Commun. Comput. Phys.
doi: 10.4208/cicp.230815.030616a

Vol. **20**, No. 5, pp. 1405-1423
November 2016

A Conservative Parallel Iteration Scheme for Nonlinear Diffusion Equations on Unstructured Meshes

Yunlong Yu[1,*], Yanzhong Yao[1], Guangwei Yuan[1] and Xingding Chen[2]

[1] *Laboratory of Computational Physics, Institute of Applied Physics and Computational Mathematics, P.O. Box 8009, Beijing 100088, P.R. China.*
[2] *Department of Mathematics, College of Science, Beijing Technology and Business University, Beijing 100048, P.R. China.*

Received 23 August 2015; Accepted (in revised version) 3 June 2016

Abstract. In this paper, a conservative parallel iteration scheme is constructed to solve nonlinear diffusion equations on unstructured polygonal meshes. The design is based on two main ingredients: the first is that the parallelized domain decomposition is embedded into the nonlinear iteration; the second is that prediction and correction steps are applied at subdomain interfaces in the parallelized domain decomposition method. A new prediction approach is proposed to obtain an efficient conservative parallel finite volume scheme. The numerical experiments show that our parallel scheme is second-order accurate, unconditionally stable, conservative and has linear parallel speed-up.

AMS subject classifications: 65M08, 65E05

Key words: Diffusion equations, conservation, parallel scheme, unstructured mesh.

1 Introduction

Nonlinear diffusion problems appear in many applications such as radiation hydrodynamics problems in high energy density physics, and pollutant transport in subsurface geological formations. The computation cost is always very expensive to solve these equations numerically, so it is necessary to solve them on parallel supercomputers. As we know, the classical explicit scheme has natural parallelism and can be implemented directly on parallel computers, but it requires severe restrictions on the time step to ensure the stability. The implicit scheme does not have the stability restrictions, however, its

*Corresponding author. *Email addresses:* yu_yunlong@iapcm.ac.cn (Y. Yu), yao_yanzhong@iapcm.ac.cn (Y. Yao), yuan_guangwei@iapcm.ac.cn (G. Yuan), chenxd@th.btbu.edu.cn (X. Chen)

 ©2016 Global-Science Press

https://doi.org/10.4208/cicp.230815.030616a Published online by Cambridge University Press

parallelism is limited and cannot be implemented directly on parallel computers. Especially, it results in a large global nonlinear algebraic system at each time step, and solving this system on parallel computers needs a lot of global communications. For the global communication is a main bottle-neck to get high parallel efficiency, it is necessary to modify the implicit scheme into a parallel scheme which can avoid the global communication.

The parallelized domain decomposition method is one of powerful numerical techniques on parallel computations [12]. In [1], a finite difference domain decomposition algorithm is proposed, in which an explicit scheme on coarse meshes is used at inner interfaces. This algorithm can take larger time step than the classical explicit scheme, but it is only conditionally stable. In [4], an unconditionally stable domain decomposition method is presented, the accuracy of which is $\mathcal{O}(\tau^{\frac{1}{2}}+h)$, where τ and h denote the time step and the mesh size respectively. The accuracy of this method is improved in [6], but the spatial accuracy remains first order. Using the prediction and correction techniques at the interfaces of subdomains, some unconditionally stable parallel difference schemes with second-order accuracy are constructed in [7, 10, 11, 14, 16]. However, these parallel schemes are not conservative since the discrete fluxes are not equal at the interfaces of subdomains.

Practical computations show that numerical errors from non-conservative schemes are accumulated dramatically through a large number of time steps, which leads to the numerical solution is far from the exact solution, especially on highly distorted meshes or in the region where the physical quantity varies intensively. Therefore, it is very important to enforce the conservation in the construction of the parallel scheme. In [2,3], the explicit/implicit conservative domain decomposition procedures are presented, in which the fluxes at the inner interfaces are calculated by the values from the previous time step. These schemes are conditionally stable, and the rigorous stability and convergence analysis can be found in [15]. A conservative parallel scheme for the linear diffusion equation on one-dimensional meshes is presented in [18], in which the unconditional stability and the second-order accuracy of the scheme are proved. However, when this scheme is directly extended to two-dimensional problems, it becomes conditionally stable. In [17], a conservative domain decomposition procedure is proposed for nonlinear diffusion problems on quadrilateral (structured) meshes. In this method, the values from the previous nonlinear iteration step are adopted as the Dirichlet boundary condition of subdomain problems at the inner interfaces, and then a correction step is implemented to ensure the conservation at the inner interfaces. Numerical experiments show that this parallel scheme is unconditionally stable, second-order accurate and conservative.

In practice a good parallel diffusion scheme is expected to satisfy the following requirements: (1) unconditionally stable, (2) second-order accurate, (3) conservative, (4) highly parallel, (5) suitable to various meshes, and no restrictions for the interface shape between subdomains. We point out that the last requirement comes from some applications such as the complex flow in Lagrangian radiation hydrodynamics problems, and pollutant transport in complex regions. For the multi-material radiation hydrodynamics problems with large variation of flow field, the Lagrangian computational meshes are

https://doi.org/10.4208/cicp.230815.030616a Published online by Cambridge University Press

highly distorted or even tangled, especially if the structured meshes are used. On such meshes, the accuracy of diffusion schemes usually used in production codes is deteriorated significantly and the iteration procedure often becomes very slow or even terminated. We remark that all of the above parallel schemes for two-dimensional diffusion problems are constructed on the structured quadrilateral meshes or rectangular meshes. To our knowledge, there is no paper considering about conservative parallel schemes on unstructured meshes, especially for nonlinear diffusion equations with discontinuous coefficients. Therefore, it is indispensable to construct an efficient parallel scheme on unstructured meshes.

In this paper, we propose a parallel finite volume scheme based on prediction and correction techniques to solve nonlinear diffusion equations on unstructured polygonal meshes. This parallel scheme is constructed by extending the idea in [17], where parallelized domain decomposition is coupled with nonlinear iterations. In our method, the global domain problem is decomposed into several subdomain problems, which are solved by a fully implicit scheme. At the inner interfaces of subdomains, we design a new prediction scheme of Jacobian type to obtain the Dirichlet boundary conditions of the subproblems at the first nonlinear iteration step. Then, in the following iteration steps, they are updated by the values of the previous iteration step. Finally, we adopt a correction step, in which the Neumann boundary conditions are used to guarantee the conservation. In our parallel scheme, the data transmission only happens between neighboring subdomains, so the global communication is avoided, which leads to the high parallel efficiency. The numerical experiments show that our scheme has good performance in terms of stability, accuracy, conservation and parallel speed-up. Besides, compared with the scheme in [17], the number of nonlinear iterations is decreased dramatically in our method.

