An Improved Linearity-Preserving Cell-Centered Scheme for Nonlinear Diffusion Problems on General Meshes

Cheng Dong\textsuperscript{1,2} and Tong Kang\textsuperscript{1,2,*}

\textsuperscript{1} School of Information and Communication Engineering, Communication University of China, Beijing 100024, China
\textsuperscript{2} School of Data Science and Media Intelligence, Communication University of China, Beijing 100024, China

Received 12 January 2020; Accepted (in revised version) 25 June 2020

Abstract. In this paper, we suggest a new vertex interpolation algorithm to improve an existing cell-centered finite volume scheme for nonlinear diffusion problems on general meshes. The new vertex interpolation algorithm is derived by applying a special limit procedure to the well-known MPFA-O method. Since the MPFA-O method for 3D cases has been addressed in some studies, the new vertex interpolation algorithm can be extended to 3D cases naturally. More interesting is that the solvability of the corresponding local system is proved under some assumptions. Additionally, we modify the edge flux approximation by an edge-based discretization of diffusion coefficient, and thus the improved scheme is free of the so-called numerical heat-barrier issue suffered by many existing cell-centered or hybrid schemes. The final scheme allows arbitrary continuous or discontinuous diffusion coefficients and can be applicable to arbitrary star-shaped polygonal meshes. A second-order convergence rate for the approximate solution and a first-order accuracy for the flux are observed in numerical experiments. In the comparative experiments with some existing vertex interpolation algorithms, the new algorithm shows obvious improvement on highly distorted meshes.

AMS subject classifications: 65M08, 65M22

Key words: Cell-centered scheme, nonlinear diffusion equation, vertex interpolation algorithm, linearity-preserving criterion, numerical heat-barrier issue, finite volume scheme.

1. Introduction

Diffusion problems arise in a wide range of applications, such as radiation hydrodynamics (RHD), magnetohydrodynamics (MHD), plasma physics, reservoir modeling,
and so on. In many applications, certain factors make it difficult to solve diffusion problem, such as the anisotropic and nonlinear diffusion coefficients, the distorted meshes and the multi-materials, etc. Finite volume method is one of the most popular methods to solve diffusion problems because of its simplicity and local conservation. A desirable finite volume scheme is expected to be able to deal with the aforementioned challenges, and to possess as many as possible the good numerical properties, including stability, high order accuracy, local stencil, positivity-preserving property, robustness, efficiency, symmetry and positive definiteness of the resulting linear system, and so on. To our knowledge, there exists no scheme satisfying all the properties mentioned above.

In recent decades, there has been an impressive amount of work on the finite volume schemes of diffusion problems. The readers are referred to [7, 12, 14, 19] and the references cited therein for recent developments. Here we are concerned with a linearity-preserving nine-point scheme (NPS) studied in [35]. NPS was originally proposed in [21] and has been used a long time in RHD applications [10, 11, 25]. NPS has both cell-centered unknowns and vertex unknowns. Generally, the vertex unknowns are treated as auxiliary ones and have to be eliminated to get a pure cell-centered scheme. Interpolation of the vertex unknowns via the cell-centered ones is the key ingredient for NPS.

To our knowledge, there exists three simple interpolation algorithms: arithmetic average interpolation, inverse distance weighted interpolation and linear interpolation [20, 21, 23, 31]. These interpolation algorithms do not satisfy the so-called linearity-preserving criterion [35], which requires that each step of the derivation of interpolation algorithm is exact or linearly exact, i.e. exact in the sense whenever the solution is a linear function and the diffusion coefficient is a constant tensor. As a result, all the above three algorithms do not have the second order accuracy and are seldom used nowadays. Least square interpolation was proposed in [9] and a generalization was suggested in [24] to get a more accurate reconstruction of the vertex unknowns on Neumann boundary. Both weights in [9, 24] won’t achieve optimal accuracy for problems with discontinuous diffusion coefficients. [16] proposed two types of linearity-preserving explicit interpolation algorithms, the second of which, called LPEW2, has been examined by many authors [8, 18, 29, 32, 33]. However, all aforementioned interpolation algorithms can not be directly extended to 3D cases. As far as we know, all existing vertex interpolation algorithms for 3D diffusion problems are based on the so-called harmonic averaging point [3, 15, 17]. It has been demonstrated in [38] that the harmonic averaging point can go outside the relevant cell face, which results in a possible loss of accuracy for discontinuous and anisotropic diffusion problems. Moreover, since the harmonic averaging point also depends on the diffusion coefficients, its location changes in each nonlinear iteration when nonlinear diffusion problems are solved, leading to a dynamic stencil for the relevant scheme.

Two kinds of vertex interpolation algorithms based on MPFA-O method and extrapolation technology were present in [35, 37]. On highly distorted meshes, some schemes employing these two algorithms, such as NPS, have low accuracy and unstable performance, and even produce totally wrong numerical results.
In this paper, we focus on designing a new linearity-preserving vertex interpolation algorithm to improve the classical NPS. We derive a new vertex interpolation algorithm by applying a special limit procedure to the interpolation algorithm in MPFA-O method [1,13]. Since the extension of MPFA-O to 3D cases is natural and has been addressed in some studies [2], the new vertex interpolation in this paper can be extended to 3D cases naturally. In the comparative experiments between the new vertex interpolation algorithm and some existing algorithms, obvious improvements are observed on highly distorted meshes.

Recently, it has been pointed out in [22] that many existing finite volume schemes for nonlinear parabolic equations, including the mimetic finite difference schemes [4, 6], suffer the so-called heat-barrier issue. A scheme with staggered discretization of diffusion coefficients was then suggested in [22] to overcome this problem. In [34], a family of vertex-centered linearity-preserving schemes for nonlinear parabolic problems on polygonal grids were proposed and they are free of the numerical heat-barrier issue. However, some cell-centered schemes such as NPS are still obsessed with the numerical heat-barrier issue.

In practical computation, the coefficient associated with diffusion tensor has to be replaced for many cell-centered schemes to avoid the numerical heat-barrier issue. For example, an average of diffusion coefficients of two neighboring cells in the arithmetical sense was applied in some studies, such as [26, 27]. Another edge-based discretization of diffusion coefficient is presented in [5]. In this paper, we modify the edge flux approximation by the edge-based discretization of diffusion coefficient in [30] to avoid the numerical heat-barrier issue.

The rest of this paper is organized as follows. In Section 2, the numerical heat-barrier issue and the motivation of the new scheme are illustrated by a 1D example. Section 3 gives the construction of the improved linearity-preserving NPS. The derivation of a new vertex interpolation algorithm is presented in Section 4 and the solvability of the relevant local system is discussed in Section 5. We give several tests in Section 6, where the numerical results verify the accuracy and effectiveness of the new scheme and the new vertex interpolation algorithm. Finally, some conclusions are given in the last section.

### 2. The numerical heat-barrier issue

In this section, we follow [34, Section 2] and add some details to give a further explanation of the numerical heat-barrier issue. Moreover, we will give the main idea of our new cell-centered scheme.

Consider the following 1D nonlinear parabolic equation:

\[
\frac{\partial u}{\partial t} - \frac{\partial}{\partial x} \left( \lambda(u) \frac{\partial u}{\partial x} \right) = f, \quad (x,t) \in (0,a) \times (T_0,T],
\]

where \(u(0,t), u(a,t), u(x,T_0)\) and \(\lambda(u)\) are the given Dirichlet boundary conditions, initial condition and the scalar nonlinear diffusion coefficient, respectively.
We consider a uniform partition of \([0, a] \times [T_0, T]\) with spatial step \(h = \frac{a}{N}\) and time step \(\Delta t = \frac{T - T_0}{N}\). Let \(x_i = ih, t_n = T_0 + n\Delta t\), and denote by \(x_{i - \frac{1}{2}}\) the midpoint (cell center) of cell \([x_{i-1}, x_i]\). The flux is defined by \(F = -\lambda(u) \frac{\partial u}{\partial x}\). Besides, \(u^n_i\) (resp. \(u^n_{i - \frac{1}{2}}\)) is the approximation of \(u\) at \((x_i, t_n)\) (resp. \((x_{i - \frac{1}{2}}, t_n)\)).

### 2.1. A cell-centered scheme

We refer to [34, Section 2.1] and give a cell-centered scheme of (2.1) as following:

\[
\frac{h}{\Delta t} \left( u^n_{i - \frac{1}{2}} - u^{n-1}_{i - \frac{1}{2}} \right) + F^n_{i,l} - F^n_{i-1,r} = \int_{x_{i-1}}^{x_i} f \, dx, \tag{2.2}
\]

where \(F^n_{i,l}\) (resp. \(F^n_{i-1,r}\)) denotes the approximation of \(F\) at \((x_i, t_n)\) (resp. \((x_{i-1}, t_n)\)) from the left (resp. right) side of \(x_i\) (resp. \(x_{i-1}\)), given by

\[
F^n_{i,l} = -\frac{1}{h} \lambda^n_i \left( u^n_{i+\frac{1}{2}} - u^n_{i-\frac{1}{2}} \right), \quad F^n_{i-1,r} = -\frac{1}{h} \lambda^n_{i-1} \left( u^n_{i-\frac{1}{2}} - u^n_{i-\frac{3}{2}} \right)
\]

with

\[
\lambda^n_i = \frac{2\lambda \left( u^n_{i+\frac{1}{2}} \right) \lambda \left( u^n_{i-\frac{1}{2}} \right) \lambda \left( u^n_{i+\frac{1}{2}} \right) \lambda \left( u^n_{i-\frac{1}{2}} \right)}{\lambda \left( u^n_{i+\frac{1}{2}} \right) + \lambda \left( u^n_{i-\frac{1}{2}} \right)}, \quad 1 \leq i \leq m - 1. \tag{2.3}
\]

It is easy to see that \(\lambda^n_i\) is the harmonic averaging of the two cell-centered diffusion coefficients, which is responsible for the so-called numerical heat-barrier issue [22,34]. We remark that the original NPS reduces to the above scheme in 1D case.

### 2.2. An improved cell-centered scheme

Now consider a different cell-centered scheme for (2.1) by redefining the diffusion coefficient \(\lambda^n_i = \lambda(u^n_i)\) at \(x_i\). Besides, we construct two one-sided fluxes from both sides of \(x_i\), given by

\[
F^n_{i,l} = -\frac{2}{h} \lambda^n_i \left( u^n_i - u^n_{i-\frac{1}{2}} \right), \quad F^n_{i,r} = -\frac{2}{h} \lambda^n_i \left( u^n_{i+\frac{1}{2}} - u^n_i \right). \tag{2.4}
\]

By imposing the flux continuity at \(x_i\), i.e., \(F^n_{i,l} = F^n_{i,r}\), we have

\[
u^n_i = \frac{1}{2} \left( u^n_{i-\frac{1}{2}} + u^n_{i+\frac{1}{2}} \right), \tag{2.5}
\]

and further

\[
\lambda^n_i = \lambda(u^n_i) = \lambda \left( \frac{1}{2} \left( u^n_{i-\frac{1}{2}} + u^n_{i+\frac{1}{2}} \right) \right). \tag{2.6}
\]

By (2.5) and (2.4), we have

\[
F^n_{i,l} = F^n_{i,r} = -\frac{1}{h} \lambda^n_i \left( u^n_{i+\frac{1}{2}} - u^n_{i-\frac{1}{2}} \right). \tag{2.7}
\]
where $\lambda^n_i$ is defined by (2.6). Substituting (2.7) into (2.2) yields an improved cell-centered scheme. A discretization of diffusion coefficient similar to (2.6) can be found in [28].

