Journal of Computational Physics 474 (2023) 111775

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

Sign-preserving second-order IMPEC time discretization and its application in compressible miscible displacement with Darcy-Forchheimer models ☆

Wenjing Feng [a], Hui Guo [a,*], Lulu Tian [a], Yang Yang [b,*]

[a] College of Science, China University of Petroleum, Qingdao 266580, China
[b] Department of Mathematical Sciences, Michigan Technological University, Houghton, MI 49931, USA

ARTICLE INFO

Article history:
Received 19 June 2022
Received in revised form 16 October 2022
Accepted 10 November 2022
Available online 17 November 2022

Keywords:
Second-order IMPEC method
Interior penalty discontinuous Galerkin method
Bound-preserving
Compressible miscible displacement
Darcy-Forchheimer model

ABSTRACT

In this paper, we construct sign-preserving second-order IMplicit Pressure Explicit Concentration (IMPEC) time methods for generalized coupled non-Darcy flow and transport problems in petroleum engineering, extending the algorithm given in [10] which is only applicable to Darcy flows. We use interior penalty discontinuous Galerkin (IPDG) methods for spatial discretization, and develop bound-preserving technique to obtain physically relevant numerical approximations. The sign-preserving second-order IMPEC method is an important innovation. The method is based on the framework of the second-order strong-stability-preserving Runge-Kutta (SSP-RK2) method. The basic idea is to treat the pressure equation implicitly and the concentration equation explicitly so as to obtain a first-order time integration. Then we introduce a correction stage to compensate the accuracy, maintaining the physical bounds of the numerical cell averages. Unfortunately, the above algorithm is not applicable to non-Darcy problems. There are two main difficulties. Firstly, since the velocity equation is nonlinear and time-independent, all variables in the equation must be calculated at the same time level. However, the treatment in [10] will yield an extremely complicated algorithm and significantly large computational cost, and some iterations whose convergence may not be available if the solutions are not smooth. In our scheme, we linearize the velocity equation and the numerical solutions are not at the same time level, leading to first-order accurate solutions. Therefore, we adopt a completely different approach from [10] to derive the correction stage, keeping the physical bounds of the numerical solutions. Secondly, though with the new correction stage, it is still not easy to solve velocity equations in some stages. In this paper, we construct a direct solver for the velocity equation to save computational cost. Numerical experiments will be given to demonstrate that the improved sign-preserving second-order IMPEC scheme can reduce the computational cost significantly compared with explicit schemes if the diffusion coefficient **D** is small in the concentration equation. The proposed method also yields much larger CFL number compared with first-order IMPEC schemes. Moreover, the effectiveness of the bound-preserving technique will also be verified.

☆ The first author was supported by Graduate Innovation Engineering Project YCX2021136. The second author was supported by the National key Nature Science Foundation of China (12131014) and the Natural Science Foundation of Shandong Province (ZR2021MA001). The third author was supported by National Natural Science Foundation of China (11801569). The last author was supported by NSF grant DMS-1818467 and Simons Foundation 961585.

* Corresponding authors.
E-mail addresses: fengwenjing27@163.com (W. Feng), sdugh@163.com (H. Guo), tianll@upc.edu.cn (L. Tian), yyang7@mtu.edu (Y. Yang).

https://doi.org/10.1016/j.jcp.2022.111775
0021-9991/

1. Introduction

In this paper, we consider the fluid mixture with two components, and study the compressible miscible displacements with general non-Darcy models

$$d(c)\frac{\partial p}{\partial t} + \nabla \cdot \mathbf{u} = q, \quad (x, y) \in \Omega,\ 0 < t \le T, \tag{1.1}$$

$$F(\mathbf{u}, c) = -\nabla p, \quad (x, y) \in \Omega,\ 0 < t \le T, \tag{1.2}$$

$$\phi\frac{\partial c}{\partial t} + b(c)\frac{\partial p}{\partial t} + \mathbf{u} \cdot \nabla c - \nabla \cdot (\mathbf{D}(\mathbf{u})\nabla c) = (\tilde{c} - c)q, \quad (x, y) \in \Omega,\ 0 < t \le T, \tag{1.3}$$

as well as its one-dimensional version, where Ω is a bounded domain and $[0, T]$ is the time interval. The unknown variables p, $\mathbf{u}$ and c represent the pressure of the fluid mixture, the velocity of the mixture and the volumetric concentration of interested species, respectively. $\phi(x, y)$ is the porosity of the porous media. (1.2) gives the relationship between the velocity $\mathbf{u}$ and the gradient of the pressure p. There are several models available and two are commonly used. For Darcy model [1,26,40], (1.2) can be written as

$$\mathbf{u} = -\frac{\kappa}{\mu(c)}\nabla p, \tag{1.4}$$

while for Darcy-Forchheimer model [11], it has the form

$$\frac{\mu(c)}{\kappa}\mathbf{u} + \beta\rho(c)|\mathbf{u}|\mathbf{u} = -\nabla p, \tag{1.5}$$

where $\kappa(x, y)$ and $\beta(x, y)$ are the permeability and dynamic viscosity, respectively. $\mu(c)$ is concentration-dependent viscosity, and a commonly used relationship is the quarter-power rule [28]

$$\mu(c) = \mu_1\left[\left(\frac{\mu_1}{\mu_2}\right)^{\frac{1}{4}} c + 1 - c\right]^{-4}. \tag{1.6}$$

Moreover, we assume $\frac{\mu(c)}{\kappa} \ge 0$. $\rho(c)$ is the density of fluid, which is given by the volume-average of the densities of the two components [28]

$$\rho(c) = \rho_1 c + \rho_2(1 - c). \tag{1.7}$$

Here μ_1, μ_2, ρ_1 and ρ_2 are all positive constants. The Darcy-Forchheimer model, whose theoretical derivation was given in [33], is more accurate than Darcy model if the velocity is high and the porosity is nonuniform. In this model, $\beta \ge 0$ is referred to as the Forchheimer number [27]. If $\beta = 0$, Darcy-Forchheimer model degenerates to Darcy model. q is the external flow rate, and $\tilde{c}$ is the concentration in the external flow, which must be specified at injection wells ($q > 0$), and is assumed to be equal to c at production wells ($q < 0$). The diffusion coefficient $\mathbf{D}$ is a 2×2 symmetric matrix, and it has the form

$$\mathbf{D}(\mathbf{u}) = \phi(x, y)(d_{mol}\mathbf{I} + d_{long}|\mathbf{u}|\mathbf{E} + d_{tran}|\mathbf{u}|\mathbf{E}^{\perp}), \tag{1.8}$$

where $\mathbf{E}(\mathbf{u}) = \frac{\mathbf{u}\mathbf{u}^T}{|\mathbf{u}|^2}$ is a 2×2 tensor representing the orthogonal projection along the velocity vector, and $\mathbf{E}^{\perp} = \mathbf{I} - \mathbf{E}$ is the orthogonal complement. d_{mol} is the molecular diffusion coefficient. d_{long} and d_{tran} show the longitudinal and transverse dispersion coefficients, respectively. In this paper, we assume $\mathbf{D}$ to be positive semidefinite. Therefore, we have $D_{11} \ge 0$, $D_{22} \ge 0$ and $D_{12} = D_{21}$. Other coefficients can be stated as follows:

$$c = c_1 = 1 - c_2, \quad d(c) = \phi\sum_{j=1}^{2} z_j c_j, \quad b(c) = \phi c_1\left\{z_1 - \sum_{j=1}^{2} z_j c_j\right\},$$

where c_j and z_j are the concentration and the compressibility factor of the jth component of the fluid mixture, respectively. In this paper, we consider homogeneous Neumann boundary conditions

$$\mathbf{u} \cdot \mathbf{n} = 0, \quad (\mathbf{D}(\mathbf{u})\nabla c - c\mathbf{u}) \cdot \mathbf{n} = 0,$$

where $\mathbf{n}$ is the unit outer normal of the boundary $\partial\Omega$. Moreover, the initial solutions are given as

$$c(x, y, 0) = c_0(x, y), \quad p(x, y, 0) = p_0(x, y), \quad (x, y) \in \Omega.$$

There are several works discussing numerical methods for miscible displacements and the Darcy-Forchheimer models, such as mixed element method [13,27,28], finite difference method [23,30–32], discontinuous Galerkin method (DG) [16,19,

38]. In many actual problems of miscible displacements, physical parameters are closely related to the concentration c. Some of the parameters are given by lab experiments and curve fits. If c is placed out of the interval $[0, 1]$, we might not obtain the parameters used in the system, and the numerical approximations will blow up, especially for problems with large gradients. In [18], Guo and Yang first proposed the bound-preserving DG methods for the two-component compressible miscible displacements. The basic idea is as follows. (1) Enforce $c_1 + c_2 = 1$ by choosing consistent numerical fluxes (see Definition 3.1) in the schemes of the pressure and concentration equations. (2) Subtract the concentration scheme from the pressure one to obtain the scheme of the second component concentration. (3) Apply the positivity-preserving techniques that proposed in [45,46] to both c_1 and c_2, respectively. The authors [18] theoretically proved the bound-preserving property of numerical cell averages. Then a slope limiter was applied to correct the numerical approximation to be within the interval $[0, 1]$. Later, Chuenjarern et al. [9] proposed high-order bound-preserving DG methods on triangular meshes for multi-component miscible displacements, Feng et al. also extended the idea to simulate flows in fractured porous media [15]. The bound-preserving finite difference methods were also discussed in [17].

Unfortunately, most of the time discretizations in the above works are based on explicit methods, such as strong-stability-preserving Runge-Kutta (SSP-RK) time methods [35,36,14], resulting in small time step sizes and large computational cost. Therefore, the above works can hardly be applied in practice. The main issue is due to the heterogeneity of the media, see e.g. [42,43,12]. In fact, in some part of the media, the permeability would be extremely high, leading to large diffusion coefficients in the pressure equation. Therefore, a straightforward alternative is to consider implicit forms of the pressure equation. Although the fully implicit scheme [24,48] has unconditional stability, such methods require a lot of computational resources in each time step since the system is fully coupled. Compared with the fully explicit methods and fully implicit methods, the IMplicit Pressure Explicit Concentration (IMPEC) [25,20,7,5,47] scheme has been a popular choice to simulate compressible flows in porous media. The basic idea is to treat the pressure equation implicitly, and update the concentration equation explicitly. This method can decouple the system and solves the equations in the system sequentially with mild time step size restrictions from the concentration equation. Hence it is straightforward to set up, efficient to implement, and occupies less computer memory per time step. Sheldon et al. [34] and Stone et al. [37] first proposed the IMPEC method. Later, it was widely applied to solve multi-phase flow problems, where it is called the IMplicit Pressure Explicit Saturation (IMPES) method [6,21,22,44]. In [3,4], the authors developed a fully mass-conservative IMPES scheme, and the saturation of each phase was proved to be bound-preserving if the time step size is smaller than a certain value. The fully mass-conservative iterative IMPEC method [2] was also proposed for multi-component compressible flow in porous media. However, above IMPEC methods have only first-order time accuracy. It is difficult to construct second-order IMPEC time method that is compatible with the bound-preserving techniques.

In [10], we first constructed a second-order IMPEC (SIPEC) time method for Darcy compressible miscible displacements. The basic idea is divided into three steps. (1) Based on the SSP-RK2 method, we treat the pressure equation implicitly and the concentration equation explicitly so as to obtain a first-order time scheme. (2) Derive the local truncation error of the above first-order scheme. (3) Following [8], a correction stage is introduced to compensate for the second-order accuracy of the above scheme, and maintain the bound-preserving property of the numerical cell averages in the meantime. However, the SIPEC method in [10] is only applicable to linear equations, i.e. $\beta = 0$.

In this paper, we extend the idea to non-Darcy models, especially Darcy-Forchheimer models, which is widely used in petroleum engineering. We will develop an improved sign-preserving SIPEC time method and use interior penalty discontinuous Galerkin (IPDG) method [29] for spatial discretization, and then apply bound-preserving techniques to obtain physically relevant numerical approximations. The improved sign-preserving SIPEC time method for non-Darcy flows is an important innovation. We emphasize that there are significant differences from [10] in the construction of the SIPEC scheme. In [10], the velocity equation is linear, so we can substitute the velocity equation into the pressure equation, and we only need to consider the time level of pressure. However, in this paper, since (1.2) is a nonlinear time-independent equation, we have to deal with the time level of velocity carefully to achieve second-order accuracy, which requires that all variables in the velocity equation must be calculated at the same time level. If we follow [10], there are two possible ways to achieve second-order time accuracy for solving the non-Darcy models. The first idea is to couple pressure equation and velocity equation to be solved iteratively, yet we cannot guarantee the convergence of the iteration. Moreover, the iterative process will lead to large CPU time, and the proposed SIPEC method may not save CPU time compared with the fully explicit methods. The second idea is to linearize the velocity equation, avoiding iterations. However, such treatment results in the degeneration of accuracy, so additional correction stages are required at each stage in the Runge-Kutta method, and the algorithm is quite complicated and CPU time is still large. Though the second-order accuracy of the above two ideas can be maintained, the proposed schemes are not applicable in practice and major revisions of the well-developed programmers based the IMPEC methods are necessary. Therefore, we adopt a completely different idea from [10] to construct a new sign-preserving SIPEC time scheme so as to reduce computational cost and only a minor revision is needed in the traditional IMPEC method to achieve second-order accuracy.

There are two main difficulties in constructing the proposed schemes. Firstly, since the velocity equation is nonlinear and time-independent, variables p, $\mathbf{u}$ and c must be treated at the same time level, otherwise the accuracy will be degenerated to 1. In our scheme, to avoid the iterations whose convergence cannot be guaranteed theoretically, we use linearization, where nonlinear terms are processed in different time levels, and a first-order one-step error in velocity is generated. Therefore, the proof of the new correction stage turns out to be more complicated. Secondly, it is not easy to solve the nonlinear equations. According to the characteristics of equation (1.5), we construct a direct solver, whose computational

cost is negligible compared with the IMPEC procedure, instead of iterated methods. The improved sign-preserving SIPEC time method is more widely applicable, not only to linear equations, but also to general nonlinear and time-independent equations. The implementation of the improved sign-preserving SIPEC method is also straightforward. The whole algorithm is simply to apply the traditional IMPEC procedure four times, with a direct solver of the nonlinear equation after the first procedure and a linear combination in the end of each time step. Numerical experiments demonstrate that the time step size used in the improved SIPEC method can be much larger than that in the IMPEC method, especially for problems with discontinuities or large gradients. This is because we use the second-order IPDG method coupled with the first-order forward Euler time scheme for the concentration equation in the IMPEC method, leading to instability if $\Delta t \sim \Delta x$ and such instability can easily be triggered for problems with singularities. Therefore, the improved SIPEC method costs compatible and even less CPU time than the IMPEC method. Moreover, the bound-preserving technique for the improved SIPEC time integration is different from that for the fully explicit methods in [18,9], and we need to do some special treatment. In the concentration equation, we treat the velocity and pressure in the convection and source terms implicitly while those in the diffusion term explicitly so as to avoid the correction of the diffusion term, otherwise it will cause anti-diffusion and the bound-preserving techniques fail to work. In the correction stage, the pressure scheme has similar form to the concentration scheme to ensure that the bound-preserving technique is applicable. In numerical experiments, we compare the improved sign-preserving SIPEC time method with the SSP-RK2 method and the first-order IMPEC method. The results show that the CPU time by the SIPEC method is significantly less than that by the SSP-RK2 method. Moreover, we can observe strong oscillations by using the first-order IMPEC method if the time step is large, while the oscillations disappear for the SIPEC method. For stability issue, we only consider second-order spatial discretization, since the time integration is developed from the SSP-RK2 method. The case with high-order spatial discretizations will be considered in the future. Finally, we point out that if the diffusion coefficient $\mathbf{D}$ is small, the sign-preserving second-order IMPEC method presents a significant advantage compared with the SSP-RK2 scheme in computational cost. However, if $\mathbf{D}$ is large, the advantage may not be significant.

The rest of the paper is organized as follows. In Section 2, we propose an improved sign-preserving SIPEC time method for general non-Darcy flows. In Section 3, we give some notations, construct fully-discrete IPDG schemes for Darcy-Forchheimer compressible miscible displacements, and consider a direct method for solving nonlinear equations. The bound-preserving techniques will be discussed in Section 4. In Section 5, some numerical experiments are presented. We will end in Section 6 with concluding remarks.

2. The sign-preserving SIPEC time integration

In this section, we will construct a sign-preserving SIPEC time integration for compressible miscible displacements with non-Darcy models. The method is derived from the SSP-RK2 method with pressure equation solving implicitly followed by a correction stage. More generally, we consider the following ordinary differential equations:

$$p_t = f_1(u, r, q), \tag{2.1}$$

$$u = f_2(p, r, u), \tag{2.2}$$

$$r_t = g_1(u, r, b) + g_2(u, r), \tag{2.3}$$

where the dependent variables are p, u and r, corresponding to pressure, velocity and concentration, respectively. It should be noted that if spatial variables are taken into account, $r(x, y, t) = \phi(x, y)c(x, y, t)$, so we use the notation r instead of c here, mainly in order to apply the bound-preserving technique in Section 4. q and b are given variables in the source terms. $g_1(u, r, b)$ stands for the convection and the source terms in the concentration equation while $g_2(u, r)$ represents the diffusion term. If (2.2) is a linear equation, then we can solve for u and substitute which into (2.1). Then the SIPEC method introduced in [10] can be applied. Unfortunately, if (2.2) is not linear in u, the idea given in [10] may not lead to less computational cost.

Let $\{t^n = n\Delta t\}_{n=0}^{M}$ be a uniform partition of the time interval $[0, T]$ with time step Δt such that $M\Delta t = T$. However, such an assumption of uniform partition is not essential. We use o^n and $o(t^n)$ ($o = p,\ u$ or r) as the numerical and exact solutions for (2.1)-(2.3) at t^n, respectively. For $n = 0$, p^0 and r^0 are L^2-projections of $p(t^0)$ and $r(t^0)$, respectively, and u^0 can be obtained by solving

$$u^0 = f_2(p^0, r^0, u^0), \tag{2.4}$$

where some iteration methods may be necessary in general. We will discuss an easier way to find u^0 in the next section without iteration for the Darcy-Forchheimer model.

