leibniz-Institut im Forschungsverbund Berlin e. V.
Mohrenstraße 39
10117 Berlin
Germany

Fax: +49 30 20372-303
E-Mail: preprint@wias-berlin.de
World Wide Web: http://www.wias-berlin.de/

An assessment of solvers for algebraically stabilized discretizations of convection-diffusion-reaction equations

Abhinav Jha, Ondřej Pártl, Naveed Ahmed, Dmitri Kuzmin

Abstract

We consider flux-corrected finite element discretizations of 3D convection-dominated transport problems and assess the computational efficiency of algorithms based on such approximations. The methods under investigation include flux-corrected transport schemes and monolithic limiters. We discretize in space using a continuous Galerkin method and $\mathbb{P}_1$ or $\mathbb{Q}_1$ finite elements. Time integration is performed using the Crank-Nicolson method or an explicit strong stability preserving Runge-Kutta method. Nonlinear systems are solved using a fixed-point iteration method, which requires solution of large linear systems at each iteration or time step. The great variety of options in the choice of discretization methods and solver components calls for a dedicated comparative study of existing approaches. To perform such a study, we define new 3D test problems for time dependent and stationary convection-diffusion-reaction equations. The results of our numerical experiments illustrate how the limiting technique, time discretization and solver impact on the overall performance.

1 Introduction

Traditional stabilization techniques for finite element discretizations of convection-diffusion-reaction(CDR) equations do not ensure the validity of discrete maximum principles [13]. As a consequence, numerical solutions may attain physically unrealistic values, and simulations may crash.

In the context of finite volume schemes and discontinuous Galerkin methods, the relevant inequality constraints are commonly enforced by using *limiters* for numerical fluxes or for slopes of piecewise-polynomial approximations. The first extensions of such schemes to continuous finite element approximations [19, 22] were based on generalizations of Zalesak's flux-corrected transport (FCT) algorithm [24].

During the last two decades, many alternatives were developed using the concept of algebraic flux correction (AFC). The AFC methodology [5, 18] is based on an algebraic splitting of a high-order target scheme into a bound-preserving low-order approximation and an antidiffusive correction term. The latter is decomposed into numerical fluxes that are limited to preserve important properties of the low-order method.

A theoretical framework for analysis and design of AFC schemes was developed in [5, 21] and used to construct improved limiter functions in [6, 21]. Further remarkable recent advances in the field include the development of a monolithic convex limiting strategy for nonlinear hyperbolic conservation laws and systems [17].

In this work, we focus on efficient numerical solution of the nonlinear discrete problems that arise from the AFC discretizations of time-dependent and stationary problems. The

DOI 10.20347/WIAS.PREPRINT.2889 Berlin 2021

overall computational cost depends on the type of the limiting strategy (predictor-corrector vs. monolithic) and on the time discretization (explicit vs. implicit). The efficiency of the implicit schemes and steady-state solvers depends on the convergence rates of the inner and outer iterations [10].

Unfortunately, these important aspects have received little attention in the AFC literature so far. We are not aware of any systematic numerical study focused on the overhead cost of the flux limiting and the performance–accuracy ratio. However, the practical use of AFC tools in simulation software for real-life applications requires a deeper understanding of such aspects. As a first step toward that end, we introduce new three-dimensional test problems and solve them using the AFC schemes proposed in [6, 12, 14, 17]. A direct comparison of CPU times for different approaches enables us to identify algorithms that offer the best performance for a certain class of problems.

This paper has the following structure: The numerical schemes that we study are described in Section 2 (for the evolutionary problems) and Section 3 (for the stationary problems). Our tests are in Section 4.

2 Evolutionary Convection-Diffusion-Reaction Equations

We consider the following initial-boundary value problem for a scalar evolutionary convection-diffusion-reaction equation: Find $u : (0,T] \times \Omega \to \mathbb{R}$ such that

$$\begin{aligned} u_t - \varepsilon\Delta u + \boldsymbol{b}\cdot\nabla u + cu &= f && \text{in } (0,T]\times\Omega, \\ u &= u_{\mathsf{D}} && \text{on } [0,T]\times\Gamma_D, \\ \varepsilon\nabla u\cdot\boldsymbol{n} &= g_{\mathsf{N}} && \text{on } [0,T]\times\Gamma_N, \\ u(0,\boldsymbol{x}) &= u_0(\boldsymbol{x}) && \forall \boldsymbol{x}\in\overline{\Omega}. \end{aligned} \tag{1}$$

Here $\Omega \subset \mathbb{R}^3$ is a bounded polyhedral domain, $\boldsymbol{n}$ is the outward pointing unit normal to the boundary $\Gamma = \Gamma_{\mathsf{D}} \cup \Gamma_{\mathsf{N}}$, $\Gamma_{\mathsf{D}} \cap \Gamma_{\mathsf{N}} = \emptyset$, and $[0,T]$ is a bounded time interval. Furthermore, ε, $0 < \varepsilon \ll 1$, is a diffusivity constant, $\boldsymbol{b} = \boldsymbol{b}(t,\boldsymbol{x})$ denotes a solenoidal velocity field, $c = c(t,\boldsymbol{x})$ stands for a nonnegative reaction coefficient, and $f = f(t,\boldsymbol{x})$ represents outer sources of the unknown scalar quantity u. On Γ_{D}, the Dirichlet boundary conditions (u_{D}) are set, and on Γ_{N}, the Neumann boundary conditions (g_{N}) are prescribed.

A standard finite element discretization of (1) with $\mathbb{P}_1$ or $\mathbb{Q}_1$ elements leads to a system of differential algebraic equations of the form

$$M_C\dot{\boldsymbol{u}} + A\boldsymbol{u} = \boldsymbol{F}, \tag{2}$$

where $M_C = \{m_{ij}\}_{i,j=1}^N$ is the consistent mass matrix, $A = \{a_{ij}\}_{i,j=1}^N$ is the stiffness matrix, and $\boldsymbol{F}$ is the corresponding right-hand side. The length of the vectors is denoted by N, which corresponds to the number of degrees of freedom. The matrix entries are given by

$$m_{ij} = (\varphi_j, \varphi_i)\,, \tag{3}$$

$$a_{ij} = \varepsilon\,(\nabla\varphi_j, \nabla\varphi_i) + (\boldsymbol{b}\cdot\nabla\varphi_j, \varphi_i) + (c\varphi_j, \varphi_i)\,, \tag{4}$$

where $(\cdot,\cdot)$ denotes the standard inner product in $L^2(\Omega)$, and $\{\varphi_i\}_{i=1}^N$ is the standard finite element basis.

DOI 10.20347/WIAS.PREPRINT.2889 Berlin 2021

The first step in the AFC methodology is to modify (2) so that we obtain an M-matrix $\mathbb{A}$ instead of A. For this purpose, we define the lumped mass matrix M_L and an artificial diffusion matrix D as follows:

$$M_L = \mathrm{diag}(m_i), \quad m_i = \sum_{j=1}^{N} m_{ij},$$

$$D = \{d_{ij}\}_{i,j=1}^{N}, \quad d_{ij} = -\max\{a_{ij}, 0, a_{ji}\} \ \text{ for } i \neq j, \quad d_{ii} = -\sum_{j=1, j\neq i}^{N} d_{ij}. \tag{5}$$

Replacing M_C by M_L and A by $\mathbb{A} = A + D$ in (2), we obtain

$$M_L \dot{\boldsymbol{u}} + \mathbb{A}\boldsymbol{u} = \boldsymbol{F}. \tag{6}$$

The temporal discretization of this equation yields a low-order scheme that is bound preserving, but overly diffusive.

To reduce this excessive diffusivity, we add an antidiffusive correction term $\boldsymbol{F}^*$ on the right-hand side of (6) to get

$$M_L \dot{\boldsymbol{u}} + \mathbb{A}\boldsymbol{u} = \boldsymbol{F} + \boldsymbol{F}^*. \tag{7}$$

To define $\boldsymbol{F}^*$, we first consider the residual difference $\boldsymbol{r}$ obtained by subtracting (2) from (6):

$$\boldsymbol{r} \;=\; (M_L - M_C)\,\dot{\boldsymbol{u}} + D\boldsymbol{u}. \tag{8}$$

Next, we decompose each component of $\boldsymbol{r}$ as

$$r_i = \sum_{j=1, j\neq i} r_{ij}, \quad \text{where} \quad r_{ij} = m_{ij}(\dot{u}_i - \dot{u}_j) + d_{ij}(u_j - u_i). \tag{9}$$

Using this decomposition, we set

$$F_i^* \;=\; \sum_{j=1, j\neq i}^{N} \alpha_{ij} r_{ij}, \tag{10}$$

where $\{\alpha_{ij}\}_{ij=1}^{N} \subset [0,1]$ are solution-dependent correction factors; algorithms for calculating them are called limiters. For $\alpha_{ij} = 1$, we revert to the standard Galerkin formulation, whereas setting $\alpha_{ij} = 0$ corresponds to the over-diffusive scheme. Note that the computation of r_{ij} is no longer required when i is a Dirichlet node.

Various definitions of α_{ij} have been proposed in the literature (see [6, 16, 17, 24]). Some flux limiters are defined at the semi-discrete level and applicable to steady-state problems as well. In other approaches, the fluxes r_{ij} and the correction factors α_{ij} are derived for a particular time-stepping method. In the next sections, we introduce the time discretizations and limiters used in our numerical studies.

2.1 Flux-Corrected Transport Algorithms

We begin with algorithms that use (generalizations of) Zalesak's FCT limiter [24] to calculate the correction factors α_{ij}. The antidiffusive fluxes r_{ij} corresponding to specific time integrators are defined below.

2.1.1 Crank-Nicolson Scheme

The Crank-Nicolson (CN) time discretization of the semi-discrete problem (7) yields the nonlinear system

$$\left[\frac{1}{\Delta t}M_L + \frac{1}{2}\mathbb{A}\right]\boldsymbol{u}^n = \left[\frac{1}{\Delta t}M_L - \frac{1}{2}\mathbb{A}\right]\boldsymbol{u}^{n-1} + \frac{1}{2}\boldsymbol{F}^n + \frac{1}{2}\boldsymbol{F}^{n-1} + \boldsymbol{F}^*(\boldsymbol{u}^n, \boldsymbol{u}^{n-1}), \tag{11}$$

where Δt is the time-step length, the superscripts denote the time levels, and the correction term $\boldsymbol{F}^*$ is assembled from limited counterparts $\alpha_{ij}r_{ij}$ of the antidiffusive fluxes [15, 18]

$$r_{ij} = \frac{m_{ij}}{\Delta t}\left[u_i^n - u_i^{n-1} - \left(u_j^n - u_j^{n-1}\right)\right] + \frac{d_{ij}}{2}\left[u_j^n + u_j^{n-1} - \left(u_i^n + u_i^{n-1}\right)\right]. \tag{12}$$

The CN-Galerkin scheme is a nondissipative high-order method, which tends to generate small ripples within the local bounds of the limiting procedure [18]. This behavior can often be cured by using a high-order linear stabilization or prelimiting [15]. In this work, we prelimit r_{ij} as follows:

$$r_{ij} = \text{minmod}\left(r_{ij},\, L\, d_{ij}(u_j - u_i)\right), \tag{13}$$

where

$$\text{minmod}(a,\, b) = \begin{cases} 0 & \text{if } \ ab < 0, \\ \min\{a, b\} & \text{if } \ a > 0 \wedge b > 0, \\ \max\{a, b\} & \text{if } \ a < 0 \wedge b < 0, \end{cases}$$

and $L = 2$ is a Lipschitz constant based on the analysis in [6].