### 2.3. A numerical example

We consider problem (2.1) with

$$a = 3, \quad \lambda(u) = u^3, \quad f = 0,$$

$$u(0, t) = \sqrt{3c^2 t + \varepsilon}, \quad u(3, t) = \sqrt{\varepsilon},$$

$$u(x, T_0) = \begin{cases} \sqrt{3c(cT_0 - x)}, & x < cT_0, \\ \sqrt{\varepsilon}, & \text{otherwise}, \end{cases}$$

where $c$ and $\varepsilon$ are two positive parameters. When $\varepsilon = 0$, problem (2.1) has the following analytic solution:

$$u(x, t) = \begin{cases} \sqrt{3c(ct - x)}, & x < ct, \\ 0, & \text{otherwise}. \end{cases} \tag{2.8}$$

We choose $c = 0.4$, $\varepsilon = 10^{-9}$, $h = \frac{1}{30}$ and $\Delta t = 0.4h^2$. When $T_0 = 0$, the corresponding example can be found in [22] and is investigated by [34] in the Section 2.3. Since $\varepsilon$ is small, (2.8) can approximately be used as the exact solution.

We first give tests for cell-centered scheme in Section 2.1. The profiles of numerical solutions and exact solutions for $T_0 = 0, 1.0, 2.0, 3.0$ and $T = 5.0$ are depicted in Fig. 1, where we can find that cell-centered scheme gives totally wrong solution profiles which are exactly blocked at the front end of initial conditions. Since

$$\frac{\lambda\left(u^n_{i+\frac{1}{2}}\right)}{\lambda\left(u^n_{i-\frac{1}{2}}\right)} \to 0$$

at the front end, we deduce from (2.3) that

$$\lambda^n_i \approx 2\lambda\left(u^n_{i+\frac{1}{2}}\right) \to 0.$$ 

It dedicates that the schemes based on the harmonic averaging of cell-centered diffusion coefficients will break down when some of these coefficients go to zero or their ratio grows in some strongly nonlinear problems.

The result of the improved cell-centered scheme in Section 2.2 is shown in Fig. 2. We can see that the modified scheme produces a quite good numerical solution. By this observation, we improve the classical NPS by applying an edge-based discretization of diffusion coefficient in the following section.
3. An improved cell-centered scheme for nonlinear diffusion problems

3.1. Problem and notations

We consider a diffusion problem on a bounded connected polygonal domain $\Omega \in \mathbb{R}^2$:

\[-\text{div}(\Lambda \nabla u) = f \quad \text{in } \Omega, \tag{3.1}\]
\[u = g_D \quad \text{on } \partial \Omega, \tag{3.2}\]
where \( f \) and \( g_D \) are given source term and Dirichlet boundary data, respectively. \( \Lambda \) is the \( 2 \times 2 \) diffusion tensor that is symmetric and positive definite (SPD), and possibly nonlinear with respect to \( u \). For simplicity of exposition, a pure Dirichlet boundary condition is considered throughout. The discussions for other types of boundary conditions are analogous.

The computational domain \( \Omega \) is partitioned into a number of non-overlapped regular polygonal cells. We denote the resulting partition by \( \mathcal{T}_h = \{ \mathcal{M}, \mathcal{P}, \mathcal{E}, \mathcal{V} \} \), where

- \( \mathcal{M} = \{ K \} \) is the set of cells, \( K \) denotes a generic cell whose center, diameter and measure are denoted as \( x_K, h_K \) and \( |K| \), respectively. Denote \( h = \max_{K \in \mathcal{M}} h_K \) as the mesh size of \( \mathcal{T}_h \). Each cell is assumed to be star-shaped with respect to its center in the sense that any ray starting from the cell center intersects the cell boundary only once;
- \( \mathcal{P} = \{ x_K, x_K \in K \} \) is the set of cell centers;
- \( \mathcal{E} = \{ \sigma \} \) is the set of edges, \( \sigma \) is a generic edge whose length is denoted by \( |\sigma| \), \( E_K \) denotes the set of edges of cell \( K \);
- \( \mathcal{V} = \{ x_\nu \} \) is the set of vertices;

Throughout this paper, we introduce the following notations:

- \( u_K \), the unknown at cell center \( x_K \);
- \( n_{K,\sigma} \), the outward unit normal of \( K \) with respect to \( \sigma \in \mathcal{E}_K \);
- \( \mathbf{F} = -\Lambda \nabla u \), the flux vector;
- \( F_{K,\sigma} \approx \int_\sigma \mathbf{F} \cdot n_{K,\sigma} \, ds \), the approximation of the normal component of \( \mathbf{F} \);
- \( \mathcal{R} \), the transformation rotating a vector clockwise to its normal direction, which means that \( \mathcal{R} \mathbf{v} = (b,-a)^T \) for a generic vector \( \mathbf{v} = (a,b)^T \).

### 3.2. Discretization of the flux

All the derivations in this paper are conducted under the following assumptions:

1. The solution and the diffusion coefficient are smooth inside each cell.
2. The possible discontinuities of solution gradient and diffusion coefficient are only allowed to appear on the edges.
3. The solution and the normal component of flux are continuous across all edges.

Consider the flux across \( \sigma \in \mathcal{E}_K \cap \mathcal{E}_L \), see Fig. 3. Let \( x_A, x_B, x_M \) be the two vertices and midpoint of edge \( \sigma \). Denote by \( u_K, u_L, u_A, u_B, u_M \) the unknowns defined at
\(x_K, x_L, x_A, x_B, x_M\), respectively. On each side of the edge \(\sigma\), we define the following approximation of the diffusion tensor

\[
\Lambda_{\alpha,\sigma} = \lim_{x \in \alpha, x \to x_M} \Lambda, \quad \alpha = K, L.
\]

(3.3)

If the diffusion tensor is piecewise constant, i.e., it is a constant tensor \(\Lambda_K\) (resp. \(\Lambda_L\)) on cell \(K\) (resp. \(L\)), then we have

\[
\Lambda_{\alpha,\sigma} = \Lambda_{\alpha}, \quad \sigma \in \mathcal{E}_\alpha, \quad \alpha = K, L.
\]

Moreover, when the diffusion coefficient is a nonlinear and continuous function of \(u\), we have

\[
\Lambda_{\alpha,\sigma} = \Lambda(u_M), \quad \alpha = K, L.
\]

(3.4)

In this case, we approximate \(u_M\) by \(\frac{1}{2}(u_A + u_B)\), where \(u_A\) and \(u_B\) can be obtained by the vertex interpolation algorithm given in Section 4. The similar discretization to (3.4) of the diffusion coefficient can be found in [30].

Now, we are ready to construct the discrete flux as follows,

\[
F_{\alpha,\sigma} = \int_{\sigma} (-\nabla u_I \cdot \mathbf{n}_{\alpha,\sigma}) \cdot \mathbf{n}_{\alpha,\sigma} \, ds, \quad \alpha = K, L,
\]

(3.5)

where \(u_I\) is the linear interpolation of \(u\) on triangle \(\triangle x_A x_B\). Note that \(|\sigma|n_{K,\sigma} = \mathcal{R}(x_A - x_B) = -|\sigma|n_{L,\sigma}\). By direct calculations, we have

\[
F_{\alpha,\sigma} = \xi_\alpha u_\alpha + \mu_\alpha u_A + \eta_\alpha u_B, \quad \alpha = K, L,
\]

(3.6)

where

\[
\xi_\alpha = -\frac{1}{2S_{\alpha,\sigma}} (\mathcal{R}(x_B - x_A))^T \Lambda_{\alpha,\sigma}^{T} \mathcal{R}(x_A - x_B),
\]

\[
\mu_\alpha = -\frac{1}{2S_{\alpha,\sigma}} (\mathcal{R}(x_\alpha - x_B))^T \Lambda_{\alpha,\sigma}^{T} \mathcal{R}(x_A - x_B),
\]

\[
\eta_\alpha = -\frac{1}{2S_{\alpha,\sigma}} (\mathcal{R}(x_A - x_\alpha))^T \Lambda_{\alpha,\sigma}^{T} \mathcal{R}(x_A - x_B),
\]
An Linearity-Preserving Cell-Centered Scheme for Nonlinear Diffusion Problems

\[ S_{K,\sigma} = \frac{1}{2} R (x_A - x_K) \cdot (x_B - x_K), \]
\[ S_{L,\sigma} = \frac{1}{2} R (x_B - x_L) \cdot (x_A - x_L). \]

To achieve the local conservation, we choose a unique expression of edge flux as following
\[ \tilde{F}_{K,\sigma} = \tilde{\xi}_K F_{K,\sigma} - \tilde{\xi}_L F_{L,\sigma}, \quad \tilde{F}_{L,\sigma} = \tilde{\xi}_L F_{L,\sigma} - \tilde{\xi}_K F_{K,\sigma}, \] (3.7)
where \( \tilde{\xi}_K \) and \( \tilde{\xi}_L \) are chosen to be
\[ \tilde{\xi}_K = \frac{\xi_L}{\xi_K + \xi_L}, \quad \tilde{\xi}_L = \frac{\xi_K}{\xi_K + \xi_L}. \] (3.8)
It is obvious that
\[ \tilde{F}_{K,\sigma} + \tilde{F}_{L,\sigma} = 0. \] (3.9)
Substituting (3.6) and (3.8) into the first equality of (3.7) and noticing
\[ \xi_\alpha = -(\mu_\alpha + \eta_\alpha), \quad \alpha = K, L, \]
we obtain the final discretization of edge flux:
\[ \tilde{F}_{K,\sigma} = -K_\sigma \left[ (u_L - u_K) - D_\sigma (u_A - u_B) \right], \] (3.10)
where
\[ K_\sigma = \frac{\xi_K \xi_L}{\xi_K + \xi_L}, \quad D_\sigma = \frac{\mu_L \eta_K - \mu_K \eta_L}{\xi_K \xi_L}. \] (3.11)
As for a Dirichlet boundary edge \( \sigma \), the discrete flux can be simply given by (3.6) directly, i.e., \( \tilde{F}_{K,\sigma} = F_{K,\sigma}, \sigma \in \partial \Omega \).

We point out that if \( \Lambda \) is piecewise constant with respect to the mesh, the scheme derived above degenerates to the classical NPS (see, e.g., (10) in [16]), otherwise, the scheme is a different one. It is obvious that the scheme we proposed above contains vertex unknowns which are treated as auxiliary unknowns and will be expressed as the linear combination of cell-centered ones. The interpolation algorithm of the vertex unknowns is given in Section 4.