This paper is organized as follows. In Section 2, we review a fully implicit finite volume scheme on unstructured polygonal meshes. In Section 3, the construction of our parallel procedure is addressed. The numerical experiments are given in Section 4, which is followed by a conclusion in Section 5.

2 Diffusion scheme on unstructured polygonal meshes

2.1 The model problem

Let Ω be an open bounded polygonal set of R^2 with boundary $\partial\Omega$. We consider the following nonlinear diffusion problem

$$\partial_t u+\nabla\cdot(\vec{F})=f,\quad \mathbf{x}\in\Omega,\ \ t\in(0,T], \tag{2.1}$$

with flux density $\vec{F}=-\kappa(u)\nabla u$. where $\mathbf{x}$ denotes the space coordinates, $u=u(\mathbf{x},t)$ is a scalar function, in the case of heat conduction, u denotes the temperature. $\kappa(u)$ is a positive scalar function, and f is the density of sources.

https://doi.org/10.4208/cicp.230815.030616a Published online by Cambridge University Press

The initial condition is

$$u(\mathbf{x},0)=\phi(\mathbf{x}),\quad \mathbf{x}\in\Omega. \tag{2.2}$$

We consider the following Dirichlet or Neumann boundary conditions

$$u=\psi,\qquad \text{on } \partial\Omega_D, \tag{2.3}$$

$$\vec{F}\cdot\vec{n}=F_1,\quad \text{on } \partial\Omega_N, \tag{2.4}$$

where $\partial\Omega=\partial\Omega_D\cup\partial\Omega_N$, and $\vec{n}$ is the unit outward normal vector on $\partial\Omega$.

The domain Ω is partitioned into unstructured polygonal meshes. We denote the mesh cell by i $(i=1,\cdots,I)$, where I is the total number of cells in Ω. The unknowns are defined at the center of cells. The computational time T is divided into N time steps. τ denotes the time interval, we get $T=N\tau$, and the nth time step $t^n=n\tau$ $(n=0,1,\cdots,N)$.

For the cell i, let l_i be the number of vertices, and $\alpha_j,i_j(j=1,\cdots,l_i)$ be the jth vertex and the jth neighboring cell, respectively. σ_j denotes the jth edge which is the common edge of the cells i and i_j (see Fig. 1).

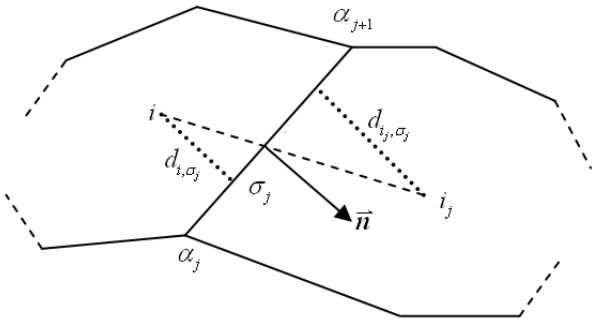


Figure 1: Notations.

Similarly with the scheme on quadrilateral meshes in [9], the fully implicit scheme on unstructured polygonal meshes for the model problem is

$$\frac{1}{\tau}(u_i^{n+1}-u_i^n)+\frac{1}{S_i}\sum_{j=1}^{l_i}F_{i,\sigma_j}^{n+1}L_{\sigma_j}=f_i^{n+1},\quad i=1,2,\cdots,I,\quad 0\le n<N, \tag{2.5}$$

where S_i is the area of cell i, L_{σ_j} is the length of edge σ_j, and F_{i,σ_j}^{n+1} is the discretization of the normal flux density on the edge σ_j of cell i at the time step t^{n+1} as follow

$$F_{i,\sigma_j}^{n+1}=-\kappa_{\sigma_j}^{n+1}\left[\frac{u_{i_j}^{n+1}-u_i^{n+1}}{d_{i_j,\sigma_j}+d_{i,\sigma_j}}-D_{\sigma_j}\left(\frac{u_{\alpha_{j+1}}^{n+1}-u_{\alpha_j}^{n+1}}{L_{\sigma_j}}\right)\right], \tag{2.6}$$

https://doi.org/10.4208/cicp.230815.030616a Published online by Cambridge University Press

and

$$\kappa_{\sigma_j}^{n+1}=\frac{(d_{i_j,\sigma_j}+d_{i,\sigma_j})\kappa_i^{n+1}\kappa_{i_j}^{n+1}}{d_{i_j,\sigma_j}\kappa_i^{n+1}+d_{i,\sigma_j}\kappa_{i_j}^{n+1}}, \tag{2.7}$$

$$D_{\sigma_j}=\frac{\left(\mathbf{x}_{i_j}-\mathbf{x}_i\right)\cdot\left(\mathbf{x}_{\alpha_{j+1}}-\mathbf{x}_{\alpha_j}\right)}{L_{\sigma_j}(d_{i_j,\sigma_j}+d_{i,\sigma_j})}, \tag{2.8}$$

where d_{i,σ_j} is the distance from the center of the cell i to the edge σ_j, $\mathbf{x}_i$ is the coordinates of the center of the cell i, $\mathbf{x}_{\alpha_j}$ is the coordinates of the vertex α_j, and κ_i^{n+1} is the diffusion coefficient defined on cell i at the time t^{n+1}. Usually, the vertex values are replaced by certain combination of their neighboring cell-centered values to reduce the number of unknowns [9], i.e.,

$$u_\alpha^{n+1}=\frac{1}{\sum_{k\in\mathcal{B}(\alpha)}\omega_{k,\alpha}^{n+1}}\sum_{k\in\mathcal{B}(\alpha)}\omega_{k,\alpha}^{n+1}u_k^{n+1}, \tag{2.9}$$

where $\mathcal{B}(\alpha)$ is the collection of these cells k that have α as a vertex. There are some simple methods to compute the weighted factors $\omega_{k,\alpha}^{n+1}$ $(k=1,\cdots,m_\alpha)$, such as the arithmetic average weighted method, i.e., $\omega_{k,\alpha}^{n+1}=\frac{1}{m_\alpha}$, and the inverse distance weighted method, i.e., $\omega_{k,\alpha}^{n+1}=\frac{\kappa_k^{n+1}}{d_{k,\alpha}}$, where m_α denotes the number of cells in $\mathcal{B}(\alpha)$ and $d_{k,\alpha}$ denotes the distance between the center of cell k and the vertex α. We should point out that these two methods have acceptable accuracy on the smooth and uniform meshes, but when the meshes are highly distorted, the accuracy will be lost [9]. In recent years, some improved methods for computing the weighted factors on distorted meshes are proposed [5,9,13]. Here we use the method in [5].