In addition to the fluxes r_{ij}, Zalesak's limiter (as presented in Section 2.1.3 below) requires a bound-preserving intermediate solution $\tilde{\boldsymbol{u}}$ of low order. For the CN version, it is defined by

$$\tilde{\boldsymbol{u}} = \boldsymbol{u}^{n-1} - \frac{\Delta t}{2}M_L^{-1}\left(\mathbb{A}\boldsymbol{u}^{n-1} - \boldsymbol{F}^{n-1}\right), \tag{14}$$

which can be viewed as the solution of (6) at time $t_{n-1/2}$ computed using the forward Euler scheme with time step $\Delta t/2$. Note that $\tilde{\boldsymbol{u}}$ should be constrained to satisfy the Dirichlet boundary conditions for nodes belonging to Γ_D.

Remark 1. *As shown in [18, 21], the explicit predictor $\tilde{\boldsymbol{u}}$ is bound preserving under a CFL-like condition.*

We test two implementations of the CN-FCT algorithm: The first one solves the system of equations for $\boldsymbol{u}^n$ using a fixed point iteration method, which means that a sparse linear system needs to be solved at each step. A brief overview of the iterative procedure is given in Section 4. In what follows, we refer to this scheme as *nonlinear Zal+CN*.

The second algorithm is the linearized CN scheme proposed in [12] that utilizes $\tilde{\boldsymbol{u}}$ defined by (14) to approximate $\boldsymbol{u}^n$ by $2\tilde{\boldsymbol{u}} - \boldsymbol{u}^{n-1}$ in formula (12). Hence, it replaces (11) by a linear system for $\boldsymbol{u}^n$. In the following sections, we refer to this scheme as *linear Zal+CN*.

2.1.2 Second-Order SSP Scheme

As an alternative to the implicit CN scheme, we consider the second-order explicit strong stability preserving (SSP) time integrator commonly known as Heun's method. For a linear or nonlinear system of the form

$$\dot{\boldsymbol{u}}(t) = \boldsymbol{G}\left(\boldsymbol{u}(t), t\right), \tag{15}$$

the numerical solution at time t_n is given by

$$\boldsymbol{u}^n = \boldsymbol{u}^{n,0} + \frac{\Delta t}{2}\left[\boldsymbol{G}\left(\boldsymbol{u}^{n,0}, t^{n,0}\right) + \boldsymbol{G}\left(\boldsymbol{u}^{n,1}, t^{n,1}\right)\right] = \frac{\boldsymbol{u}^{n,0} + \boldsymbol{u}^{n,2}}{2},$$

where

$$t^{n,0} = t^{n-1}, \quad t^{n,1} = t^n, \tag{16}$$

$$\boldsymbol{u}^{n,0} = \boldsymbol{u}^{n-1}, \quad \boldsymbol{u}^{n,s} = \boldsymbol{u}^{n,s-1} + \Delta t \boldsymbol{G}\left(\boldsymbol{u}^{n,s-1}, t^{n,s-1}\right), \qquad s = 1, 2. \tag{17}$$

The application of this method to (7) requires two explicit Euler updates of the form

$$\boldsymbol{u}^{\text{new}} = \boldsymbol{u} + \Delta t M_L^{-1}\left(\boldsymbol{F} + \boldsymbol{F}^*(\boldsymbol{u}) - \mathbb{A}\boldsymbol{u}\right). \tag{18}$$

At each stage, the flux limiting is performed using Zalesak's algorithm with the low-order predictor

$$\tilde{\boldsymbol{u}} = \boldsymbol{u} + \Delta t M_L^{-1}\left(\boldsymbol{F} - \mathbb{A}\boldsymbol{u}\right) \tag{19}$$

and the antidiffusive fluxes

$$r_{ij} = m_{ij}\left(\dot{u}_i^L - \dot{u}_j^L\right) + d_{ij}\left(u_j - u_i\right), \tag{20}$$

where

$$\dot{\boldsymbol{u}}^L = M_L^{-1}\left(\boldsymbol{F} - \mathbb{A}\boldsymbol{u}\right) \tag{21}$$

stands for the low-order approximation given by (6). As shown in [15], the fluxes defined by (20) do not require prelimiting because the use of the low-order time derivatives introduces high-order linear stabilization.

2.1.3 Zalesak's Limiter

Given a low-order predictor $\tilde{\boldsymbol{u}}$ and an array of antidiffusive fluxes r_{ij}, our FCT schemes use Zalesak's limiter [24] to calculate the correction factors α_{ij} as follows:

1 Compute

$$P_i^+ = \sum_{j=1, j\neq i}^{N} \max\{r_{ij}, 0\}, \qquad P_i^- = \sum_{j=1, j\neq i}^{N} \min\{r_{ij}, 0\}.$$

2 Compute

$$\begin{aligned} Q_i^+ &= \max\left\{0, \max_{j=1,\ldots,N, j\neq i}(\tilde{u}_j - \tilde{u}_i)\right\}, \\ Q_i^- &= \min\left\{0, \min_{j=1,\ldots,N, j\neq i}(\tilde{u}_j - \tilde{u}_i)\right\}. \end{aligned}$$

3 Compute

$$R_i^+ = \min\left\{1, \frac{m_i Q_i^+}{\Delta t P_i^+}\right\}, \qquad R_i^- = \min\left\{1, \frac{m_i Q_i^-}{\Delta t P_i^-}\right\}.$$

If P_i^+ or P_i^- is zero, we set $R_i^+ = 1$ or $R_i^- = 1$, respectively. We also set $R_i^+ = R_i^- = 1$ if i is a Dirichlet node.

4 Compute

$$\alpha_{ij} = \begin{cases} \min\{R_i^+, R_j^-\} & \text{if } r_{ij} > 0, \\ \min\{R_i^-, R_j^+\} & \text{otherwise.} \end{cases}$$

The CN and SSP versions of Zalesak's FCT scheme differ in the definition of r_{ij} and $\tilde{\boldsymbol{u}}$.

2.2 Monolithic Convex (MC) Limiter

A potential drawback of FCT-like approaches is their dependence on the particular time-stepping method. As an alternative that is also applicable to stationary problems, we consider the monolithic convex (MC) limiting algorithm proposed in [17]. Flux limiters of this kind use definition (20) of r_{ij} for the semi-discrete scheme (7). The limited antidiffusive term of the CN scheme (11) is given by

$$F_i^*(\boldsymbol{u}^n, \boldsymbol{u}^{n-1}) = \frac{1}{2} \sum_{j=1, j\neq i}^{N} \left(\alpha_{ij}(\boldsymbol{u}^n) r_{ij}(\boldsymbol{u}^n) + \alpha_{ij}(\boldsymbol{u}^{n-1}) r_{ij}(\boldsymbol{u}^{n-1}) \right),$$

while the SSP version performs forward Euler updates (18) using

$$F_i^*(\boldsymbol{u}) = \sum_{j=1, j\neq i}^{N} \alpha_{ij}(\boldsymbol{u}) r_{ij}(\boldsymbol{u}).$$

The corresponding nonlinear space discretizations are of the form (15) and reduce to $\boldsymbol{G}(\boldsymbol{u}) = 0$ at steady state. At each fixed-point iteration or Runge-Kutta stage, the limited fluxes $\alpha_{ij} r_{ij} = r_{ij}^*$ are defined by

$$r_{ij}^* = \begin{cases} \min\left\{ r_{ij}, \min\left\{ 2d_{ij}\left(\bar{u}_{ij} - u_i^{\max}\right), 2d_{ij}\left(u_j^{\min} - \bar{u}_{ji}\right) \right\} \right\} & \text{if } r_{ij} > 0, \\ \max\left\{ r_{ij}, \max\left\{ 2d_{ij}\left(\bar{u}_{ij} - u_i^{\min}\right), 2d_{ij}\left(u_j^{\max} - \bar{u}_{ji}\right) \right\} \right\} & \text{otherwise,} \end{cases} \tag{22}$$

where $\bar{u}_{ij}$ are intermediate states defined by

$$2d_{ij}\bar{u}_{ij} = d_{ij}(u_i + u_j) + a_{ij}(u_j - u_i)$$

and

$$u_i^{\max} = \max_{j\in N_i} u_j, \quad u_i^{\min} = \min_{j\in N_i} u_j. \tag{23}$$

In the last formula, $N_i = \{j \in \{1, \ldots, N\} : m_{ij} \neq 0\}$ is the integer set containing the indices of node i and its nearest neighbors. Note that the definition of $u_i^{\max}$ and $u_i^{\min}$ can be changed to ensure linearity preservation [17, Section 6.1].

Remark 2. *Note that the MC limiter defined by* (22) *was designed for hyperbolic conservation laws. When applying this limiter to convection-diffusion-reaction (CDR) equations, we perform algebraic flux correction for the semi-discrete problem corresponding to* $\varepsilon = 0,\ c = 0$ *and add the unlimited discretization of* $\varepsilon\Delta u - cu$ *on the right-hand side of the resulting system.*

However, a proper extension of the MC limiter to CDR problems should include the diffusive and reactive terms in a manner that ensures preservation of local bounds. The development of such extensions is beyond the scope of the present work which is mainly focused on solver aspects.

3 Stationary Convection-Diffusion-Reaction Equations

In this section, we present AFC schemes for the stationary counterpart of (1), the boundary value problem

$$\begin{aligned} -\varepsilon\Delta u + \boldsymbol{b}\cdot\nabla u + cu &= f && \text{in } \Omega, \\ u &= u_{\mathsf{D}} && \text{on } \Gamma_D, \\ \varepsilon\nabla u\cdot\boldsymbol{n} &= g_{\mathsf{N}} && \text{on } \Gamma_N. \end{aligned} \tag{24}$$

Flux-limited discretizations of such problems always lead to a nonlinear system of equations. We solve these systems via the fixed point iteration studied in [10, 11] and outlined at the beginning of Section 4. This iterative solver requires the solution of a system of linear equations at each step.

When deriving our numerical schemes for (24), we use the same procedure as in Section 2, but the time derivatives vanish. Hence, the antidiffusive term $\boldsymbol{r}$ reduces to $\boldsymbol{r}^{\mathrm{ss}}$, where

$$r_i^{\mathrm{ss}} = \sum_{j=1, j\neq i} r_{ij}^{\mathrm{ss}} = \sum_{j=1, j\neq i} d_{ij}(u_j - u_i). \tag{25}$$

3.1 Monolithic Convex (MC) Limiter

The MC limiter presented in Section 2.2 is directly applicable to stationary problems. At each fixed-point iteration, the antidiffusive fluxes $r_{ij}^{\mathrm{ss}} = d_{ij}(u_j - u_i)$ are limited using formula (22). The validity of a discrete maximum principle for the converged steady-state solution was shown in [17, Theorem A.3].

3.2 Monolithic Upwind (MU) Limiter

This limiter was proposed in [14] and analyzed in [5]. Using the notation

$$r_{ij}^{\mathrm{ss},+} = \max\left\{r_{ij}^{\mathrm{ss}}, 0\right\}, \quad r_{ij}^{\mathrm{ss},-} = \min\left\{r_{ij}^{\mathrm{ss}}, 0\right\}, \tag{26}$$

the correction factors α_{ij} are computed as follows:

1 Compute

$$P_i^+ = \sum_{j=1,\, a_{ji}\leq a_{ij}}^{N} r_{ij}^{\mathrm{ss},+}, \qquad P_i^- = \sum_{j=1,\, a_{ji}\leq a_{ij}}^{N} r_{ij}^{\mathrm{ss},-}.$$

2 Compute

$$Q_i^+ = -\sum_{j=1}^{N} r_{ij}^{\mathrm{ss},-}, \qquad Q_i^- = -\sum_{j=1}^{N} r_{ij}^{\mathrm{ss},+}.$$

3 Compute

$$R_i^+ = \min\left\{1, \frac{Q_i^+}{P_i^+}\right\}, \qquad R_i^- = \min\left\{1, \frac{Q_i^-}{P_i^-}\right\}.$$

If i is a Dirichlet node or if P_i^+ or P_i^- is zero, the corresponding R_i^+ or R_i^- is set to 1.