4. Interpolation of the vertex unknowns

4.1. Introduction and notations

We consider the local mesh structure with \( n \) cells around a generic interior vertex \( x_\nu \) and introduce the following notations, some of which are shown in Fig. 4(a) (an example with \( n = 4 \)):

- \( u_\nu \), the unknown at \( x_\nu \);
Figure 4: The notations and local structure around an interior vertex.

- $K_i$, the cells that are ordered anticlockwise around $x_\nu$. The cell center and the corresponding cell-centered unknown are denoted as $x_{K_i}$ and $u_i$, respectively;
- $\sigma_i$, the common edge of $K_i$ and $K_{i-1}$, whose endpoints are $x_{\nu_i}$ and $x_{\nu}$;
- $\Lambda_i$, the value of $\Lambda$ at $x_{K_i}$;
- $n_i$, the unit outward normal of $K_i$ with respect to $\sigma_i$.

Throughout, $i$ will be understood as a periodic index given by mapping $i \mapsto ((i + n - 1) \mod n) + 1$, so that $x_{\nu_{n+1}} = x_{\nu_1}$, $u_0 = u_n$, etc. A vertex interpolation algorithm aims to find the following formula

$$u_\nu = \sum_{i=1}^{n} \omega_i u_i,$$  \hspace{1cm} (4.1)

where $\omega_i$ is so-called interpolation weight whose expression will be given in the later discussion.

### 4.2. An existing interpolation algorithm for edge unknowns

On the edge $\sigma_i$, we introduce an interior point $x_{\sigma_i}$ (see Fig. 4(b)), given by

$$x_{\sigma_i} = (1 - \tau_i)x_\nu + \tau_i x_{\nu_i}, \quad i = 1, \ldots, n,$$  \hspace{1cm} (4.2)

where $\tau_i \in (0,1)$ is a parameter. Let $\bar{u}_i$ denote the edge unknown at $x_{\sigma_i}$. In the well-known MPFA-O method [1], the edge unknowns are interpolated by the cell-centered unknowns. Since our interpolation algorithm for vertex unknowns is based on this existing algorithm, a brief introduction is required here. Let

$$\mathbf{U} = (\bar{u}_1, \ldots, \bar{u}_n)^T, \quad \mathbf{E}_i = (e_i, e_{i+1}),$$

$$\mathbf{U} = (u_1, \ldots, u_n)^T, \quad \mathbf{E}_+ = \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix},$$  \hspace{1cm} (4.3)
where \( e_i \) denotes the \( i \)-th column vector of the \( n \times n \) unit matrix. Obviously, \( E_i \) is an \( n \times 2 \) matrix and
\[
\begin{align*}
E_i^T \bar{U} &= \begin{pmatrix}
\bar{u}_i \\
\bar{u}_{i+1}
\end{pmatrix}, \\
E_i E_i^T U &= \begin{pmatrix}
u_i \\
u_i
\end{pmatrix}.
\end{align*}
\]
(4.4)

For the two discrete fluxes on edges \( \sigma_i \) and \( \sigma_{i+1} \) of cell \( K_i \), we have
\[
F_{K_i, \sigma_i} = \int_{\sigma_i} -\Lambda_i \nabla u_{I,i} \cdot n_i \, ds, \\
F_{K_i, \sigma_{i+1}} = \int_{\sigma_{i+1}} -\Lambda_i \nabla u_{I,i} \cdot (-n_{i+1}) \, ds,
\]
where \( u_{I,i} \) denotes the standard \( P_1 \) interpolation on \( \triangle x_{K_i} x_{\sigma_i} x_{\sigma_{i+1}} \) whose nodal values are \( u_i, \bar{u}_i \) and \( \bar{u}_{i+1} \), respectively. By direct calculations, we have
\[
\nabla u_{I,i} = \frac{1}{2S_i} ((\bar{u}_i - u_i) \mathcal{R}(x_{\sigma_{i+1}} - x_{K_i}) - (\bar{u}_{i+1} - u_i) \mathcal{R}(x_{\sigma_i} - x_{K_i})),
\]
where \( 2S_i = (x_{\sigma_{i+1}} - x_{K_i}) \cdot \mathcal{R}(x_{\sigma_i} - x_{K_i}) \). Noticing that \( |\sigma_i| n_i = \mathcal{R}(x_{\nu_i} - x_{\nu}) \) for \( i = 1, \ldots , n \), we obtain
\[
\begin{pmatrix}
F_{K_i, \sigma_i} \\
F_{K_i, \sigma_{i+1}}
\end{pmatrix} = A_i \begin{pmatrix}
\bar{u}_i - u_i \\
\bar{u}_{i+1} - u_i
\end{pmatrix},
\]
(4.5)
where
\[
A_i = \begin{pmatrix}
\alpha_{11}^{(i)} & \alpha_{12}^{(i)} \\
\alpha_{21}^{(i)} & \alpha_{22}^{(i)}
\end{pmatrix}
\]
(4.6)
and
\[
a_{jk}^{(i)} = \frac{(-1)^{j+k}}{2S_i} \left( A_i \mathcal{R}(x_{\sigma_{i+2-k}} - x_{K_i}) \right) \cdot \mathcal{R}(x_{\nu_{i+j-1}} - x_{\nu}), \quad j, k = 1, 2.
\]
(4.7)

We should keep in mind that \( a_{jk}^{(i)} \) is a function of \( \tau_i \). By (4.4), we rewrite (4.5) as
\[
\begin{pmatrix}
F_{K_i, \sigma_i} \\
F_{K_i, \sigma_{i+1}}
\end{pmatrix} = A_i \left( E_i^T \bar{U} - E_i E_i^T U \right).
\]
(4.8)

Imposing the flux continuity condition across the edge \( \sigma_{i+1} \), i.e.,
\[
F_{K_{i+1}, \sigma_{i+1}} + F_{K_i, \sigma_{i+1}} = 0, \quad i = 1, \ldots , n,
\]
(4.9)
or equivalently,
\[
\sum_{i=1}^n E_i \begin{pmatrix}
F_{K_i, \sigma_i} \\
F_{K_i, \sigma_{i+1}}
\end{pmatrix} = 0,
\]
we arrive at
\[
M \bar{U} = NU,
\]
(4.10)
where
\[
M = \sum_{i=1}^n E_i A_i E_i^T, \quad N = \sum_{i=1}^n E_i A_i E_i^T.
\]
(4.11)

In MPFA-O method, by solving (4.10), the edge unknowns can be expressed by the cell-centered unknowns. However, in constructing the new vertex interpolation algorithm, we just need the local system (4.10) and do not solve it, which will be seen clear in the later discussion.
4.3. A new interpolation algorithm for the vertex unknowns

From (4.2), it is easily to see that
\[
\lim_{\tau_i \to 0} x_{\sigma_i} = x_\nu, \quad \lim_{\tau_i \to 0} u_i = u_\nu, \quad i = 1, \ldots, n.
\]
Therefore, it is a natural idea to construct an interpolation algorithm for \( u_\nu \) by applying the above limit procedure to (4.10). To this end, we rewrite \( \tau_i \) in (4.2) as
\[
\tau_i = \bar{\tau}_i \tau
\]
with \( \bar{\tau}_i \in (0, +\infty) \) is a constant and \( \bar{\tau}_i \tau \in (0, 1) \), where \( \tau \) will be used in the limit procedure. For simplicity of exposition, we denote
\[
\bar{x}_{\nu_i} = x_{\nu_i} - x_\nu, \quad \bar{x}_{K_i} = x_{K_i} - x_\nu.
\]

We first figure out the explicit dependence of \( \tau \) for \( a_{j,k}^{(i)} \). By (4.2), we have
\[
x_{\sigma_{i+j}} - x_{K_i} = -\bar{x}_{K_i} + \bar{\tau}_{i+j} \tau \bar{x}_{\nu_{i+j}}, \quad j = 0, 1,
\]
and
\[
2S_i = \tau^2 \bar{\tau}_{i+1} (\mathcal{R} \bar{x}_{K_i}) \cdot \bar{x}_{\nu_{i+1}} + 2\tau(S_{i,1} + S_{i,2}),
\]
where
\[
S_{i,1} = -\frac{1}{2} \bar{\tau}_{i+1} (\mathcal{R} \bar{x}_{K_i}) \cdot \bar{x}_{\nu_{i+1}}, \quad S_{i,2} = -\frac{1}{2} \bar{\tau}_i (\mathcal{R} \bar{x}_{\nu_i}) \cdot \bar{x}_{K_i}.
\]
Then we can rewrite (4.7) as
\[
a_{j,k}^{(i)} = \frac{1}{\tau} \left( a_{j,k}^{(i,1)} + \tau a_{j,k}^{(i,2)} \right), \quad j, k = 1, 2,
\]
where
\[
a_{j,k}^{(i,1)} = \frac{(-1)^{j+k+1} (\Lambda_i \mathcal{R} \bar{x}_{K_i}) \cdot \mathcal{R} \bar{x}_{\nu_{i+j-1}}}{\tau \bar{\tau}_{i+1} (\mathcal{R} \bar{x}_{\nu_i}) \cdot \bar{x}_{\nu_{i+1}} + 2(S_{i,1} + S_{i,2})},
\]
\[
a_{j,k}^{(i,2)} = \frac{(-1)^{j+k} \bar{\tau}_{i+2-k} (\Lambda_i \mathcal{R} \bar{x}_{\nu_{i+2-k}}) \cdot \mathcal{R} \bar{x}_{\nu_{i+j-1}}}{\tau \bar{\tau}_{i+1} (\mathcal{R} \bar{x}_{\nu_i}) \cdot \bar{x}_{\nu_{i+1}} + 2(S_{i,1} + S_{i,2})}.
\]

Note that \( a_{j,k}^{(i,1)}, a_{j,k}^{(i,2)} \) are functions of \( \tau, \bar{\tau}_1, \ldots, \bar{\tau}_n \). Hereafter, we will use notations \( (\tau), (\tau, \bar{\tau}_1, \ldots, \bar{\tau}_n), (\tau, \bar{\tau}_1, \ldots, \bar{\tau}_n) \) for any quantities associated with \( \tau, \bar{\tau}_1, \ldots, \bar{\tau}_n \) to indicate that they are the functions of the corresponding variables. These notations will be dropped whenever there is no ambiguity. Denote
\[
\mathcal{A}_{i,l} = \begin{pmatrix} a_{11}^{(i,l)} & a_{12}^{(i,l)} \\ a_{21}^{(i,l)} & a_{22}^{(i,l)} \end{pmatrix}, \quad l = 1, 2.
\]
It follows from (4.11) that
\[
\mathcal{M} = \frac{1}{\tau} (\mathcal{M}_1 + \tau \mathcal{M}_2), \quad \mathcal{N} = \frac{1}{\tau} (\mathcal{N}_1 + \tau \mathcal{N}_2),
\]
where
\[
\mathcal{M}_1 = \begin{pmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{pmatrix}, \quad \mathcal{M}_2 = \begin{pmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{pmatrix},
\]
\[
\mathcal{N}_1 = \begin{pmatrix} N_{11} & N_{12} \\ N_{21} & N_{22} \end{pmatrix}, \quad \mathcal{N}_2 = \begin{pmatrix} N_{11} & N_{12} \\ N_{21} & N_{22} \end{pmatrix}.
\]
where
\[ M_l = \sum_{i=1}^{n} E_i A_{i,l} E_i^T, \quad N_l = \sum_{i=1}^{n} E_i A_{i,l} E_+ E_i^T, \quad l = 1, 2. \]

