LOCAL AND PARALLEL FINITE ELEMENT ALGORITHMS BASED ON 
THE PARTITION OF UNITY FOR THE STOKES PROBLEM *

JIAPING YU†, FENG SHI‡, AND HAIIBAO ZHENG§

Abstract. By combining the techniques of two-grid method and the partition of unity, two local and 
parallel finite element algorithms are presented for the Stokes problem. The most interesting features of these 
algorithms are: (1) the partition of unity technique introduces a framework for domain decomposition, (2) only 
a series of local residual problems need to be solved on these subdomains in parallel, meanwhile require very 
little communication, (3) a globally continuous finite element solution is constructed by combining all the local 
solutions via the partition of unity functions.

The optimal error estimates in $L^2$ and energy norms are proved under some assumptions. Also, several 
numerical simulations are presented to demonstrate the effectiveness and flexibility of the new algorithms.

Key words. Stokes, Local and parallel, Partition of unity, Oversampling.

AMS subject classifications. 65N30, 65M55, 76D07, 76M10.

1. Introduction. In the numerical simulation of incompressible flows, finite element meth-
ods have been one class of the most successful methods [1, 2, 3]. However, constructing higher 
accurate and more effective finite element algorithms still offers many challenges. Many re-
searchers have derived several new techniques and algorithms in the past several decades, such 
as, domain decomposition methods, nonlinear Galerkin methods, multi-grid and two grid/two 
level post-processing Galerkin methods (see [4, 5, 6, 7, 8, 9]). Among these methods, the two-
grid methods are motivated by the observation that for a solution, low frequency components 
could be approximated by a relatively coarse grid and high frequency components are computed 
on a fine grid. Later, Xu et al [10] and He et al [11] recognize that, the global behavior of a 
solution is mainly governed by low frequency components while the local behavior is mostly 
determined by high frequency components, therefore, they improve the two-grid method by 
capturing the high frequency components locally, that is, solving the residual equations (which 
mostly contain high frequencies, c.f. [4, 5]) on a fine grid by some local and parallel procedures. 
These approaches base on local finite element discretization and require less communication 
between blocks than classical domain decomposition methods. Meanwhile, a similar approach 
and implementation details are developed by Bank and Holst [12, 13]. He et al extend this idea 
to the Navier-Stokes equations [14, 15], Liu et al apply it to the time-dependent convection-
diffusion equations [16].

There are still some ways to improve these local and parallel algorithms, such as, imple-
menting domain decomposition in a better way, accomplishing the parallel procedures at the 
element level and constructing a global finite element solution. In this paper, we aim to improve 
the parallel algorithm by the partition of unity technique.

The partition of unity method(PUM) [17, 18] introduces us a flexible and controllable 
way to implement domain decomposition, and to construct a global solution. Many authors 
have paid their attentions to PUM. For example, Bacuta et al [19] propose a partition of 
unity refinement(PUR) method to improve the local approximations of elliptic boundary value

---

* H. Zheng was partially supported by NSF of China with Grant No. 11201369, 11171269, 11271298 and 
NCET. And he is also subsidized by the Fundamental Research Funds for the Central Universities (Grant No. 
08142003 and No. 08143007). F. Shi was partially supported by Guangdong Natural Science Foundation with 
Grant No. S201204007760, and the NSF of China (Tianyuan Fund for Mathematics, No. 11226314).

†School of Science, Donghua University, Shanghai, China. (yujiaping.yjp@gmail.com)
‡School of Mechanical Engineering and Automation, Harbin Institute of Technology, Shenzhen Graduate 
School, Shenzhen 518055, China. (shifeng@hit.edu.cn)
§Corresponding author. School of Mathematics and Statistics, Xi’an Jiaotong University, Xi’an, China. 
(hbzhang13@gmail.com)
problems in the regions of interest. Holst [20, 21] constructs the parallel partition of unity method (PPUM) by a combination of PUM with the parallel adaptive algorithm. Larson et al [22] and Song et al [23] also use PUM as the localization technique in post-processing procedure.

In this paper, based on the two-grid method, using the partition of unity technique, we design some local and parallel finite element algorithms for the Stokes problems.

In brief, firstly, we compute the global lower frequency approximation by the standard finite element method on a global coarse grid. Secondly, we derive a partition of unity on a regular triangulation, which is used to guide us to decompose the entire domain into several overlapping subdomains. Then, we get an approximation of the higher frequency component by solving a series of locally defined approximate residual problems with homogeneous Dirichlet boundary conditions on some finer grids. Finally, we construct a globally continuous finite element solution by assembling all the local solutions together using the partition of unity subordinate to the overlapping subdomains.