2.2 The Picard iteration method

For the sake of simplicity, we omit the superscript $n+1$ of the value at the present time level, i.e., u_i denotes u_i^{n+1}. The superscript n of u_i^n at the previous time level is preserved to avoid confusion.

We apply the Picard iteration method to linearize the scheme (2.5). Let s be the nonlinear iteration index, and the iteration formula is as follows,

$$\frac{1}{\tau}(u_i^{(s+1)}-u_i^n)+\frac{1}{S_i}\sum_{j=1}^{l_i}F_{i,\sigma_j}^{(s+1)}L_{\sigma_j}=f_i^{(s)},\quad s=0,1,\cdots, \tag{2.10}$$

where

$$F_{i,\sigma_j}^{(s+1)}=-\kappa_{\sigma_j}^{(s)}\left[\frac{u_{i_j}^{(s+1)}-u_i^{(s+1)}}{d_{i_j,\sigma_j}+d_{i,\sigma_j}}-D_{\sigma_j}\left(\frac{u_{\alpha_{j+1}}^{(s)}-u_{\alpha_j}^{(s)}}{L_{\sigma_j}}\right)\right], \tag{2.11}$$

https://doi.org/10.4208/cicp.230815.030616a Published online by Cambridge University Press

$$\kappa_{\sigma_j}^{(s)} = \frac{(d_{i_j,\sigma_j} + d_{i,\sigma_j})\kappa_i^{(s)}\kappa_{i_j}^{(s)}}{d_{i_j,\sigma_j}\kappa_i^{(s)} + d_{i,\sigma_j}\kappa_{i_j}^{(s)}}, \tag{2.12}$$

$$\kappa_i^{(s)} = \kappa(\mathbf{x}_i, u_i^{(s)}, t^{n+1}), \tag{2.13}$$

$$u_{\alpha_j}^{(s)} = \frac{1}{\sum_{k\in\mathcal{B}(\alpha_j)}\omega_{k,\alpha_j}^{(s)}} \sum_{k\in\mathcal{B}(\alpha_j)} \omega_{k,\alpha_j}^{(s)} u_k^{(s)}, \tag{2.14}$$

$$f_i^{(s)} = f(\mathbf{x}_i, u_i^{(s)}, t^{n+1}). \tag{2.15}$$

In other words the diffusion coefficient κ and the source term f are linearized by taking the values at the previous nonlinear iteration step. Moreover, the tangent flux term $\frac{1}{L_{\sigma_j}}(u_{\alpha_{j+1}} - u_{\alpha_j})$ is also set to be the values at the previous nonlinear iteration step so that it is convenient to solve the linear system.

3 Parallel scheme on unstructured meshes

In this section, we construct the parallel scheme for diffusion equations on the unstructured polygonal meshes.

3.1 Sketch of the parallel scheme

The computational domain Ω is divided into several non-overlapping subdomains, and each subdomain is partitioned by polygonal meshes. The common boundaries of neighboring subdomains are called the inner interfaces, and we assume that the inner interfaces are coincided with the edges of the cells.

The parallel scheme is a combination of domain decomposition with the nonlinear iteration. At the first nonlinear iteration step, the Dirichlet boundary conditions at inner interfaces are predicted by an explicit scheme of Jacobian type. At the following iteration steps, they are updated by the values from the previous iteration step. After the nonlinear iteration converges, the subproblems are solved once more, in which the same Neumann boundary conditions are imposed at the common inner interface for neighboring subdomains.

Note that at each iteration step the global problem is always decoupled into several independent subdomain problems, and each of them is computed by the fully implicit scheme.

It is well known that overlapping subdomain methods usually result in fast iterative convergence, but with the increase of the number of freedom degree such methods need more computational cost. In the construction of our scheme we use non-overlapping subdomain methods due to the following two considerations. Firstly, in solving the diffusion equation from radiation hydrodynamics problems it is necessary to reduce computational complexity and difficulty, so we should avoid introducing new freedom degree

https://doi.org/10.4208/cicp.230815.030616a Published online by Cambridge University Press

as much as possible. Secondly, in some applications the radiation diffusion process is always coupled with multi-material fluid flow. For the problems we are concerned, Lagrangian method is used and then the computational meshes move with the time and the material interfaces are naturally set as the inner interfaces in the parallel computations. When overlapping subdomain methods are used, it cannot ensure that there exists only one material in each subdomain. In other words, there exists at least one subdomain in which there are at least two materials. According to our experience of numerical simulation, when the physical properties (e.g., density and temperature) of the two materials differ significantly, the convergence of the nonlinear iteration in such subdomain will be very slow due to the conditional number being very poor. And then it will directly reduce the calculational efficiency in solving the whole domain problem, for which the synchronization is needed for all the processors.

3.2 The prediction of cell-centered values near the inner interface

For the sake of clarity, the global domain Ω is decomposed into two non-overlapping subdomains Ω_1 and Ω_2, and the inner interface is denoted by $\Gamma=\partial\Omega_1\bigcap\partial\Omega_2$ (see Fig. 2). After the partition of the domain Ω, we need to define the boundary condition on Γ to solve each subdomain problem independently. Assume cell $i\in\Omega_1$. If cell i is not adjacent to Γ, i.e., there is no edge of cell i lying at the inner interface, the iteration scheme (2.10)-(2.11) is used. If the edge σ_j of cell i lies at the inner interface, the neighboring cell in Ω_2 is denoted by i_j (see Fig. 2). When we use the scheme (2.10)-(2.11) at cell i, the value $u_{i_j}^{(s+1)}$ of cell i_j at the $(s+1)$th Picard iteration step is not known for the moment, and it has to be replaced by a predicted value $\tilde{u}_{i_j}^{(s+1)}$, which will be estimated by the following strategy.

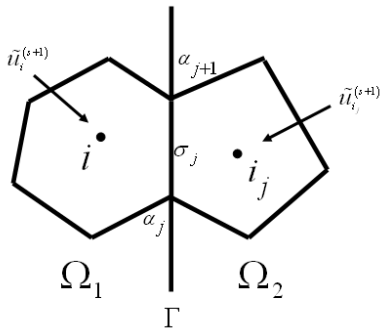


Figure 2: Dirichlet condition at the inner interface.