Noticing that
\[ a_{j1}^{(1)} + a_{j2}^{(1)} = 0, \quad j = 1, 2, \]
we have
\[ M_1 = 0, \quad N_1 = O, \]
where \( O \) is a square matrix with zero entries, and \( 1 \) (resp. \( 0 \)) is a vector with all entries equal to 1 (resp. 0). Noticing that \( E_i^T 1 = E_+ E_i^T 1 = (1, 1)^T \), from (4.11) we have \( M_1 = N_1 \). Then, (4.10) implies that
\[ M(U - u_\nu 1) = N(U - u_\nu 1). \]

It follows from (4.17) and (4.19) that
\[ (M_1 + \tau M_2)(U - u_\nu 1) = \tau N_2(U - u_\nu 1). \]

Note that the solution is continuous and differentiable along the cell edges \( \sigma_i \) (\( 1 \leq i \leq n \)). By Taylor expansion, we have
\[ U - u_\nu 1 = \tau \Gamma + O(\tau^2), \]
where \( \Gamma \) is a constant vector independent of \( \tau \) and \( O(\tau^2) \) denotes a vector with all components proportional to \( O(\tau^2) \). Substituting (4.21) into (4.20), we get
\[ \tau M_1 \Gamma + \tau^2 M_2 \Gamma + (M_1 + \tau M_2) O(\tau^2) = \tau N_2(U - u_\nu 1). \]

Now, dividing both sides of the above equation by \( \tau \) and then letting \( \tau \to 0 \), we reach
\[ M_{10} \Gamma = N_{20}(U - u_\nu 1), \]
where \( M_{10} \) (resp. \( N_{20} \)) denotes \( M_1 \) (resp. \( N_2 \)) with \( \tau = 0 \). Thus,
\[ M_{10} = \sum_{i=1}^{n} E_i A_{i,1}(0) E_i^T, \quad N_{20} = \sum_{i=1}^{n} E_i A_{i,2}(0) E_+ E_i^T, \]
where \( A_{i,1}(0), A_{i,2}(0) \) are given by (4.15) and (4.16) with \( \tau = 0 \). Specifically, the entries of \( A_{i,1}(0) \) and \( A_{i,2}(0) \) are given by
\[ a_{j1}^{(1)}(0) = \frac{(-1)^{j+k+1} (\Lambda_i \mathcal{R} \bar{x}_{K_i}) \cdot \mathcal{R} \bar{x}_{\nu_{i+1}},}{2(S_{i,1} + S_{i,2})}, \]
\[ a_{j1}^{(2)}(0) = \frac{(-1)^{j+k} (\Lambda_i \mathcal{R} \bar{x}_{\nu_{i+2}},) \cdot \mathcal{R} \bar{x}_{\nu_{i+1}}}{2(S_{i,1} + S_{i,2})}. \]
Now, denote the $i$-th column vector of $M_{10}$ by $m_i$ and define
\[
\tilde{M} = (m_1, m_2, \ldots, m_{n-1}, N_{20}1),
\] (4.26)
which means that $\tilde{M}$ is the $n \times n$ matrix obtained from $M_{10}$ by replacing its last column with $N_{20}1$ and keeping the rest columns unchanged. By multiplying $(0, \ldots, 0, 1)\tilde{M}^{-1}$ on both sides of (4.26), we can simply get
\[
(0, \ldots, 0, 1)\tilde{M}^{-1}(m_1, m_2, \ldots, m_{n-1}) = 0_{n-1}^T,
\] (4.27)
and
\[
(0, \ldots, 0, 1)\tilde{M}^{-1}N_{20}1 = 1.
\] (4.28)
Set
\[
(\omega_1, \omega_2, \ldots, \omega_n) = (0, \ldots, 0, 1)\tilde{M}^{-1}N_{20}.
\] (4.29)
Then (4.28) implies that
\[
\sum_{i=1}^{n} \omega_i = 1,
\]
while (4.27) and the first equality in (4.19) lead to
\[
(0, \ldots, 0, 1)\tilde{M}^{-1}M_{10} = 0^T.
\] Combing these results with (4.29) and (4.22), we finally reach
\[
\sum_{i=1}^{n} \omega_i u_i - u_\nu = (\omega_1, \omega_2, \ldots, \omega_n)(U - u_\nu 1) = (0, \ldots, 0, 1)\tilde{M}^{-1}M_{10}1 = 0,
\]
which leads to a new interpolation algorithm given by (4.1) and (4.29).

**Remark 4.1.** An approximate interpolation algorithm for the vertex unknowns can be directly obtained by choosing a small $\tau_i$ in (4.10) and averaging the resulting weights for the edge unknowns. In this alternative approach, $\tau_i = 10^{-8}$ seems to be a very good choice for the computations in double precision.

**4.4. The comparison with existing algorithms**

Since the vertex interpolation algorithms in [35] (WDGY10 for short), [37] (ZK17 for short) and the limit weight (LW for short) given by (4.29) are all based on the MPFA-O method, we give a simple comparison among them in this section. A brief derivation of algorithms WDGY10 and ZK17 will be given first.

Using the notations in Sections 4.1 and 4.2, we can get the following formula by extrapolation
\[
u^{(i)}_\nu = u_i \lambda_i^{(1)}(x_\nu) + \bar{u}_i \lambda_i^{(2)}(x_\nu) + \bar{u}_{i+1} \lambda_i^{(3)}(x_\nu), \quad i = 1, \ldots, n,
\] (4.30)
where $\lambda_i^{(k)}(k = 1, 2, 3)$ are the area coordinates on triangle $x_K, x_{\sigma_i}, x_{\sigma_{i+1}}$ corresponding to $x_K, x_{\sigma_i}, x_{\sigma_{i+1}}$, respectively. Suppose that $M$ is invertible, by solving (4.10) and
substituting $\mathbf{U} = \mathbf{M}^{-1} \mathbf{N} \mathbf{U}$ into (4.30), we can express $u^{(k)}_\nu (k = 1, \ldots, n)$ in terms of $u_i$ as follows

$$u^{(k)}_\nu = \sum_{i=1}^{n} \omega^{(k)}_i u_i, \quad k = 1, \ldots, n. \quad (4.31)$$

Then choosing $\omega_i = \omega^{(k_0)}_i (i = 1, \ldots, n)$ according to $\sum_{i=1}^{n} |\omega^{(k)}_i|$ and the variance of $\omega^{(k)}_i (i = 1, \ldots, n)$ (see (3.17) and (3.18) in [35]) gives the rise of the algorithm WDGY10. The algorithm ZK17 simply approximates $u_\nu$ by $\frac{1}{n} \sum_{k=1}^{n} u^{(k)}_\nu$ and gives the weight in the following

$$\omega_i = \sum_{k=1}^{n} \frac{\omega^{(k)}_i}{n}, \quad i = 1, \ldots, n. \quad (4.32)$$

We list the key differences between WDGY10, ZK17 and LW below:

- Both the algorithms WDGY10, ZK17 and LW have to solve a local linear system to get the weight. WDGY10 and ZK17 solve the linear system (4.10) from the MPFA-O procedure. However, it is totally unnecessary for LW to solve the linear system of MPFA-O. The corresponding coefficient matrix $\tilde{\mathbf{M}}$ of the linear system of LW which is defined by (4.26) is different from the one of the MPFA-O procedure (see $\mathbf{M}$ in (4.11)) in general cases. Thus the invertibility of $\tilde{\mathbf{M}}$ requires new discussion which is presented in the following section;

- WDGY10 and ZK17 have to employ extrapolation technology to approximate the value of the interpolation vertex, but this procedure is redundant for LW;

- Suppose that $\tau_i = \tau$ (resp. $\bar{\tau}_i = \tau$) for WDGY10 and ZK17 (resp. LW), where $\tau$ is a constant, then WDGY10 and ZK17 lead to different weight with different $\tau$ for general vertex (see Examples 6.3 and 6.4 in Section 6 of [35] for WDGY10). However, the weight given by LW is independent of $\tau$, which can be simply verified from the expression (4.29) of the weight;

- For isotropic homogeneous diffusion problems on square meshes, WDGY10 produces weight $(0, \frac{1}{2}, 0, \frac{1}{2})$, whereas LW and ZK17 produce ($\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}$).

We conclude that LW is a totally different vertex interpolation algorithm from WDGY10 and ZK17 in general cases. Moreover, the numerical performances of LW, WDGY10 and ZK17 are quite different on highly distorted meshes and the results are shown in Test 3 of Section 6.

5. Invertibility of $\tilde{\mathbf{M}}$ defined in (4.26)

It is obvious that the invertibility of $\tilde{\mathbf{M}}$ plays an important role in the new interpolation algorithm defined by (4.1) and (4.29). For specific $\tau_i$, say, $\tau_i = 0.5$, the solvability of the local linear system (4.10) in MPFA-O method is still open theoretically. If all $\tilde{\tau}_i$
in (4.12) are chosen the same value and \( \tau \) is treated as a variable, it has been proved in [35] that (4.10) can always be solved by selecting a proper \( \tau \). The most interesting part of this paper is that \( \bar{\tau}_i \) can take different values so that the invertibility of \( \bar{M} \) can be proved theoretically under certain assumptions. In this section, we shall first given an algorithm for \( \bar{\tau}_i \) and then prove the invertibility of \( \bar{M} \).

For simplicity, denote by \( N_n \) the integer set \( \{1, \ldots, n\} \) and define

\[
N_n^+ = \{ i \in N_n : (\Lambda_i \cdot R \bar{x}_{\nu_i}) \cdot R \bar{x}_{\nu_{i+1}} > 0 \},
\]

and \( m = \text{Card}(N_n^+) \) is the number of elements in \( N_n^+ \). In addition, denote

\[
\begin{align*}
r_i &= \frac{(\Lambda_i \cdot R \bar{x}_{\nu_i}) \cdot R \bar{x}_{\nu_{i+1}}}{(\Lambda_i \cdot R \bar{x}_{\nu_{i+1}}) \cdot R \bar{x}_{\nu_i}}, & s_i &= \frac{(\Lambda_i \cdot R \bar{x}_{\nu_i}) \cdot R \bar{x}_{\nu_{i+1}}}{(\Lambda_i \cdot R \bar{x}_{\nu_{i+1}}) \cdot R \bar{x}_{\nu_i}}, \quad i \in N_n^+, \\
\end{align*}
\]

and when \( m = n \) let

\[
\bar{r}_i = \prod_{j=i+1}^n r_j, \quad \bar{s}_i = \prod_{j=i+1}^n s_j, \quad i = 0, \ldots, n-1.
\]

We propose the algorithm for \( \bar{\tau}_i \) in two cases.