Given the numerical solutions p^n, u^n and r^n with $n \geq 0$, we discuss how to find p^{n+1}, u^{n+1} and r^{n+1}. The traditional sign-preserving IMPEC method, namely IMPEC-SP method, gives

$$p^{n+1} = p^n + \Delta t f_1(u^{n+1}, r^n, q^n), \quad u^{n+1} = f_2(p^{n+1}, r^n, u^n), \tag{2.5}$$

$$r^{n+1} = r^n + \Delta t \left(g_1(u^{n+1}, r^n, b^n) + g_2(u^n, r^n)\right). \tag{2.6}$$

Unfortunately, the above scheme is only first-order accurate. We will construct second-order ones. For simplicity, in the rest of this section, the error we will discuss is one step error and the accuracy is the global one unless otherwise stated. For example, the forward Euler time integration has error $O(\Delta t^2)$ in one step and the accuracy is 1. Following [10] with some minor changes, we can get a SIPEC scheme, namely SIPEC-C method, whose detailed proof is omitted:

$$p^{(1)} = p^n + \Delta t f_1(u^{(1)}, r^n, q^n), \quad u^{(1)} = f_2(p^{(1)}, r^n, u^n), \tag{2.7}$$
$$r^{(1)} = r^n + \Delta t \left(g_1(u^{(1)}, r^n, b^n) + g_2(u^n, r^n)\right), \tag{2.8}$$
$$\breve{u}^{(1)} = f_2(p^{(1)}, r^{(1)}, \breve{u}^{(1)}), \tag{2.9}$$
$$\breve{p}^{(1)} = p^n + \Delta t f_1(\breve{\breve{u}}^{(1)}, r^n, q^n), \quad \breve{\breve{u}}^{(1)} = f_2(\breve{p}^{(1)}, r^{(1)}, \breve{u}^{(1)}), \tag{2.10}$$
$$\breve{r}^{(1)} = r^n + \Delta t \left(g_1(\breve{\breve{u}}^{(1)}, r^n, b^n) + g_2(u^n, r^n)\right), \tag{2.11}$$
$$p^{(2)} = p^{(1)} + \Delta t f_1(u^{(2)}, r^{(1)}, q^{n+1}), \quad u^{(2)} = f_2(p^{(2)}, r^{(1)}, u^{(1)}), \tag{2.12}$$
$$r^{(2)} = r^{(1)} + \Delta t \left(g_1(u^{(2)}, r^{(1)}, b^{n+1}) + g_2(\breve{u}^{(1)}, r^{(1)})\right), \tag{2.13}$$
$$\breve{u}^{(2)} = f_2(p^{(2)}, r^{(2)}, \breve{u}^{(2)}), \tag{2.14}$$
$$\breve{p}^{(2)} = \breve{p}^{(1)} + \Delta t f_1(\breve{\breve{u}}^{(2)}, \breve{r}^{(1)}, q^{n+1}), \quad \breve{\breve{u}}^{(2)} = f_2(\breve{p}^{(2)}, r^{(2)}, \breve{u}^{(2)}), \tag{2.15}$$
$$\breve{r}^{(2)} = \breve{r}^{(1)} + \Delta t \left(g_1(\breve{\breve{u}}^{(2)}, \breve{r}^{(1)}, b^{n+1}) + g_2(\breve{u}^{(1)}, \breve{r}^{(1)})\right), \tag{2.16}$$
$$p^{(3)} = \frac{1}{2}p^n + \frac{1}{2}\breve{p}^{(2)}, \tag{2.17}$$
$$r^{(3)} = \frac{1}{2}r^n + \frac{1}{2}\breve{r}^{(2)}, \tag{2.18}$$

as well as the correction stage

$$p^{cor,1} = p^n + \Delta t f_1(u^{cor,1}, r^{(3)}, q^{n+1}), \quad u^{cor,1} = f_2(p^{cor,1}, r^{(1)}, \breve{u}^{(1)}), \tag{2.19}$$
$$p^{cor,2} = \breve{p}^{(1)} + \Delta t f_1(u^{cor,2}, r^{(3)}, q^{n+1}), \quad u^{cor,2} = f_2(p^{cor,2}, r^{(2)}, \breve{u}^{(2)}), \tag{2.20}$$
$$\begin{aligned} p^{n+1} &= p^{(3)} + \Delta t \left(f_1(u^{cor,1}, r^{(3)}, q^{n+1}) - f_1(u^{cor,2}, r^{(3)}, q^{n+1})\right) \\ &= p^{(3)} + \breve{p}^{(1)} - p^{cor,2} + p^{cor,1} - p^n, \end{aligned} \tag{2.21}$$
$$r^{n+1} = r^{(3)} + \Delta t \left(g_1(u^{cor,1}, r^{(3)}, b^{n+1}) - g_1(u^{cor,2}, r^{(3)}, b^{n+1})\right), \tag{2.22}$$
$$u^{n+1} = f_2(p^{n+1}, r^{n+1}, u^{n+1}), \tag{2.23}$$

where $q^{n+1} = q(t^{n+1})$ and $b^{n+1} = b(t^{n+1})$. (2.1) and (2.2) are solved implicitly while (2.3) is solved explicitly in the above scheme. Note that the velocity equation (2.2) has been linearized in (2.7) and (2.12), which may degenerate the accuracy of $u^{(1)}$ and $u^{(2)}$, i.e. $u^{(1)} = u(t^{n+1}) + O(\Delta t)$ and $u^{(2)} = u(t^{n+2}) + O(\Delta t)$. To fix this problem, we construct $\breve{u}^{(1)}$, $\breve{u}^{(2)}$, $\breve{\breve{u}}^{(1)}$ and $\breve{\breve{u}}^{(2)}$ such that $\breve{u}^{(1)} = u(t^{n+1}) + O(\Delta t^2)$, $\breve{u}^{(2)} = u(t^{n+2}) + O(\Delta t^2)$, $\breve{\breve{u}}^{(1)} = u(t^{n+1}) + O(\Delta t^2)$ and $\breve{\breve{u}}^{(2)} = u(t^{n+2}) + O(\Delta t^2)$. By the same analysis given above, we construct the correction stage (2.19) and (2.20) by using $\breve{u}^{(1)}$ and $\breve{u}^{(2)}$. Moreover, we need to solve the nonlinear equations (2.9), (2.14) and (2.23), which will be discussed in the next section for Darcy-Forchheimer models. By the SIPEC-C scheme (2.7)-(2.23), we have $p^{n+1} = p(t^{n+1}) + O(\Delta t^3)$, $u^{n+1} = u(t^{n+1}) + O(\Delta t^3)$, $r^{n+1} = r(t^{n+1}) + O(\Delta t^3)$.

The detailed theoretical analysis of the above scheme can be found in [10] with some minor changes. Hence we omit it. However, the computational cost of the above scheme would be extremely large, since algorithm is too complicated and we need to update p and r six times. We can also confirm this point in numerical experiments. We still hope to find an improved sign-preserving second-order IMPEC method, namely SIPEC-S method, to save computational cost, and the new algorithm is given as follows.

$$p^{(1)} = p^n + \Delta t f_1(u^{(1)}, r^n, q^n), \quad u^{(1)} = f_2(p^{(1)}, r^n, u^n), \tag{2.24}$$
$$r^{(1)} = r^n + \Delta t \left(g_1(u^{(1)}, r^n, b^n) + g_2(u^n, r^n)\right), \tag{2.25}$$
$$\breve{u}^{(1)} = f_2(p^{(1)}, r^{(1)}, \breve{u}^{(1)}), \tag{2.26}$$
$$p^{(2)} = p^{(1)} + \Delta t f_1(u^{(2)}, r^{(1)}, q^{n+1}), \quad u^{(2)} = f_2(p^{(2)}, r^{(1)}, u^{(1)}), \tag{2.27}$$
$$r^{(2)} = r^{(1)} + \Delta t \left(g_1(u^{(2)}, r^{(1)}, b^{n+1}) + g_2(\breve{u}^{(1)}, r^{(1)})\right), \tag{2.28}$$

$$p^{(3)} = \frac{1}{2}p^n + \frac{1}{2}p^{(2)}, \tag{2.29}$$

$$r^{(3)} = \frac{1}{2}r^n + \frac{1}{2}r^{(2)}, \tag{2.30}$$

and correction stage

$$p^{cor,1} = p^n + \Delta t f_1(u^{cor,1}, r^{(3)}, q^{n+1}), \quad u^{cor,1} = f_2(p^{cor,1}, r^{(3)}, \breve{u}^{(1)}), \tag{2.31}$$

$$p^{cor,2} = p^{(2)} + \Delta t f_1(u^{cor,2}, r^{(3)}, q^{n+1}), \quad u^{cor,2} = f_2(p^{cor,2}, r^{(3)}, u^{(1)}), \tag{2.32}$$

$$\begin{aligned} p^{n+1} &= p^{(3)} + \frac{1}{2}\Delta t\left(f_1(u^{cor,1}, r^{(3)}, q^{n+1}) - f_1(u^{cor,2}, r^{(3)}, q^{n+1})\right) \\ &= p^{(3)} + \frac{1}{2}(p^{(2)} - p^{cor,2} + p^{cor,1} - p^n), \end{aligned} \tag{2.33}$$

$$r^{n+1} = r^{(3)} + \frac{1}{2}\Delta t\left(g_1(u^{cor,1}, r^{(3)}, b^{n+1}) - g_1(u^{cor,2}, r^{(3)}, b^{n+1})\right), \tag{2.34}$$

$$u^{n+1} = f_2(p^{n+1}, r^{n+1}, u^{n+1}). \tag{2.35}$$

Different from SIPEC-C scheme, although accuracy of $u^{(1)}$ and $u^{(2)}$ degenerates, we do not make further processing in schemes (2.24)-(2.30). Instead, we directly construct a new correction stage (2.31)-(2.35) to guarantee the second-order accuracy of p^{n+1}, r^{n+1} and u^{n+1}. The above algorithm is much easier than the SIPEC-C method, as we only need to update p and u four time, and the complexity is the same as that given in [10].

Now, we can state the following theorem.

Theorem 2.1. *The SIPEC-S scheme* (2.24)-(2.35) *is second-order accurate in time.*

Proof. Let's start with the traditional SSP-RK2 method [36].

$$\tilde{p}^{(1)} = p^n + \Delta t f_1(u^n, r^n, q^n), \tag{2.36}$$

$$\tilde{r}^{(1)} = r^n + \Delta t\left(g_1(u^n, r^n, b^n) + g_2(u^n, r^n)\right), \tag{2.37}$$

$$\tilde{u}^{(1)} = f_2(\tilde{p}^{(1)}, \tilde{r}^{(1)}, \tilde{u}^{(1)}), \tag{2.38}$$

$$\tilde{p}^{(2)} = \tilde{p}^{(1)} + \Delta t f_1(\tilde{u}^{(1)}, \tilde{r}^{(1)}, q^{n+1}), \tag{2.39}$$

$$\tilde{r}^{(2)} = \tilde{r}^{(1)} + \Delta t\left(g_1(\tilde{u}^{(1)}, \tilde{r}^{(1)}, b^{n+1}) + g_2(\tilde{u}^{(1)}, \tilde{r}^{(1)})\right), \tag{2.40}$$

$$\tilde{p}^{n+1} = \frac{1}{2}p^n + \frac{1}{2}\tilde{p}^{(2)}, \tag{2.41}$$

$$\tilde{r}^{n+1} = \frac{1}{2}r^n + \frac{1}{2}\tilde{r}^{(2)}, \tag{2.42}$$

$$\tilde{u}^{n+1} = f_2(\tilde{p}^{n+1}, \tilde{r}^{n+1}, \tilde{u}^{n+1}). \tag{2.43}$$

By the SSP-RK2 scheme, we have $\tilde{p}^{n+1} = p(t^{n+1}) + O(\Delta t^3)$, $\tilde{u}^{n+1} = u(t^{n+1}) + O(\Delta t^3)$ and $\tilde{r}^{n+1} = r(t^{n+1}) + O(\Delta t^3)$ after one step.

Next, we derive the relationship between SIPEC-S and SSP-RK2 schemes. We first obtain

$$p^{(1)} = \tilde{p}^{(1)} - \Delta t f_1(u^n, r^n, q^n) + \Delta t f_1(u^{(1)}, r^n, q^n) \tag{2.44}$$

by (2.24) and (2.36). It's easy to see that $p^{(1)} = p(t^{n+1}) + O(\Delta t^2)$. We use (2.27), (2.39) and (2.44) to get

$$p^{(2)} = \tilde{p}^{(2)} + \Delta t\left(f_1(u^{(2)}, r^{(1)}, q^{n+1}) + f_1(u^{(1)}, r^n, q^n) - f_1(u^n, r^n, q^n) - f_1(\tilde{u}^{(1)}, \tilde{r}^{(1)}, q^{n+1})\right). \tag{2.45}$$

Using (2.29), (2.41) and (2.45), we have

$$\tilde{p}^{n+1} = p^{(3)} + \frac{1}{2}\Delta t R_1 = p(t^{n+1}) + O(\Delta t^3), \tag{2.46}$$

where

$$R_1 = f_1(u^n, r^n, q^n) + f_1(\tilde{u}^{(1)}, \tilde{r}^{(1)}, q^{n+1}) - f_1(u^{(2)}, r^{(1)}, q^{n+1}) - f_1(u^{(1)}, r^n, q^n).$$

Similarly, it's easy to obtain

$$\tilde{r}^{n+1} = r^{(3)} + \frac{1}{2}\Delta t R_2 = r(t^{n+1}) + O(\Delta t^3), \tag{2.47}$$

where

$$R_2 = g_1(u^n, r^n, q^n) + g_1(\tilde{u}^{(1)}, \tilde{r}^{(1)}, b^{n+1}) - g_1(u^{(2)}, r^{(1)}, b^{n+1}) - g_1(u^{(1)}, r^n, b^n) + g_2(\tilde{u}^{(1)}, \tilde{r}^{(1)}) - g_2(\check{u}^{(1)}, r^{(1)}).$$

Next, let us deal with R_1 and R_2. We can substitute the expressions of $u^{(1)}$, $u^{(2)}$ and $\tilde{u}^{(1)}$ into R_1 to get that

$$\begin{aligned}
R_1 =& f_1\Big(f_2(p^n, r^n, u^n), r^n, q^n\Big) - f_1\Big(f_2(p^{(1)}, r^n, u^n), r^n, q^n\Big) \\
&+ f_1\Big(f_2(\tilde{p}^{(1)}, \tilde{r}^{(1)}, \tilde{u}^{(1)}), \tilde{r}^{(1)}, q^{n+1}\Big) - f_1\Big(f_2(p^{(2)}, r^{(1)}, u^{(1)}), r^{(1)}, q^{n+1}\Big) \\
=& f_1\Big(f_2\big(p(t^n), r(t^n), u(t^n)\big), r(t^n), q^n\Big) - f_1\Big(f_2\big(p(t^{n+1}), r(t^n), u(t^n)\big), r(t^n), q^n\Big) + O(\Delta t^2) \\
&+ f_1\Big(f_2\big(p(t^{n+1}), r(t^{n+1}), u(t^{n+1})\big), r(t^{n+1}), q^{n+1}\Big) - f_1\Big(f_2\big(p(t^{n+2}), r(t^{n+1}), u^{(1)}\big), r(t^{n+1}), q^{n+1}\Big) \\
:=& R_{11} - R_{12} + R_{13} - R_{14} + O(\Delta t^2).
\end{aligned}$$

Here we have used the fact that $p^{(1)}$, $\tilde{p}^{(1)}$, $\tilde{r}^{(1)}$, $\tilde{u}^{(1)}$, $p^{(2)}$, $r^{(1)}$ are locally second-order approximations (globally first-order accurate) of the exact solutions at the corresponding time levels since we only march one step in time. Notice that $u^{(1)} = u(t^{n+1}) + O(\Delta t)$, so we leave it here for the moment. Then, with Taylor's expansion, R_{11} and R_{12} are further transformed into

$$\begin{aligned}
R_{11} :=& f_1\Big(f_2\big(p(t^n), r(t^n), u(t^n)\big), r(t^n), q^n\Big) \\
=& f_1\Big(f_2\big(p(t^n), r(t^n), u(t^n)\big), r(t^{n+1}), q^{n+1}\Big) \\
&- F_r r_t(t^{n+1})\Delta t - F_q q_t(t^{n+1})\Delta t + O(\Delta t^2),
\end{aligned} \tag{2.48}$$

and

$$\begin{aligned}
R_{12} :=& f_1\Big(f_2\big(p(t^{n+1}), r(t^n), u(t^n)\big), r(t^n), q^n\Big) \\
=& f_1\Big(f_2\big(p(t^{n+1}), r(t^n), u(t^n)\big), r(t^{n+1}), q^{n+1}\Big) \\
&- f_{1_r}\Big(f_2\big(p(t^{n+1}), r(t^n), u(t^n)\big), r(t^{n+1}), q^{n+1}\Big) r_t(t^{n+1})\Delta t \\
&- f_{1_q}\Big(f_2\big(p(t^{n+1}), r(t^n), u(t^n)\big), r(t^{n+1}), q^{n+1}\Big) q_t(t^{n+1})\Delta t + O(\Delta t^2),
\end{aligned} \tag{2.49}$$

where

$$\begin{aligned}
F_r :=& f_{1_r}\Big(f_2\big(p(t^n), r(t^n), u(t^n)\big), r(t^{n+1}), q^{n+1}\Big) \\
=& f_{1_r}\Big(f_2\big(p(t^{n+1}), r(t^n), u(t^n)\big), r(t^{n+1}), q^{n+1}\Big) + O(\Delta t),
\end{aligned} \tag{2.50}$$

and

$$\begin{aligned}
F_q :=& f_{1_q}\Big(f_2\big(p(t^n), r(t^n), u(t^n)\big), r(t^{n+1}), q^{n+1}\Big) \\
=& f_{1_q}\Big(f_2\big(p(t^{n+1}), r(t^n), u(t^n)\big), r(t^{n+1}), q^{n+1}\Big) + O(\Delta t).
\end{aligned} \tag{2.51}$$