4 For all i, j such that $a_{ji} \leq a_{ij}$, set

$$\alpha_{ij} = \begin{cases} R_i^+ & \text{if } r_{ij}^{ss} > 0, \\ 1 & \text{if } r_{ij}^{ss} = 0, \\ R_i^- & \text{if } r_{ij}^{ss} < 0, \end{cases} \qquad \alpha_{ji} = \alpha_{ij}. \tag{27}$$

Similarly to the MC limiter, this algorithm was designed for the hyperbolic case. It exploits the skew symmetry of the discrete convection operator and requires a careful extension to transport problems with diffusion and/or reaction.

3.3 Linearity Preserving (LP) Limiter

This limiter, which makes the AFC scheme linearity preserving, was introduced in [6]. It is custom-made for $\mathbb{P}_1$ elements. The correction factors α_{ij} are computed as follows:

1 Compute

$$P_i^+ = \sum_{j=1, j\neq i}^{N} r_{ij}^{\mathrm{ss},+}, \qquad P_i^- = \sum_{j=1, j\neq i}^{N} r_{ij}^{\mathrm{ss},-},$$

where $r_{ij}^{\mathrm{ss},+}$ and $r_{ij}^{\mathrm{ss},-}$ are given by (26).

2 Compute

$$Q_i^+ = q_i(u_i^{\max} - u_i), \qquad Q_i^- = q_i(u_i^{\min} - u_i),$$

where the bounds $u_i^{\max}$ and $u_i^{\min}$ are defined as in (23), and

$$q_i = -\sum_{j \in N_i} \gamma_i d_{ij}$$

for a positive constant γ_i depending only on the shape of the spatial grid in the nearest vicinity of the node i. We define γ_i as in [6, Rem. 6.2].

3 Compute

$$R_i^+ = \min\left\{1, \frac{Q_i^+}{P_i^+}\right\}, \qquad R_i^- = \min\left\{1, \frac{Q_i^-}{P_i^-}\right\}.$$

If i is a Dirichlet node or if P_i^+ or P_i^- is zero, the corresponding R_i^+ or R_i^- is set to 1.

4 For all i, j define

$$\overline{\alpha}_{ij} = \begin{cases} R_i^+ & \text{if } r_{ij}^{ss} > 0, \\ 1 & \text{if } r_{ij}^{ss} = 0, \\ R_i^- & \text{if } r_{ij}^{ss} < 0. \end{cases}$$

For each combination of non-Dirichlet nodes i and j, set

$$\alpha_{ij} = \min\left\{\overline{\alpha}_{ij}, \overline{\alpha}_{ji}\right\}.$$

For each combination of a Dirichlet node j and a non-Dirichlet node i, set

$$\alpha_{ij} = \overline{\alpha}_{ij}.$$

The boundary conditions are taken into account by setting $a_{ij} = 0$ for each combination of a non-Dirichlet node i and a Dirichlet node j.

4 Numerical Studies

In this section, we perform numerical studies for stationary and evolutionary convection-diffusion-reaction problems in 3D domains to evaluate the accuracy and efficiency of the above AFC methods.

The nonlinear systems corresponding to flux-corrected schemes from Sections 2 and 3 were solved using a fixed point method with dynamic damping. We give a brief overview of the scheme for the stationary problem. The same solution strategy is used for the evolutionary problem. We refer the reader to [10] for a detailed explanation.

The matrix form of the nonlinear schemes under investigation is given by

$$\mathbb{A}\boldsymbol{u} = \boldsymbol{F} + \boldsymbol{F}^*(\boldsymbol{u}).$$

We solve such nonlinear systems using fixed-point iterations of the form

$$\mathbb{A}\boldsymbol{u}^{\nu+1} = \boldsymbol{F} + \omega\boldsymbol{F}^*(\boldsymbol{u}^{\nu}),$$

where ν is the ν-th iteration step, and ω is a dynamic damping parameter. Here, the matrix $\mathbb{A}$ is a constant M-matrix, and thus must be factorized once, and the factorization can be reused in the iteration loop. For the CN discretization of a time-dependent problem, the iteration matrix is $((\Delta t)^{-1}M_L + \frac{1}{2}\mathbb{A})$. This is also a constant M-matrix, and hence can be reused in the iteration process.

To evaluate the efficiency of different limiters, we compare the computation times using the following parameters:

1 **Choice of solvers:** We have tested several direct and iterative solvers for linear systems available in the Portable library, Extensible Toolkit for Scientific Computation Toolkit for Advanced Optimization (PETSc), Release version 3.14.3 [2–4]. The solvers, along with the PETSc arguments and abbreviations used in the simulations, are listed in Tables 1 and 2. They are all used with the default settings, except that we set $\texttt{rtol} = 0$, and we use various values of `atol`. This means that the stopping criterion for the linear solvers was

$$R < \texttt{atol},$$

where R is the Euclidean norm of the residual. We also specified various maximum numbers of iterations.

Table 1: Iterative solvers and the corresponding PETSc arguments.

Name	**PETSc arguments** (`-ksp_type`)	**Abbreviation**	**Marker**
flexible GMRES	`fgmres`	FGMRES	•
loose GMRES	`lgmres`	LGMRES	■
stabilized BiCG	`bcgs`	BCGS	▲

2 **Choice of preconditioners**: The iterative solvers of Table 1 are used in combination with various preconditioners from PETSc, see the list of preconditioners along with the corresponding PETSc arguments and abbreviations in Table 3.

Table 2: Direct solvers and the corresponding PETSc arguments.

Name	**PETSc arguments** (`-pc_factor_mat_solver_type`)	**Abbreviation**
LU factorization	`mumps`	LU
UMFPACK	`umfpack`	UMFPACK

Table 3: List of preconditioners and PETSc arguments.

Name	**PETSc arguments** (`-pc_type`)	**Abbreviation**	**Color**
Jacobi	`jacobi`	Jac	gold
point block Jacobi	`bjacobi`	BJac	red
successive over relaxation	`sor`	SOR	green
additive Schwarz method	`asm`	ASM	blue
multigrid	`mg`	MG	—

3 **Effect of parallelization**: The numerical simulations are performed both sequentially and in parallel and compared in terms of the resulting computing times. The parallel calculations were performed using the *Open Run-Time Environment* (OpenRTE), version 1.10.7.0.5e373bf1fd [8], which originated from the Open MPI project.

In our representation, the number of processors is denoted by NP. We consider NP $= 4$, 8, 16, and 32. The sequential case is denoted by NP $= 1$.

Note that the definitions of the preconditioners BJac, SOR, and ASM depend on the number of processors by default (e.g., there is one block for each processor in BJac). We considered this dependence to be so natural that we kept it when we changed the number of processors. So, strictly speaking, if we changed the number of processors, we were no longer using the same solver.

To check the accuracy of the schemes, we compare the L^1 and L^2 norms (denoted by $\|\cdot\|_1$ and $\|\cdot\|_2$) of the error $(u - u_h)$, the plots of the solution, and the time evolution of the solution for a given point.

In the presentation of the numerical results, the following is always the same:

- The spatial grid is always generated by successive uniform refinement starting from an initial grid. The number of refinement steps performed is called the *(refinement) level*.
- Unless otherwise specified, the termination criterion for the nonlinear solver is

$$R < \sqrt{\#\mathrm{DOF}} \cdot \mathtt{tol}, \tag{28}$$

  where $\#\mathrm{DOF}$ is the number of degrees of freedom, and `tol` is a positive number close to zero.
- We always use the same pattern when representing the computing times in figures (see, e.g., Figure 6, page 23): The lines for BJac are always red, the lines for BCGS are always dotted and so on. Tables 1 and 3 contain the colors and the indices used in this work to denote the solvers and preconditioners.

The simulations were performed using the in-house code ParMooN [23] on the computer HPE Synergy 660 Gen10 with 2 Xeon eighteen-core processors, 3000 MHz and 768 GB RAM.

4.1 Time-dependent problems

4.1.1 Rotating shapes

Our first example is inspired by the well-known two-dimensional transport problem with rotating bodies [20]. As far as we know, this example has not been treated in the literature before. This problem aims to investigate the accuracy of the newly introduced MC limiter and to compare the results with the Zalesak limiter.

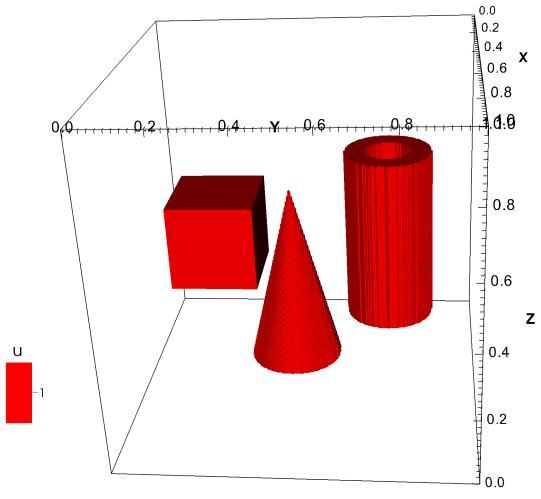
Description of Problem

We consider (1) with $\Omega = (0,1)^3$, $\varepsilon = 0$, $\boldsymbol{b} = (0.5-y, x-0.5, 0)^{\mathsf{T}}$, $c = 0$, $f = 0$, $T = 2\pi$, $\Gamma_{\mathsf{D}} = \Gamma$, $\Gamma_{\mathsf{N}} = \emptyset$, $g_{\mathsf{N}} = 0$, and $u_{\mathsf{D}} = 0$. The initial condition u_0 is depicted in Figure 1, i.e., u_0 takes the value 1 in the volumes enclosed by the red surfaces and zero everywhere else. These volumes have the following properties:

- cube: edges of length 0.25 parallel to coordinate axes, center at $(0.5, 0.25, 0.5)^T$;
- cone: height 0.5 and bottom surface of radius 0.125 and center at $(0.75, 0.5, 0.25)^T$ parallel to plane $z = 0$;
- hollow cylinder: top and bottom surfaces parallel to plane $z = 0$, inner radius 0.0625, outer radius 0.125, height 0.5, bottom surface with center at $(0.5, 0.75, 0.25)^T$.

The velocity field $\boldsymbol{b}$ makes the spatial shapes rotate around the axis $x = y = 0.5$. One full rotation takes $t = 2\pi$. Since ε and f both are zero, one should obtain a solution similar to the initial solution after one full rotation.

Figure 1: Example 4.1.1: Initial condition.



Numerical simulations were performed with the uniform cube mesh with the edge length 2^{-8} and a fixed time step length $\Delta t = 10^{-3}$. The implicit schemes used LGMRES with SOR as the preconditioner.

Finally, we set $\texttt{atol} = 10^{-25}$ for the PETSc solver. The nonlinear solver stopped if $R < 10^{-20}$ or after 50 iterations. The reason for this maximum number of iterations is explained below.

Discussion of Results

The initial conditions and the solutions after a complete rotation in the plane $z = 0.5$ are shown in Figure 2. The latter were calculated using all of the schemes mentioned in Section 2.