When $s=0$, we give the predicted value $\tilde{u}_{i_j}^{(1)}$ by the following Jacobi scheme, in which the surrounding cell-centered unknowns u_k^{n+1} of cell i_j are replaced by $2u_k^n-u_k^{n-1}$,

$$\frac{1}{\tau}(\tilde{u}_{i_j}^{(1)}-u_{i_j}^n)+\frac{1}{S_{i_j}}\sum_{k=1}^{l_{i_j}}\tilde{F}_{i_j,\sigma_k}L_{\sigma_k}=f_{i_j}^n, \tag{3.1}$$

https://doi.org/10.4208/cicp.230815.030616a Published online by Cambridge University Press

where

$$\tilde{F}_{i_j,\sigma_k}=-\kappa^n_{\sigma_k}\left[\frac{(2u^n_k-u^{n-1}_k)-\tilde{u}^{(1)}_{i_j}}{d_{i_j,\sigma_k}+d_{k,\sigma_k}}-D_{\sigma_k}\left(\frac{u^n_{\alpha_{k+1}}-u^n_{\alpha_k}}{L_{\sigma_k}}\right)\right]. \tag{3.2}$$

At the first time step, i.e., $n=0$, let $u^{n-1}_k=u^0_k$.

When $s\geq 1$, we take the values obtained from the previous Picard iteration step of the subproblem in Ω_2 as the predicted ones,

$$\tilde{u}^{(s+1)}_{i_j}=u^{(s)}_{i_j},\quad s=1,2,\cdots. \tag{3.3}$$

With the predicted values $\{\tilde{u}^{(s+1)}_{i_j},s=0,1,\cdots\}$, the discrete flux across the inner interface σ_j of cell i in the scheme can be constructed as follow

$$\tilde{F}^{(s+1)}_{i,\sigma_j}=-\kappa^{(s)}_{\sigma_j}\left[\frac{\tilde{u}^{(s+1)}_{i_j}-u^{(s+1)}_i}{d_{i_j,\sigma_j}+d_{i,\sigma_j}}-D_{\sigma_j}\left(\frac{u^{(s)}_{\alpha_{j+1}}-u^{(s)}_{\alpha_j}}{L_{\sigma_j}}\right)\right],\quad s=0,1,\cdots. \tag{3.4}$$

Similarly for the cell i_j, we get the predicted value $\tilde{u}^{(s+1)}_i$ of cell i and the discrete flux $\tilde{F}^{(s+1)}_{i_j,\sigma_j}$ across σ_j at the $(s+1)$th Picard iteration step as follow

$$\tilde{F}^{(s+1)}_{i_j,\sigma_j}=-\kappa^{(s)}_{\sigma_j}\left[\frac{\tilde{u}^{(s+1)}_i-u^{(s+1)}_{i_j}}{d_{i_j,\sigma_j}+d_{i,\sigma_j}}-D_{\sigma_j}\left(\frac{u^{(s)}_{\alpha_j}-u^{(s)}_{\alpha_{j+1}}}{L_{\sigma_j}}\right)\right],\quad s=0,1,\cdots. \tag{3.5}$$

When $s\geq 1$, $\tilde{u}^{(s+1)}_i=u^{(s)}_i$, which comes from the solutions of the subproblem in Ω_1; when $s=0$, $\tilde{u}^{(1)}_i$ is calculated by the following equations,

$$\frac{1}{\tau}(\tilde{u}^{(1)}_i-u^n_i)+\frac{1}{S_i}\sum_{\tilde{k}=1}^{l_i}\tilde{F}_{i,\sigma_{\tilde{k}}}L_{\sigma_{\tilde{k}}}=f^n_i, \tag{3.6}$$

$$\tilde{F}_{i,\sigma_{\tilde{k}}}=-\kappa^n_{\sigma_{\tilde{k}}}\left[\frac{(2u^n_{\tilde{k}}-u^{n-1}_{\tilde{k}})-\tilde{u}^{(1)}_i}{d_{i,\sigma_{\tilde{k}}}+d_{\tilde{k},\sigma_{\tilde{k}}}}-D_{\sigma_{\tilde{k}}}\left(\frac{u^n_{\alpha_{\tilde{k}+1}}-u^n_{\alpha_{\tilde{k}}}}{L_{\sigma_{\tilde{k}}}}\right)\right]. \tag{3.7}$$

With the predicted fluxes $\tilde{F}^{(s+1)}_{i,\sigma_j}$ and $\tilde{F}^{(s+1)}_{i_j,\sigma_j}$ in formulae (3.4) and (3.5), respectively, the problem in the global domain is decoupled into some independent subproblems, which can be solved on parallel computers in parallel.

Note that at the first Picard iteration step, a simple prediction scheme is $\tilde{u}^{(1)}_i=u^n_i$. This prediction approach can ensure the unconditional stability of the parallel scheme, but it leads to a number of Picard iterations because of its low accuracy. Compared with the simple prediction scheme, the Jacobian type scheme (3.1)-(3.2) is a more accurate explicit scheme with the approximate accuracy $\mathcal{O}(\tau+(\frac{\tau}{h})^2)$. It can decrease the number of the Picard iterations. Meanwhile, it can also ensure the unconditional stability of the parallel scheme.

https://doi.org/10.4208/cicp.230815.030616a Published online by Cambridge University Press

3.3 Conservative correction

According to the expressions (3.4) and (3.5) of the discretization fluxes, it is obvious that the parallel scheme is not conservative, since the numerical flux $\tilde{F}_{i,\sigma_j}^{(s+1)}$ for the cell i is not same as the minus of $\tilde{F}_{i_j,\sigma_j}^{(s+1)}$ for the cell i_j across the common edge σ_j. In order to ensure the conservation, we need to implement the following correction step after the Picard iteration converges.

Assume the edge σ_j lies in the inner interface. For the cell i, the flux across σ_j is reconstructed as

$$F_{i,\sigma_j}^{(s+1)} = -\kappa_{\sigma_j}^{(s)} \left[\frac{u_{i_j}^{(s)} - u_i^{(s)}}{d_{i_j,\sigma_j} + d_{i,\sigma_j}} - D_{\sigma_j} \left(\frac{u_{\alpha_{j+1}}^{(s)} - u_{\alpha_j}^{(s)}}{L_{\sigma_j}} \right) \right]. \tag{3.8}$$

And for the cell i_j, the flux across σ_j is set to be

$$F_{i_j,\sigma_j}^{(s+1)} = -F_{i,\sigma_j}^{(s+1)}. \tag{3.9}$$

It is well-known that the physical solutions of the diffusion problem in different subdomains are strongly correlated, so we glue them properly by updating the inner boundary conditions at each nonlinear iteration step. In our scheme, we use the convergent cell-centered values in the neighboring subdomains to explicitly obtain a flux across the inner interfaces, and then take the new flux as the Neumann boundary condition to solve neighboring subdomain problems once more. From (3.9), it means the numerical fluxes are the same at the common edge σ_j between the subdomains Ω_1 and Ω_2. It follows that the local conservation is guaranteed across the inner interfaces by this correction step. Obviously the scheme (2.10)-(2.11) is conservative in each subdomain. Therefore, the scheme (2.10)-(2.11), (3.4)-(3.5), (3.8) is a Conservative Parallel Scheme on Unstructured Polygonal Meshes, which is abbreviated as CPSUPM.