Therefore, we have

$$\begin{aligned}
R_{11} - R_{12} =& f_1\Big(f_2\big(p(t^n), r(t^n), u(t^n)\big), r(t^{n+1}), q^{n+1}\Big) \\
&- f_1\Big(f_2\big(p(t^{n+1}), r(t^n), u(t^n)\big), r(t^{n+1}), q^{n+1}\Big) + O(\Delta t^2) \\
=& f_1\Big(f_2\big(p(t^n), r(t^{n+1}), u(t^{n+1})\big), r(t^{n+1}), q^{n+1}\Big) \\
&- f_1\Big(f_2\big(p(t^{n+1}), r(t^{n+1}), u(t^{n+1})\big), r(t^{n+1}), q^{n+1}\Big) + O(\Delta t^2),
\end{aligned} \tag{2.52}$$

where in the second step we used Taylor's expansion similar to (2.48)-(2.51). Now we proceed to consider R_{14}. We have

$$\begin{aligned} R_{14} =& f_1\Big(f_2\big(p(t^{n+2}), r(t^{n+1}), u^{(1)}\big), r(t^{n+1}), q^{n+1}\Big) \\ =& f_1\Big(f_2\big(p(t^{n+2}), r(t^{n+1}), f_2\big(p(t^{n+1}), r(t^n), u(t^n)\big)\big), r(t^{n+1}), q^{n+1}\Big) + O(\Delta t^2), \end{aligned} \tag{2.53}$$

where in the last step, we used the fact that $u^{(1)} = f_2(p^{(1)}, r^n, u^n) = f_2(p(t^{n+1}), r(t^n), u(t^n)) + O(\Delta t^2)$. Combining (2.52) and (2.53), R_1 is rewritten as

$$\begin{aligned} R_1 =& f_1\Big(f_2\big(p(t^n), r(t^{n+1}), f_2\big(p(t^{n+1}), r(t^{n+1}), u(t^{n+1})\big)\big), r(t^{n+1}), q^{n+1}\Big) \\ & - f_1\Big(f_2\big(p(t^{n+2}), r(t^{n+1}), f_2\big(p(t^{n+1}), r(t^n), u(t^n)\big)\big), r(t^{n+1}), q^{n+1}\Big) + O(\Delta t^2) \\ =& f_1\Big(f_2\big(p(t^{n+1}), r(t^{n+1}), f_2\big(p(t^{n+1}), r(t^{n+1}), u(t^{n+1})\big)\big), r(t^{n+1}), q^{n+1}\Big) \\ & - f_1\Big(f_2\big(p(t^{n+3}), r(t^{n+1}), f_2\big(p(t^{n+1}), r(t^n), u(t^n)\big)\big), r(t^{n+1}), q^{n+1}\Big) + O(\Delta t^2) \\ =& f_1\Big(f_2\big(p(t^{n+1}), r(t^{n+1}), u(t^{n+1})\big), r(t^{n+1}), q^{n+1}\Big) \\ & - f_1\Big(f_2\big(p(t^{n+3}), r(t^{n+1}), u^{(1)}\big), r(t^{n+1}), q^{n+1}\Big) + O(\Delta t^2), \end{aligned} \tag{2.54}$$

where in the second step, we once again used Taylor's expansion similar to (2.48)-(2.52). Similarly, R_2 can be written as

$$\begin{aligned} R_2 =& g_1\Big(f_2\big(p(t^{n+1}), r(t^{n+1}), u(t^{n+1})\big), r(t^{n+1}), b^{n+1}\Big) \\ & - g_1\Big(f_2\big(p(t^{n+3}), r(t^{n+1}), u^{(1)}\big), r(t^{n+1}), b^{n+1}\Big) + O(\Delta t^2). \end{aligned} \tag{2.55}$$

Notice that the g_2 term disappears in (2.55) because $g_2(\tilde{u}^{(1)}, \tilde{r}^{(1)}) - g_2(\breve{u}^{(1)}, r^{(1)}) = O(\Delta t^2)$.

Substituting (2.54), (2.55) into (2.46) and (2.47) respectively, we can obtain that

$$\begin{aligned} p(t^{n+1}) =& p^{(3)} + \frac{1}{2}\Delta t\, f_1\Big(f_2\big(p(t^{n+1}), r(t^{n+1}), u(t^{n+1})\big), r(t^{n+1}), q^{n+1}\Big) \\ & - \frac{1}{2}\Delta t\, f_1\Big(f_2\big(p(t^{n+3}), r(t^{n+1}), u^{(1)}\big), r(t^{n+1}), q^{n+1}\Big) + O(\Delta t^3), \end{aligned}$$

and

$$\begin{aligned} r(t^{n+1}) =& r^{(3)} + \frac{1}{2}\Delta t\, g_1\Big(f_2\big(p(t^{n+1}), r(t^{n+1}), u(t^{n+1})\big), r(t^{n+1}), b^{n+1}\Big) \\ & - \frac{1}{2}\Delta t\, g_1\Big(f_2\big(p(t^{n+3}), r(t^{n+1}), u^{(1)}\big), r(t^{n+1}), b^{n+1}\Big) + O(\Delta t^3). \end{aligned}$$

By the correction stage (2.31) and (2.32), we construct $p^{cor,1}$, $u^{cor,1}$, $p^{cor,2}$ and $u^{cor,2}$, which are locally second-order approximations of the exact solutions at the corresponding time levels. So we can conclude that $p^{n+1} = p(t^{n+1}) + O(\Delta t^3)$, $u^{n+1} = u(t^{n+1}) + O(\Delta t^3)$ and $r^{n+1} = r(t^{n+1}) + O(\Delta t^3)$ in (2.33), (2.34) and (2.35). □

Remark 2.1. In proof of the Theorem 2.1, we can find that the construction of correction stage (2.31)-(2.35) is completely different from the correction stage in [10]. This is mainly because the velocity equation (2.2) is nonlinear and time-independent. After calculating $p^{(3)}$ and $r^{(3)}$ by schemes (2.24)-(2.30), the second-order accuracy of p^{n+1}, r^{n+1} and u^{n+1} cannot be guaranteed if we use the correction stage in [10]. Moreover, the second step is necessary in (2.54), otherwise we need to use p^{n-1} to construct the correction stage, and this multi-step scheme is not what we want. In the correction stage, $r(t^{n+1})$ must be approximated by $r^{(3)}$, ensuring that the bound-preserving technique can be applied. Finally, (2.33) and (2.34) are similar, though written in different forms.

Remark 2.2. It seems to be impossible to extend the proposed algorithm to construct third-order sign-preserving IMPEC method. Actually, to obtain third-order accuracy, we need to correct g_2, the diffusion term, leading to anti-diffusion, and the

Table 1
Comparison the four time integrations.

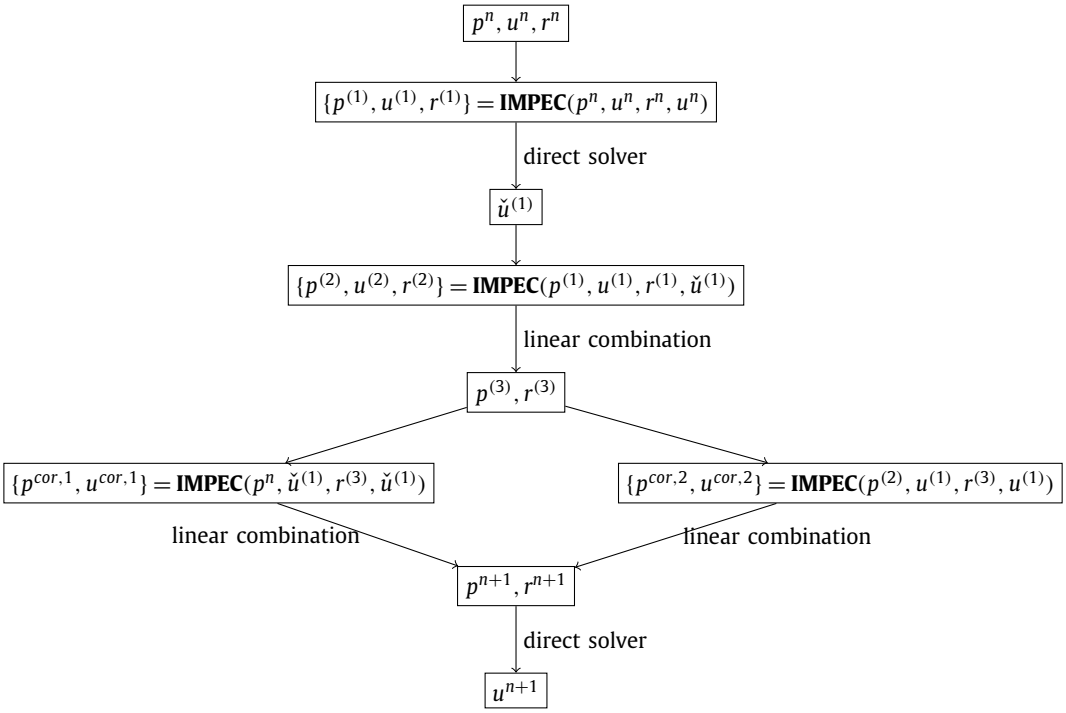
Method	Update p and r	Nonlinear solver	Global accuracy
SSP-RK2	2	2	2
IMPEC-SP	1	0	1
SIPEC-C	6	3	2
SIPEC-S	4	2	2

bound-preserving technique may fail to work. Though for problems with $g_2 = 0$, this issue disappears, the correction stage is still too difficult to construct.

Remark 2.3. As a summary, we explain the main ideas of the SIPEC-C method in (2.7)-(2.23) and the SIPEC-S method in (2.24)-(2.35). For the SIPEC-C method, we first treat the pressure equation implicitly and the concentration equation explicitly based on the framework of the SSP-RK2 method, so we construct (2.7), (2.8), (2.12) and (2.13). However, due to the linearization of the velocity equation, the velocity accuracy decreases. To fix this problem, we construct $\breve{u}^{(1)}$, $\breve{u}^{(2)}$ to update the pressure p and concentration r in (2.10), (2.11), (2.15) and (2.16), and we obtain first-order accurate in time. By comparing the scheme (2.7)-(2.18) with the traditional SSP-RK2 scheme (2.36)-(2.43), the truncation error between them is obtained. And we introduce a correction stage (2.19)-(2.23) to compensate for the second-order accuracy, maintaining the bound-preserving property of the numerical cell averages in the meantime. For SIPEC-S method, although the velocity accuracy decreases, we do not adjust it immediately. We derive the truncation error between scheme (2.24)-(2.30) and SSP-RK2 scheme directly, as shown in (2.46) and (2.47). By further derivation for truncation error, we get (2.54) and (2.55), therefore a new correction stage (2.31) -(2.35) is constructed. Compared with the SIPEC-C method, the SIPEC-S method has fewer number of stages and nonlinear solvers.

We compare the IMPEC-SP, SSP-RK2, SIPEC-C, SIPEC-S methods in Table 1 where the second and third columns give how many times we need to update p and r, and apply the nonlinear solver for (2.2), respectively. From the table, we can see that though IMPEC-SP has the least complexity, its accuracy is only one. Numerical experiments in Section 5 will demonstrate that the IMPEC-SP method requires a much smaller CFL number than the SIPEC-S method, otherwise, the IMPEC-SP method is not stable. Similar argument also works for SSP-RK2 method, which suffers from extremely limited time step sizes. Moreover, we would like to mention that the two correction stages in SIPEC-S scheme, see (2.31) and (2.32), and parallelizable leading to even smaller computational cost.

Finally, we summarize the SIPEC-S scheme in the following flow chart, where a typical IMPEC algorithm is given as (2.27)-(2.28):



Note that by the last two IMPEC schemes, we can obtain

$$
\begin{aligned}
r^{cor,1} &= r^n + \Delta t g_1(u^{cor,1}, r^{(3)}, b^{n+1}),\\
r^{cor,2} &= r^{(2)} + \Delta t g_1(u^{cor,2}, r^{(3)}, b^{n+1}),\\
r^{n+1} &= r^{(3)} + \frac{1}{2}(r^{(2)} - r^{cor,2} + r^{cor,1} - r^n),
\end{aligned}
$$

which is exactly the same as equation (2.34), and we just abbreviate these stages in the SIPEC-S scheme.

Next, we introduce sign-preserving definition of general ODE system (2.1)-(2.3).

Definition 2.1. When $0 \le r^n \le \Phi$, if $0 \le r^{n+1} \le \Phi$, the time discretization method of the ODE system (2.1)-(2.3) is said to be sign-preserving, where Φ is a constant.

To illustrate the sign-preserving property of the SIPEC-S method, we make the following assumptions for the ODE equations (2.1)-(2.3).

1. If $r^{*1} \ge 0$, then we have $\theta_1\, r^{*1} + \theta_2\, \Delta t\, g_1(u^{*2}, r^{*1}, b^{*1}) \ge 0$, $(1-\theta_1)\, r^{*1} + \theta_2\, \Delta t\, g_2(u^{*3}, r^{*1}) \ge 0$, where θ_1, θ_2 are two positive constants, and $0 \le \theta_1 \le 1$.
2. If
$$
\begin{cases}
p^{*1} = p^{*2} + \Delta t\ f_1(u^{*3}, r^{*2}, q^{*2}),\\
r^{*1} = r^{*2} + \Delta t\ \left(g_1(u^{*3}, r^{*2}, b^{*2}) + g_2(u^{*4}, r^{*2})\right),
\end{cases}
\tag{2.56}
$$
then we can get $r_2^{*1} = r_2^{*2} + \Delta t\ \left(g_1(u^{*3}, r_2^{*2}, b^{*2}) + g_2(u^{*4}, r_2^{*2})\right)$, where $r_2 = \Phi - r$.
3. There exists $\tilde{u}$ such that $g_1(u^{*2}, r^{*1}, b^{*1}) - g_1(u^{*3}, r^{*1}, b^{*1}) = g_1(\tilde{u}, r^{*1}, b^{*1})$, $f_1(u^{*2}, r^{*1}, q^{*1}) - f_1(u^{*3}, r^{*1}, q^{*1}) = f_1(\tilde{u}, r^{*1}, q^{*1})$.

Here variables such as p^{*1}, r^{*2}, u^{*2} represent the corresponding function values at any stage. Assumption 2 requires p and r on the right side of the equation (2.56) to be in the same stage, and the variable u in the functions f_1 and g_1 must be in the same stage. Based on the above assumptions, we state the following theorem.

Theorem 2.2. *The SIPEC-S scheme (2.24)-(2.35) is sign-preserving.*

Proof. Let's assume that $0 \le r^n \le \Phi$. Firstly, by (2.25) and assumption 1, we get

$$
r^{(1)} = \theta\, r^n + \Delta t\, g_1(u^{(1)}, r^n, b^n) + (1-\theta)\, r^n + \Delta t\, g_2(u^n, r^n) \ge 0.
$$

And using (2.24), (2.25) and assumption 2, we obtain

$$
r_2^{(1)} = r_2^n + \Delta t\ \left(g_1(u^{(1)}, r_2^n, b^n) + g_2(u^n, r_2^n)\right) \ge 0
$$

where $r_2^{(1)} = \Phi - r^{(1)}$. Therefore, $0 \le r^{(1)} \le \Phi$. Similarly, we have $0 \le r^{(2)} \le \Phi$ and $0 \le r^{(3)} \le \Phi$. In correction stage, we can find

$$
\begin{aligned}
r^{n+1} &= r^{(3)} + \frac{1}{2}\Delta t\ g_1(\tilde{u}^{cor}, r^{(3)}, b^{n+1}),\\
p^{n+1} &= p^{(3)} + \frac{1}{2}\Delta t\ f_1(\tilde{u}^{cor}, r^{(3)}, q^{n+1}),
\end{aligned}
$$

by assumption 3, so $0 \le r^{n+1} \le \Phi$ by similar analysis. □

It should be noted that we only consider time discretization for Definition 2.1 and Theorem 2.2, and if spatial discretization is also considered, it is called bound-preserving technique. We need to do the analysis in two steps. (1) When $0 \le r^n(x, y) \le \Phi(x, y)$, we can prove $0 \le \bar{r}^{n+1} \le \bar{\Phi}$, where $\bar{r}$, $\bar{\Phi}$ cell average of r, Φ, respectively. (2) By using a slope limiter, we can obtain $0 \le r^{n+1}(x, y) \le \Phi(x, y)$, and the slope limiter does not change the numerical cell averages. Moreover, assumptions 1 and 2 can be proved to be valid by using some spatial techniques such as consistent fluxes (see Definition 3.1), and assumption 3 is natural for equation (1.1)-(1.3). The above sign-preserving analysis is abstract, please refer to Section 4 for more details.

3. The fully-discrete IPDG schemes

In this section, we introduce the notations used throughout the paper and construct fully-discrete IPDG scheme for Darcy-Forchheimer compressible miscible displacements (1.1), (1.5) and (1.3).

3.1. Basic notations

For simplicity, we only consider rectangular meshes, and the techniques for triangular meshes can be obtained following [9]. Let $\Omega = [0, 2\pi] \times [0, 2\pi]$ be the computational domain and define the grid points as

$$0 = x_{\frac{1}{2}} < x_{\frac{3}{2}} < \cdots < x_{N_x-\frac{1}{2}} < x_{N_x+\frac{1}{2}} = 2\pi,$$
$$0 = y_{\frac{1}{2}} < y_{\frac{3}{2}} < \cdots < y_{N_y-\frac{1}{2}} < y_{N_y+\frac{1}{2}} = 2\pi.$$

Define $I_i = (x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}})$ and $J_j = (y_{j-\frac{1}{2}}, y_{j+\frac{1}{2}})$. Let

$$K_{ij} = I_i \times J_j,\ i = 1, \ldots, N_x,\ j = 1, \ldots, N_y$$

be the i, j-th rectangular cell. Denote $\Omega_h = \cup_{i,j} K_{ij}$ as a partition of Ω. If not otherwise stated, we always use K as the cell. The grid sizes in the x and y directions are given as

$$\Delta x_i = x_{i+\frac{1}{2}} - x_{i-\frac{1}{2}},\ \Delta y_j = y_{j+\frac{1}{2}} - y_{j-\frac{1}{2}}.$$

For simplicity, we assume uniform partition and denote $\Delta x = \Delta x_i,\ \Delta y = \Delta y_j$. However, this assumption is not essential. Moreover, we denote Γ as the set of all element interfaces and $\Gamma_0 = \Gamma \backslash \partial\Omega$. $\forall e \in \Gamma$, $|e|$ is the length of e. We choose $\boldsymbol{\beta} = (1, 1)^T$ to be a fixed vector that is not parallel to any normals of the cell interfaces. $\mathbf{n}_e$ is the unit normal of $e \in \Gamma_0$ such that $\boldsymbol{\beta} \cdot \mathbf{n}_e > 0$. Furthermore, we define

$$\partial\Omega_+ = \{e \in \partial\Omega : \boldsymbol{\beta} \cdot \mathbf{n} > 0\},\ \partial\Omega_- = \partial\Omega \backslash \partial\Omega_+,$$

where $\mathbf{n}$ is the unit outer normal of $\partial\Omega$. Following [10], we will develop a second-order IPDG scheme, and the finite element space is chosen as

$$W_h = \{z : z|_K \in Q^1(K),\ \forall K \in \Omega_h\},$$

where $Q^1(K)$ denotes the tensor product space of linear polynomials in cell K. Moreover, given $e \in \Gamma_0$, it is shared by two elements K_ℓ and K_r, where $\boldsymbol{\beta} \cdot \mathbf{n}_\ell > 0$ and $\boldsymbol{\beta} \cdot \mathbf{n}_r < 0$, with $\mathbf{n}_\ell$ and $\mathbf{n}_r$ being the unit outer normal of K_ℓ and K_r. $\forall z \in W_h$, we define $z^- = z|_{\partial K_l}$ and $z^+ = z|_{\partial K_r}$. And we use

$$[z] = z^+ - z^-,\ \{z\} = \frac{1}{2}(z^+ + z^-)$$

as the jump and average of z at the cell interfaces, respectively. For $\mathbf{s} \in \mathbf{W}_h = W_h \times W_h$, we define $[\mathbf{s}]$ and $\{\mathbf{s}\}$ analogously.