Important criteria for assessing the quality of the solutions are the smearing of the layers and the size of the undershoots and overshoots. This is compared in Figure 2 that shows the following:

The nonlinear Zal+CN smeared the solution the least, but the layers are uneven. This is probably due to our low limit on the number of the nonlinear iterations; the linear Zal+CN would also smear the layer very unevenly if we decreased `atol`.

The other schemes smeared the solution quite uniformly; but the schemes with the MC limiter or SSP produced slightly more uniform smearing than the linear Zal+CN.

None of the schemes produced undershoots or overshoots that were larger than the machine precision.

Figure 3 compares the time evolution of the solution at the point $(0.5, 0.25, 0.5)^T$ (i.e., the initial center of the cube): We can see that the schemes with the MC limiter produced essentially the same results. The results by the linear Zal+CN are almost identical to them, but they are a bit better in some parts: e.g., near the times 0.6 or 2.6. The results of Zal+SSP are very similar to the linear Zal+CN. The least diffusive results are produced by the nonlinear Zal+CN. However, all of the results are very similar. The errors in $\|\cdot\|_2$ presented in Table 4 confirm these observation.

Table 4: Example 4.1.1: Errors at final time measured in $\|\cdot\|_2$ and computing times.

	MC+SSP	**MC+CN**	**linear Zal+CN**	**nonlinear Zal+CN**	**Zal+SSP**
error	5.85e-2	5.85e-02	5.76e-2	4.58e-2	5.76e-2
computing time (hrs)	29.9	106.6	17.7	254.4	22.7

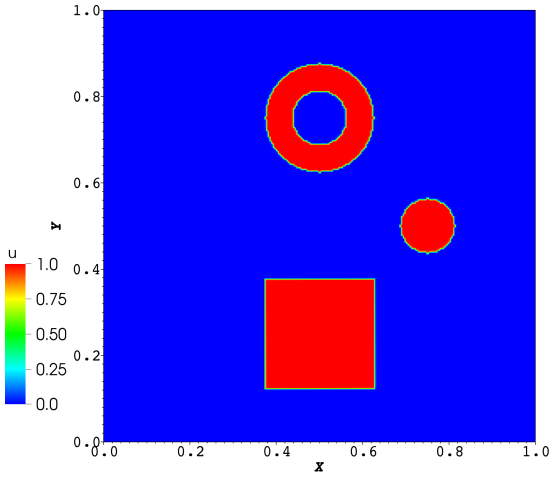
The largest difference between the schemes is in the computing times (Table 4): The computing time by the nonlinear Zal+CN is by far the worst. The reason is that the rate of the decrease in the residual during the nonlinear loop steadily decreases in the course of the simulations. (For MC+CN, on the contrary, this rate is roughly constant.) This property makes the scheme impractical for large problems and long simulations; and because of this, we decided to limit the number of the nonlinear iterations by 50. Near the end of the simulation, the final norm of the residual in the nonlinear loop was around 10^{-13}.

When comparing the nonlinear Zal+CN with the other schemes, we believe that the slightly higher precision of the solution does not compensate for the excessively long computing time at all.

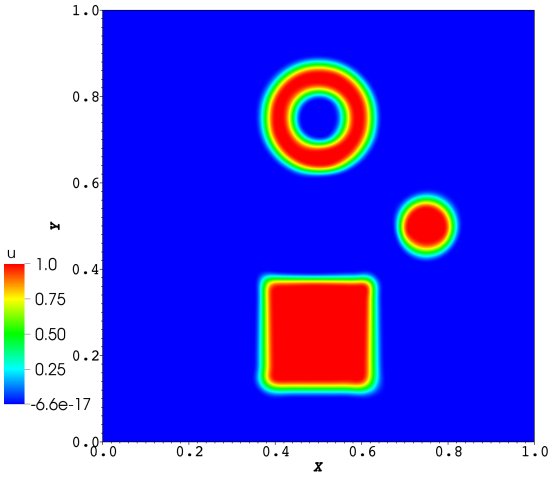
One might be surprised that the explicit solvers are not the fastest ones because they do not require any (non)linear systems to be solved. The reason is that one has to do the assembling

Figure 2: Example 4.1.1: $u_0(\boldsymbol{x})$ and the resulting $u(T, \boldsymbol{x})$ for $\boldsymbol{x}$ in the plane given by $z = 0.5$.

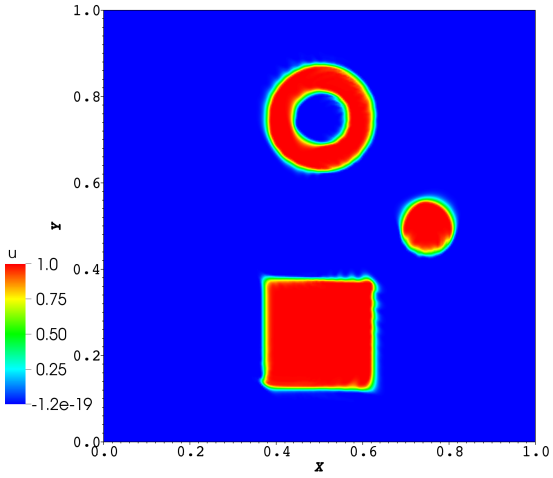
(a) Initial condition.



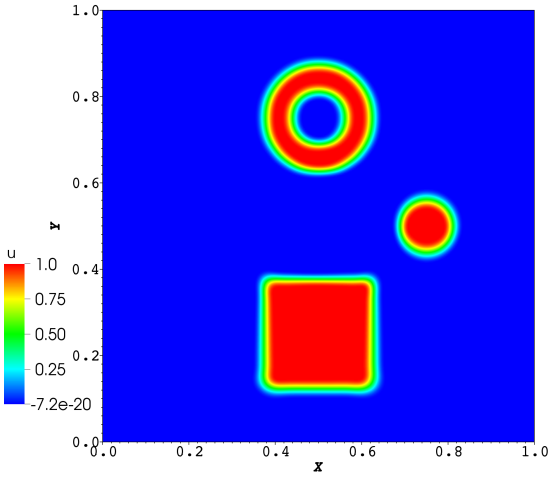
(b) Linear Zal+CN.



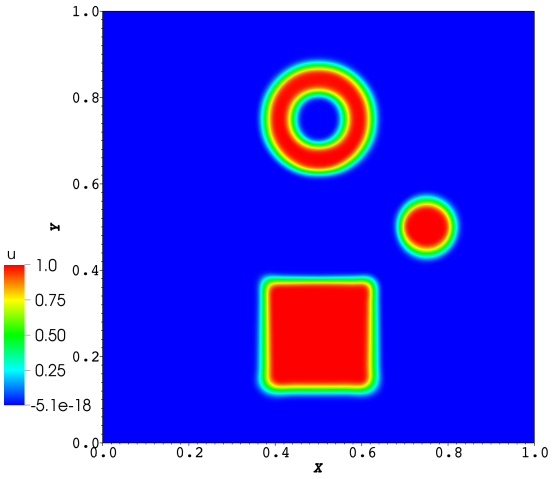
(c) Nonlinear Zal+CN.



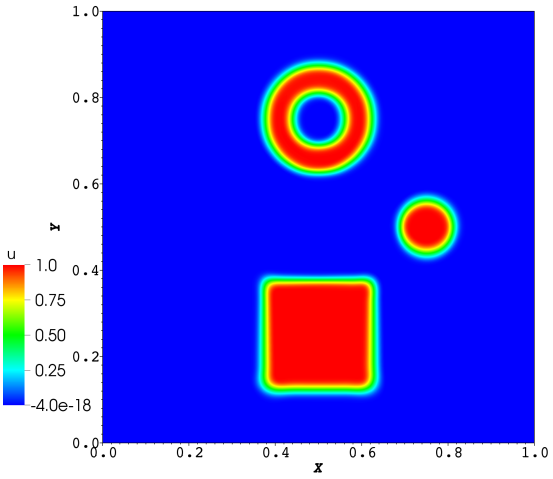
(d) Zal+SSP.



(e) MC+CN.



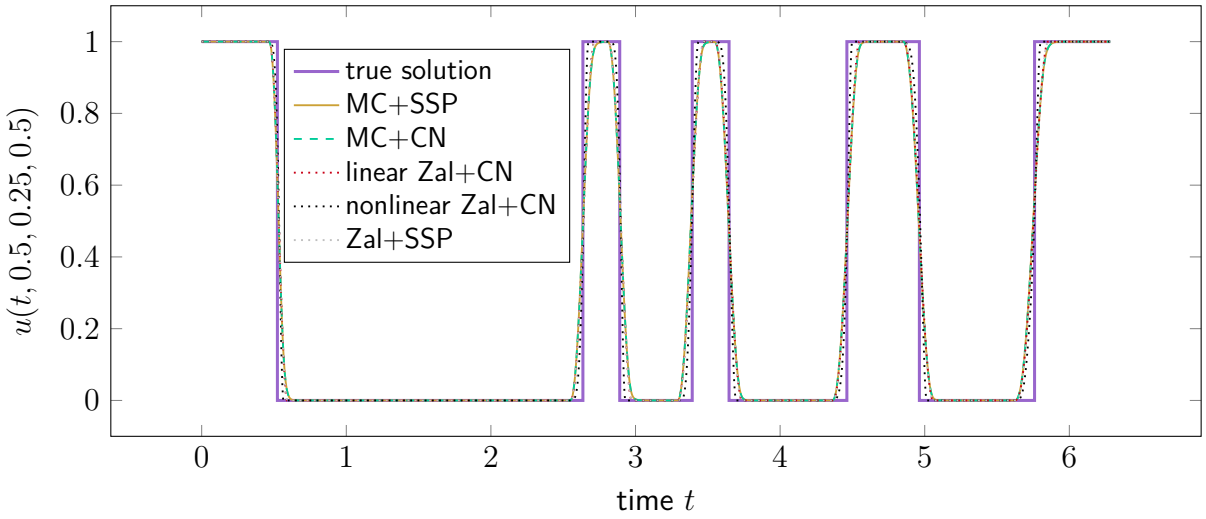
(f) MC+SSP.



twice in each time step (once for each SSP stage), and in our programs, this assembling is more expensive than solving the linear system.

Finally, Zal+SSP was faster than MC+SSP. This is because of the complexity of the assembling in the schemes with the MC limiter (see Remark 2, page 6).

Figure 3: Example 4.1.1: Time evolution of the numerical solution and the true solution at the point $(0.5, 0.25, 0.5)^T$ (i.e., the initial center of the cube).



4.1.2 Concentration of species

This example was proposed in [13] and can be also found in [1, 12]. We use it to compare the efficiency of the solvers employed to solve the linear systems arising from the AFC schemes. We are also interested in the comparison between the Zalesak and the MC limiter.

Description of Problem

This example models a typical situation where a species enters the domain at the entrance, travels through the domain while breeding and leaves it at the exit.

We consider (1) for $\Omega = (0,1)^3$, $\varepsilon = 10^{-6}$, $\boldsymbol{b} = (1, -1/4, -1/8)^T$, $f = 0$ and

$$c(x) = \begin{cases} 1 & \text{if distance}(x, g) \leq 0.1, \\ 0 & \text{otherwise,} \end{cases} \tag{29}$$

where g stands for the line segment with the endpoints $(0, 11/16, 11/16)^\mathsf{T}$ and $(1, 7/16, 9/16)^\mathsf{T}$. These endpoints are the centers of the inlet $\Gamma_{\text{in}} = \{0\} \times [5/8, 6/8] \times [5/8, 6/8]$ and the outlet $\{1\} \times (3/8, 4/8) \times (4/8, 5/8) = \Gamma_{\text{N}}$, where $\Gamma_{\text{D}} = \Gamma \setminus \Gamma_{\text{N}}$. Note the differences in the closeness of the above intervals that define the inlet and outlet.