In the CPSUPM, the conservative correction step with Neumann inner boundary conditions is implemented after the convergence of the Picard iterations, which does not affect the unconditional stability of the parallel scheme. However, if the prediction step is passed over and only the Neumann boundary condition is taken at the inner interfaces in each Picard iteration, then the resulting parallel scheme becomes conditionally stable.

We must point out that, besides the nonlinear Picard iteration as that in the sequential scheme to ensure the convergence, there is no extra nonlinear iterations among subdomains in our parallel scheme. Moreover, at each step of the Picard iteration, only one local communication between neighboring subdomains is needed.

4 Numerical experiments

In this section, we present some numerical experiments to examine the performance of CPSUPM. The computational domain Ω is a square $(0,1)\times(0,1)$ for the test problems

https://doi.org/10.4208/cicp.230815.030616a Published online by Cambridge University Press

in Subsections 4.1, 4.2, 4.3, and a quarter circle for the problem in Subsection 4.4. The terminate criterion of the Picard iteration is

$$\max_{1\le i\le I}\left(\left|u_i^{(s+1)}-u_i^{(s)}\right|\right)<\varepsilon, \tag{4.1}$$

and the maximum norm and L_2-norm of numerical errors are defined by

$$\|u-u_h\|_\infty=\max_{1\le i\le I}\left|u(\mathbf{x}_i,T)-u_i^N\right|, \tag{4.2}$$

$$\|u-u_h\|_2=\left(\sum_{i=1}^{I}\left|u(\mathbf{x}_i,T)-u_i^N\right|^2 S_i\right)^{\frac{1}{2}}, \tag{4.3}$$

where $u(\mathbf{x}_i,T)$ and u_i^N are the exact solution and the numerical solution of cell i at the time T, respectively.

For the exact solution $u(\mathbf{x},t)$, it holds

$$\int_\Omega u(\mathbf{x},T)dS-\int_\Omega u(\mathbf{x},0)dS+\int_0^T dt\int_{\partial\Omega}\vec{F}\cdot\vec{n}dl-\int_0^T dt\int_\Omega f dS=0. \tag{4.4}$$

We use the following discrete analogy of (4.4) to measure the conservation of the numerical solution from the initial time to the current time T,

$$e_{con}=\left|\sum_{i=1}^{I}u_i^N S_i-\sum_{i=1}^{I}u_i^0 S_i+\sum_{n=1}^{N}\Big(\sum_{\sigma\in\partial\Omega}F_\sigma^n L_\sigma\Big)\tau-\sum_{n=1}^{N}\Big(\sum_{i=1}^{I}f_i^n S_i\Big)\tau\right|. \tag{4.5}$$

In the following numerical experiments, the number of subdomains is equal to the number of processors. We use the BiCRSTAB (bi-conjugate residual stabilized) algorithm [8] to solve the linear system in each subdomain.

4.1 Linear diffusion problem

Consider the linear diffusion problem (2.1) with $\kappa=1$. The exact solution is $u(x,y,t)=e^t(2+\sin(\pi x)\sin(\pi y))$. The source term is given by $f=\partial_t u-\nabla\cdot\nabla u$ and the Neumann boundary condition is adopted.

For this problem, we choose $\varepsilon=10^{-8}$ as the terminate criterion in the Picard iteration. Table 1 gives the parallel efficiency of CPSUPM by using different numbers of processors on the unstructured triangular meshes (see Fig. 3).

In the first column of Table 1, $m\times n$ means that we use m processors in the x direction and n processors in the y direction. It shows that our CPSUPM obtains very high parallel efficiency. Moreover, when the number of processors increases, the numerical errors $\|u-u_h\|_\infty$ and $\|u-u_h\|_2$ are almost the same as those of the fully implicit sequential scheme.

Some numerical results on 150850 mesh-cells are obtained by using more processors, see Table 2, where $time_{total}$ is the total time used for this problem, $time_{lin}$ is the total time

https://doi.org/10.4208/cicp.230815.030616a Published online by Cambridge University Press

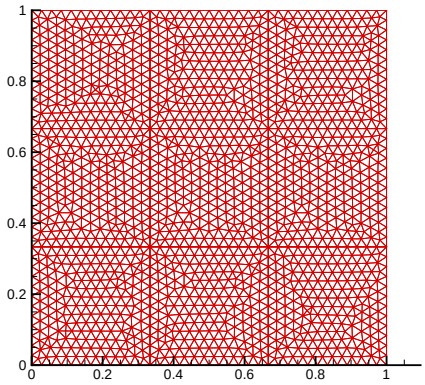


Figure 3: The unstructured triangular meshes.

Table 1: The parallel efficiency of CPSUPM for the linear diffusion problem on 49494 triangular meshes (T=0.01, $\tau=10^{-6}$).

processors	$\|u-u_h\|_2$	$\|u-u_h\|_\infty$	time(sec)	speed-up
1×1	3.0587e-6	1.7396e-5	5544.0	1
3×1	3.0614e-6	1.7395e-5	1872.7	98.6%
3×3	3.0654e-6	1.7395e-5	610.9	100.8%
6×3	3.0650e-6	1.7382e-5	294.1	104.7%
6×6	3.0681e-6	1.7380e-5	146.4	105.1%

Table 2: The scalability of CPSUPM for the linear diffusion problem on 150850 triangular meshes ($T=0.01$, $\tau=10^{-6}$).

processors	$time_{total}$	speed-up	$time_{lin}$	$time_{average}$
18	1394.1	1	833.9	1.67e-2
36	610.7	114.1%	389.6	7.79e-3
72	289.1	120.5%	176.5	3.53e-3
144	139.4	125.0%	80.1	1.60e-3
288	68.4	127.3%	33.4	6.68e-4

used for solving the linear system, and $time_{average}$ is the average time used for solving the linear system once. The speed-up is higher than 100%, and the speed-up increases as the number of processors increases, which shows that our scheme has good scalability. From the last two columns in Table 2, we can find that the calculation of the linear system occupies approximately half of the total time, and the decreasing rate of the $time_{average}$ is super-linear as the number of mesh-cells in one processor decreases.

https://doi.org/10.4208/cicp.230815.030616a Published online by Cambridge University Press

Table 3: The convergence order of CPSUPM for the linear diffusion problem on triangular meshes by 3×3 processors ($T=0.01$).