**Algorithm 5.1**

1. **Case** \( m < n \). Without loss of generality, we assume that \( (\Lambda_n \cdot R \bar{x}_{\nu_n}) \cdot R \bar{x}_{\nu_1} \leq 0 \). In this case, \( \bar{\tau}_i \) is given by the following recursive formula

\[
\bar{\tau}_1 = 1, \quad \bar{\tau}_{i+1} = \bar{\tau}_i c_i, \quad i = 1, \ldots, n-1,
\]

where

\[
c_i = \begin{cases} 
\frac{1}{2} (r_i + s_i), & i \in N_n^+, \\
1, & i \in N_n \setminus N_n^+.
\end{cases}
\]

2. **Case** \( m = n \). In this case, we suggest that

\[
\bar{\tau}_1 = 1, \quad \bar{\tau}_{i+1} = \frac{1}{2} \max \left\{ r_i \bar{\tau}_i, \frac{1}{s_i} \right\} + \frac{1}{2} \min \left\{ s_i \bar{\tau}_i, \frac{1}{r_i} \right\}, \quad i = 1, \ldots, n-1.
\]

In the following, we prove the invertibility of \( \bar{M} \). First, introduce the following assumptions:

**H1** We have either \( m < n \) or

\[
\begin{cases} 
m = n, \\
\bar{r}_0 < 1 < \bar{s}_0.
\end{cases}
\]

**H2** We have

\[
(\Lambda_i \cdot R \bar{x}_{\nu_i}) \cdot R \bar{x}_{\nu_{i+j-1}} > 0, \quad i = 1, \ldots, n, \quad j = 1, 2.
\]
Lemma 5.1. Assume that $0 < a_i < b_i$ ($i = 1, 2$) and $a_1 a_2 < c < b_1 b_2$. Let

$$\bar{c} = \frac{1}{2} \max \left\{ a_1, \frac{c}{b_2} \right\} + \frac{1}{2} \min \left\{ b_1, \frac{c}{a_2} \right\}.$$ 

Then, we have

$$a_1 < \bar{c} < b_1, \quad a_2 < \frac{c}{\bar{c}} < b_2.$$ 

Proof. Since $a_1 a_2 < c < b_1 b_2$ and $0 < a_i < b_i$ ($i = 1, 2$), we have

$$a_1 < \frac{c}{a_2}, \quad \frac{c}{b_2} < b_1, \quad \frac{c}{b_2} < \frac{c}{a_2},$$

which implies that

$$\max \left\{ a_1, \frac{c}{b_2} \right\} < \min \left\{ b_1, \frac{c}{a_2} \right\}.$$ 

As a result,

$$\max \left\{ a_1, \frac{c}{b_2} \right\} < \bar{c} < \min \left\{ b_1, \frac{c}{a_2} \right\},$$

which leads to the desired inequalities and completes the proof.

Lemma 5.2. Under assumption (H1), we have

$$\Lambda_i \mathcal{R} \bar{x}_{\nu_i} \cdot (\mathcal{R} \bar{x}_{\nu_{i+1}} - \tau_i \mathcal{R} \bar{x}_{\nu_i}) < 0, \quad (5.7)$$

$$\Lambda_i \mathcal{R} \bar{x}_{\nu_{i+1}} \cdot (\mathcal{R} \bar{x}_{\nu_i} - \tau_{i+1} \mathcal{R} \bar{x}_{\nu_{i+1}}) < 0, \quad (5.8)$$

where $\tau_i$ is given by Algorithm 5.1, $i \in \mathcal{N}_n$ and $\tau_{n+1} = \bar{\tau}$.

Proof. Since $\Lambda_i$ is SPD and $\bar{x}_{\nu_i} \neq 0$ for star-shaped meshes, we have

$$(\Lambda_i \mathcal{R} \bar{x}_{\nu_i}) \cdot \mathcal{R} \bar{x}_{\nu_{i+1}} = (\Lambda_i \mathcal{R} \bar{x}_{\nu_{i+1}} \cdot \mathcal{R} \bar{x}_{\nu_i} \cdot (\Lambda_i \mathcal{R} \bar{x}_{\nu_i}) \cdot \mathcal{R} \bar{x}_{\nu_{i+1}} > 0, \quad i \in \mathcal{N}_n. \quad (5.9)$$

Noting that $\bar{x}_{\nu_i}$ and $\bar{x}_{\nu_{i+1}}$ are not collinear and by Schwarz' inequality, we have

$$0 < r_i = \frac{(\Lambda_i \mathcal{R} \bar{x}_{\nu_{i+1}} \cdot \mathcal{R} \bar{x}_{\nu_i})}{(\Lambda_i \mathcal{R} \bar{x}_{\nu_{i+1}} \cdot \mathcal{R} \bar{x}_{\nu_{i+1}}) < \frac{(\Lambda_i \mathcal{R} \bar{x}_{\nu_i}) \cdot \mathcal{R} \bar{x}_{\nu_{i+1}}}{(\Lambda_i \mathcal{R} \bar{x}_{\nu_i}) \cdot \mathcal{R} \bar{x}_{\nu_{i+1}}} = s_i, \quad i \in \mathcal{N}_n^+. \quad (5.10)$$

In the following discussion, we verify the two cases, i.e., $m < n$ and $m = n$, one by one. For $m < n$, the proof is obvious if $i \in \mathcal{N}_n \setminus \mathcal{N}_n^+$. When $i \in \mathcal{N}_n^+$, recalling Algorithm 5.1 and (5.10), we have

$$r_i < \frac{1}{2} (r_i + s_i) = \frac{\tau_{i+1}}{\tau_i} < s_i,$$

which proves (5.7) and (5.8).

As for the case $m = n$, (5.7) and (5.8) hold if and only if

$$r_i < \frac{\tau_{i+1}}{\tau_i} < s_i, \quad i = 1, \ldots, n.$$
To prove the above inequalities, it suffices to prove that
\[ r_i < \frac{\bar{\tau}_{i+1}}{\bar{\tau}_i} < s_i, \quad \bar{r}_i < \frac{1}{\bar{\tau}_{i+1}} < \bar{s}_i, \quad i = 1, \ldots, n - 1, \tag{5.11} \]
since the second inequality of (5.11) with \( i = n - 1 \) implies that
\[ r_n = \bar{r}_{n-1} < \frac{1}{\bar{\tau}_n} \leq \frac{\bar{\tau}_{n+1}}{\bar{\tau}_n} < \bar{s}_{n-1} = s_n. \]

We claim that (5.11) can be proved by the mathematical induction. In fact, by (5.3) and (5.10), we have
\[ 0 < \bar{r}_i < \bar{s}_i, \quad i = 1, \ldots, n - 1. \]
From (5.6), we obtain \( r_1 \bar{r}_1 < 1 < s_1 \bar{s}_1 \). Since \( \bar{\tau}_1 = 1 \), we have
\[ \bar{\tau}_2 = \frac{1}{2} \max \left\{ r_1, \frac{1}{s_1} \right\} + \frac{1}{2} \min \left\{ s_1, \frac{1}{r_1} \right\}. \]
Then, by Lemma 5.1, we get
\[ r_1 < \bar{\tau}_2 < s_1, \quad \bar{r}_1 < \frac{1}{\bar{\tau}_2} < \bar{s}_1. \]
Assume that
\[ r_i < \frac{\bar{\tau}_{i+1}}{\bar{\tau}_i} < s_i, \quad \bar{r}_i < \frac{1}{\bar{\tau}_{i+1}} < \bar{s}_i. \]
It follows that
\[ 0 < r_{i+1} \bar{r}_{i+1} = \bar{r}_i < \frac{1}{\bar{\tau}_{i+1}} < \bar{s}_i = s_{i+1} \bar{s}_{i+1}. \]
Recalling that
\[ \frac{\bar{\tau}_{i+2}}{\bar{\tau}_{i+1}} = \frac{1}{2} \max \left\{ r_{i+1}, \frac{1}{\bar{\tau}_{i+1} \bar{s}_{i+1}} \right\} + \frac{1}{2} \min \left\{ s_{i+1}, \frac{1}{\bar{\tau}_{i+1} \bar{r}_{i+1}} \right\} \]
and using Lemma 5.1 once again, we get
\[ r_{i+1} < \frac{\bar{\tau}_{i+2}}{\bar{\tau}_{i+1}} < s_{i+1}, \quad \bar{r}_{i+1} < \frac{1}{\bar{\tau}_{i+2}} < \bar{s}_{i+1}, \]
which proves (5.11) and completes the proof. \( \square \)

**Lemma 5.3.** Under assumption (H1), all entries of \( \mathbb{N}_{20} \mathbf{1} \) are negative with \( \bar{\tau}_i \) given by Algorithm 5.1, i.e.,
\[ e_i^T \mathbb{N}_{20} \mathbf{1} < 0, \quad i = 1, \ldots, n, \tag{5.12} \]
where \( \mathbb{N}_{20} \) is defined in (4.22).
Proof. By the definition of $\mathbb{N}_{20}$, we have

\[
e^{T}_{i}N_{20}1 = \sum_{i' = 1}^{n} e^{T}_{i}E_{i'}A_{i',2}(0)E_{i'}^{T}1
\]

\[
= \sum_{i' = 1}^{n} (\delta_{i',i}, \delta_{i'+1,i}) \begin{pmatrix} a_{11}^{(i',2)}(0) & a_{12}^{(i',2)}(0) \\ a_{21}^{(i',2)}(0) & a_{22}^{(i',2)}(0) \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix}
\]

\[
= a_{11}^{(i,2)}(0) + a_{12}^{(i,2)}(0) + a_{21}^{(i-1,2)}(0) + a_{22}^{(i-1,2)}(0),
\]

where $\delta_{ij}$ denotes the Kronecker delta. By (4.25), we conclude further

\[
e^{T}_{i}N_{20}1 = (\Lambda_{i}R\bar{x}_{\nu_{i}+1} - \bar{x}_{\nu_{i}}) \cdot \frac{R(\bar{\tau}_{i} - 1 \bar{x}_{\nu_{i}})}{2(S_{i,1} + S_{i,2})} + (\Lambda_{i-1}R\bar{x}_{\nu_{i}}) \cdot \frac{R(\bar{\tau}_{i-1} \bar{x}_{\nu_{i-1}} - \bar{x}_{\nu_{i}})}{2(S_{i-1,1} + S_{i-1,2})}.
\]

Considering $S_{i,1}$ (resp. $S_{i,2}$) is the area of $\Delta x_{\nu_{i}}x_{K_{i}}$ (resp. $\Delta x_{\nu_{i+1}}x_{K_{i}}$) for star-shaped meshes and is consequently positive number, we obtain (5.12) immediately from Lemma 5.2.