We use the following boundary conditions: At Γ_{in}, we prescribe

$$u_{\mathsf{D}}(t) = \begin{cases} \sin(\pi t/2) & \text{if } t \in [0,1], \\ 1 & \text{if } t \in (1,2], \\ \sin(\pi(t-1)/2) & \text{if } t \in (2,3]. \end{cases} \tag{30}$$

At Γ_{N}, we prescribe $g_{\mathsf{N}} = 0$. At $\Gamma_{\mathsf{D}} \setminus \Gamma_{\mathsf{in}}$, we set $u_{\mathsf{D}} = 0$.

The initial condition is $u_0 = 0$, i.e., there are no species within the domain.

In the time interval $(0,1)$, the inflow increases, and the injected species is transported to the outlet. Then, in the time interval $(1,2)$, the inflow is constant, and the species reaches the outlet. Finally, in $(2,3)$, the influx decreases.

The simulations were performed for the refinement levels 5–7, where the initial grid consisted of a single cube. The time step length was $\Delta t = 5 \cdot 10^{-3}$. Finally, we set $\texttt{tol} = 10^{-14}$ in (28) and $\texttt{atol} = 10^{-16}$ for the PETSc solver.

Our choice of `tol` and `atol` means that the scheme MC+CN (unlike the other schemes) produces small undershoots, i.e., negative values of the magnitude larger than the machine precision. This can be prevented by decreasing these tolerances. However, we did not decrease them because it would lead to excessive long computing times of the nonlinear Zal+CN (as in Section 4.1.1). Instead, we replaced all negative components of the solution by zero in each time step.

We considered all of the solvers and preconditioners listed in Tables 1–3. Simulations were performed with 4, 8, 16 and 32 processors for the refinement levels 5–7. However, in the sequential case, we performed the simulations only for the refinement level 5. Note that we do not list results of the parallel solvers for BCGS + Jac below because the PETSc solver always crashed for this combination.

Discussion of Results

Tables 5–7 (pages 16–17), 8–10 (pages 17–18) and 11–13 (pages 19–20) show the dependence of the computing time on the number of processors for the nonlinear and linear Zal+CN and for MC+CN, respectively. We believe that the reduction in computation time by doubling the number of processors is acceptable.

We can clearly see that, generally, the best solver is FGMRES, and the worst solver is BCGS. The order of preconditioners is (from the best one): Jac, SOR, BJac, ASM and MG, where ASM performs much worse than the first three preconditioners, and MG performs much worse than ASM.

As expected, the computing times for the nonlinear Zal+CN are several times longer than those for MC+CN.

In the above discussion, we compared only the parallel algorithms because we expected them to perform better than their sequential counterparts for the considered sizes of the problems. The correctness of this conjecture is illustrated by Tables 14–16 (pages 20–21) in which we listed the computing times for sequential and parallel (4 processors) solvers on the coarse refinement level. The parallel versions are clearly better.

These tables also show the computing time for the direct solvers LU and UMFPACK. Although UMFPACK is clearly competitive in the sequential case, the computing time for parallel versions

Table 5: Example 4.1.2: Computing time in seconds of the solver with the nonlinear Zal+CN for refinement level 5.

		time (sec)			
Solver	**PC**	NP = 4	NP = 8	NP = 16	NP = 32
BCGS	BJac	2079	1222	760	512
	SOR	1986	1167	726	496
	ASM	2398	1452	987	680
	MG	2540	1476	1081	895
LGMRES	BJac	2079	1229	763	518
	SOR	1989	1178	726	499
	ASM	2378	1444	986	678
	Jac	1901	1130	698	485
	MG	2534	1512	1115	955
FGMRES	BJac	2003	1185	737	499
	SOR	1907	1126	706	476
	ASM	2290	1398	950	650
	Jac	1860	1107	684	471
	MG	2270	1402	1067	957

Table 6: Example 4.1.2: Computing time in seconds of the solver with the nonlinear Zal+CN for refinement level 6.

		time (sec)			
Solver	**PC**	NP = 4	NP = 8	NP = 16	NP = 32
BCGS	BJac	12488	6670	3772	2248
	SOR	12004	6314	3596	2124
	ASM	14144	7555	4463	2760
	MG	17302	8763	5187	3022
LGMRES	BJac	12459	6626	3787	2254
	SOR	11948	6389	3612	2128
	ASM	14077	7528	4448	2736
	Jac	11427	6115	3445	2046
	MG	17225	8786	5239	3081
FGMRES	BJac	11682	6232	3515	2090
	SOR	11068	5939	3354	2002
	ASM	13297	7162	4217	2561
	Jac	10846	5873	3280	1964
	MG	14636	7603	4470	2689

are by far worse than those for the iterative solvers. Therefore, we decided not to include them in our further comparisons. Also note the interesting fact that the computing time with UMFPACK actually shots up when running the solver in parallel.

As a measure of accuracy, the authors of [12] proposed to compare the time evolution of the solution at the center of the outlet, i.e., at the point $(1, 7/16, 9/16)^{\mathsf{T}}$. This is done in Figure 4.

The implicit schemes are represented only by the combination FGMRES + SOR; the other combinations produced very similar results. Contrary to the implicit schemes, the explicit

Table 7: Example 4.1.2: Computing time in seconds of the solver with the nonlinear Zal+CN for refinement level 7.

		time (sec)			
Solver	**PC**	NP = 4	NP = 8	NP = 16	NP = 32
BCGS	BJac	52053	29114	15469	8941
	SOR	51732	29042	15504	9209
	ASM	58398	32978	17694	10356
	MG	89549	54378	31340	16829
LGMRES	BJac	52176	29198	15598	9023
	SOR	52203	29426	15508	9293
	ASM	58760	32953	17689	10361
	Jac	50720	27788	14815	9035
	MG	91239	47312	26151	17150
FGMRES	BJac	47523	26532	14117	8141
	SOR	46234	25695	13666	8010
	ASM	54173	30610	16409	9519
	Jac	45626	24908	13356	7846
	MG	64826	36046	20632	12251

Table 8: Example 4.1.2: Computing time in seconds of the solver with the linear Zal+CN for refinement level 5.

		time (sec)			
Solver	**PC**	NP = 4	NP = 8	NP = 16	NP = 32
BCGS	BJac	82	46	30	21
	SOR	82	46	30	20
	ASM	87	48	33	21
	MG	87	48	33	22
LGMRES	BJac	82	46	31	20
	SOR	82	45	30	20
	ASM	86	48	33	21
	Jac	81	44	30	19
	MG	89	49	32	22
FGMRES	BJac	81	45	30	19
	SOR	82	45	30	20
	ASM	86	48	32	21
	Jac	81	45	30	21
	MG	85	48	32	21

schemes were used with the time step $\Delta t = 10^{-3}$ (because of the CFL condition). The curves representing the time evolutions probably converge to some curve, but it is not clear which one is better. Note that the schemes with the MC limiter produced essentially the same results.

Table 9: Example 4.1.2: Computing time in seconds of the solver with the linear Zal+CN for refinement level 6.

		time (sec)			
Solver	**PC**	NP = 4	NP = 8	NP = 16	NP = 32
BCGS	BJac	615	339	194	114
	SOR	616	331	194	115
	ASM	626	343	202	121
	MG	665	362	215	125
LGMRES	BJac	624	342	198	116
	SOR	617	336	194	115
	ASM	650	351	207	123
	Jac	615	328	189	112
	MG	701	370	214	128
FGMRES	BJac	598	326	192	114
	SOR	604	329	188	113
	ASM	633	338	198	120
	Jac	605	327	188	112
	MG	650	351	202	122

Table 10: Example 4.1.2: Computing time in seconds of the solver with the linear Zal+CN for refinement level 7.

		time (sec)			
Solver	**PC**	NP = 4	NP = 8	NP = 16	NP = 32
BCGS	BJac	4905	2612	1406	794
	SOR	4870	2600	1433	808
	ASM	4974	2696	1468	822
	MG	5747	3142	1797	1019
LGMRES	BJac	5045	2664	1440	813
	SOR	5097	2677	1467	846
	ASM	5198	2820	1554	878
	Jac	4827	2597	1381	803
	MG	5971	3224	1882	1020
FGMRES	BJac	4831	2606	1417	796
	SOR	4823	2566	1387	794
	ASM	4976	2696	1481	834
	Jac	4706	2565	1397	783
	MG	5465	2981	1650	935

4.2 Stationary problems

4.2.1 Example with non-constant convection

This example was proposed in [7]. We use it to compare the computing times of different solvers.

Table 11: Example 4.1.2: Computing time in seconds of the solver with MC+CN for refinement level 5.

		time (sec)			
Solver	**PC**	NP = 4	NP = 8	NP = 16	NP = 32
BCGS	BJac	184	102	66	42
	SOR	177	99	64	42
	ASM	196	110	75	49
	MG	201	113	73	48
LGMRES	BJac	184	102	66	43
	SOR	178	99	64	42
	ASM	195	110	76	50
	Jac	173	97	63	41
	MG	201	111	74	50
FGMRES	BJac	177	99	65	42
	SOR	175	98	64	41
	ASM	198	109	74	48
	Jac	173	96	63	41
	MG	212	107	70	46

Table 12: Example 4.1.2: Computing time in seconds of the solver with MC+CN for refinement level 6.

		time (sec)			
Solver	**PC**	NP = 4	NP = 8	NP = 16	NP = 32
BCGS	BJac	1349	736	416	247
	SOR	1329	717	407	240
	ASM	1429	777	453	276
	MG	1604	852	490	284
LGMRES	BJac	1364	731	420	249
	SOR	1332	721	411	243
	ASM	1448	790	460	276
	Jac	1310	732	403	239
	MG	1608	857	496	287
FGMRES	BJac	1320	706	404	239
	SOR	1281	689	396	234
	ASM	1396	762	444	267
	Jac	1271	694	393	234
	MG	1447	779	450	264

Description of Problem

We consider (24) with $\Omega = \Omega_1 \setminus \overline{\Omega}_2$, where $\Omega_1 = (0,5) \times (0,2) \times (0,2)$ and $\Omega_2 = (0.5, 0.8) \times (0.8, 1.2) \times (0.8, 1.2)$, $\boldsymbol{b} = (1, l(x), l(x))^T$ with $l(x) = (0.19x^3 - 1.42x^2 - 2.38x)/4$, $c = 0$, $f = 0$ and the convection-dominated case of $\varepsilon = 10^{-3}$. An illustration of the solution is given in Figure 5.