I	$\|u-u_h\|_2$	rate	$\|u-u_h\|_\infty$	rate
838	2.7629e-4	-	1.1877e-3	-
3152	5.8662e-5	2.23	3.0049e-4	1.97
12368	1.2057e-5	2.28	6.8346e-5	2.13
48796	2.6671e-6	2.16	1.9126e-5	1.83

Table 4: The stability of CPSUPM for the linear diffusion problem on 200×200 orthogonal meshes by using 64 processors ($T=1.0$).

τ	λ	$\|u-u_h\|_2$	$\|u-u_h\|_\infty$
6.25e-4	25	1.3873e-3	1.4234e-3
1.25e-3	50	2.7001e-3	2.7489e-3
2.5e-3	100	5.3031e-3	5.3772e-3
5.0e-3	200	1.0482e-2	1.0607e-2
1.0e-2	400	2.0840e-2	2.1064e-2

Table 5: The conservation errors for the linear diffusion problem on 49494 triangular meshes.

processors	CPSUPM	NCPSUPM
3×1	1.0138e-14	1.0157e-9
3×3	1.0441e-14	1.0907e-9
6×3	1.0285e-14	1.7348e-9
6×6	1.2844e-14	2.7546e-9

We test the accuracy of CPSUPM on the triangular meshes with 9 processors. Since the finite volume scheme has the temporal accuracy of first order, we choose a suitable time step $\tau=10^{-7}$. Table 3 shows that CPSUPM has second-order spatial accuracy.

Define $\lambda=\tau/h^2$. Table 4 shows that our scheme has good performance under different λ. Notice that the stability constraint for the fully explicit scheme is $\lambda\leq\frac{1}{4}$, and our scheme is still stable for $\lambda=400$, which indicates that our scheme is unconditionally stable.

From Table 5, we can see that the conservation errors of the CPSUPM are almost zero. When we drop the conservation correction step, the conservation errors achieved by the non-conservative parallel scheme on the unstructured polygonal meshes (NCPSUPM) are larger than those of the CPSUPM.

4.2 Diffusion problem with discontinuous coefficients

Consider the diffusion problem (2.1) with the discontinuous coefficient

https://doi.org/10.4208/cicp.230815.030616a Published online by Cambridge University Press

$$\kappa(x,y,t)=\begin{cases} 4, & (x,y,t)\in\Omega_1\times(0,T], \\ 1, & (x,y,t)\in\Omega_2\times(0,T], \end{cases} \tag{4.6}$$

where $\Omega_1=[0,\frac{2}{3}]\times(0,1)$, $\Omega_2=(\frac{2}{3},1]\times(0,1)$. The exact solution is chosen to be

$$u(x,y,t)=\begin{cases} e^{\frac{1}{1+t}}\sin(\pi x)\sin(2\pi y), & (x,y,t)\in\Omega_1\times(0,T], \\ e^{\frac{1}{1+t}}\sin(4\pi x)\sin(2\pi y), & (x,y,t)\in\Omega_2\times(0,T]. \end{cases} \tag{4.7}$$

We consider the Dirichlet boundary condition and the source is calculated by the exact solution.

This problem is used to test the scalability and the accuracy of CPSUPM on the polygonal meshes (see Fig. 4).

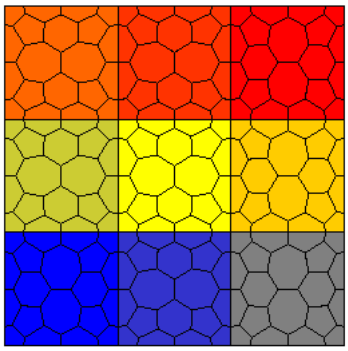


Figure 4: The mixed pentagonal and hexagonal meshes in 9 subdomains.

Table 6 gives the results by using different numbers of processors on mixed pentagonal and hexagonal meshes with 55691 cells. From this table, we can see that our CPSUPM has good scalability and the speed-up is super-linear, which is similar to the linear problem in last section. Although the diffusion coefficient is discontinuous, the parallel scheme has second-order accuracy on unstructured polygonal meshes, see Table 7. The final time is $T=0.01$ and the time step is $\tau=10^{-6}$ for this test.

Table 6: The scalability of CPSUPM for the diffusion problem with discontinuous coefficients on 55691 polygonal meshes.

processors	$time_{total}$	speed-up	$time_{lin}$	$time_{average}$
18	5930.5	1	3517.1	1.06e-2
36	2519.7	117.7%	1613.1	5.01e-3
72	1111.8	133.6%	670.1	2.36e-3
144	518.6	142.9%	280.1	1.12e-3
288	225.9	163.9%	90.1	4.02e-4

https://doi.org/10.4208/cicp.230815.030616a Published online by Cambridge University Press

Table 7: The convergence order of CPSUPM for the diffusion problem with discontinuous coefficients by 3×3 processors.

I	$\|u-u_h\|_2$	rate	$\|u-u_h\|_\infty$	rate
536	2.1616e-2	-	1.0091e-1	-
1801	5.9620e-3	2.12	2.8818e-2	2.05
6625	1.5048e-3	2.11	8.7820e-3	1.82
25271	3.8389e-4	2.03	2.7813e-3	1.72

4.3 Nonlinear diffusion problem

Consider the nonlinear diffusion problem (2.1) with $\kappa = u$. Let the exact solution be $u(x,y,t) = e^{-2\pi^2 t}(2+\sin(\pi x)\sin(\pi y))$. The source term f and the Neumann boundary condition are set by the exact solution.

For this problem, we choose $\varepsilon = 10^{-6}$ as the terminate criterion in the Picard iteration. Table 8 gives the parallel efficiency of our scheme on 144×144 random meshes (see Fig. 5) by using different numbers of processors. The random disturbance range is 70%.

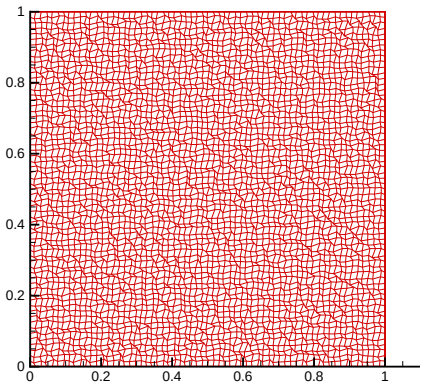


Figure 5: The random meshes.

From Table 8, it can be seen that, for arbitrary quadrilateral meshes, CPSUPM also has good parallel efficiency and achieves the same accuracy as that of the fully implicit scheme. To check the scalability of our scheme, we increase the number of processors to more than 1000 and the meshes scale to 800×800, and the results are listed in Table 9. Just as before, our scheme obtains good scalability and more than 100% speed-up for this nonlinear problem, mainly due to that the $time_{average}$ decreases at a faster speed than decreasing rate of the number of cells in one processor. Table 10 shows that the conservation errors of CPSUPM is obviously less than those of NCPSUPM.