3.2. The fully-discrete IPDG schemes

We first rewrite (1.3) into the conservative form:

$$\phi \frac{\partial c}{\partial t} + \nabla \cdot (\mathbf{u}c) - \nabla \cdot (\mathbf{D}(\mathbf{u})\nabla c) = \tilde{c}q - \phi c z_1 p_t. \tag{3.1}$$

Following [18], we state the following key points for the bound-preserving technique.

1. Approximate $r = \phi c$ instead of c. We take the test function to be 1 to obtain the cell average of r.
2. p_t is used as a source to apply the positivity-preserving technique in (3.1).
3. Choose a consistent flux pair (see Definition 3.1) for equations (1.1) and (3.1) to ensure that the cell averages of r are less than or equal to the cell averages of ϕ.
4. Project the porosity ϕ into the finite element space, denoted as Φ.
5. Construct a slope limiter to keep the cell average $\bar{r}$ and modify the numerical approximations of r such that $0 \le r \le \Phi$. Then we compute $c = P\left(\frac{r}{\Phi}\right) \in [0, 1]$, where $P(u)\,|_K$ is the interpolation of u at the four vertices of cell K.

If not otherwise stated, we use p, $\mathbf{u}$, c as the numerical approximations from now on. The IPDG scheme for (1.1), (1.5) and (3.1) is to find $p,\ r \in W_h$, $\mathbf{u} \in \mathbf{W}_h$ such that for any $\xi,\ \zeta \in W_h$ and $\boldsymbol{\eta} \in \mathbf{W}_h$,

$$(\tilde{d}(r)p_t, \xi) = \mathcal{P}(\mathbf{u}, \xi) + (q, \xi), \tag{3.2}$$

$$(a(c)\mathbf{u}, \boldsymbol{\eta}) + (\beta\rho(c)|\mathbf{u}|\mathbf{u}, \boldsymbol{\eta}) = \mathcal{K}(p, \boldsymbol{\eta}), \tag{3.3}$$

$$(r_t, \zeta) = \mathcal{L}^c(\mathbf{u}, c, \zeta) + \mathcal{L}^d(\mathbf{u}, c, \zeta) + (\tilde{c}q - rz_1 p_t, \zeta), \tag{3.4}$$

where $c = P\left(\frac{r}{\Phi}\right)$, $\tilde{d}(r) = z_1 r + z_2(\Phi - r)$, $a(c) = \frac{\mu(c)}{\kappa}$, $(u, v) = \int_\Omega uv dxdy$, and

$$\mathcal{P}(\mathbf{u}, \xi) = (\mathbf{u}, \nabla\xi) + \sum_{e\in\Gamma_0}\int_e \hat{\mathbf{u}}\cdot\mathbf{n}_e[\xi]ds, \tag{3.5}$$

$$\mathcal{K}(p, \boldsymbol{\eta}) = (p, \nabla\cdot\boldsymbol{\eta}) + \sum_{e\in\Gamma}\int_e \hat{p}[\boldsymbol{\eta}\cdot\mathbf{n}_e]ds, \tag{3.6}$$

$$\mathcal{L}^c(\mathbf{u}, c, \zeta) = (\mathbf{u}c, \nabla\zeta) + \sum_{e\in\Gamma_0}\int_e \widehat{\mathbf{u}c}\cdot\mathbf{n}_e[\zeta]ds, \tag{3.7}$$

$$\begin{aligned}\mathcal{L}^d(\mathbf{u}, c, \zeta) = &-(\mathbf{D}(\mathbf{u})\nabla c, \nabla\zeta)\\ &- \sum_{e\in\Gamma_0}\int_e \left(\{\mathbf{D}(\mathbf{u})\nabla c\cdot\mathbf{n_e}\}[\zeta] + \{\mathbf{D}(\mathbf{u})\nabla\zeta\cdot\mathbf{n_e}\}[c] + \frac{\tilde{\alpha}}{|e|}[c][\zeta]\right)ds.\end{aligned} \tag{3.8}$$

In (3.5)-(3.7), $\hat{p}$, $\hat{\mathbf{u}}$ *and* $\widehat{\mathbf{u}c}$ are the numerical fluxes. We use alternating fluxes for the diffusion terms, $\forall e \in \Gamma_0$

$$\hat{p}|_e = p^-|_e, \quad \hat{\mathbf{u}}|_e = \mathbf{u}^+|_e,$$

and we take

$$\hat{p}|_e = p^-|_e,\ \forall e \in \partial\Omega_+, \quad \hat{p}|_e = p^+|_e,\ \forall e \in \partial\Omega_-.$$

For the convection term in (3.7), we use

$$\widehat{\mathbf{u}c} = \mathbf{u}^+c^+ - \alpha[c]\mathbf{n}_e.$$

Here α and $\tilde{\alpha}$ are two positive constants to be chosen by the bound-preserving technique.

Next, we would like to introduce the definition of consistent fluxes, which is required by the bound-preserving technique [18].

Definition 3.1. $\widehat{\mathbf{u}c}$ and $\hat{\mathbf{u}}$ are said to be consistent if $\widehat{\mathbf{u}c} = \hat{\mathbf{u}}$ by taking $c = 1$ in Ω.

Obviously, the numerical flux $\widehat{\mathbf{u}c}$ and $\hat{\mathbf{u}}$ are consistent. We can also discuss the following consistent fluxes for the bound-preserving technique. The proofs are similar, so we only list some of them below without more details. We use them in different numerical examples in Section 5.

- $\hat{p} = p^+,\ \hat{\mathbf{u}} = \mathbf{u}^-,\ \widehat{\mathbf{u}c} = \mathbf{u}^-c^- - \alpha[c]\mathbf{n}_e$.
- $\hat{p} = \frac{1}{2}(p^+ + p^-),\ \hat{\mathbf{u}} = \frac{1}{2}(\mathbf{u}^+ + \mathbf{u}^-),\ \widehat{\mathbf{u}c} = \frac{1}{2}(\mathbf{u}^+c^+ + \mathbf{u}^-c^-) - \alpha[c]\mathbf{n}_e$.

Now, let us propose the fully-discrete SIPEC-S-IPDG schemes. By (2.24)-(2.30), we can obtain $p^{(1)}$, $\mathbf{u}^{(1)}$, $r^{(1)}$, $\check{\mathbf{u}}^{(1)}$, $p^{(2)}$, $\mathbf{u}^{(2)}$, $r^{(2)}$, $p^{(3)}$, $r^{(3)}$ such that for any $\xi,\ \zeta \in W_h$ and $\boldsymbol{\eta} \in \mathbf{W}_h$,

$$(\tilde{d}(r^n)p^{(1)}, \xi) = (\tilde{d}(r^n)p^n, \xi) + \Delta t\left(\mathcal{P}(\mathbf{u}^{(1)}, \xi) + (q^n, \xi)\right), \tag{3.9}$$

$$(a(c^n)\mathbf{u}^{(1)}, \boldsymbol{\eta}) + (\beta\rho(c^n)|\mathbf{u}^n|\mathbf{u}^{(1)}, \boldsymbol{\eta}) = \mathcal{K}(p^{(1)}, \boldsymbol{\eta}), \tag{3.10}$$

$$(r^{(1)}, \zeta) = (r^n, \zeta) + \Delta t\left(\mathcal{L}^c(\mathbf{u}^{(1)}, c^n, \zeta) + \mathcal{L}^d(\mathbf{u}^n, c^n, \zeta) + (\tilde{c}^n q^n - r^n z_1 p_t^{(1)}, \zeta)\right), \tag{3.11}$$

$$(a(c^{(1)})\check{\mathbf{u}}^{(1)}, \boldsymbol{\eta}) + (\beta\rho(c^{(1)})|\check{\mathbf{u}}^{(1)}|\check{\mathbf{u}}^{(1)}, \boldsymbol{\eta}) = \mathcal{K}(p^{(1)}, \boldsymbol{\eta}), \tag{3.12}$$

$$(\tilde{d}(r^{(1)})p^{(2)}, \xi) = (\tilde{d}(r^{(1)})p^{(1)}, \xi) + \Delta t\left(\mathcal{P}(\mathbf{u}^{(2)}, \xi) + (q^{n+1}, \xi)\right), \tag{3.13}$$

$$(a(c^{(1)})\mathbf{u}^{(2)}, \boldsymbol{\eta}) + (\beta\rho(c^{(1)})|\mathbf{u}^{(1)}|\mathbf{u}^{(2)}, \boldsymbol{\eta}) = \mathcal{K}(p^{(2)}, \boldsymbol{\eta}), \tag{3.14}$$

$$(r^{(2)}, \zeta) = (r^{(1)}, \zeta) + \Delta t\left(\mathcal{L}^c(\mathbf{u}^{(2)}, c^{(1)}, \zeta) + \mathcal{L}^d(\check{\mathbf{u}}^{(1)}, c^{(1)}, \zeta) + (\tilde{c}^{(1)} q^{n+1} - r^{(1)} z_1 p_t^{(2)}, \zeta)\right), \tag{3.15}$$

$$p^{(3)} = \frac{1}{2}p^n + \frac{1}{2}p^{(2)}, \tag{3.16}$$

$$r^{(3)} = \frac{1}{2}r^n + \frac{1}{2}r^{(2)}, \tag{3.17}$$

where $p_t^{(1)} = \frac{p^{(1)}-p^n}{\Delta t}$, $p_t^{(2)} = \frac{p^{(2)}-p^{(1)}}{\Delta t}$, and we use $c = P\left(\frac{r}{\Phi}\right)$ to obtain c^n, $c^{(1)}$, $c^{(2)}$, $c^{(3)}$. Then we can obtain p^{n+1}, r^{n+1} and $\mathbf{u}^{n+1}$ by the correction stage

$$(\tilde{d}(r^{(3)})p^{cor,1}, \xi) = (\tilde{d}(r^{(3)})p^n, \xi) + \Delta t\left(\mathcal{P}(\mathbf{u}^{cor,1}, \xi) + (q^{n+1}, \xi)\right), \tag{3.18}$$

$$(a(c^{(3)})\mathbf{u}^{cor,1}, \boldsymbol{\eta}) + (\beta\rho(c^{(3)})|\breve{\mathbf{u}}^{(1)}|\mathbf{u}^{cor,1}, \boldsymbol{\eta}) = \mathcal{K}(p^{cor,1}, \boldsymbol{\eta}), \tag{3.19}$$

$$(\tilde{d}(r^{(3)})p^{cor,2}, \xi) = (\tilde{d}(r^{(3)})p^{(2)}, \xi) + \Delta t\left(\mathcal{P}(\mathbf{u}^{cor,2}, \xi) + (q^{n+1}, \xi)\right), \tag{3.20}$$

$$(a(c^{(3)})\mathbf{u}^{cor,2}, \boldsymbol{\eta}) + (\beta\rho(c^{(3)})|\mathbf{u}^{(1)}|\mathbf{u}^{cor,2}, \boldsymbol{\eta}) = \mathcal{K}(p^{cor,2}, \boldsymbol{\eta}), \tag{3.21}$$

$$(\tilde{d}(r^{(3)})p^{n+1}, \xi) = (\tilde{d}(r^{(3)})p^{(3)}, \xi) + \frac{1}{2}\Delta t\left(\mathcal{P}(\mathbf{u}^{cor,1}, \xi) - \mathcal{P}(\mathbf{u}^{cor,2}, \xi)\right), \tag{3.22}$$

$$(r^{n+1}, \zeta) = (r^{(3)}, \zeta) + \frac{1}{2}\Delta t\left(\mathcal{L}^c(\mathbf{u}^{cor,1} - \mathbf{u}^{cor,2}, c^{(3)}, \zeta) - \left(r^{(3)}z_1(p_t^{cor,1} - p_t^{cor,2}), \zeta\right)\right), \tag{3.23}$$

$$(a(c^{n+1})\mathbf{u}^{n+1}, \boldsymbol{\eta}) + (\beta\rho(c^{n+1})|\mathbf{u}^{n+1}|\mathbf{u}^{n+1}, \boldsymbol{\eta}) = \mathcal{K}(p^{n+1}, \boldsymbol{\eta}), \tag{3.24}$$

where $p_t^{cor,2} = \dfrac{p^{cor,2} - p^{(2)}}{\Delta t}$ and $p_t^{cor,1} = \dfrac{p^{cor,1} - p^n}{\Delta t}$. In practice, (3.22) can actually be simplified to $p^{n+1} = p^{(3)} + \frac{1}{2}\left(p^{(2)} - p^{cor,2} + p^{cor,1} - p^n\right)$.

Next let us discuss the method for solving nonlinear equations. In fact, we can use a direct method since the expression of equation (1.2) is relatively simple. Let $\mathbf{A}_c(\mathbf{u}) = a(c)\mathbf{u} + \beta\rho(c)|\mathbf{u}|\mathbf{u}$. Taking (3.12) as an example, the first step is to find $\mathbf{A}_{c^{(1)}}(\breve{\mathbf{u}}^{(1)}) = a(c^{(1)})\breve{\mathbf{u}}^{(1)} + \beta\rho(c^{(1)})|\breve{\mathbf{u}}^{(1)}|\breve{\mathbf{u}}^{(1)} \in \mathbf{W}_h$ such that for any $\boldsymbol{\eta} \in \mathbf{W}_h$,

$$\left(\mathbf{A}_{c^{(1)}}(\breve{\mathbf{u}}^{(1)}), \boldsymbol{\eta}\right) = \mathcal{K}(p^{(1)}, \boldsymbol{\eta}).$$

The second step is to find the value of $\breve{\mathbf{u}}^{(1)}$ at arbitrary Gaussian quadrature point (x_G, y_G). We define the values of $\mathbf{u}(x_G, y_G)$, $c(x_G, y_G)$, $a(c^{(1)}(x_G, y_G))$, $\rho(c^{(1)}(x_G, y_G))$, $\mathbf{A}_{c^{(1)}}(\breve{\mathbf{u}}^{(1)}(x_G, y_G))$ as $\mathbf{u}_G$, c_G, a_G, ρ_G, $\mathbf{A}_G$, respectively. We can solve $\breve{\mathbf{u}}_G^{(1)}$ by the following equation,

$$a_G\breve{\mathbf{u}}_G^{(1)} + \beta\rho_G|\breve{\mathbf{u}}_G^{(1)}|\breve{\mathbf{u}}_G^{(1)} = \mathbf{A}_G. \tag{3.25}$$

In fact, by the above equation, we can get

$$\left(a_G + \beta\rho_G|\breve{\mathbf{u}}_G^{(1)}|\right)^2|\breve{\mathbf{u}}_G^{(1)}|^2 = |\mathbf{A}_G|^2.$$

When $\beta\rho_G \geq 0$, dropping the negative term, we have

$$|\breve{\mathbf{u}}_G^{(1)}| = \frac{-a_G + \sqrt{a_G^2 + 4\beta\rho_G|\mathbf{A}_G|}}{2\beta\rho_G}. \tag{3.26}$$

Substituting (3.26) into (3.25), $\breve{\mathbf{u}}_G^{(1)}$ can be written as follows,

$$\breve{\mathbf{u}}_G^{(1)} = \frac{2\mathbf{A}_G}{a_G + \sqrt{a_G^2 + 4\beta\rho_G|\mathbf{A}_G|}}.$$

The third step is to take L^2-projection of $\frac{2\mathbf{A}}{a+\sqrt{a^2+4\beta\rho|\mathbf{A}|}}$ into $\mathbf{W}_h$, i.e. find $\breve{\mathbf{u}}^{(1)} \in \mathbf{W}_h$ such that for any $\boldsymbol{\eta} \in \mathbf{W}_h$,

$$(\breve{\mathbf{u}}^{(1)}, \boldsymbol{\eta}) = \left(\frac{2\mathbf{A}}{a + \sqrt{a^2 + 4\beta\rho|\mathbf{A}|}}, \boldsymbol{\eta}\right) = \sum_G \breve{\mathbf{u}}_G^{(1)}\boldsymbol{\eta}(x_G, y_G)w_G,$$

where w_G is the Gaussian quadrature weight in a cell. Through the above three steps, we obtain $\breve{\mathbf{u}}^{(1)}$ for (3.12). The method for equation (3.24) is similar, so we omit the details.

Remark 3.1. In practice, we only need to obtain the values of $\mathbf{u}$ at the Gaussian points, and $\beta\rho(c) \geq 0$ can easily be satisfied. Actually, we get $0 \leq c \leq 1$ by bound-preserving technique, and then $\beta\rho(c) = \beta(\rho_1 c + \rho_2(1-c)) \geq 0$, where $\beta \geq 0$, ρ_1 and ρ_2 are positive constants. For the general form of f_2, we can use an iterative method. For example, we can solve (2.4) using the following equation,

$$u^{0,l+1} = f_2(p^0, r^0, u^{0,l}).$$

This iterative method starts with random number $u^{0,0}$ and stops when $|u^{0,l+1} - u^{0,l}| < \epsilon$, with $\epsilon = 10^{-10}$. And then we can obtain $u^0 \approx u^{0,l+1}$.

4. Bound-preserving technique

4.1. Second-order bound-preserving

In this subsection, we will develop the bound-preserving technique for the sign-preserving SIPEC-S-IPDG scheme (3.9)-(3.24) in $\mathbb{R}^2$. For simplicity, we only consider the cells away from $\partial\Omega$, while the boundary cells can be analyzed similarly with minor changes [18]. We use o_{ij} for the numerical approximation o in cell K_{ij} with cell average $\bar{o}_{ij}$. We approximate the above integrals by 2-point Gaussian quadratures. The Gaussian quadrature points on I_i and J_j are denoted as $\{x_i^1, x_i^2\}$ and $\{y_j^1, y_j^2\}$, respectively. The corresponding weights on the interval $[-\frac{1}{2}, \frac{1}{2}]$ are represented as w_1 and w_2. Moreover, $o^+_{i+\frac{1}{2},j,\beta}$, $o^+_{i,j+\frac{1}{2},\beta}$ and $o^{++}_{i+\frac{1}{2},j+\frac{1}{2}}$ represent the values of $o(x^+_{i+\frac{1}{2}}, y_j^\beta)$, $o(x_i^\beta, y^+_{j+\frac{1}{2}})$ and $o(x^+_{i+\frac{1}{2}}, y^+_{j+\frac{1}{2}})$, respectively.