As for the boundary conditions, we set $\Gamma_{\mathsf{N}} := \{5\} \times (0,2) \times (0,2)$ and $\Gamma_{\mathsf{D}} = \partial\Omega \setminus \Gamma_{\mathsf{N}}$, and we

Table 13: Example 4.1.2: Computing time in seconds of the solver with MC+CN for refinement level 7.

		time (sec)			
Solver	**PC**	NP = 4	NP = 8	NP = 16	NP = 32
BCGS	BJac	10127	5449	2938	1677
	SOR	10174	5428	2934	1702
	ASM	10842	5774	3141	1778
	MG	13790	7209	3898	2360
LGMRES	BJac	10374	5566	2985	1702
	SOR	10446	5569	2996	1739
	ASM	11072	5926	3220	1845
	Jac	10311	5387	2924	1700
	MG	14058	7295	3954	2387
FGMRES	BJac	9924	5278	2837	1602
	SOR	9762	5490	2812	1603
	ASM	10554	5661	3069	1742
	Jac	9692	5144	2779	1593
	MG	11987	6267	3386	2024

Table 14: Example 4.1.2: Nonlinear Zal+CN: Computing time in seconds for NP $= 1, 4$ and refinement level 5.

		time (sec)				time (sec)	
Solver	**PC**	NP = 1	NP = 4	**Solver**	**PC**	NP = 1	NP = 4
BCGS	BJac	5740	2079	FGMRES	BJac	5288	2003
	SOR	5355	1986		SOR	4958	1907
	ASM	6271	2398		ASM	5821	2290
	Jac	5224	—		Jac	4958	1860
	MG	6618	2540		MG	6013	2270
LGMRES	BJac	5915	2079	LU	—	85567	48247
	SOR	5248	1989	UMFPACK	—	5073	48987
	ASM	6325	2378				
	Jac	5077	1901				
	MG	6599	2534				

prescribe $g_{\mathsf{N}} = 0$ and $u_{\mathsf{D}} = 0$ on $\partial\Omega_2$ and $u_{\mathsf{D}} = 1$ on the remainder of Γ_{D}.

The region Ω is covered by unstructured tetrahedral grids corresponding to the refinement levels 3, 4 and 5. The coarsest one (in Figure 5) was obtained by GMSH [9]. It consists of 226 tetrahedra. The cell diameters of our grids are in the ranges [0.0718, 0.2313], [0.0510, 0.1548] and [0.0180, 0.0578], respectively. Finally, we consider `atol` $= 10^{-14}$ and `tol` $= 10^{-10}$.

Simulations were performed for all solvers and preconditioners listed in Tables 1 and 3 for all NP listed in item 3 on page 10. We note that certain combinations of solvers and preconditioners did not work; in particular, the PETSc solver with the preconditioner MG always either crashed or did not converge. Similarly, the PETSc solver always crashed when combining BCGS with Jac or SOR, or when combining LGMRES with SOR.

Table 15: Example 4.1.2: Linear Zal+CN: Computing time in seconds for NP $= 1, 4$ and refinement level 5.

		time (sec)				time (sec)	
Solver	**PC**	NP $= 1$	NP $= 4$	**Solver**	**PC**	NP $= 1$	NP $= 4$
BCGS	BJac	252	82	FGMRES	BJac	249	81
	SOR	248	82		SOR	247	82
	ASM	253	87		ASM	254	86
	Jac	248	—		Jac	248	81
	MG	261	87		MG	255	85
LGMRES	BJac	252	82	LU	—	1009	513
	SOR	249	82	UMFPACK	—	248	516
	ASM	255	86				
	Jac	250	81				
	MG	261	89				

Table 16: Example 4.1.2: MC+CN: Computing time in seconds for NP $= 1, 4$ and refinement level 5.

		time (sec)				time (sec)	
Solver	**PC**	NP $= 1$	NP $= 4$	**Solver**	**PC**	NP $= 1$	NP $= 4$
BCGS	BJac	553	184	FGMRES	BJac	543	177
	SOR	543	177		SOR	532	175
	ASM	585	196		ASM	560	198
	Jac	538	—		Jac	534	173
	MG	593	201		MG	564	212
LGMRES	BJac	558	184	LU	—	3750	2066
	SOR	541	178	UMFPACK	—	513	2078
	ASM	571	195				
	Jac	538	173				
	MG	590	201				

Discussion of Results

Results regarding the comparison of computing times for different limiters, refinement levels and number of processors (4, 8, 16 and 32) are compared in Figures 6–8, pages 23–25.

First, we believe that the shortening of the computing time by doubling the number of processors is acceptable. Note that for the lowest refinement level, the problem was obviously too small for 32 processors, and the computing time even increased sometimes.

Regarding the solvers, we note that FGMRES is clearly the most efficient. Further, BCGS seems to be slightly better than LGMRES.

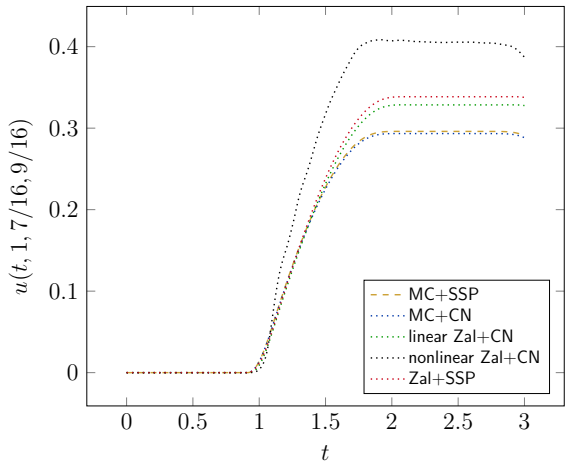
As for the preconditioners, the best one is clearly Jac, but only when used for very fine grids: when coarsening the grid, it becomes less and less efficient. The second best are BJac and SOR: Sometimes, BJac is better than SOR; sometimes SOR is the better one.

The worst preconditioners are clearly Jac (when used for coarse grids) and ASM.

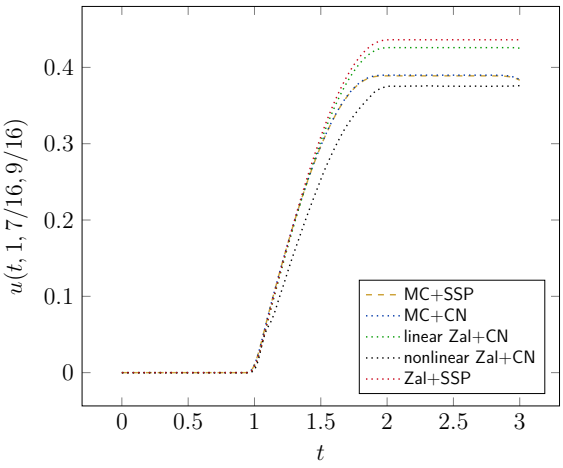
The computing times of the scheme with the LP limiter are about an order of magnitude larger

Figure 4: Example 4.1.2: Time evolution of u at $(1, 7/16, 9/16)^{\mathsf{T}}$. The MC limiter produces essentially the same result for both time discretizations.

(a) Refinement level 5.



(b) Refinement level 6.



(c) Refinement level 7.

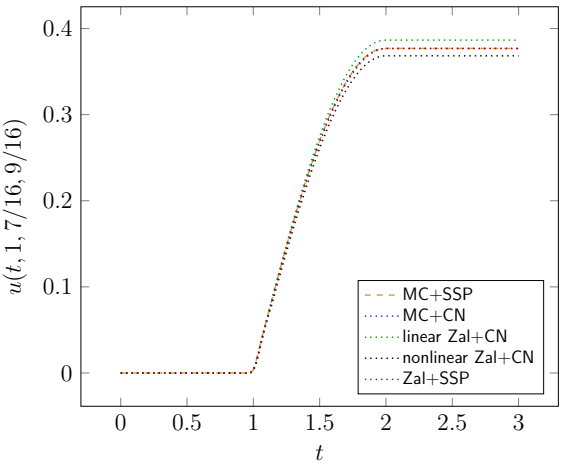
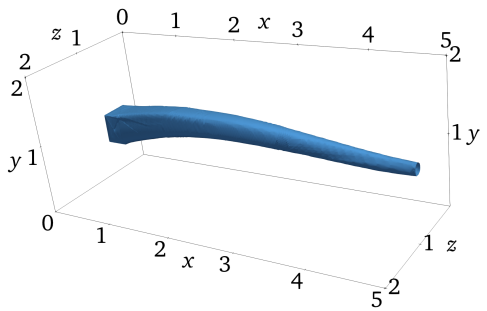


Figure 5: Example 4.2.1: Isosurface for $u = 0.05$ of the solution computed using the MU limiter for the level 5, and sketch of the coarsest grid (level 0).



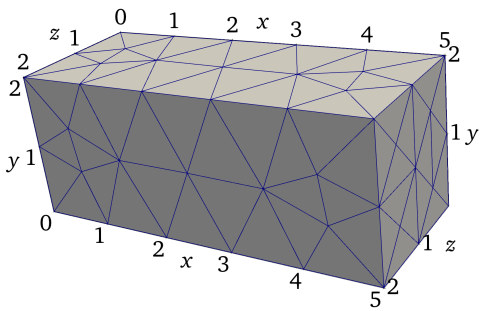
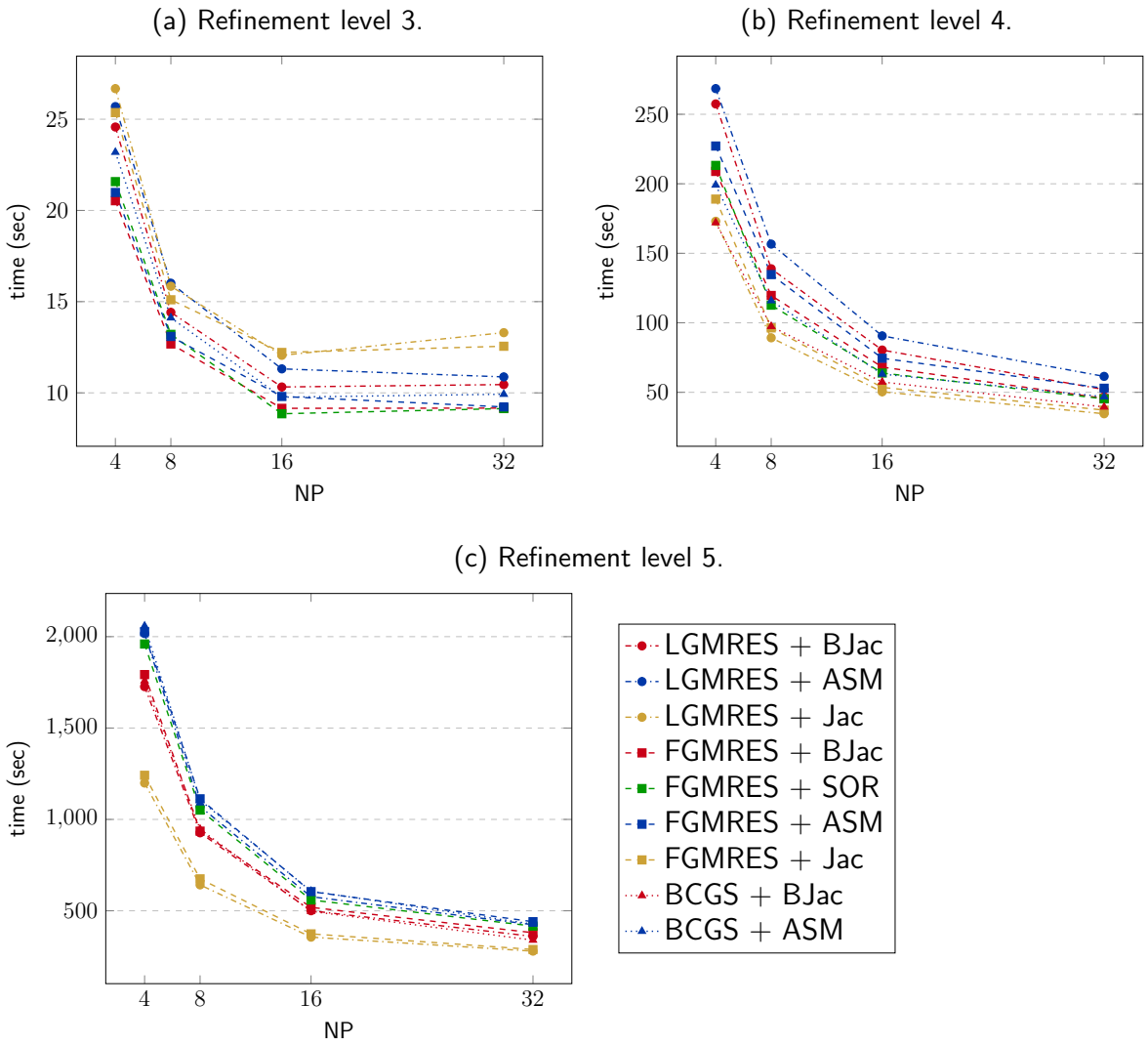


Figure 6: Example 4.2.1: Efficiency of different solvers and preconditioners with respect to the number of processors for the LP limiter.



than those of the other runtimes. The same is true for the number of iterations (not shown here), which is the reason for these long runtimes. This seems to be the price to pay for better accuracy. This point is discussed in details in the next section.