From our numerical experiences for parallel schemes, we can find that different prediction approaches yield different computational efficiency and accuracy. An elaborate

https://doi.org/10.4208/cicp.230815.030616a Published online by Cambridge University Press

Table 8: The parallel efficiency of CPSUPM for the nonlinear diffusion problem on 144×144 random meshes ($T=0.01$, $\tau=10^{-6}$).

processors	$\|u-u_h\|_2$	$\|u-u_h\|_\infty$	$time(sec)$	speed-up
1×1	1.5772e-5	6.0711e-5	3889.1	1
3×1	1.5774e-5	6.0714e-5	1308.8	99.0%
3×3	1.5783e-5	6.0714e-5	424.3	101.8%
6×3	1.5766e-5	6.0714e-5	210.7	102.5%
6×6	1.5789e-5	6.0714e-5	97.5	110.8%

Table 9: The scalability of CPSUPM for the nonlinear diffusion problem on 800×800 random meshes ($T=0.01$, $\tau=10^{-6}$).

processors	$time_{total}$	speed-up	$time_{lin}$	$time_{average}$
100	10433.9	1	6462.4	1.47e-2
200	4219.1	126.4%	2749.0	6.41e-3
400	1694.9	153.8%	1112.1	2.59e-3
800	814.6	160.2%	570.8	1.25e-3
1280	462.8	176.1%	281.1	5.86e-3

Table 10: The conservation errors for the nonlinear diffusion problem on 144×144 random meshes.

processors	CPSUPM	NCPSUPM
3×1	7.21644e-16	3.59343e-7
3×3	1.0441e-14	7.04273e-7
6×3	1.0285e-14	1.17845e-6
6×6	1.80411e-15	1.45172e-6

Table 11: The average number of Picard iterations for the nonlinear diffusion problem on 144×144 random meshes by 36 processors ($T=0.1$, $\tau=10^{-5}$).

algorithm	$it^{\#}$	time(sec)	$\|u-u_h\|_2$	$\|u-u_h\|_\infty$
method 1	3.63	141.7	1.9281e-4	2.2243e-4
method 2	6.08	201.3	2.9257e-4	3.3153e-4

prediction for the values at the inner interfaces is likely to decrease the number of nonlinear iterations and enhance the accuracy. For the parallel scheme constructed in [17], the values of cells near the inner interfaces at the previous time step are directly taken as the predicted ones at the first Picard iteration step, i.e., $\tilde{u}_{i_j}^{(1)}=u_{i_j}^n$. In CPSUPM, we adopt the prediction formula (3.1)-(3.2) which is denoted by "method 1". The prediction approach in [17] is denoted by "method 2". These two methods are compared in Table 11, which

https://doi.org/10.4208/cicp.230815.030616a Published online by Cambridge University Press

shows that both of the average number of Picard iterations ($it^{\#}$) and the used time of our CPSUPM are less than those of the method 2. The numerical errors of the method 1 are also less than those of the method 2.

4.4 Results of the circle problem on mixed polygonal meshes

Consider the following nonlinear problem

$$\begin{cases} \partial_t u-\nabla\cdot((1+u)\nabla u)=0, & (x,y,t)\in\Omega\times(0,T], \\ u(x,y,0)=1.0, & (x,y)\in\Omega, \\ (1+u)\nabla u\cdot\vec{n}=T_b, & x^2+y^2=1, t\in(0,T], \\ (1+u)\nabla u\cdot\vec{n}=0, & x=0, y\in[0,1], t\in(0,T], \\ (1+u)\nabla u\cdot\vec{n}=0, & y=0, x\in[0,1], t\in(0,T], \end{cases} \tag{4.8}$$

where the computational domain is a quarter of circle, i.e.,

$$\Omega=\{(x,y)|\sqrt{x^2+y^2}\leq 1,\ x\geq 0,\ y\geq 0\},$$

the $T_b=-100t^2+30t+1$, and its variation curve is shown in Fig. 6.

In this test problem, the different subdomains of Ω are partitioned by different types of meshes, see Fig. 6 (1600 quadrilaterals and 1914 triangles). We choose $\tau=10^{-5}$, $T=0.2$, and $\varepsilon=10^{-6}$. We test the parallel efficiency of our scheme under the complicated mesh partition.

Table 12 shows the our CPSUPM also obtains high parallel efficiency, although it is a little lower than those in the previous examples. Note that the numbers of meshes are different in different subdomains, which may lead to unbalanced load on parallel computers. The average number of Picard iterations increases slightly as the number of subdomains increases (see Table 12), and we will further study this problem in future.

Table 12: The parallel efficiency of CPSUPM for the test problem 4 on the complicated meshes.

processors	time(sec)	speed-up	$it^{\#}$
1	1043.6	1	2.0
2	542.6	96.1%	2.0
4	306.4	85.1%	2.45
8	159.6	81.7%	2.66
16	74.6	87.4%	2.73

The exact solution is not known for this problem, so we compare the numerical solutions obtained by our parallel scheme on 16 processors with those by single processor. Fig. 7 presents the variation curves of u at two different points: $p_1(0.0592,0.0598)$,

https://doi.org/10.4208/cicp.230815.030616a Published online by Cambridge University Press

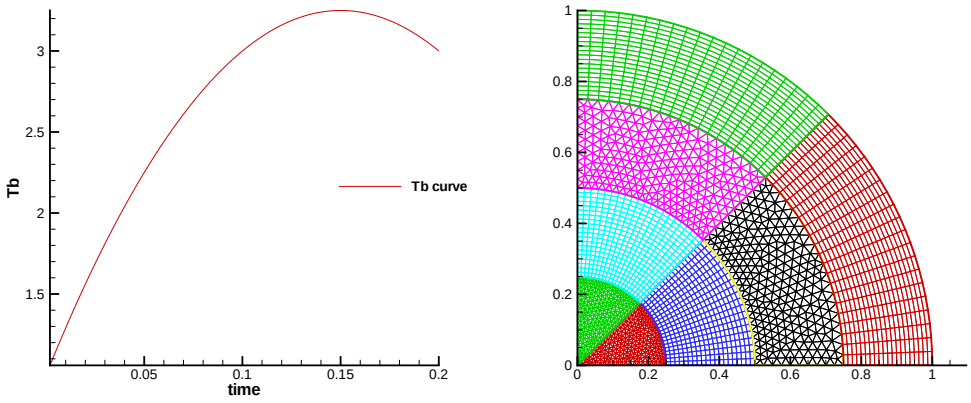


Figure 6: Variation curve for T_b and the mesh partition.