In [18], we have derived the bound-preserving analyses for IPDG schemes with forward Euler time integration, which can be directly extended to $r^{(1)}$ and $r^{(2)}$ in (3.9)-(3.15). Therefore, we conclude that $0 \leq \bar{r}^{(1)} \leq \bar{\Phi}$, $0 \leq \bar{r}^{(2)} \leq \bar{\Phi}$ and $0 \leq \bar{r}^{(3)} \leq \bar{\Phi}$ under different conditions. We take $r^{(1)}$ as an example and summarize the following theorem without further proof.

Theorem 4.1. *Suppose $0 \leq r^n \leq \Phi$, and the parameters α and $\tilde{\alpha}$ satisfy*

$$\alpha \geq \max_{\substack{2\leq i\leq N_x-1,\\ 2\leq j\leq N_y-1,\\ \beta=1,2}} \{u_1{}^{(1)+}_{i+\frac{1}{2},j,\beta},\ u_2{}^{(1)+}_{i,j+\frac{1}{2},\beta},\ 0\}, \tag{4.1}$$

$$\tilde{\alpha} \geq \max\left\{\frac{\Delta y}{2\Delta x}D_{11}^M + \sqrt{3}D_{12}^M,\ \frac{\Delta x}{2\Delta y}D_{22}^M + \sqrt{3}D_{21}^M\right\}, \tag{4.2}$$

where $D_{mn}^M = \max_{(x,y)\in\Omega} |D_{mn}(\mathbf{u}^n)(x,y)|$ $(m,n=1,2)$. Moreover, if the fluxes $\widehat{\mathbf{u}}c$ and $\widehat{\mathbf{u}}$ are consistent, then $0 \leq \bar{r}^{(1)} \leq \bar{\Phi}$ under the conditions

$$\frac{\Delta t}{\Delta x} + \frac{\Delta t}{\Delta y} \leq \frac{1}{6}\min\left\{\frac{\Phi_m}{\alpha},\ A_1^{(1)},\ A_2^{(1)}\right\}, \tag{4.3}$$

$$D_{11}^M\frac{\Delta t}{\Delta x^2} + 2(\tilde{\alpha} + D_{12}^M)\frac{\Delta t}{\Delta x\Delta y} \leq \frac{1}{12}\Phi_m, \tag{4.4}$$

$$D_{22}^M\frac{\Delta t}{\Delta y^2} + 2(\tilde{\alpha} + D_{21}^M)\frac{\Delta t}{\Delta x\Delta y} \leq \frac{1}{12}\Phi_m, \tag{4.5}$$

$$\Delta t \leq \frac{1}{6}\min\left\{\frac{1}{z_1 p_M^{(1)}},\ \frac{1}{z_2 p_M^{(1)}},\ \frac{\Phi_m}{q_M}\right\}, \tag{4.6}$$

where

$$\Phi_m = \min_{(x,y)\in\Omega}\Phi(x,y), \quad p_M^{(1)} = \max_{\substack{i,j,\\ \beta,\gamma=1,2}}\left\{p_t^{(1)}(x_i^\beta, y_j^\gamma),\ 0\right\}, \quad q_M^n = \max_{\substack{i,j,\\ \beta,\gamma=1,2}}\left\{-q^n(x_i^\beta, y_j^\gamma),\ 0\right\},$$

$$A_1^{(1)} = \min_{\substack{2\leq i\leq N_x-1,\\ 2\leq j\leq N_y-1,\\ \beta=1,2}} \frac{\Phi^{+\mp}_{i-\frac{1}{2},j\pm\frac{1}{2}}}{\alpha - u_1{}^{(1)+}_{i-\frac{1}{2},j,\beta}}, \quad A_2^{(1)} = \min_{\substack{2\leq i\leq N_x-1,\\ 2\leq j\leq N_y-1,\\ \beta=1,2}} \frac{\Phi^{\mp+}_{i\pm\frac{1}{2},j-\frac{1}{2}}}{\alpha - u_2{}^{(1)+}_{i,j-\frac{1}{2},\beta}}.$$

Next, we consider the bound-preserving technique for the correction stage. In (3.23), we take $\zeta = 1$ in K_{ij} to obtain

$$\bar{r}_{ij}^{n+1} = H_{ij}^c(r, \mathbf{u}, c) + H_{ij}^s(r, p_t),$$

where

$$H_{ij}^{c}(r,\mathbf{u},c)=\frac{1}{2}\bar{r}_{ij}^{(3)}+\frac{1}{2}\lambda\left(\int_{J_j}\widehat{u_1^{cor}c^{(3)}}_{i-\frac{1}{2},j}-\widehat{u_1^{cor}c^{(3)}}_{i+\frac{1}{2},j}dy+\int_{I_i}\widehat{u_2^{cor}c^{(3)}}_{i,j-\frac{1}{2}}-\widehat{u_2^{cor}c^{(3)}}_{i,j+\frac{1}{2}}dx\right),$$

$$H_{ij}^{s}(r,p_t)=\frac{1}{2}\bar{r}_{ij}^{(3)}-\frac{1}{2}\Delta t z_1\overline{r_{ij}^{(3)}p_t^{cor}},$$

and

$$\mathbf{u}^{cor}=\mathbf{u}^{cor(1)}-\mathbf{u}^{cor(2)},$$
$$p_t^{cor}=p_t^{cor(1)}-p_t^{cor(2)},$$

with $\lambda=\frac{\Delta t}{\Delta x\Delta y}$ and $\mathbf{u}^{cor}=(u_1^{cor},u_2^{cor})^T$. We first consider the source term H_{ij}^{s}, and then analyze the convection term H_{ij}^{c}. The results are shown below.

Lemma 4.1. *Suppose $r^{(3)}>0$ ($c^{(3)}>0$), then $H_{ij}^{s}(r,p_t)\geq 0$ under the condition*

$$\Delta t\leq\frac{1}{z_1 p_M^{cor}},\tag{4.7}$$

where

$$p_M^{cor}=\max_{\substack{i,j,\\ \beta,\gamma=1,2}}\left\{p_t^{cor}(x_i^{\beta},y_j^{\gamma}),0\right\}.\tag{4.8}$$

Lemma 4.2. *Suppose $r^{(3)}>0$ ($c^{(3)}>0$), then $H_{ij}^{c}(r,\mathbf{u},c)\geq 0$ if α and the time step Δt satisfy*

$$\alpha\geq\max_{\substack{2\leq i\leq N_x-1,\\ 2\leq j\leq N_y-1,\\ \beta=1,2}}\{{u_1}^{cor+}_{i+\frac{1}{2},j,\beta},\ {u_2}^{cor+}_{i,j+\frac{1}{2},\beta},\ 0\},\tag{4.9}$$

and

$$\frac{\Delta t}{\Delta x}+\frac{\Delta t}{\Delta y}\leq\frac{1}{2}\min\left\{\frac{\Phi_m}{\alpha},\ A_1^{cor},\ A_2^{cor}\right\},\tag{4.10}$$

where $\Phi_m=\min_{(x,y)\in\Omega}\Phi(x,y)$ and

$$A_1^{cor}=\min_{\substack{2\leq i\leq N_x-1,\\ 2\leq j\leq N_y-1,\\ \beta=1,2}}\frac{\Phi^{+\mp}_{i-\frac{1}{2},j\pm\frac{1}{2}}}{\alpha-{u_1}^{cor+}_{i-\frac{1}{2},j,\beta}},\quad A_2^{cor}=\min_{\substack{2\leq i\leq N_x-1,\\ 2\leq j\leq N_y-1,\\ \beta=1,2}}\frac{\Phi^{\mp+}_{i\pm\frac{1}{2},j-\frac{1}{2}}}{\alpha-{u_2}^{cor+}_{i,j-\frac{1}{2},\beta}}.$$

The proof of the above lemmas can be found in [10], which guarantees $\bar{r}^{n+1}\geq 0$. Next, we discuss how to prove $\bar{r}^{n+1}\leq\bar{\Phi}$.

Theorem 4.2. *Suppose the conditions in Lemma 4.1 and Lemma 4.2 are satisfied. Moreover, we assume $0\leq r^{(3)}\leq\Phi$ and the flux pair $(\widehat{\mathbf{u}c},\widehat{\mathbf{u}})$ is consistent, then $0\leq\bar{r}^{n+1}\leq\bar{\Phi}$ under another condition*

$$\Delta t\leq\frac{1}{z_2 p_M^{cor}},\tag{4.11}$$

where p_M^{cor} is given in (4.8).

Proof. Since the flux $\widehat{\mathbf{u}c}$ is consistent with $\widehat{\mathbf{u}}$, then $\widehat{\mathbf{u}}-\widehat{\mathbf{u}c}=\widehat{\mathbf{u}c_2}$ where $c_2=1-c$. Take $\xi=\zeta$ in (3.22), and subtract (3.23) from (3.22) to get

$$(r_2^{n+1},\zeta)=(r_2^{(3)},\zeta)+\frac{1}{2}\Delta t\left((\mathbf{u}^{cor}c_2^{(3)},\nabla\zeta)+\sum_{e\in\Gamma_0}\int_e\widehat{\mathbf{u}^{cor}c_2^{(3)}}\cdot\mathbf{n}_e[\zeta]ds-(r_2^{(3)}z_2p_t^{cor},\zeta)\right),\tag{4.12}$$

where $r_2 = \Phi - r$. We can easily see that (4.12) is exactly (3.23) when r, c *and* z_1 are replaced by r_2, c_2 *and* z_2, respectively. Following the same analyses in Lemma 4.1 and Lemma 4.2, we can show $\bar{r}_2^{n+1} \geq 0$, which further implies $\bar{r}^{n+1} \leq \bar{\Phi}$. □

Remark 4.1. When we prove the bound-preserving of $\bar{r}^{(1)}$ (or $\bar{r}^{(2)}$), the diffusion term satisfies (4.2), (4.4), (4.5). In practice, **D** would be very small, and the time step size restrictions are very mild unless the meshes are extremely refined. It means that the SIPEC-S method is suitable, since concentration equation (1.3) is convection-dominant. In the numerical experiments, we will choose $\Delta t \sim \Delta x$.

4.2. Slope limiter

By Theorem 4.2, we have proved that $0 \leq \bar{r}^{n+1} \leq \bar{\Phi}$. However, the numerical approximations of r may be negative or larger than Φ. Therefore, we need to apply a slope limiter to r. As discussed in [9], the procedure is given as follows.

1. Define $\hat{S} = \{(x, y) \in K : r(x, y) \leq 0\}$. Take
$$\hat{r} = r + \theta\left(\frac{\bar{r}}{\bar{\Phi}}\Phi - r\right), \quad \theta = \max_{(x,y)\in\hat{S}}\left\{\frac{-r(x,y)\bar{\Phi}}{\bar{r}\Phi(x,y) - r(x,y)\bar{\Phi}}, 0\right\};$$
2. Set $r_2 = \Phi - \hat{r}$, and repeat the above step for r_2 to get $\hat{r}_2$;
3. Take $\tilde{r} = \Phi - \hat{r}_2$ as the new approximation.

After the above three steps, we have $0 \leq \tilde{r} \leq \Phi$. It is easy to check that the limiter does not change the numerical cell averages, i.e. $\int_{K_{ij}} \tilde{r}(x)dx = \int_{K_{ij}} r(x)dx$. Moreover, the limiter does not affect the accuracy. See [9] for more information.

5. Numerical experiments

In this section, we give six numerical examples to illustrate the accuracy and capability of the bound-preserving IPDG method with the sign-preserving SIPEC-S time discretization for Darcy-Forchheimer compressible miscible displacements. Unless otherwise stated, we take $z_1 = z_2 = \phi = \kappa = \mu(c) = 1$.

5.1. One dimensional case

In this subsection, we solve problems in one space dimension. In the first example, we construct analytical solutions and test the accuracy and CPU time of the IPDG methods with the IMPEC-SP (2.5)-(2.6), SIPEC-S (2.24)-(2.35), SIPEC-C (2.7)-(2.23) and SSP-RK2 (2.36)-(2.43) time discretizations, respectively.

Example 5.1. We first consider the problem with $\beta = \rho = 1$, $D(u) = \gamma$, where γ is a constant. We choose the initial conditions as
$$c(x, 0) = \frac{1}{2}(1 - \cos(x)), \quad p(x, 0) = \cos(x) + \frac{1}{2}(\sin(x)\cos(x) - x),$$
and source parameters are taken as
$$q = -e^{-2t}(\sin(x)\cos(x) - x),$$
$$\tilde{c}q = \frac{1}{2}\left(e^{-(\gamma+1)t}\sin^2(x) - e^{-2t}(\sin(x)\cos(x) - x) + e^{-(\gamma+2)t}\left(\cos^2(x)\sin(x) - x\cos(x)\right)\right).$$
It is easy to verify that the exact solutions are
$$c(x, t) = \frac{1}{2}\left(1 - e^{-\gamma t}\cos(x)\right), \quad x \in [0, \pi],$$
$$p(x, t) = e^{-t}\cos(x) + \frac{e^{-2t}}{2}(\sin(x)\cos(x) - x), \quad x \in [0, \pi],$$
$$u(x, t) = e^{-t}\sin(x), \quad x \in [0, \pi].$$

We take $\Delta t = 0.47\Delta x$ ($\Delta x = \frac{\pi}{N}$), final time $T = 1$, and $\gamma = 10^{-5}$. We first use the SIPEC-S-IPDG method (3.9)-(3.24). The error of c in the L^2-norm, convergence rate and CPU time are listed in Table 2. We can observe optimal convergence rates regardless of whether the bound-preserving technique is used. Therefore, the bound-preserving technique does not degenerate the convergence order for one dimensional case.

Moreover, we test the accuracy and CPU time by using SIPEC-C (2.7)-(2.23) and SSP-RK2 (2.36)-(2.43) time methods without the bound-preserving limiter, where the maximum time steps are $\Delta t = 0.33\Delta x$ and $0.12\Delta x^2$, respectively. The

Table 2
Example 5.1: Accuracy test of c for SIPEC-S-IPDG schemes.

N	With limiter			No limiter		
	L^2 error	order	CPU time(s)	L^2 error	order	CPU time(s)
5	2.13e-02	-	0.07	1.55e-02	-	0.06
10	4.22e-03	2.34	0.17	3.50e-03	2.14	0.06
20	9.26e-04	2.19	0.38	8.39e-04	2.06	0.11
40	2.20e-04	2.08	0.91	2.09e-04	2.01	0.58
80	5.30e-05	2.05	1.89	5.17e-05	2.01	1.66
160	1.29e-05	2.04	9.14	1.28e-05	2.02	7.75

Table 3
Example 5.1: Accuracy test of c for IPDG method with SIPEC-C time discretization.

N	L^2 error	order	CPU time(s)
5	1.83e-02	-	0.25
10	4.32e-03	2.08	0.29
20	1.07e-03	2.01	0.25
40	2.63e-04	2.03	0.47
80	6.63e-05	1.99	3.03
160	1.65e-05	2.00	18.81

Table 4
Example 5.1: Accuracy test of c for fully-discrete SSPRK2-IPDG schemes.

N	L^2 error	order	CPU time(s)
5	1.54e-02	-	0.06
10	3.61e-03	2.10	0.08
20	8.79e-04	2.04	0.14
40	2.17e-04	2.02	2.92
80	5.39e-05	2.01	20.31
160	1.34e-05	2.01	155.28

Table 5
Example 5.1: Accuracy test of c for IMPEC-SP-IPDG schemes.

N	$\Delta t = 2.8\Delta x^2$			$\Delta t = 0.33\Delta x$		
	L^2 error	order	CPU time(s)	L^2 error	order	CPU time(s)
5	5.09e-01	-	0.03	7.53e-02	-	0.02
10	1.65e-01	1.62	0.14	3.71e-02	1.02	0.03
20	2.70e-02	2.61	0.42	1.86e-02	1.00	0.03
40	6.21e-03	2.12	0.88	9.33e-03	1.00	0.16
80	1.55e-03	2.00	1.86	4.68e-03	1.00	0.89
160	3.87e-04	2.00	24.23	2.34e-03	1.00	3.05

results are given in Table 3 and Table 4. By comparing the CPU time, it can be seen that the SIPEC-S method has the fastest computing speed. In addition, we also test this example using the IMPEC-SP method without limiter, where $\Delta t = 2.8\Delta x^2$ and $\Delta t = 0.33\Delta x$ are used. The results are shown in Table 5. By comparing Table 2 and Table 5, we find that the CPU time of SIPEC-S method is significantly smaller than that of IMPEC-SP method when the same accuracy is achieved. Though the IMPEC-SP method may yield less CPU time than the SIPEC-S method under the same resolution. However, to achieve the same error, the SIPEC-S method is also much faster than the IMPEC-SP method. Actually, we need $N = 160$ to make the error to be $2.34e-3$ and the CPU time is 3.05 s, while for the SIPEC-S method, we only need to take $N = 20$ to make the error even less, and the CPU time is only 0.11 s.