In the discussion above, we compared only the parallel algorithms because we assumed that they would perform better than their sequential counterparts on the problem sizes considered. The correctness of this assumption is illustrated by Tables 17–19, where we have listed the sequential computing times and the parallel computing times for 4 processors for our lowest refinement level. The parallel implementations are clearly better.

Finally, Figure 9 compares the solution values along the cut-line defined by $y = 1$ and $z = 1$. We can see that the solutions do not contain any overshoots or undershoots. They are also approximately the same for $x \leq 4.0$. However, they significantly differ for $x > 4.0$.

Figure 7: Example 4.2.1: Efficiency of different solvers and preconditioners with respect to the number of processors for the MU limiter.

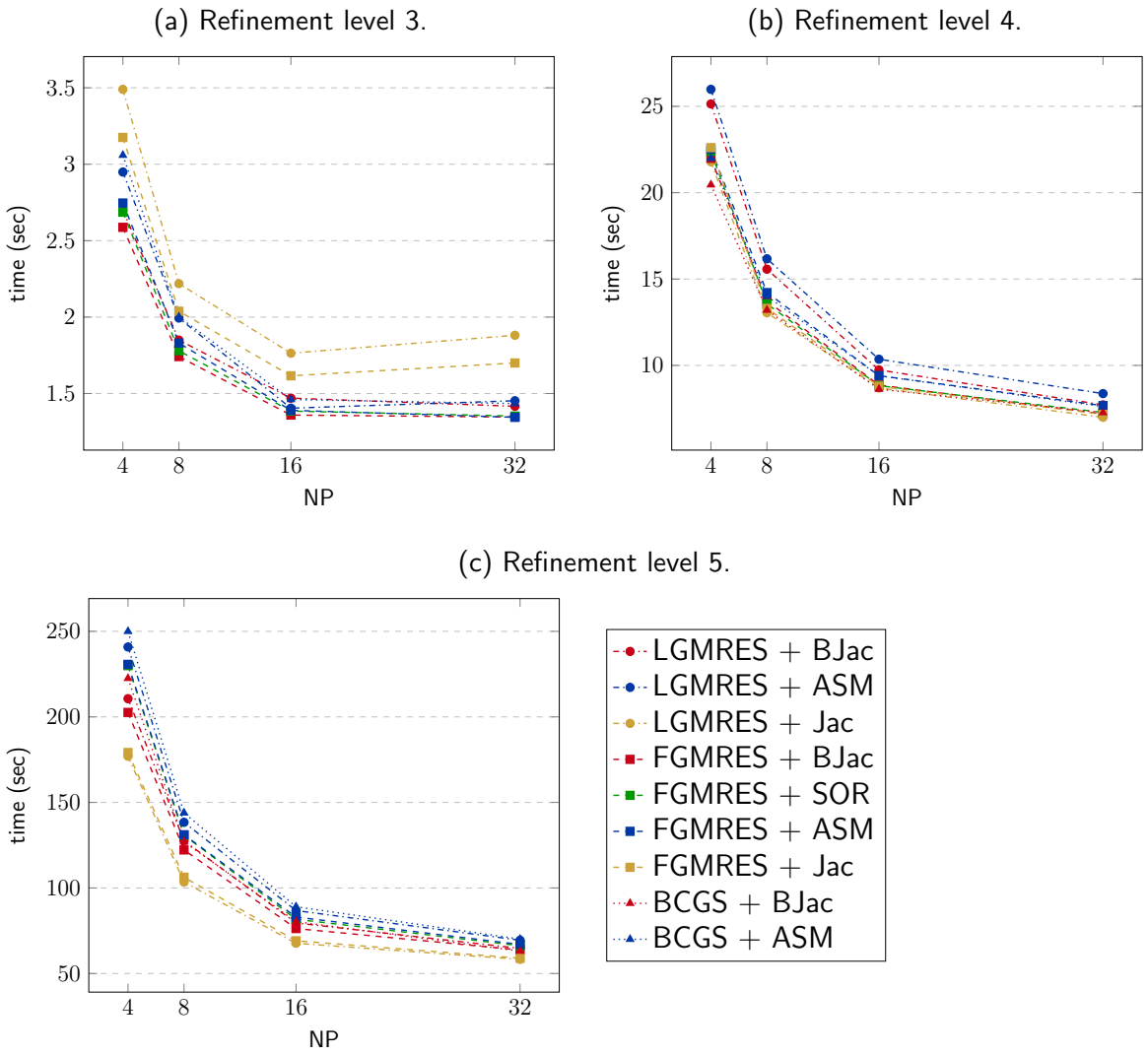


Table 17: Example 4.2.1: Computing times in seconds of the solver with the LP limiter for refinement level 3.

		time (sec)				time (sec)	
Solver	**PC**	NP = 1	NP = 4	**Solver**	**PC**	NP = 1	NP = 4
LGMRES	BJac	74.6	24.6	BCGS	BJac	63.1	21.7
	SOR	79.6	—		SOR	66.8	—
	ASM	79.8	25.7		ASM	68.4	23.2
	Jac	91.0	26.7				
FGMRES	BJac	60.8	20.5				
	SOR	62.9	21.6				
	ASM	65.3	21.0				
	Jac	84.9	25.4				

Figure 8: Example 4.2.1: Efficiency of different solvers and preconditioners with respect to the number of processors for the MC limiter.

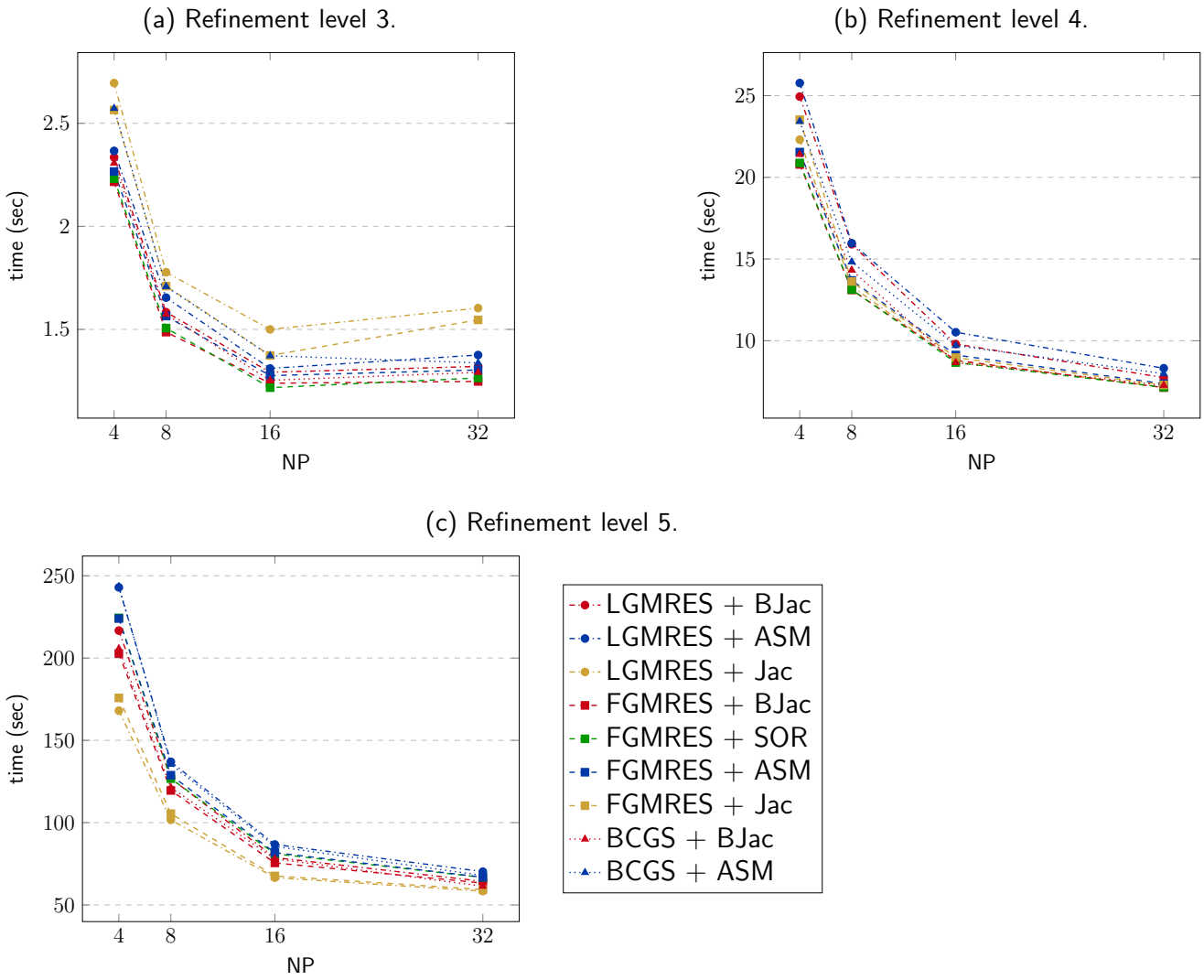


Table 18: Example 4.2.1: Computing times in seconds of the solver with the MU limiter for refinement level 3.

		time (sec)				time (sec)	
Solver	**PC**	NP = 1	NP = 4	**Solver**	**PC**	NP = 1	NP = 4
LGMRES	BJac	8.2	2.7	BCGS	BJac	7.9	2.9
	SOR	8.5	—		SOR	8.1	—
	ASM	8.7	2.9		ASM	8.3	3.1
	Jac	11.1	3.5				
FGMRES	BJac	7.6	2.6				
	SOR	7.8	2.7				
	ASM	8.0	2.7				
	Jac	10.0	3.2				

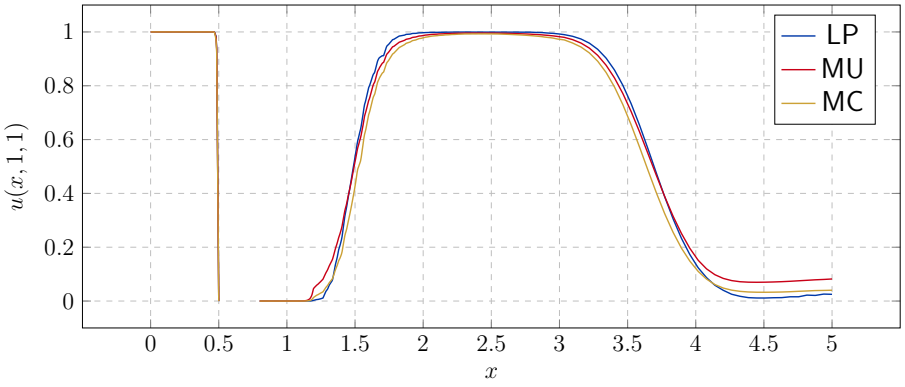
4.2.2 Circular convection

This example is an extension of the 2D example with the same name from [17]. We use it to compare the accuracy of the solutions computed with different limiters.