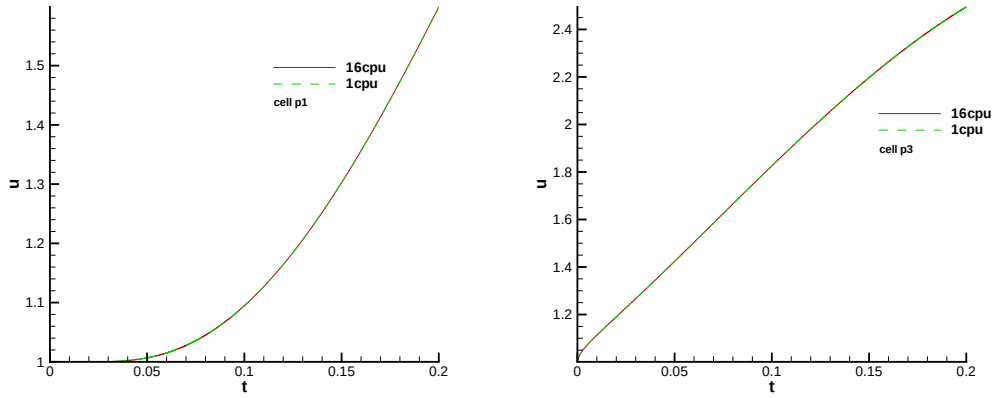


Figure 7: Variation curve for u at 2 positions.

$p_2(0.9934,0.0191)$. The solid lines are obtained by our scheme on 16 processors, and the dashed lines are computed by the fully implicit scheme on single processor. We can see that the both lines are completely coincident during the whole computational phase, which show that our CPSUPM has the same accuracy as that of the fully implicit scheme.

5 Conclusion

In this paper, we propose an efficient parallel scheme for the nonlinear diffusion equations on unstructured meshes. This parallel scheme has three main features. Firstly, we use a new prediction approach to decrease the number of the nonlinear iteration and improve the computational efficiency. Secondly, a correction step is adopted to ensure the

https://doi.org/10.4208/cicp.230815.030616a Published online by Cambridge University Press

conservation of the method. Thirdly, our scheme does not need any particular restrictions for the mesh partitions and the shape of the inner interfaces, and it can be naturally implemented on the parallel computers. The numerical experiments show that our scheme is second-order accurate, unconditionally stable and completely conservative. Moreover, this scheme can achieve the linear parallel speed-up.

Acknowledgments

The work of Y. Yao was supported by the National Natural Science Foundation of China (abbreviated as NNSFC) 11671049. The work of G. Yuan was supported by the NNSFC 11571048 and the Science Challenge Program. The work of X. Chen was supported by the NNSFC 11401015, the General Program of Science and Technology Development Project of Beijing Municipal Education Commission KM201310011006, and the Program of the Young People of Outstanding Ability for the Construction of the Teachers Procession YETP1445.

References

[1] C.N. DAWSON, Q. DU, T.F. DUPONT, *A finite difference domain decomposition algorithm for numerical solution of the heat equation*, Math. Comput., 57(1991), 63-71.
[2] C.N. DAWSON, T.F. DUPONT, *Explicit/implicit conservative Galerkin domain decomposition procedures for parabolic problems*, Math. Comput., 58(1992), 21-34.
[3] C.N. DAWSON, T.F. DUPONT, *Explicit/implicit conservative domain decomposition procedures for parabolic problems based on block-centered finite differences*, SIAM J. Numer. Anal., 31(1994), 1045-1061.
[4] M. DRYJA, *Substructreing methods for parabolic problems, Fourth International Symposium on Domain Decomposition Methods for Partial Differential Equations*, SIAM, Philadelphia, PA, (1991), 264-271.
[5] Z.M. GAO, J.M WU, *A linearity-preserving cell-centered scheme for the heterogeneous and anisotropic diffusion equations on general meshes*, Int. J. Numer. Meth. Fluids, 67(2011), 2157-2183.
[6] YU.M.LAEVAKY, O.V. RUDENKO, *Splitting methods for parabolic problems in nonrectangular domains*, Appl.Math.Lett., 8:6(1995), 9-14.
[7] H.L. LIAO, H.S. SHI, Z.Z. SUN, *Corrected explicit-implicit domain decomposition algorithms for two-dimensional semilinear parabolic equations*, Sci. China Ser. A, 52:8 (2009), 2362-2388.
[8] Y.SAAD, *Iterative Method for Sparse Linear Systems*, PWS Pub. Co., New York, 1996.
[9] Z.Q. SHENG, G.W. YUAN, *A nine point scheme for the approximation of diffusion operators on distorted quadrilateral meshes*, SIAM J. Sci. Comput., 30 (2008), 1341-1361.
[10] Z.Q. SHENG, G.W. YUAN, X. D. HANG, *Unconditional stability of parallel difference schemes with second order accuracy for parabolic equation*, Appl. Math. Comput., 184 (2007), 1015-1031.
[11] H.S. SHI, H.L. LIAO, *Unconditional stability of corrected explicit-implicit domain decomposition algortihtms for parrallel approximation of heat equations*, SIAM J. Numer. Anal., 44 (2006), 1584-1611.

https://doi.org/10.4208/cicp.230815.030616a Published online by Cambridge University Press

[12] A. TOSELLI, O. WIDLUND, *Domain Decomposition Methods-Algorithms and Theory*, Springer - Verlag, Berlin, 2005.
[13] J.M WU, Z.H. DAI, Z.M. GAO, G.W. YUAN, *Linearity preserving nine-point schemes for diffusion equation on distorted quadrilateral meshes*, J. Comput. Phys., 229(2010), 3382-3401.
[14] G.W. YUAN, F.L. ZUO, *Parallel differences schemes for heat conduction equation*, Int. J. Comput. Math., 80:8 (2003), 995-999.
[15] G.W. YUAN, L.J. SHEN, *Stability and convergence of explicit-implicit conservative domain decomposition procedure for parabolic problems*, Appl. Math. Comput., 47(2004), 793-801.
[16] G.W. YUAN, X.D. HANG, Z.Q. SHENG, *The unconditional stability of parallel difference schemes with second order convergence for nonlinear parabolic system*, J. Partial Diff. Eqs., Vol.20, No.1(2007), 1-22.
[17] G.W. YUAN, Y.Z. YAO, L. YIN, *A conservative domain decomposition procedure for nonlinear diffusion problems on arbitrary quadrilateral grids*, SIAM J. Sci. Comput., 33:3 (2011), 1352-1368.
[18] S.H. ZHU, *Conservative domain decomposition procedure with unconditional stability and second-order accuracy*, Appl. Math. Comput., 216 (2010), 3275-3282.

https://doi.org/10.4208/cicp.230815.030616a Published online by Cambridge University Press