Lemma 5.4. Let matrix $A = (a_{jk})_{n \times n}$ be defined by

\[
A = \sum_{i=1}^{n} E_{i} \begin{pmatrix} a_{i} & -a_{i} \\ -b_{i} & b_{i} \end{pmatrix} E_{i}^{T}, \tag{5.13}
\]

where $a_{i}, b_{i}(1 \leq i \leq n)$ are positive numbers and $E_{i}$ is defined in (4.3). Then,

\[
a_{*j}^{*} = a_{j,n}^{*}, \quad d a_{j}^{*} > 0, \quad 1 \leq j, \quad k \leq n, \tag{5.14}
\]

where $a_{jk}^{*}$ denotes the algebraic cofactor corresponding to $a_{jk}$.

Proof. From (5.13), we find that

\[
A1 = \sum_{i=1}^{n} E_{i} \begin{pmatrix} a_{i} & -a_{i} \\ -b_{i} & b_{i} \end{pmatrix} E_{i}^{T}1 = \sum_{i=1}^{n} E_{i} \begin{pmatrix} a_{i} & -a_{i} \\ -b_{i} & b_{i} \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = 0.
\]

By the definitions of $a_{jk}^{*}$ and $a^{*}_{j}n$, one can easily verify the first part of (5.14). What remains is the proof of $a_{j}^{*} > 0 (1 \leq j \leq n)$, and we first prove $a_{*n}^{*} > 0$. From the definition of $A$, we find that

\[
a_{*n}^{*} = \det A_{nn}, \quad A_{nn} = \sum_{i=1}^{n} \bar{I}_{(n-1)}E_{i} \begin{pmatrix} a_{i} & -a_{i} \\ -b_{i} & b_{i} \end{pmatrix} E_{i}^{T} \bar{I}_{(n-1)}^{T}.
\]

\[
\]
where $\mathbb{I}_{(n-1)}$ denotes the first $n - 1$ rows of $n \times n$ unit matrix. By straightforward calculations, we have

$$\det A_{nn} = \sum_{i=1}^{n-2} b_0 b_1 \ldots b_{i-1} a_{i+1} \ldots a_{n-1} + \prod_{i=0}^{n-2} b_i + \prod_{i=1}^{n-1} a_i,$$  \hspace{1cm} (5.16)

where $b_0 = b_n$. Thus, $a_{nn}^* > 0$ is verified immediately.

As for the proof of $a_{nn}^* > 0$ with $j \neq n$, we can simply verify that

$$a_{jj}^* = \det \left( \tilde{V}_j^T \mathbb{I}_{(n-1)} \tilde{V}_j^T \mathbb{A} \mathbb{V}_j \mathbb{I}_{(n-1)} \tilde{V}_j \right) = \det \left( \mathbb{I}_{(n-1)} \tilde{V}_j^T \mathbb{A} \mathbb{V}_j \mathbb{I}_{(n-1)} \tilde{V}_j \right),$$  \hspace{1cm} (5.17)

where

$$\mathbb{V}_j = \left( \begin{array}{cc} \mathbb{O} & \mathbb{I}_j \\ \mathbb{I}_{n-j} & \mathbb{O} \end{array} \right), \quad \tilde{\mathbb{V}}_j = \left( \begin{array}{cc} \mathbb{O} & \mathbb{I}_{n-j} \\ \mathbb{I}_{j-1} & \mathbb{O} \end{array} \right)$$

and $\mathbb{I}_k$ denotes the $k \times k$ identity matrix. Note that

$$\mathbb{V}_j^T \mathbb{A} \mathbb{V}_j = \sum_{i=1}^{n} \mathbb{V}_j^T \mathbb{E}_i \left( \begin{array}{cc} a_i & -a_i \\ -b_i & b_i \end{array} \right) \mathbb{E}_i^T \mathbb{V}_j$$

$$= \sum_{i=1}^{n} \mathbb{E}_{n+i-j} \left( \begin{array}{cc} a_i & -a_i \\ -b_i & b_i \end{array} \right) \mathbb{E}_{n+i-j}^T$$

$$= \sum_{i=1}^{n} \mathbb{E}_i \left( \begin{array}{cc} a_{i+j-n} & -a_{i+j-n} \\ -b_{i+j-n} & b_{i+j-n} \end{array} \right) \mathbb{E}_i^T.$$  \hspace{1cm} (5.19)

Here the indices $n + i - j$ and $i + j - n$ are understood in sense of periodic ones. Hence, the rest argument is essentially the same as that of $a_{nn}^* > 0$.

Now we are ready to present the main result of this section.

**Theorem 5.1.** Under assumptions (H1) and (H2), the coefficient matrix $\tilde{\mathbb{M}}$ defined by (4.26) is nonsingular if $\bar{\tau}_i$ is specified by Algorithm 5.1.

**Proof.** From (4.18) and (4.23), we have

$$\mathbb{M}_{10} = \sum_{i=1}^{n} \mathbb{E}_i \left( \begin{array}{cc} a_{11}^{(i,1)}(0) & -a_{11}^{(i,1)}(0) \\ -a_{22}^{(i,1)}(0) & a_{22}^{(i,1)}(0) \end{array} \right) \mathbb{E}_i^T.$$  \hspace{1cm} (5.18)

By (4.24) and (H2), we see that

$$a_{jj}^{(i,1)}(0) = \frac{(-\Lambda_i \mathbb{R} \mathbb{x}_K) \cdot \mathbb{R} \mathbb{x}_{\nu_{i+j-1}}}{2(S_{i,1} + S_{i,2})} < 0, \quad j = 1, 2, \quad i \in \mathcal{N}_n.$$  \hspace{1cm} (5.19)

Let $\Lambda = (\lambda_{ij}) = -\mathbb{M}_{10}$. We have $a_{ii}^* > 0 (1 \leq i \leq n)$ by Lemma 5.4. By the definition of $\tilde{\mathbb{M}}$, and making use of Lemma 5.3, we have

$$\text{Det}(-\tilde{\mathbb{M}}) = \sum_{i=1}^{n} \left( -\varepsilon_i^T \mathbb{N}_{20} \right) a_{ii}^* > 0,$$

which implies that $\tilde{\mathbb{M}}$ is nonsingular.  \hspace{1cm} \(\square\)
Note that the assumptions (H1) and (H2) are sufficient conditions for the existence of $\bar{\tau}_i$. In practical computation, we have never encountered a singular $\tilde{M}$ so far for general $\bar{\tau}_i$.

**Remark 5.1.** If $\Lambda_i = \lambda_i I_2$, where $\lambda_i$ is a positive constant, or alternatively, $\Lambda$ is a constant tensor, i.e., $\Lambda_i = \Lambda \ (i = 1, \ldots, n)$, then we have

$$\bar{r}_0 \bar{s}_0 = \prod_{i=1}^n \frac{\Lambda_i \bar{R} \bar{x}_{\nu_i+1} \cdot \bar{R} \bar{x}_{\nu_i}}{\Lambda_i \bar{R} \bar{x}_{\nu_i} \cdot \bar{R} \bar{x}_{\nu_i+1}} = 1.$$  

On the other hand, from (5.10), we find that

$$\bar{r}_0 = \prod_{i=1}^n \frac{\Lambda_i \bar{R} \bar{x}_{\nu_i+1} \cdot \bar{R} \bar{x}_{\nu_i}}{\Lambda_i \bar{R} \bar{x}_{\nu_i} \cdot \bar{R} \bar{x}_{\nu_i+1}} < \prod_{i=1}^n \frac{\Lambda_i \bar{R} \bar{x}_{\nu_i} \cdot \bar{R} \bar{x}_{\nu_i}}{\Lambda_i \bar{R} \bar{x}_{\nu_i} \cdot \bar{R} \bar{x}_{\nu_i+1}} = \bar{s}_0.$$  

Hence (H1) always holds in these cases.

**Remark 5.2.** In the case where the diffusion coefficient is a scalar, the geometric meaning of (H2) is that the two angles $\angle x_{\nu_i} x_{\nu_i} x_{\nu_i}$ and $\angle x_{\nu_i} x_{\nu_i+1}$ are acute ones.

### 6. Numerical experiments

In this section, we present some tests to investigate the performance of the improved finite volume scheme (INPS for short) in Section 3 with new linearity-preserving limit vertex interpolation (LW for short) in Section 4 on several mesh families. We also give the performance of INPS with LPEW2 (EW2 for short) given in [16] as a comparison in some tests. We shall employ $\tilde{\tau}_i = 1$ for simplicity in all tests.

Solution errors and edge normal flux errors are evaluated by the following expressions:

$$E_u(h) = \left( \sum_{K \in M} |K| \left( u(x_K) - u_K \right)^2 \right)^{\frac{1}{2}},$$

$$E_q(h) = \left( \sum_{K \in M} \sum_{\sigma \in E_K} \frac{1}{d_{\sigma}} \left( (u(x_K) - u(x_L)) - (u_K - u_L) \right)^2 \right)^{\frac{1}{2}},$$

where $d_{\sigma} = \sum_{K \in M, \sigma \in E_K} d_{K,\sigma}$ and $d_{K,\sigma}$ is the distance from $x_K$ to the line containing $\sigma$.

The rate of convergence $R_{\alpha}$ ($\alpha = u, q$) is obtained by the following formula

$$R_{\alpha} = \frac{\log \left[ E_{\alpha}(h_2)/E_{\alpha}(h_1) \right]}{\log (h_2/h_1)}, \quad \alpha = u, q,$$

(6.1)

where $h_i \ (i = 1, 2)$ denote the mesh sizes of the two successive meshes. All tests are performed in double precision, and we use BICGSTAB for solving linear systems with stopping tolerance 1.0E-15.
6.1. Test 1: Anisotropy problem

We consider the problem (3.1) and (3.2) on \( \Omega = [0,1]^2 \). The diffusion coefficient and exact solution are given by

\[
\Lambda = \begin{pmatrix} 1.5 & 0.5 \\ 0.5 & 1.5 \end{pmatrix}, \quad u(x, y) = 16x(1-x)y(1-y)\exp[-(x-0.5)^2-(y-0.5)^2],
\]

where the exact solution is located in the interval \([0,1]\). This numerical test is a similar one to \([19, \text{Test 1.1}]\). We apply a sequence of six mesh types (see Fig. 5) to this test.

The numerical results of INPS-LW are close to INPS-EW2 on Mesh1, Mesh3, Mesh5, Mesh6, and are given in Table 1, where optimal convergence rates, i.e., second order for solution errors and first order for flux errors can be seen. On Mesh2 and Mesh4, INPS-LW performs better than INPS-EW2 and the results of this two schemes are given in Tables 2 and 3. We observe that on both Mesh2 and Mesh4 INPS-LW has smaller
errors and higher convergence orders for solution and flux than INPS-EW2.