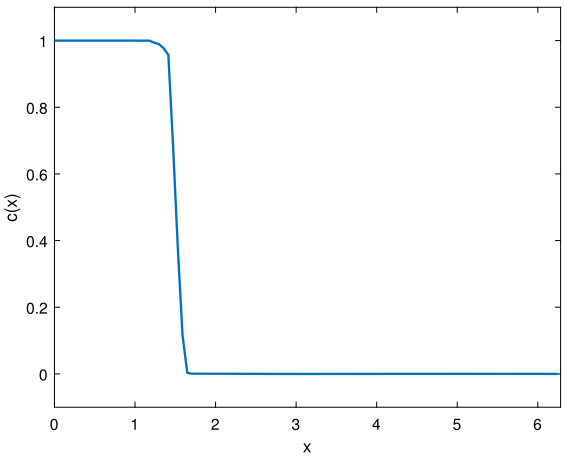
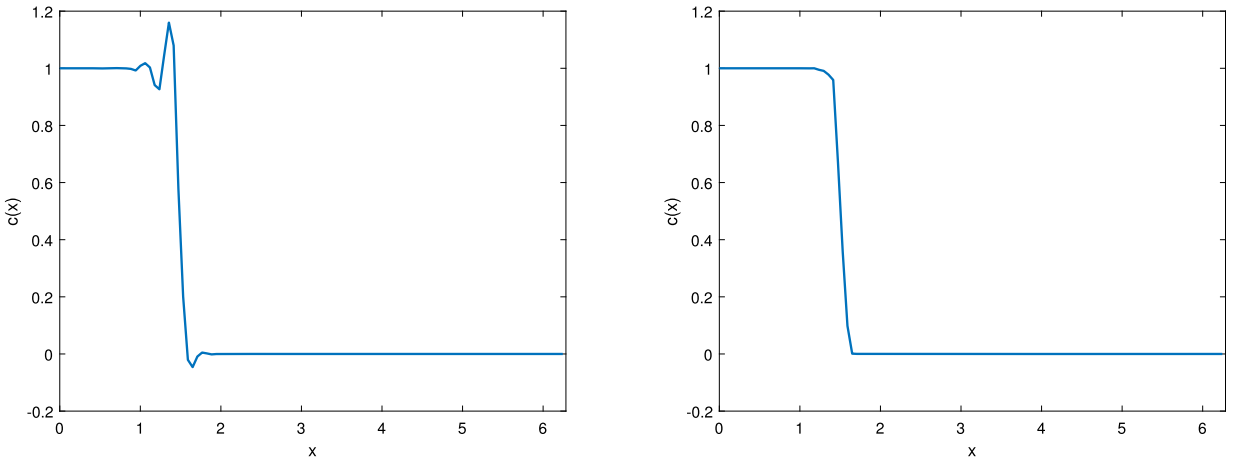


Fig. 1. Example 5.2: Numerical approximations of c at $T = 1$.



(a) c without limiter (b) c with limiter

Fig. 2. Example 5.2: Numerical approximations of c at $T = 1$ with and without bound-preserving limiter.

Example 5.2. We consider a problem with discontinuous initial solutions and test the necessity of the bound-preserving technique. Refer to [18], we set the initial conditions as

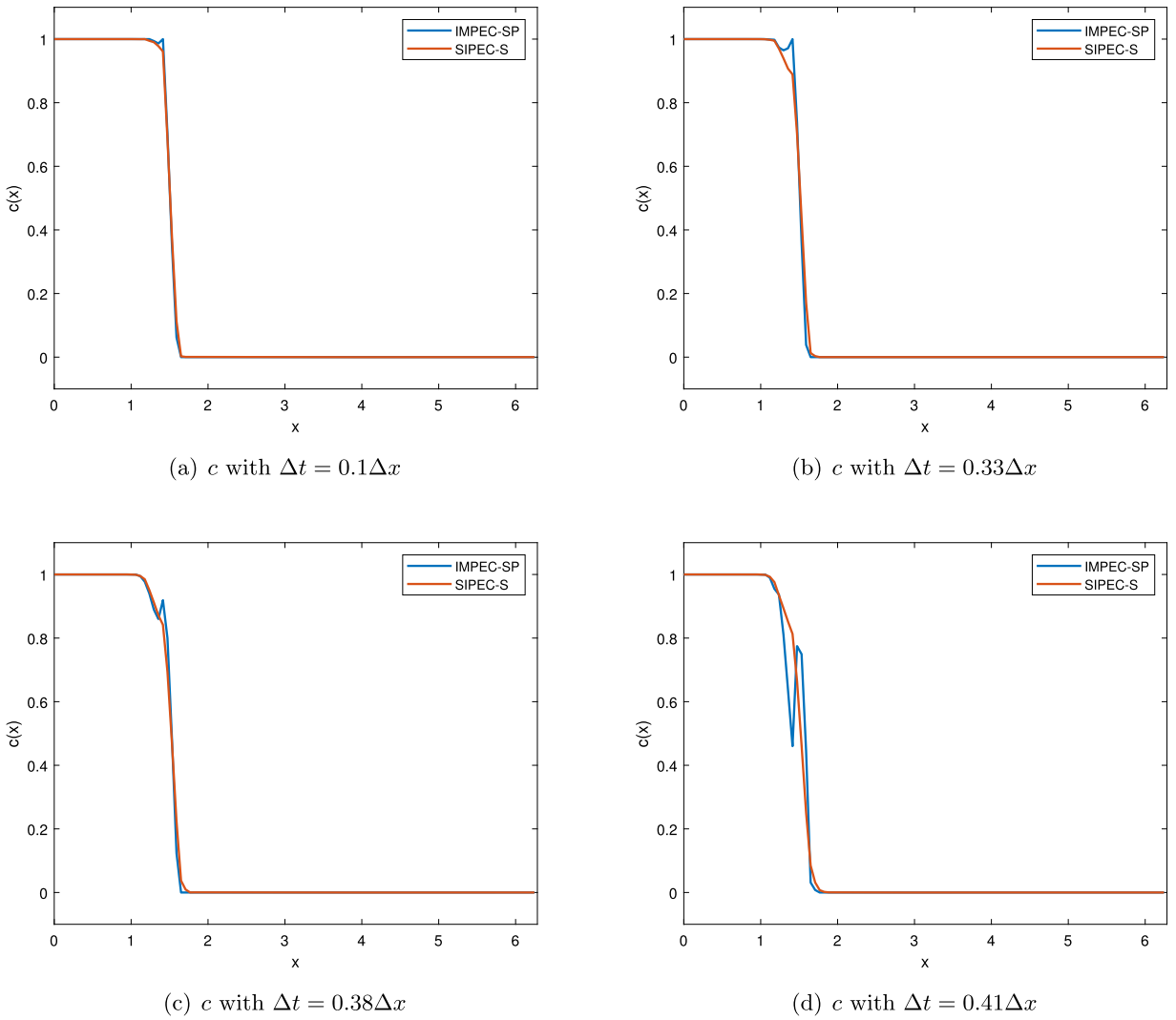
$$c(x,0) = \begin{cases} 1, & x < 1, \\ 0, & 1 \le x \le 2\pi, \end{cases} \qquad p(x,0) = \begin{cases} 5, & x < 1, \\ 0, & 1 \le x \le 2\pi. \end{cases}$$

Other parameters are taken as

$$q(x,t) = 0,\ z_1 = 0.1,\ z_2 = 1,\ \beta = 0.5,\ \rho = 1,\ D(u) = 0.$$

We compute up to $T = 1$ with $N = 80$ and $\Delta t = 0.13\Delta x$ ($\Delta x = \frac{2\pi}{N}$). We solve the problem by using schemes (3.9)-(3.24) with the bound-preserving limiter. The numerical approximation of c is shown in Fig. 1. We can observe that the numerical result is between 0 and 1. Then we solve the problem without the bound-preserving limiter, and the numerical approximation blows up at $T \approx 0.02$ s even though we take time step size as small as $\Delta t = 0.0001\Delta x$. Next, we change $\beta = 1$ and $\Delta t = 0.05\Delta x$. The results are shown in Fig. 2. We can observe oscillations and physically irrelevant values in Fig. 2(a). The above results demonstrate the necessity and effectiveness of the bound-preserving technique for Darcy-Forchheimer compressible miscible displacements.

Moreover, we also use SSP-RK2 time discretization to simulate the example. The results show that the CPU time by the SSP-RK2 method is about 38 s with maximum time step size $\Delta t = 0.002\Delta x$, while the CPU time by the SIPEC-S method is about 3.6 s with time step size $\Delta t = 0.13\Delta x$. Therefore, the CPU time of the SIPEC-S method is significantly less than that of SSP-RK2 method.



(a) c with $\Delta t = 0.1\Delta x$ (b) c with $\Delta t = 0.33\Delta x$

(c) c with $\Delta t = 0.38\Delta x$ (d) c with $\Delta t = 0.41\Delta x$

Fig. 3. Example 5.2: Concentrations c by the SIPEC-S (red) and IMPEC-SP (blue) time discretizations at $T = 1$ with different time step size Δt. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

We also test the IMPEC-SP method, where the backward and forward Euler methods are used to discretize time derivatives of pressure and concentration, respectively. The results are shown in Fig. 3. We can see that when the time step size is small, the difference between the SIPEC-S and IMPEC-SP methods is tiny. However, the larger the time step size, the more significant the difference. Though in the SIPEC-S method, we need to repeat the IMPEC-SP method 4 times, the CFL number for the SIPEC-S is more than 4 times larger than that for the IMPEC-SP method. Therefore, we expect the SIPEC-S method can yield compatible and even less CPU time than IMPEC-SP method, especially for problems with strong discontinuities.

In addition to the above, we compare the influence of β on the numerical solution of concentration. Since the initial pressure contains a discontinuity, the velocity can be large near the singularity. Therefore, the Darcy-Forchheimer model may be more accurate. In this example, we numerically test the difference between the Darcy and Darcy-Forchheimer model. We choose $\Delta t = 0.1\Delta x$, $\beta = 0,\ 10,\ 20,\ 40$. The results are shown in Fig. 4. We can see that when β is small, say $\beta = 10$, the Darcy model ($\beta = 0$) and the Darcy-Forchheimer model yield similar solutions. However, when β is large, the interfaces of the large gradient from the two models are different.

5.2. Two dimensional case

In this subsection, we solve equations (1.1)-(1.3) on the computational domain $\Omega = [0, 2\pi] \times [0, 2\pi]$. For simplicity, we take $N_x = N_y = N$ and $\Delta x = \Delta y = \frac{2\pi}{N}$. In the following example, we test the accuracy and CPU time of the bound-preserving IPDG schemes with different time methods.

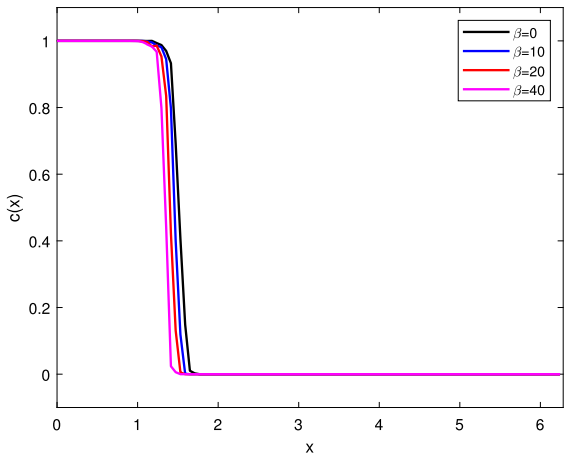
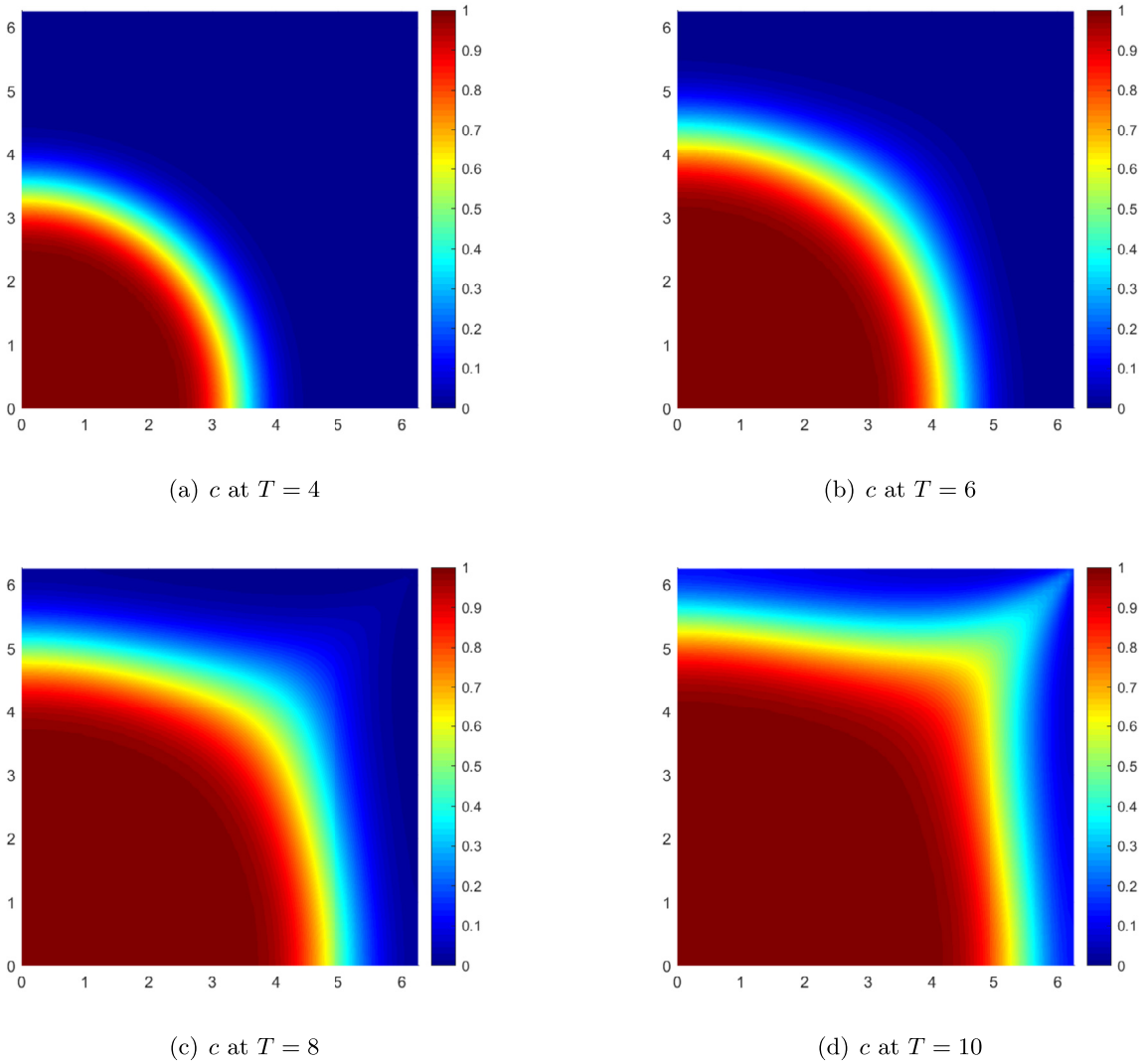


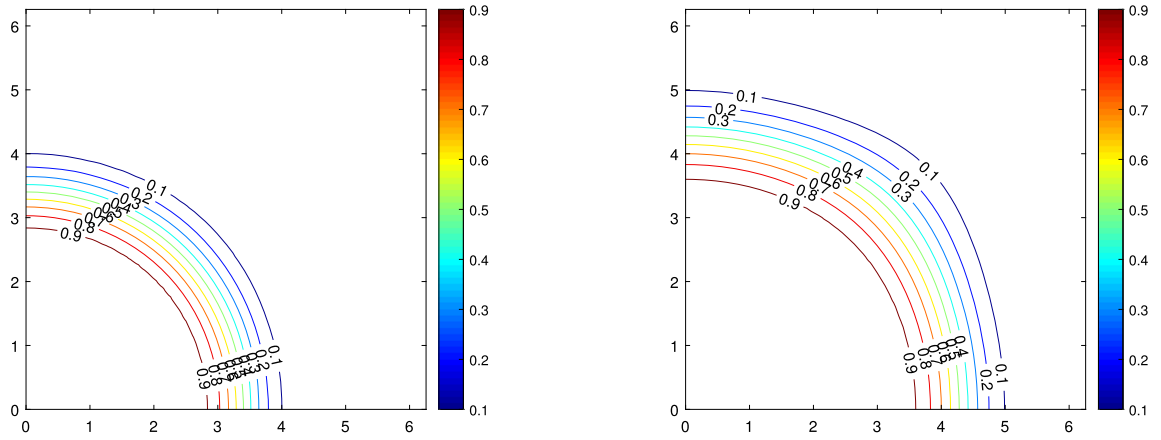
Fig. 4. Example 5.2: Numerical approximations of c with different β at $T = 1$.



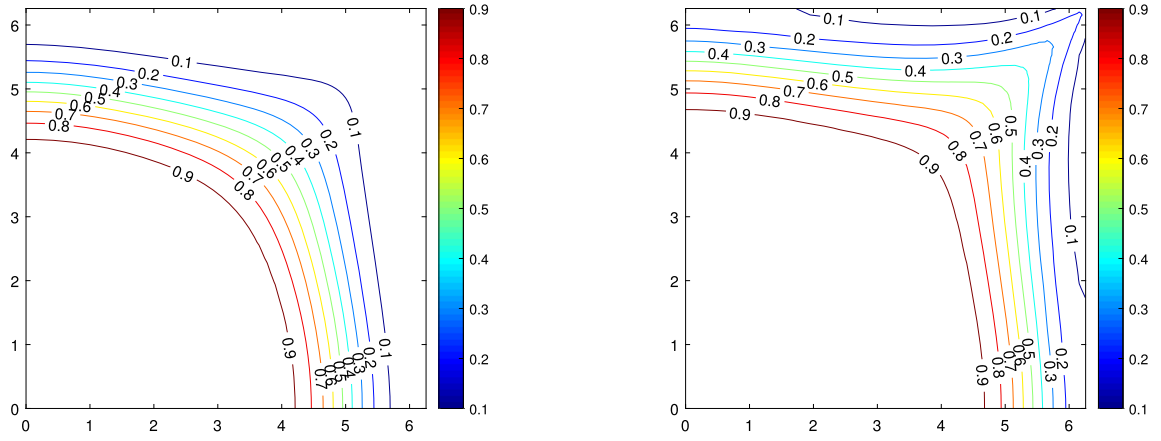
(a) c at $T = 4$ (b) c at $T = 6$

(c) c at $T = 8$ (d) c at $T = 10$

Fig. 5. Example 5.4: Distribution of concentration c with bound-preserving limiter at different time.



(a) c at $T = 4$ (b) c at $T = 6$



(c) c at $T = 8$ (d) c at $T = 10$

Fig. 6. Example 5.4: Contour plots of concentration c with bound-preserving limiter at different time.