Table 19: Example 4.2.1: Computing times in seconds of the solver with the MC limiter for refinement level 3.

		time (sec)				time (sec)	
Solver	**PC**	NP = 1	NP = 4	**Solver**	**PC**	NP = 1	NP = 4
LGMRES	BJac	6.2	2.3	BCGS	BJac	5.8	2.3
	SOR	6.4	—		SOR	6.0	—
	ASM	6.6	2.4		ASM	6.2	2.6
	Jac	8.1	2.7				
FGMRES	BJac	5.7	2.2				
	SOR	5.8	2.2				
	ASM	6.0	2.3				
	Jac	7.5	2.6				

Figure 9: Example 4.2.1: $u(x,1,1)$ for $x \in [0,0.5] \cup [0.8,5]$ computed using different limiters.



Description of Problem

We consider equation (24) with $\Omega = (0,1)^3$, $\varepsilon = 0$, $\boldsymbol{b} = (y,-x,0)^{\mathsf{T}}$, $c = 0$, and $f = 0$.

As for the boundary conditions, Γ_D is the union of the faces with $y = 1$, $x = 0$ and $x = 1$, and $\Gamma_N = \Gamma \setminus \Gamma_D$, where $g_{\mathsf{N}} = 0$. The values of u_{D} correspond to the exact solution

$$u(x,y,z) = \begin{cases} 1 & \text{if } \ 0.15 \le r(x,y) \le 0.45, \\ \cos^2\left(10\pi \frac{r(x,y)-0.7}{3}\right) & \text{if } \ 0.55 \le r(x,y) \le 0.85, \\ 0 & \text{otherwise,} \end{cases} \tag{31}$$

where $r(x,y) = \sqrt{x^2+y^2}$. An approximation of this solution is in Figure 10.

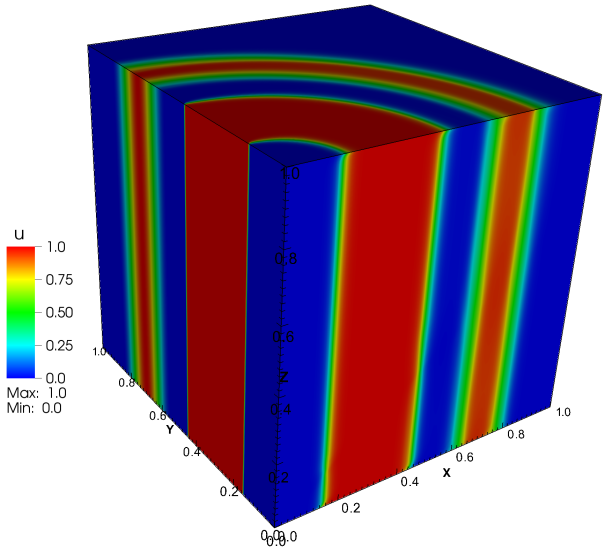
The domain Ω is covered by tetrahedral grids corresponding to the refinement levels 5–7, with the initial grid consisting of 6 tetrahedra. The maximum cell diameters of these grids are approximately $0.0765, 0.0383$, and 0.0191, respectively.

Since we wanted to obtain the best approximation of the solution for a given combination of limiter and refinement level, our stopping criterion for the nonlinear solver was

$$|R_{\text{new}} - R_{\text{old}}|/R_{\text{new}} < \texttt{tol}_2 \ \text{ and } \ u(\boldsymbol{x}) \ge -10^{-16} \ \forall \boldsymbol{x} \in \overline{\Omega}, \tag{32}$$

where R_{new} and R_{old} stand for the Euclidean norms of the new and old residues, respectively. That is, the solver should stop when the solution stops improving. We set $\texttt{tol}_2 = 10^{-6}$ and $\texttt{atol} = 10^{-14}$.

Figure 10: Example 4.2.2: Solution for the refinement level 7 computed by the scheme with the MC limiter.



Discussion of Results

Table 20, page 27 shows the resulting errors of the numerical solution, measured in $\|\cdot\|_1$ and $\|\cdot\|_2$, which denote the standard norms in $L^1(\Omega)$ and $L^2(\Omega)$, respectively. Note that we do not specify which combination of linear systems solver and preconditioner we use. The reason for this is that we tested all the solvers and preconditioners listed in Tables 1 and 3, and, as expected, they all gave approximately the same results (with the exception of some combinations, for which the PETSc solver always crashed). In addition, we only ran our solvers in parallel.

The errors clearly indicate convergence. However, the L^1-error decreases much faster than the L^2-error. For each refinement level, the results obtained with the LP limiter are clearly the best. The results obtained with the MC and MU limiters are similar.

Due to (32), none of the schemes produced undershoots or overshoots that were larger than the machine precision.

Table 20: Example 4.2.2: Errors measured in $\|\cdot\|_1$ and $\|\cdot\|_2$.

(a) Error in $\|\cdot\|_1$.

Level	MC	LP	MU
5	8.11e-02	4.14e-02	9.24e-02
6	3.39e-02	1.86e-02	3.62e-02
7	1.62e-02	8.31e-03	1.56e-02

(b) Error in $\|\cdot\|_2$.

Level	MC	LP	MU
5	1.46e-01	9.76e-02	1.61e-01
6	8.15e-02	6.64e-02	8.52e-02
7	5.81e-02	4.59e-02	5.60e-02

5 Conclusions

The numerical studies of AFC schemes in this work indicate that the costs of the following procedures must be taken into account when designing high-performance algorithms: (i) calculation of the correction factors for limited antidiffusive fluxes, (ii) matrix/residual assembly, and (iii) iterative solution.

The need for high accuracy and efficiency becomes particularly pronounced in applications of AFC to 3D problems. As a rule, schemes equipped with more diffusive limiters converge faster than more accurate approaches. Thus a fair comparison of different algorithms should be based on CPU times that are required to attain a certain level of accuracy.

An interesting observation is that implicit schemes can outperform their explicit counterparts even in numerical simulations of transient processes using small time steps. We hope that our comparison of different limiters and linear algebra tools provides useful insights for finding combinations of methods that offer the best overall performance in terms of accuracy and efficiency.

References

[1] N. Ahmed and V. John. Adaptive time step control for higher order variational time discretizations applied to convection-diffusion-reaction equations. *Computer Methods in Applied Mechanics and Engineering*, 285:83–101, 2015.

[2] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, A. Dener, V. Eijkhout, W. D. Gropp, D. Karpeyev, D. Kaushik, M. G. Knepley, D. A. May, L. C. McInnes, R. T. Mills, T. Munson, K. Rupp, P. Sanan, B. F. Smith, S. Zampini, H. Zhang, and H. Zhang. PETSc Web page. `https://www.mcs.anl.gov/petsc`, 2019.

[3] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, A. Dener, V. Eijkhout, W. D. Gropp, D. Karpeyev, D. Kaushik, M. G. Knepley, D. A. May, L. C. McInnes, R. T. Mills, T. Munson, K. Rupp, P. Sanan, B. F. Smith, S. Zampini, H. Zhang, and H. Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 3.14, Argonne National Laboratory, 2020.

[4] S. Balay, W. D. Gropp, L. C. M., and B. F. Smith. Efficient management of parallelism in object oriented numerical software libraries. In E. Arge, A. M. Bruaset, and H. P. Langtangen, editors, *Modern Software Tools in Scientific Computing*, pages 163–202. Birkhäuser Press, 1997.

[5] G. R. Barrenechea, V. John, and P. Knobloch. Analysis of algebraic flux correction schemes. *SIAM J. Numer. Anal.*, 54(4):2427–2451, 2016.

[6] G. R. Barrenechea, V. John, and P. Knobloch. An algebraic flux correction scheme satisfying the discrete maximum principle and linearity preservation on general meshes. *Math. Models Methods Appl. Sci.*, 27(3):525–548, 2017.

[7] G. R. Barrenechea, V. John, P. Knobloch, and R. Rankin. A unified analysis of algebraic flux correction schemes for convection-diffusion equations. *SeMA J.*, 75(4):655–685, 2018.

[8] R. H. Castain, T. S. Woodall, D. J. Daniel, J. M. Squyres, B. Barrett, and G. E. Fagg. The Open Run-Time Environment (OpenRTE): A transparent multi-cluster environment for high-performance computing. In *Proceedings, 12th European PVM/MPI Users' Group Meeting*, Sorrento, Italy, September 2005.

[9] C. Geuzaine and J. F. Remacle. Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities. *Internat. J. Numer. Methods Engrg.*, 79(11):1309–1331, 2009.

[10] A. Jha and V. John. A study of solvers for nonlinear AFC discretizations of convection-diffusion equations. *Comput. Math. Appl.*, 78(9):3117–3138, 2019.

[11] A. Jha and V. John. On basic iteration schemes for nonlinear AFC discretizations. In G. R. Barrenechea and J. Mackenzie, editors, *Boundary and Interior Layers, Computational and Asymptotic Methods BAIL 2018*, pages 113–128, Cham, 2020. Springer International Publishing.

[12] V. John and J. Novo. On (essentially) non-oscillatory discretizations of evolutionary convection-diffusion equations. *J. Comput. Phys.*, 231(4):1570–1586, 2012.

[13] V. John and E. Schmeyer. Finite element methods for time-dependent convection-diffusion-reaction equations with small diffusion. *Comput. Methods Appl. Mech. Engrg.*, 198(3-4):475–494, 2008.

[14] D. Kuzmin. Algebraic flux correction for finite element discretizations of coupled systems. *Computational Methods for Coupled Problems in Science and Engineering II*, 01 2007.

[15] D. Kuzmin. Explicit and implicit FEM-FCT algorithms with flux linearization. *J. Comput. Phys.*, 228(7):2517–2534, 2009.

[16] D. Kuzmin. Linearity-preserving flux correction and convergence acceleration for constrained Galerkin schemes. *J. Comput. Appl. Math.*, 236(9):2317–2337, 2012.

[17] D. Kuzmin. Monolithic convex limiting for continuous finite element discretizations of hyperbolic conservation laws. *Computer Methods in Applied Mechanics and Engineering*, 361:112804, 2020.

[18] D. Kuzmin and M. Möller. Algebraic flux correction. I. Scalar conservation laws. In *Flux-corrected transport*, Sci. Comput., pages 155–206. Springer, Berlin, 2005.

[19] D. Kuzmin and S. Turek. Flux correction tools for finite elements. *Journal of Computational Physics*, 175(2):525–558, 2002.

[20] R. J. Leveque. High-resolution conservative algorithms for advection in incompressible flow. *SIAM Journal on Numerical Analysis*, 33(2):627–665, 1996.

[21] C. Lohmann. *Physics-compatible finite element methods for scalar and tensorial advection problems*. Springer, 2019.

[22] R. Löhner, K. Morgan, J. Peraire, and M. Vahdati. Finite element flux-corrected transport (FEM–FCT) for the Euler and Navier–Stokes equations. *International Journal for Numerical Methods in Fluids*, 7(10):1093–1109, 1987.

[23] U. Wilbrandt, C. Bartsch, N. Ahmed, N. Alia, F. Anker, L. Blank, A. Caiazzo, S. Ganesan, S. Giere, G. Matthies, R. Meesala, A. Shamim, J. Venkatesan, and V. John. Parmoon - a modernized program package based on mapped finite elements. *Computers and Mathematics with Applications*, 74:74–88, 2016.

[24] S. T. Zalesak. Fully multidimensional flux-corrected transport algorithms for fluids. *J. Comput. Phys.*, 31(3):335–362, 1979.