The numerical results of INPS-LW on all mesh types are graphically depicted in Fig. 6 as log-log plots of the discrete errors versus the characteristic mesh size $\frac{1}{h}$. 
Table 3: Test 1: computational results of INPS-EW2 and INPS-LW on Mesh4.

| Mesh level | INPS-EW2 | | INPS-LW | |
|------------|----------|-----------------|----------|---------------------|-----------------|
|            | $E_u$    | $R_u$ | $E_q$ | $R_q$ | $E_u$ | $R_u$ | $E_q$ | $R_q$ |
| 1          | 3.123E-02 | -     | 1.017E+00 | -      | 1.451E-02 | -     | 5.467E-01 | -      |
| 2          | 2.105E-02 | 0.98  | 7.550E-01 | 0.74   | 8.351E-03 | 1.37 | 3.461E-01 | 1.13   |
| 3          | 1.446E-02 | 1.31  | 5.523E-01 | 1.09   | 5.161E-03 | 1.68 | 2.247E-01 | 1.51   |
| 4          | 1.039E-02 | 1.48  | 4.146E-01 | 1.29   | 3.461E-03 | 1.79 | 1.549E-01 | 1.67   |
| 5          | 7.778E-03 | 1.59  | 3.201E-01 | 1.42   | 2.471E-03 | 1.85 | 1.126E-01 | 1.76   |

6.2. Test 2: Discontinuous problem

We deal with the problem (3.1) and (3.2) on $\Omega = [0,1]^2$. The diffusion coefficient and exact solution are given below:

$$\Lambda = \begin{cases} I_2, & x \leq 0.5, \\ kI_2, & x > 0.5, \end{cases}$$

$$u(x,y) = \begin{cases} 1 + x + y + (x - 0.5)^2 e^{x+y}, & x \leq 0.5, \\ \frac{3k-1}{2k} + \frac{x}{k} + y + (x - 0.5)^2 e^{x+y}, & x > 0.5 \end{cases}$$

with $k = 10^{-6}$ and $I_2$ is $2 \times 2$ unit matrix. Convergence investigations of the schemes INPS-EW2 and INPS-LW are conducted on Mesh1 and Mesh2. Since the mesh lines have to be aligned with the discontinuities of diffusion coefficients, the vertices on $x = 0.5$ are not disturbed for Mesh1. The numerical behaviors are shown in Tables 4 and 5. INPS-LW and INPS-EW2 have similar behaviors for this test. We can find that both solution and flux of INPS-LW achieve optimal convergence rate for discontinuous problem.
An Linearity-Preserving Cell-Centered Scheme for Nonlinear Diffusion Problems

Table 4: Test 2: behaviors of INPS-EW2 and INPS-LW on Mesh1.

<table>
<thead>
<tr>
<th>Mesh level</th>
<th>INPS-EW2</th>
<th></th>
<th>INPS-LW</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$E_u$</td>
<td>$R_u$</td>
<td>$E_q$</td>
<td>$R_q$</td>
</tr>
<tr>
<td>1</td>
<td>1.156E-02</td>
<td>2.087E-01</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>3.396E-03</td>
<td>1.78</td>
<td>1.036E-01</td>
<td>1.02</td>
</tr>
<tr>
<td>3</td>
<td>8.900E-04</td>
<td>2.24</td>
<td>4.394E-02</td>
<td>1.43</td>
</tr>
<tr>
<td>4</td>
<td>2.343E-04</td>
<td>1.97</td>
<td>2.292E-02</td>
<td>0.96</td>
</tr>
<tr>
<td>5</td>
<td>7.942E-05</td>
<td>1.57</td>
<td>1.053E-02</td>
<td>1.13</td>
</tr>
</tbody>
</table>

Table 5: Test 2: behaviors of INPS-EW2 and INPS-LW on Mesh2.

<table>
<thead>
<tr>
<th>Mesh level</th>
<th>NPS-LPEWII</th>
<th></th>
<th>NPS-LPLW</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$E_u$</td>
<td>$R_u$</td>
<td>$E_q$</td>
<td>$R_q$</td>
</tr>
<tr>
<td>1</td>
<td>1.770E-02</td>
<td>3.136E-01</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>4.884E-03</td>
<td>2.19</td>
<td>1.251E-01</td>
<td>1.56</td>
</tr>
<tr>
<td>3</td>
<td>1.369E-03</td>
<td>1.98</td>
<td>4.803E-02</td>
<td>1.49</td>
</tr>
<tr>
<td>4</td>
<td>3.760E-04</td>
<td>1.93</td>
<td>1.714E-02</td>
<td>1.54</td>
</tr>
<tr>
<td>5</td>
<td>9.868E-05</td>
<td>1.96</td>
<td>5.859E-03</td>
<td>1.58</td>
</tr>
</tbody>
</table>

6.3. Test 3: Poisson problem

We consider Poisson problem $-\Delta u = f$ on $\Omega = [0, 1]^2$ with exact solution given by

$$u(x, y) = \sin(\pi x) \sin(\pi y),$$

and the source $f$ is determined by $u$ accordingly.

We introduce the vertex interpolation algorithms WDGY10 and ZK17 in Section 4.4. Both the two and LW are based on the MPFA-O method and thus we illustrate the difference between them through this simple problem. The dynamic parameter $\tau_i$ (see Section 4.4 and (4.2)) is chosen to be 0.5 for both WDGY10 and ZK17 in this test.

We test the truncation errors of these three vertex interpolation algorithms and the performances of INPS combined with the corresponding vertex interpolation algorithm. Both the results of truncation error test and the performances of INPS employing these three algorithms are quite close on Mesh1, Mesh3, Mesh5 and Mesh6. We will only show the results on Mesh2 and Mesh4.

The following discrete norms are defined to test the truncation error first

$$E_{\text{max}}(h) = \max_{x_\nu \in \mathcal{V}^{\text{int}}} |e_\nu|, \quad E_{L_2}(h) = \left( \sum_{x_\nu \in \mathcal{V}^{\text{int}}} e_{x_\nu}^2 S_\nu \right)^{\frac{1}{2}},$$

where $\mathcal{V}^{\text{int}}$ is the set of internal vertices, $e_\nu$ is the difference between the value given by the corresponding interpolation algorithm and the exact value at $x_\nu$, and $S_\nu$ is the area of the dual cell formed by the surrounding cell centers and edge midpoints of vertex $x_\nu$. The rate of convergence $R_\alpha$ ($\alpha = \max, L_2$) can be obtained by (6.1) analogously.
The results of truncation error test are shown in Tables 6 and 7. We can see that LW has smaller errors than WDGY10 and ZK17 for both $E_{\text{max}}$ and $E_{L_2}$ on Mesh2. On Mesh4, LW has similar performance to WDGY10 and ZK17 for $E_{\text{max}}$ and better performance close to $O(h^{\frac{5}{2}})$ than WDGY10 and ZK17 for $E_{L_2}$.

We give the numerical results of INPS-LW, INPS-WDGY10 and INPS-ZK17 in Tables 8 and 9, where the mark “*” means that the linear iterative solver (here we use BICGSTAB) fails to converge for the final finite volume linear system in the corresponding tests. We can see that INPS-LW produces perfect numerical results for both cases. INPS-WDGY10 gets unreasonable result for flux at mesh level 2 and totally wrong result at mesh level 5 on Mesh2. The numerical results are all unreasonable at mesh level higher than 2 for INPS-ZK17 on Mesh2. The linear iterative solver even cannot con-

<table>
<thead>
<tr>
<th>Mesh Level</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>LW</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$E_{\text{max}}$</td>
<td>7.150E-02</td>
<td>1.890E-02</td>
<td>4.800E-03</td>
<td>1.207E-03</td>
<td>3.027E-04</td>
</tr>
<tr>
<td>$R_{\text{max}}$</td>
<td>2.26</td>
<td>2.13</td>
<td>3.06</td>
<td>2.03</td>
<td></td>
</tr>
<tr>
<td>$E_{L_2}$</td>
<td>2.200E-02</td>
<td>5.150E-03</td>
<td>1.272E-03</td>
<td>3.186E-04</td>
<td>7.991E-05</td>
</tr>
<tr>
<td>$R_{L_2}$</td>
<td>2.47</td>
<td>2.17</td>
<td>2.07</td>
<td>2.03</td>
<td></td>
</tr>
<tr>
<td>WDGY10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$E_{\text{max}}$</td>
<td>8.819E-02</td>
<td>3.302E-02</td>
<td>8.737E-03</td>
<td>2.284E-03</td>
<td>6.056E-04</td>
</tr>
<tr>
<td>$R_{\text{max}}$</td>
<td>1.67</td>
<td>2.06</td>
<td>2.03</td>
<td>1.95</td>
<td></td>
</tr>
<tr>
<td>$E_{L_2}$</td>
<td>3.181E-02</td>
<td>8.623E-03</td>
<td>2.260E-03</td>
<td>5.818E-04</td>
<td>1.477E-04</td>
</tr>
<tr>
<td>$R_{L_2}$</td>
<td>2.22</td>
<td>2.08</td>
<td>2.03</td>
<td>2.01</td>
<td></td>
</tr>
<tr>
<td>ZK17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$E_{\text{max}}$</td>
<td>1.205E-01</td>
<td>5.767E-02</td>
<td>1.561E-02</td>
<td>4.007E-03</td>
<td>1.007E-03</td>
</tr>
<tr>
<td>$R_{\text{max}}$</td>
<td>1.25</td>
<td>2.03</td>
<td>2.03</td>
<td>2.03</td>
<td></td>
</tr>
<tr>
<td>$E_{L_2}$</td>
<td>3.557E-02</td>
<td>1.052E-02</td>
<td>2.923E-03</td>
<td>7.672E-04</td>
<td>1.962E-04</td>
</tr>
<tr>
<td>$R_{L_2}$</td>
<td>2.07</td>
<td>1.99</td>
<td>2.00</td>
<td>2.00</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Mesh Level</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>LW</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$E_{\text{max}}$</td>
<td>1.560E-02</td>
<td>6.999E-03</td>
<td>3.946E-03</td>
<td>2.530E-03</td>
<td>1.758E-03</td>
</tr>
<tr>
<td>$R_{\text{max}}$</td>
<td>1.99</td>
<td>2.00</td>
<td>2.00</td>
<td>2.00</td>
<td></td>
</tr>
<tr>
<td>$E_{L_2}$</td>
<td>3.254E-03</td>
<td>1.991E-03</td>
<td>5.868E-04</td>
<td>3.401E-04</td>
<td>2.184E-04</td>
</tr>
<tr>
<td>$R_{L_2}$</td>
<td>2.50</td>
<td>2.47</td>
<td>2.45</td>
<td>2.44</td>
<td></td>
</tr>
<tr>
<td>WDGY10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$E_{\text{max}}$</td>
<td>1.548E-02</td>
<td>6.966E-03</td>
<td>3.930E-03</td>
<td>2.521E-03</td>
<td>1.752E-03</td>
</tr>
<tr>
<td>$R_{\text{max}}$</td>
<td>1.99</td>
<td>2.00</td>
<td>2.00</td>
<td>2.00</td>
<td></td>
</tr>
<tr>
<td>$E_{L_2}$</td>
<td>3.661E-03</td>
<td>1.472E-03</td>
<td>7.812E-04</td>
<td>4.819E-04</td>
<td>3.260E-04</td>
</tr>
<tr>
<td>$R_{L_2}$</td>
<td>2.26</td>
<td>2.21</td>
<td>2.17</td>
<td>2.15</td>
<td></td>
</tr>
<tr>
<td>ZK17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$E_{\text{max}}$</td>
<td>1.764E-02</td>
<td>8.096E-03</td>
<td>4.598E-03</td>
<td>2.958E-03</td>
<td>2.058E-03</td>
</tr>
<tr>
<td>$R_{\text{max}}$</td>
<td>1.93</td>
<td>1.98</td>
<td>1.98</td>
<td>1.99</td>
<td></td>
</tr>
<tr>
<td>$E_{L_2}$</td>
<td>5.114E-03</td>
<td>2.346E-03</td>
<td>1.340E-03</td>
<td>8.657E-04</td>
<td>6.048E-04</td>
</tr>
<tr>
<td>$R_{L_2}$</td>
<td>1.93</td>
<td>1.96</td>
<td>1.96</td>
<td>1.97</td>
<td></td>
</tr>
</tbody>
</table>
Table 8: Test 3: computational results of INPS-LW, INPS-WDG10 and INPS-ZK17 on Mesh2.