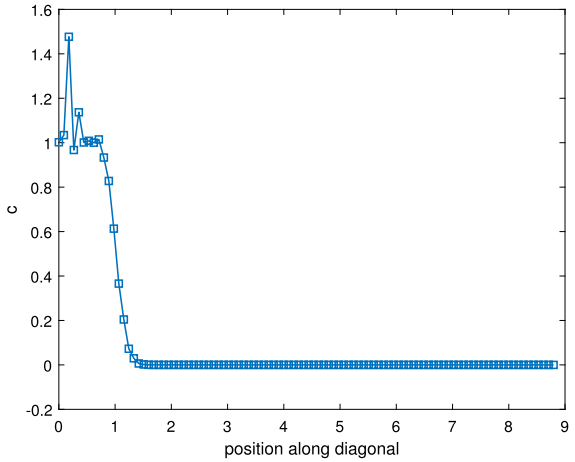
Example 5.3. We take the initial conditions

$$c(x, y, 0) = \frac{1}{2}\left(1 - \cos(x)\cos(y)\right), \ p(x, y, 0) = \cos(x)\cos(y) - 1.$$

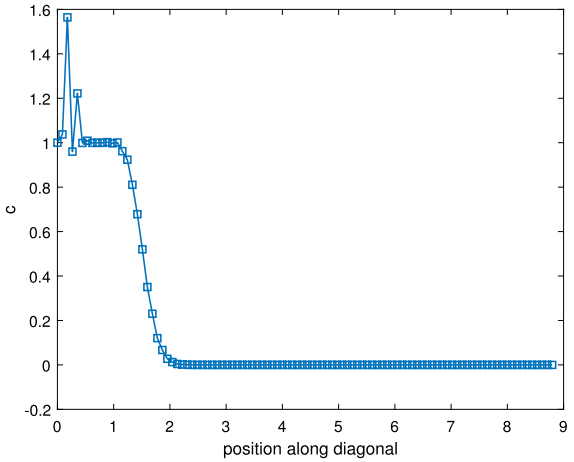
The source parameters are chosen as

$$\begin{aligned} q &= 2e^{-2t}, \\ \tilde{c} &= \frac{1}{2}\left(e^{-2\gamma t}\left(\frac{1}{2}\sin^2(x)\cos^2(y) + \frac{1}{2}\cos^2(x)\sin^2(y) - \cos(x)\cos(y)\right) + 1\right), \\ \mathbf{g} &= e^{-4t}\sqrt{\sin^2(x)\cos^2(y) + \cos^2(x)\sin^2(y)}\left(\sin(x)\cos(y),\ \cos(x)\sin(y)\right)^T, \end{aligned}$$

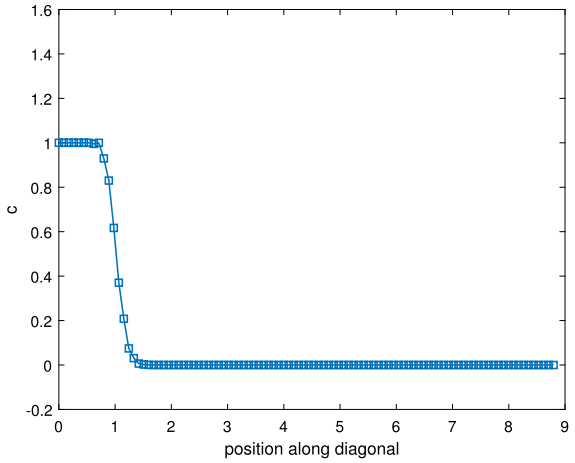
where we add a source term $\mathbf{g}$ to the velocity equation. We consider $\beta = \rho = 1$ and constant matrix $\mathbf{D}(\mathbf{u}) = \begin{pmatrix} \gamma & 0 \\ 0 & \gamma \end{pmatrix}$. It is easy to see that the exact solutions on $\Omega = [0, 2\pi] \times [0, 2\pi]$ are



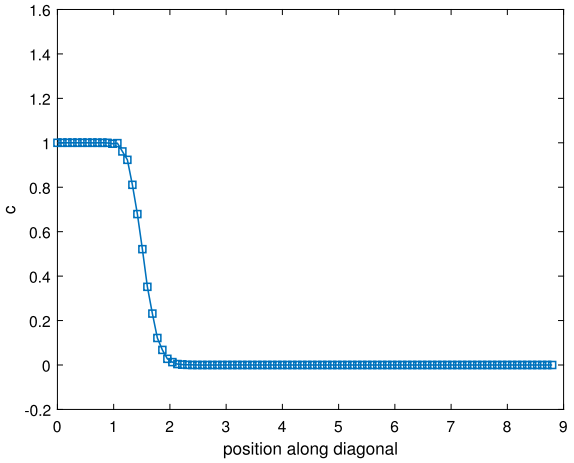
(a) c at $T = 0.5$ without limiter



(b) c at $T = 1$ without limiter



(c) c at $T = 0.5$ with limiter



(d) c at $T = 1$ with limiter

Fig. 7. Example 5.4: Concentration c with and without bound-preserving limiter.

$$c(x, y, t) = \frac{1}{2}\Big(1 - e^{-2\gamma t}\cos(x)\cos(y)\Big),$$
$$p(x, y, t) = e^{-2t}\Big(\cos(x)\cos(y) - 1\Big),$$
$$\mathbf{u}(x, y, t) = e^{-2t}\Big(\sin(x)\cos(y), \cos(x)\sin(y)\Big)^T.$$

We simulate the problem up to $T = 0.1$ with $\Delta t = 0.2\Delta x$ and $\gamma = 10^{-5}$ by using schemes (3.9)-(3.24). The L^2 error, convergence orders and CPU time for c are given in Table 6. We can observe second-order accuracy with and without the bound-preserving limiter. Therefore, the limiter does not kill the accuracy for two dimensional case.

Moreover, we use SIPEC-C (2.7)-(2.23) and SSP-RK2 (2.36)-(2.43) time discretizations to test the accuracy and CPU time without the bound-preserving limiter. The maximum time step of the SIPEC-C scheme is $\Delta t = 0.15\Delta x$, and the results are shown in Table 7. The maximum time step of SSP-RK2 scheme is $0.07\Delta x^2$, and the results are shown in Table 8. From the tables, we can see that the computing speed of the SIPEC-S method is faster than the other two methods for two dimensional case. We also test IMPEC-SP method without limiter, and we use $\Delta t = 0.5\Delta x^2$ and $\Delta t = 0.15\Delta x$. We can find the results in Table 9, where we observe second-order and first-order accuracy using different time step size. By comparing Table 6 and Table 9, the CPU time of SIPEC-S method is significantly smaller than that of IMPEC-SP method when the same accuracy is achieved. Though the IMPEC-SP method with first-order accuracy may spend less CPU time than the SIMPE-S method under the same resolution, to achieve the same error, it costs more CPU time than the SIPEC-S method.

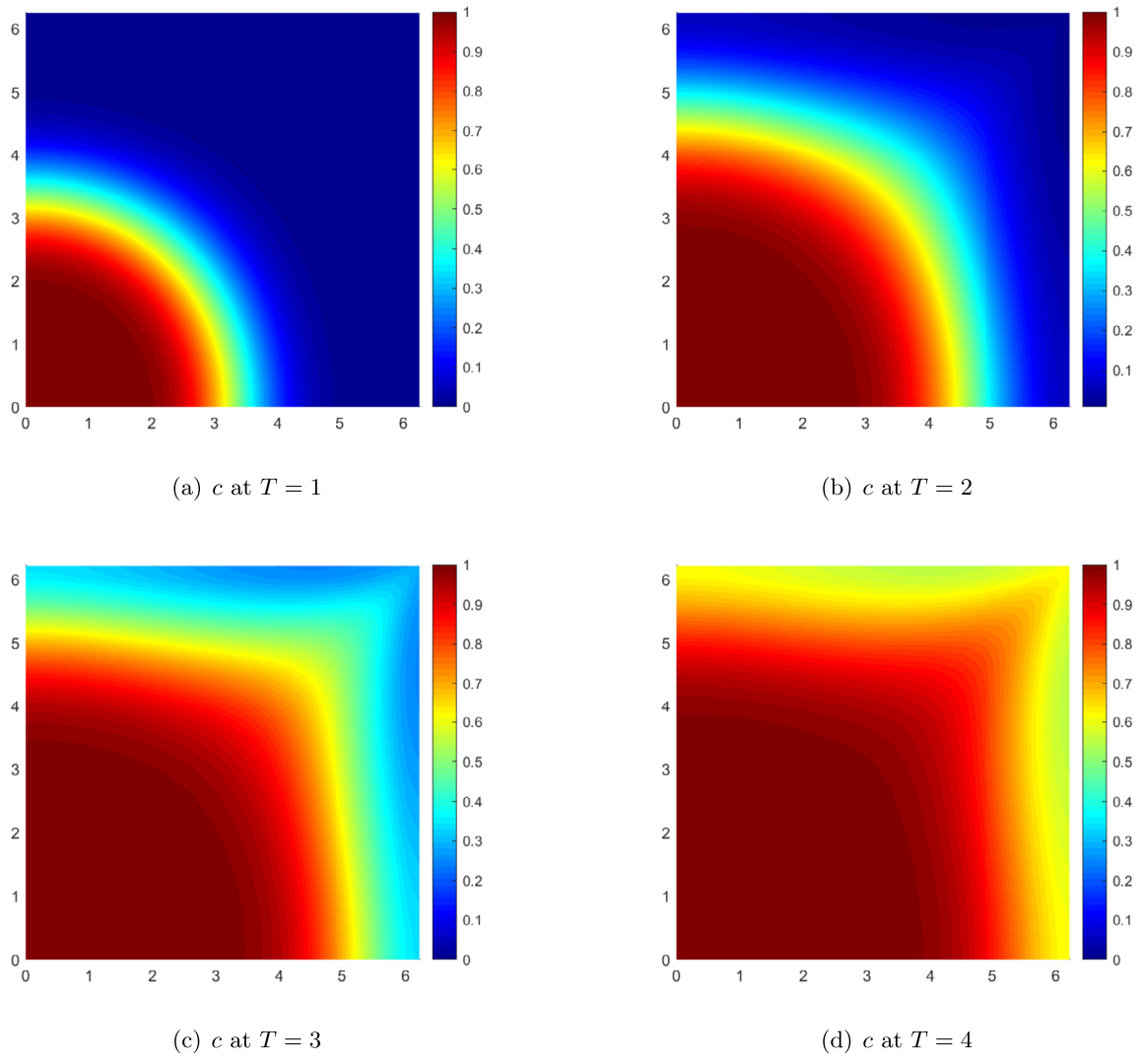


Fig. 8. Example 5.5: Distribution of concentration c with bound-preserving limiter at different time.

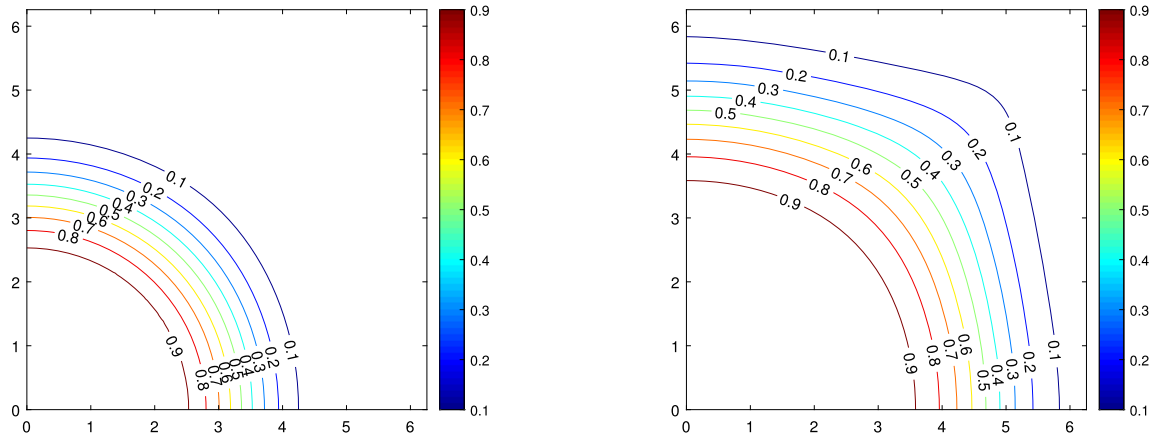
Table 6
Example 5.3: Accuracy test of c for the SIPEC-S-IPDG schemes.

N	With limiter			No limiter		
	L^2 error	order	CPU time(s)	L^2 error	order	CPU time(s)
5	1.94e-01	-	0.38	1.69e-01	-	0.16
10	5.15e-02	1.92	0.50	4.24e-02	1.99	0.26
20	1.12e-02	2.20	1.25	1.05e-02	2.01	0.98
40	2.66e-03	2.07	8.95	2.62e-03	2.01	10.47
80	6.58e-04	2.02	110.34	6.55e-04	2.00	111.11
160	1.64e-04	2.01	1904.60	1.64e-04	2.00	1905.10

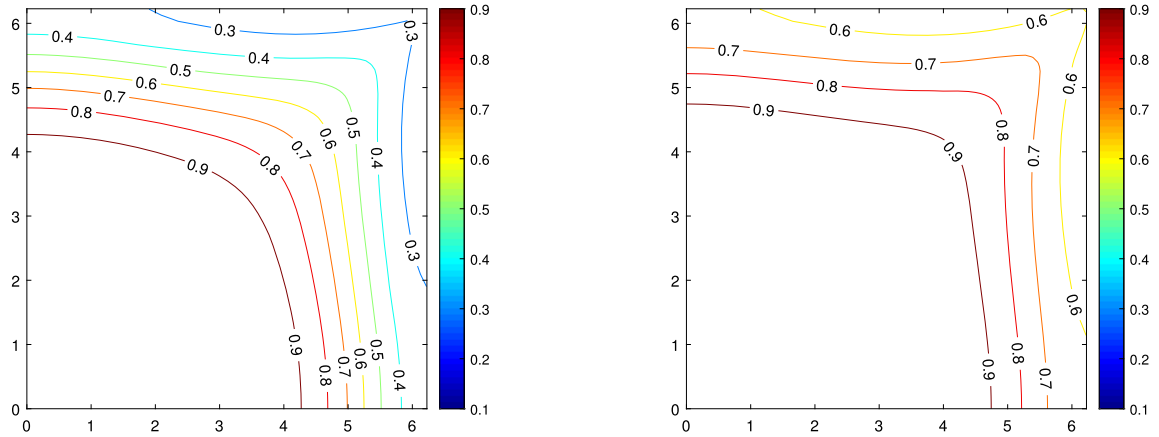
Example 5.4. In the example, we simulate the injection-production problem in oil recovery. We choose the initial conditions as $c_0 = 0,\ p_0 = 0$. Other parameters are taken as

$$z_1 = z_2 = 1,\ \phi(x, y) = 0.3,\ \mathbf{D}(\mathbf{u}) = 0.02\mathbf{I},$$

$$\mu_1 = 0.4,\ \mu_2 = 0.5,\ \kappa = 2, \beta = 5, \rho_1 = 0.9, \rho_2 = 1.$$



(a) c at $T = 1$ (b) c at $T = 2$



(c) c at $T = 3$ (d) c at $T = 4$

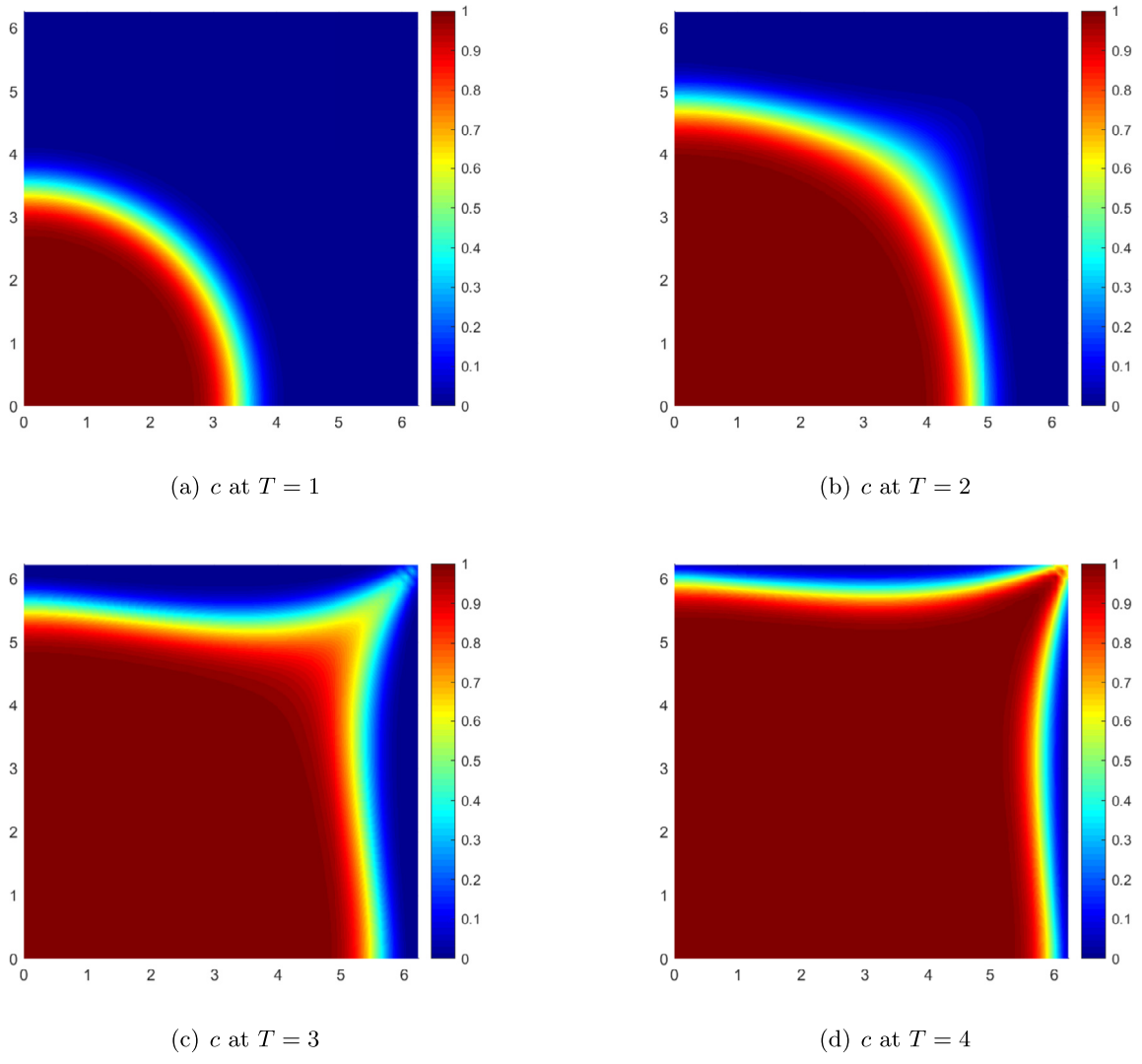
Fig. 9. Example 5.5: Contour plots of concentration c with bound-preserving limiter at different time.

Table 7
Example 5.3: Accuracy test of c for IPDG method with SIPEC-C time discretization.