<table>
<thead>
<tr>
<th>Mesh Level</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>INPS-LW</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$E_u$</td>
<td>3.77E-02</td>
<td>1.40E-02</td>
<td>3.98E-03</td>
<td>1.00E-03</td>
<td>2.50E-04</td>
</tr>
<tr>
<td>$R_u$</td>
<td>-</td>
<td>1.68</td>
<td>1.96</td>
<td>2.06</td>
<td>2.04</td>
</tr>
<tr>
<td>$E_q$</td>
<td>8.53E-01</td>
<td>3.94E-01</td>
<td>1.40E-01</td>
<td>4.57E-02</td>
<td>1.52E-02</td>
</tr>
<tr>
<td>$R_q$</td>
<td>-</td>
<td>1.32</td>
<td>1.61</td>
<td>1.67</td>
<td>1.61</td>
</tr>
<tr>
<td>INPS-WDG10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$E_u$</td>
<td>3.72E-02</td>
<td>2.04E-02</td>
<td>9.74E-03</td>
<td>2.24E-03</td>
<td>1.1E+139</td>
</tr>
<tr>
<td>$R_u$</td>
<td>-</td>
<td>1.02</td>
<td>1.14</td>
<td>2.19</td>
<td>-478.71</td>
</tr>
<tr>
<td>$E_q$</td>
<td>8.03E-01</td>
<td>1.41E+00</td>
<td>6.30E-01</td>
<td>1.30E-01</td>
<td>1.84E+142</td>
</tr>
<tr>
<td>$R_q$</td>
<td>-</td>
<td>-0.95</td>
<td>1.25</td>
<td>2.36</td>
<td>-483.66</td>
</tr>
<tr>
<td>INPS-ZK17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$E_u$</td>
<td>4.44E-02</td>
<td>6.72E-02</td>
<td>1.42E-02</td>
<td>8.98E+143</td>
<td>6.63E-01</td>
</tr>
<tr>
<td>$R_u$</td>
<td>-</td>
<td>-0.71</td>
<td>2.41</td>
<td>-501.46</td>
<td>486.97</td>
</tr>
<tr>
<td>$E_q$</td>
<td>9.90E-01</td>
<td>2.84</td>
<td>1.167</td>
<td>2.71E+146</td>
<td>3.40E+01</td>
</tr>
<tr>
<td>$R_q$</td>
<td>-</td>
<td>-1.79</td>
<td>1.38</td>
<td>-503.41</td>
<td>489.57</td>
</tr>
</tbody>
</table>

Table 9: Test 3: computational results of INPS-LW, INPS-WDG10 and INPS-ZK17 on Mesh4.

<table>
<thead>
<tr>
<th>Mesh Level</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>INPS-LW</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$E_u$</td>
<td>1.393E-02</td>
<td>7.988E-03</td>
<td>4.938E-03</td>
<td>3.314E-03</td>
<td>2.368E-03</td>
</tr>
<tr>
<td>$R_u$</td>
<td>-</td>
<td>1.38</td>
<td>1.68</td>
<td>1.79</td>
<td>1.85</td>
</tr>
<tr>
<td>$E_q$</td>
<td>5.244E-01</td>
<td>3.306E-01</td>
<td>2.145E-01</td>
<td>1.480E-01</td>
<td>1.075E-01</td>
</tr>
<tr>
<td>$R_q$</td>
<td>-</td>
<td>1.14</td>
<td>1.51</td>
<td>1.67</td>
<td>1.75</td>
</tr>
<tr>
<td>INPS-WDG10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$E_u$</td>
<td>2.234E-02</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$R_u$</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$E_q$</td>
<td>1.784</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$R_q$</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>INPS-ZK17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$E_u$</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>$R_u$</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$E_q$</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>$R_q$</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

verge at many mesh levels on Mesh4 for both INPS-WDG10 and INPS-ZK17. We conclude that INPS-LW has a more accurate and stable performance than INPS-WDG10 and INPS-ZK17 in this case.

6.4. Test 4: Linearity-preserving verification

In this test, the domain $\Omega$ is composed of three subdomains:

$$\Omega_1 = \{(x, y) \in \Omega; \phi_1(x, y) < 0\},$$
$$\Omega_2 = \{(x, y) \in \Omega; \phi_1(x, y) > 0, \phi_2(x, y) < 0\},$$
$$\Omega_3 = \{(x, y) \in \Omega; \phi_2(x, y) > 0\}$$
with \( \phi_1(x, y) = y - \delta(x - 0.5) - 0.475 \) and \( \phi_2(x, y) = \phi_1(x, y) - 0.05 \). We take the \( \delta = 0.2 \) and choose the exact solution \( u(x, y) = 2 - x - \delta y \). The diffusion tensor is chosen to be

\[
\Lambda = \begin{pmatrix}
\cos \theta & -\sin \theta \\
\sin \theta & \cos \theta
\end{pmatrix}
\begin{pmatrix}
k_1 & 0 \\
0 & k_2
\end{pmatrix}
\begin{pmatrix}
\cos \theta & \sin \theta \\
-\sin \theta & \cos \theta
\end{pmatrix}
\]

with \( \theta = \arctan \delta \), \( k_1 = 100 \), \( k_2 = 10 \) on \( \Omega_2 \) and \( k_1 = 1 \), \( k_2 = 0.1 \) on \( \Omega_1 \cup \Omega_3 \). This test is a benchmark in [19]. We approximate this test on non-conforming mesh (see Fig. 7).

The numerical results are given in Table 10. The \( u_{\min} \) and \( u_{\max} \) present minimum value and maximum value of the approximate solution at the cell centers, respectively. We can find that both INPS-EW2 and INPS-LW are exactly reproduce piecewise linear solutions, such that INPS-EW2 and INPS-LW are linearity-preserving in this test as the exact solution is affine on the whole domain.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>( u_{\min} )</th>
<th>( u_{\max} )</th>
<th>( E_u )</th>
<th>( E_q )</th>
</tr>
</thead>
<tbody>
<tr>
<td>INPS-EW2</td>
<td>0.85385</td>
<td>1.94615</td>
<td>4.319E-015</td>
<td>4.932E-014</td>
</tr>
<tr>
<td>INPS-LW</td>
<td>0.85385</td>
<td>1.94615</td>
<td>2.285E-015</td>
<td>3.717E-014</td>
</tr>
</tbody>
</table>

6.5. Test 5: Spherical nonlinear heat wave

Consider a spherically symmetric heat wave, propagating in a cold medium and governed by the following nonlinear parabolic equation:

\[
\frac{\partial T}{\partial t} = \frac{1}{r^2} \frac{\partial}{\partial r} \left( r^2 \Lambda \frac{\partial T}{\partial r} \right),
\]  

(6.3)
where $T$ stands for the temperature and depends only on the spherical radius $r$. The analytic solution is given by [36] in terms of the radial position of the wave front $r_f$ and the temperature at the center $T_c$,

$$
T = \begin{cases} 
T_c \left(1 - \frac{r^2}{r_f^2}\right)^{\frac{1}{2}}, & \text{if } r \leq r_f, \\
0, & \text{otherwise},
\end{cases}
$$

(6.4)

where $T_c = 2^{\frac{3}{8}}\xi t^{-\frac{3}{8}}, r_f = \xi t^{\frac{1}{8}},$ and $\xi = 2^{\frac{7}{8}}\pi^{-\frac{3}{2}}$ is a dimensionless constant.

In order to solve this problem with the new scheme directly, we simply turn to solve the following equation in Cartesian coordinate system:

$$
\frac{\partial u}{\partial t} - \text{div} \left( \Lambda(u) \nabla u \right) + \frac{u}{8t} = 0
$$

(6.5)

with $r = \sqrt{x^2 + y^2}, u = T$ and $\Lambda(u) = T^2$. The computational domain is chosen to be $[-1, 1]^2$ and the point source is placed at the center $(0, 0)$.

Random mesh is used in this test, which aims to check whether the new scheme suffers the numerical heat-barrier issue. The simulation starts at $t = 10^{-8}$ and stops at $t = 0.3$. Meanwhile, the time step is a variable one, i.e., it takes $4E-10 \times h^2$ at the beginning and in the following computation it is enlarged (resp. shortened) twenty percent if the number of nonlinear iteration is no more than 5 (resp. greater than 20). The results on $24 \times 24$, $48 \times 48$ and $96 \times 96$ random meshes are shown in Figs. 8-10, respectively. The left parts of the figures show the meshes and contours of the numerical results while the right parts are the profiles of the numerical solutions with respect to radius $r$. One can see that INPS-LW maintains the spherical symmetry quite well.

![Figure 8: Results for INPS-LW on 24 × 24 random mesh at t = 0.3.](image-url)
7. Conclusion

This article introduced a linearity-preserving vertex interpolation algorithm LW to improve the classical NPS for nonlinear diffusion problems on general meshes. An edge-based discretization of diffusion coefficients is employed to improve the classical NPS, and thus the improved scheme can overcome the numerical heat-barrier issue well when solving strong nonlinear diffusion problems. The numerical experiments show that the new vertex interpolation algorithm brings obvious improvement for the final scheme on highly distorted meshes. Since LW is motivated by classical MPFA-O method which has been extend to 3D cases, the LW interpolation algorithm can be extended to 3D cases naturally which will be addressed in following work.
Acknowledgments

This work was partially supported by the National Key Research and Development Program of China (No. 2018YFC0603500), the National Natural Science Foundation of China (Nos. 11871009, 41725017, 41590864) and the Fundamental Research Funds for the Central Universities.

References


[31] D. Shepard, A two-dimensional interpolation function for irregularly-spaced data, in: Pro-