N	L^2 error	order	CPU time(s)
5	1.73e-01	-	0.23
10	4.32e-02	2.00	0.54
20	1.06e-02	2.03	3.02
40	2.63e-03	2.01	13.67
80	6.56e-04	2.00	161.38
160	1.64e-04	2.00	2553.70

The viscosity $\mu(c)$ and density $\rho(c)$ are calculated using (1.6) and (1.7). The injection well is located at the lower-left corner with $q = \frac{1}{\Delta x \Delta y}$ and $\tilde{c} = 1$, and production well is located at the upper-right corner with $q = -\frac{1}{\Delta x \Delta y}$. The above parameters are obtained from [28].

We choose $\Delta t = 0.03\Delta x$, $N_x = N_y = 50$, and compute c at final time $T = 4$, 6, 8, 10 with the bound-preserving limiter. The numerical concentration distribution and contour are plotted in Figs. 5 and 6, respectively. The figures agree with that from [28]. From the figures we can see that the invading fluid moves faster along the flow direction of the reservoir, and the concentration c is between 0 and 1, which are all physically reasonable. However, the numerical approximations without



(a) c at $T = 1$ (b) c at $T = 2$

(c) c at $T = 3$ (d) c at $T = 4$

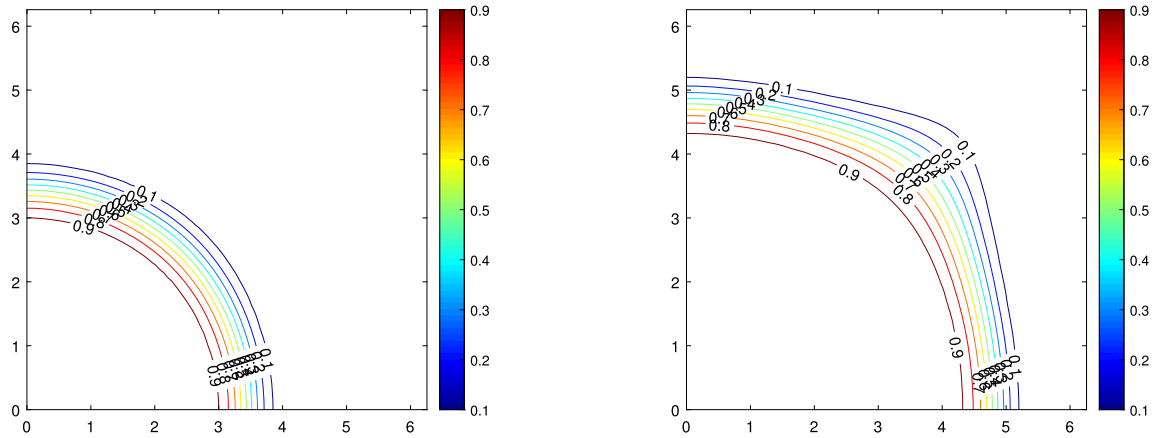
Fig. 10. Example 5.6: Distribution of concentration c with bound-preserving limiter at different time.

Table 8
Example 5.3: Accuracy test of c for fully-discrete SSPRK2-IPDG schemes.

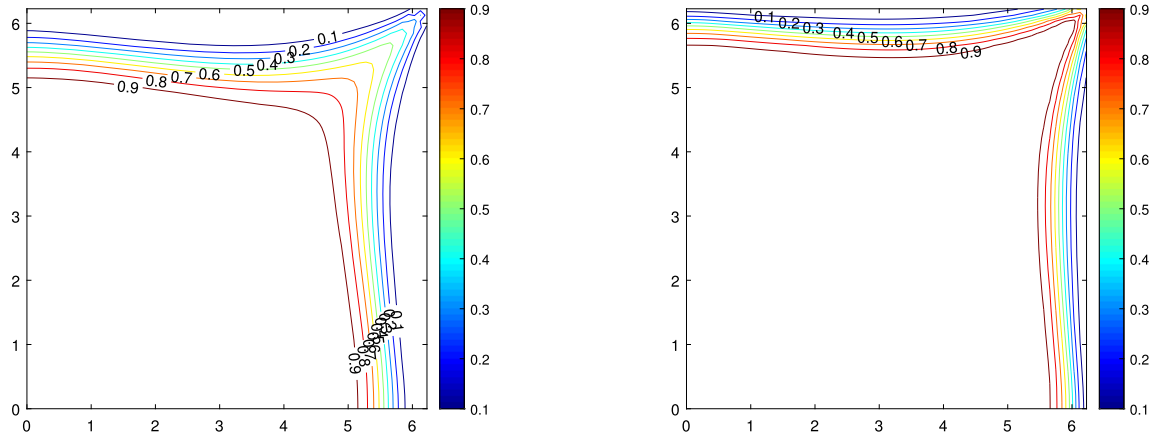
N	L^2 error	order	CPU time(s)
5	1.66e-01	-	0.17
10	4.23e-02	1.97	0.38
20	1.06e-02	2.00	3.50
40	2.64e-03	2.00	16.23
80	6.60e-04	2.00	247.13
160	1.65e-04	2.00	4030.80

bound-preserving limiter blow up at $T \approx 1.5$ s if we take the same time step as before. To test the effectiveness of the bound-preserving technique for two dimensional case, we simulate the distributions of c along diagonal at time $T = 0.5,\ 1$ with and without bound-preserving limiter. The results are shown in Fig. 7, from which we can observe strong oscillations and physically irrelevant values if the bound-preserving limiter is missing. The numerical results imply the effectiveness of the bound-preserving technique.

To test the good performance of the SIPEC-S method, we simulate the example by using the SSP-RK2 method at $T = 1$ with the bound-preserving technique. The CPU time by the SSP-RK2 method is about 2999 s with maximum time step size



(a) c at $T = 1$ (b) c at $T = 2$



(c) c at $T = 3$ (d) c at $T = 4$

Fig. 11. Example 5.6: Contour plots of concentration c with bound-preserving limiter at different time.

Table 9
Example 5.3: Accuracy test of c for the IMPEC-SP-IPDG schemes.

N	$\Delta t = 0.5\Delta x^2$			$\Delta t = 0.15\Delta x$		
	L^2 error	order	CPU time(s)	L^2 error	order	CPU time(s)
5	1.69e-01	-	0.14	1.67e-01	-	0.05
10	4.63e-02	1.86	0.56	4.58e-02	1.86	0.06
20	1.39e-02	1.74	0.95	1.42e-02	1.69	0.41
40	3.48e-03	2.00	8.09	5.73e-03	1.31	4.33
80	8.72e-04	2.00	123.82	2.84e-03	1.01	70.09
160	2.19e-04	2.00	2677.50	1.41e-03	1.01	1374.10

$\Delta t = 0.01\Delta x$, while the CPU time by the SIPEC-S method is about 715 s with time step size $\Delta t = 0.06\Delta x$. Through the comparison, we can conclude that the SIPEC-S method is superior to the traditional SSP-RK2 method.

Example 5.5. We set

$$\phi(x, y) = 0.1,\ \mathbf{D}(\mathbf{u}) = 0.05\mathbf{I},\ \mu_1 = \mu_2 = \beta = \rho_1 = \rho_2 = 1,\ \kappa = 25.$$

The other parameters are the same as in the Example 5.4. We choose $\Delta t = 0.01\Delta x$, $N_x = N_y = 50$ and final time $T = 1, 2, 3, 4$. The distribution and contour plots of the concentration c are given in Figs. 8 and 9. Similarly, we can observe that the wave front of the concentration moves from the injection well to the production well, and all the numerical approximations of c are between 0 and 1. We also simulate the example without the bound-preserving limiter. The numerical approximations blow up at about 0.05 s.

Moreover, we compare the CPU time of the SIPEC-S and SSP-RK2 methods at $T = 1$ with the bound-preserving limiter. The maximum time step of SSP-RK2 method is $\Delta t = 0.002\Delta x$ and the CPU time is about 13800 s, while the maximum time step of the SIPEC-S method is $\Delta t = 0.014\Delta x$ and the CPU time is about 3037 s. Therefore, the conclusion that the SIPEC-S method is superior to SSP-RK2 method is verified again.

Example 5.6. We consider a more practical problem, and calculate $\mathbf{D}(\mathbf{u})$ using (1.8) with $d_{mol} = 0$, $d_{long} = 0.5$ and $d_{tran} = 0.1$. Other parameters are the same as those given in Example 5.5.

We compute c at time $T = 1, 2, 3, 4$ with $N_x = N_y = 50$ and $\Delta t = 0.01\Delta x$. The numerical results of c at different time are shown in Figs. 10 and 11. From the figures we can see that the concentration c are between 0 and 1. Moreover, due to the difference in longitudinal versus transverse dispersion, the wave front of the concentration moves much faster along diagonal direction than it did in Figs. 8 and 9 [39,41].

6. Concluding remarks

In this paper, we constructed SIPEC-S time integration and applied it to two-component compressible miscible displacements with Darcy-Forchheimer models in porous media. We used the IPDG method for spatial discretization. Bound-preserving technique has been applied to the problems to obtain physically relevant numerical approximations. The SIPEC-S time method is applicable not only to linear equations, but also to nonlinear and time-independent equations. Numerical experiments were given to demonstrate the effectiveness of the bound-preserving technique, as well as the superiority of the SIPEC-S method by comparing it with the traditional SSP-RK2 and IMPEC-SP methods. Finally, we point out that the SIPEC-S method presents a significant advantage if D is small. However, if D is large, the advantage of SIPEC-S method may not be significant compared with the traditional SSP-RK2, which will be discussed in future work.

CRediT authorship contribution statement

Wenjing Feng: Investigation, Software, Writing – original draft. **Hui Guo:** Conceptualization, Funding acquisition, Writing – review & editing. **Lulu Tian:** Software. **Yang Yang:** Conceptualization, Funding acquisition, Investigation, Methodology, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

[1] K. Aziz, A. Settari, Petroleum Reservoir Simulation, Applied Science Publishers LTD, 1979.
[2] H. Chen, X. Fan, S. Sun, A fully mass-conservative iterative IMPEC method for multicomponent compressible flow in porous media, J. Comput. Appl. Math. 362 (2019) 1–21.
[3] H. Chen, J. Kou, S. Sun, T. Zhang, Fully mass-conservative IMPES schemes for incompressible two-phase flow in porous media, Comput. Methods Appl. Mech. Eng. 350 (2019) 641–663.
[4] H. Chen, S. Sun, A new physics-preserving IMPES scheme for incompressible and immiscible two-phase flow in heterogeneous porous media, J. Comput. Appl. Math. 381 (2021) 113035.
[5] Z. Chen, Reservoir Simulation: Mathematical Techniques in Oil Recovery, SIAM, Philadelphia, PA, USA, 2007.
[6] Z. Chen, G. Huan, B. Li, An improved IMPES method for two-phase flow in porous media, Transp. Porous Media 54 (2004) 361–376.
[7] Z. Chen, G. Huan, Y. Ma, Computational Methods for Multiphase Flows in Porous Media, SIAM, Philadelphia, PA, USA, 2006.
[8] A. Chertock, S. Cui, A. Kurganov, T. Wu, Steady state and sign preserving semi-implicit Runge-Kutta methods for ODEs with stiff damping term, SIAM J. Numer. Anal. 53 (2015) 2008–2029.
[9] N. Chuenjarern, Z. Xu, Y. Yang, High-order bound-preserving discontinuous Galerkin methods for compressible miscible displacements in porous media on triangular meshes, J. Comput. Phys. 378 (2019) 110–128.
[10] W. Feng, H. Guo, Y. Kang, Y. Yang, Bound-preserving discontinuous Galerkin methods with second-order implicit pressure explicit concentration time marching for compressible miscible displacements in porous media, J. Comput. Phys. 463 (2022) 111240.
[11] P. Forchheimer, Wasserbewegung durch Boden, Z. Ver. Dtsch. Ing. 45 (1901) 1782–1788.
[12] G. Fu, Y. Yang, A hybrid-mixed finite element method for single-phase Darcy flow in fractured porous media, Adv. Water Resour. 161 (2022) 104129.
[13] V. Girault, M.F. Wheeler, Numerical discretization of a Darcy-Forchheimer model, Numer. Math. 110 (2008) 161–198.

[14] S. Gottlieb, C.-W. Shu, E. Tadmor, Strong stability-preserving high-order time discretization methods, SIAM Rev. 43 (2001) 89–112.
[15] H. Guo, W. Feng, Z. Xu, Y. Yang, Conservative numerical methods for the reinterpreted discrete fracture model on non-conforming meshes and their applications in contaminant transportation in fractured porous media, Adv. Water Resour. 153 (2021) 103951.
[16] H. Guo, R. Jia, L. Tian, Y. Yang, Stability analysis and error estimates of fully-discrete local discontinuous Galerkin method for simulating wormhole propagation with Darcy-Forchheimer model, J. Comput. Appl. Math. 409 (2022) 114158.
[17] H. Guo, X. Liu, Y. Yang, High-order bound-preserving finite difference methods for miscible displacements in porous media, J. Comput. Phys. 406 (2020) 109219.
[18] H. Guo, Y. Yang, Bound-preserving discontinuous Galerkin method for compressible miscible displacement in porous media, SIAM J. Sci. Comput. 39 (2017) A1969–A1990.
[19] H. Guo, Q. Zhang, Y. Yang, A combined mixed finite element method and local discontinuous Galerkin method for miscible displacement problem in porous media, Sci. China Math. 57 (2014) 2301–2320.
[20] H. Hoteit, A. Firoozabadi, Multicomponent fluid flow by discontinuous Galerkin and mixed methods in unfractured and fractured media, Water Resour. Res. 41 (2005) W11412.
[21] H. Hoteit, A. Firoozabadi, Numerical modeling of two-phase flow in heterogeneous permeable media with different capillarity pressures, Adv. Water Resour. 31 (2008) 56–73.
[22] H. Hoteit, A. Firoozabadi, An efficient numerical model for incompressible two-phase flow in fractured media, Adv. Water Resour. 31 (2008) 891–905.
[23] W. Liu, J. Cui, A two-grid block-centered finite difference algorithm for nonlinear compressible Darcy-Forchheimer model in porous media, J. Sci. Comput. 74 (2018) 1786–1815.
[24] J.E.P. Monteagudo, A. Firoozabadi, Comparison of fully implicit and IMPES formulations for simulation of water injection in fractured and unfractured media, Int. J. Numer. Methods Eng. 69 (2007) 698–728.
[25] J. Moortgat, Adaptive implicit finite element methods for multicomponent compressible flow in heterogeneous and fractured porous media, Water Resour. Res. 53 (2017) 73–92.
[26] S.P. Neuman, Theoretical derivation of Darcy's law, Acta Mech. 25 (1977) 153–170.
[27] H. Pan, H. Rui, Mixed element method for two-dimensional Darcy-Forchheimer model, J. Sci. Comput. 52 (2012) 563–587.
[28] H. Pan, H. Rui, A mixed element method for Darcy-Forchheimer incompressible miscible displacement problem, Comput. Methods Appl. Mech. Eng. 264 (2013) 1–11.
[29] B. Rivière, Discontinuous Galerkin Methods for Solving Elliptic and Parabolic Equations: Theory and Implementation, SIAM, 2008.
[30] H. Rui, W. Liu, A two-grid block-centered finite difference method for Darcy-Forchheimer flow in porous media, SIAM J. Numer. Anal. 53 (2015) 1941–1962.
[31] H. Rui, H. Pan, A block-centered finite difference method for the Darcy-Forchheimer model, SIAM J. Numer. Anal. 50 (2012) 2612–2631.
[32] H. Rui, H. Pan, A block-centered finite difference method for slightly compressible Darcy-Forchheimer flow in porous media, J. Sci. Comput. 73 (2017) 70–92.
[33] D. Ruth, H. Ma, On the derivation of the Forchheimer equation by means of the averaging theorem, Transp. Porous Media 7 (1992) 255–264.
[34] J.W. Sheldon, B. Zondek, W.T. Cardwell, One-dimensional, incompressible, noncapillary, two-phase fluid flow in a porous medium, Trans. SPE AIME 216 (1959) 290–296.
[35] C.-W. Shu, Total-variation-diminishing time discretizations, SIAM J. Sci. Stat. Comput. 9 (1988) 1073–1084.
[36] C.-W. Shu, S. Osher, Efficient implementation of essentially non-oscillatory shock-capturing schemes, J. Comput. Phys. 77 (1988) 439–471.
[37] H.L. Stone, A.O. Garder Jr., Analysis of gas-cap or dissolved-gas reservoirs, Trans. SPE AIME 222 (1961) 92–104.
[38] L. Tian, H. Guo, R. Jia, Y. Yang, An h-adaptive local discontinuous Galerkin method for simulating wormhole propagation with Darcy-Forcheiner model, J. Sci. Comput. 82 (2020) 43.
[39] H. Wang, D. Liang, R.E. Ewing, S.L. Lyons, G. Qin, An approximation to miscible fluid flows in porous media with point sources and sinks by an Eulerian-Lagrangian localized adjoint method and mixed finite element methods, SIAM J. Sci. Comput. 22 (2000) 561–581.
[40] S. Whitaker, Flow in porous media I: a theoretical derivation of Darcy's law, Transp. Porous Media 1 (1985) 3–25.
[41] W. Xu, D. Liang, H. Rui, X. Li, A multipoint flux mixed finite element method for Darcy-Forchheimer incompressible miscible displacement problem, J. Sci. Comput. 82 (2020) 2.
[42] Z. Xu, Y. Yang, The hybrid dimensional representation of permeability tensor: a reinterpretation of the discrete fracture model and its extension on nonconforming meshes, J. Comput. Phys. 415 (2020) 109523.
[43] Z. Xu, Z. Huang, Y. Yang, The hybrid-dimensional Darcy's law: a non-conforming reinterpreted discrete fracture model (RDFM) for single-phase flow in fractured media, J. Comput. Phys. 473 (2023) 111749.
[44] L.C. Young, R.E. Stephenson, A generalized compositional approach for reservoir simulation, SPE J. 23 (1983) 727–742.
[45] X. Zhang, C.-W. Shu, On maximum-principle-satisfying high order schemes for scalar conservation laws, J. Comput. Phys. 229 (2010) 3091–3120.
[46] Y. Zhang, X. Zhang, C.-W. Shu, Maximum-principle-satisfying second order discontinuous Galerkin schemes for convection-diffusion equations on triangular meshes, J. Comput. Phys. 234 (2013) 295–316.
[47] A. Zidane, A. Firoozabadi, An efficient numerical model for multicomponent compressible flow in fractured porous media, Adv. Water Resour. 74 (2014) 127–147.
[48] A. Zidane, A. Firoozabadi, An implicit numerical model for multicomponent compressible two-phase flow in porous media, Adv. Water Resour. 85 (2015) 64–78.