The most interesting features of our parallel algorithms are, the finite element basis functions on a given triangulation derive a class of partition of unity, which induces decomposition of the domain; the localization procedure could be implemented at the element level, and require far less communication than current approaches. Besides, to make up the lack of global continuity caused by the artificial homogeneous Dirichlet boundary conditions enforced on the local problems, a coarse grid correction procedure coupling with enlarging subdomains, called oversampling, is also introduced. Finally, a coarse grid correction procedure is also used to improve the smoothness of the finite element solution.

Our work improves the results of He et al [11], especially in implementation. Firstly, we give a framework of domain decomposition on the partition of unity for the localization procedures, which could ensure us to run the computation in parallel at the element level. Secondly, they derive the optimal error estimates by introducing a mesh-dependent norm. For our local and parallel algorithms, we give two global error estimates by employing the characteristics of the partition of unity and improving some recent local estimates. Finally, we obtain the globally continuous finite element solution, which is not involved in [11].

The outline of the paper is as follows. Section 2 introduces the Stokes problem, the notations and some well-known results for the finite element methods. The local and parallel finite element algorithms based on the partition of unity are discussed in section 3. In section 4, the implementation and some numerical simulations are presented to illustrate the efficiency of our methods. A short conclusion will be given at last in section 5.

2. The Stokes problem. We consider the incompressible Stokes problem in this paper of the form

\[\begin{align*}
-\nu \Delta u + \nabla p &= f \quad \text{in } \Omega, \\
\nabla \cdot u &= 0 \quad \text{in } \Omega, \\
u u &= 0 \quad \text{on } \partial \Omega,
\end{align*}\]

(2.1)

where \(\Omega\) represents a convex polyhedral domain in \(\mathbb{R}^d\), \(d = 2, 3\) with boundary \(\partial \Omega\), \(u\) the velocity vector, \(p\) the pressure, \(f\) the prescribed body force, and \(\nu > 0\) the kinematic viscosity, which is inversely proportional to the Reynolds number \(Re\). For simplicity, we set \(\nu = 1\).

For a bounded domain \(\Omega \subset \mathbb{R}^d\), we use the standard notations. For example, \((\cdot, \cdot)\) denotes the inner product in \(L^2(\Omega)\) or its vector/matrix-valued version. The norm and seminorm in \(H^k(\Omega)\) are denoted by \(||\cdot||_{k, \Omega}\) and \(||\cdot||_{k, \Omega}\), respectively. The space \(H^1_0(\Omega)\) is equipped with the norm \(||\nabla \cdot \||_{0, \Omega}\) or its equivalent norm \(||\cdot||_{1, \Omega}\) due to the Poincare’s inequality. The space \(H^{-1}(\Omega)\) is the dual of \(H^1_0(\Omega)\). Spaces consisting of vector-valued functions are denoted in boldface. For sub-domains \(D \subset G \subset \Omega\), notation \(D \subset \subset G\) means that \(\text{dist}(\partial D \setminus \partial \Omega, \partial G \setminus \partial \Omega) > 0\). Throughout this paper we use \(C\) to denote a generic positive constant whose value may change from place to place but remains independent of the mesh parameter \(h\).
The standard variational formulation of (2.1) is given by: find \((u, p) \in (X, Q)\) satisfying
\[
(2.2) \quad \mathcal{C}(u, p; v, q) = (f, v) \quad \forall (v, q) \in (X, Q),
\]
where
\[
X = H^1_0(\Omega) \text{ and } Q = L_0^2(\Omega) = \{ q \in L^2(\Omega); \int_{\Omega} q \, dx = 0 \},
\]
and
\[
\mathcal{C}(u, p; v, q) = (\nabla u, \nabla v) - (\nabla \cdot v, p) - (\nabla \cdot u, q).
\]

It is well-known \([1, 2]\) that when \((X, Q)\) satisfies the following inf-sup condition with a positive constant \(\beta_0\),
\[
(2.3) \quad \inf \sup_{q \in Q, v \in X} \frac{(\nabla v, q)}{||v||_{2, \Omega} ||q||_{0, \Omega}} \geq \beta_0,
\]
which implies that the weak inf-sup condition holds with a positive constant \(\beta_1\),
\[
(2.4) \quad \sup_{(v, q) \in (X, Q)} \frac{\mathcal{C}(u, p; v, q)}{||v||_{2, \Omega}||q||_{0, \Omega}} \geq \beta_1(||u||_{1, \Omega} + ||p||_{0, \Omega}) \quad \forall (u, p) \in (X, Q),
\]
thus, the problem (2.2) has a unique solution.

A further assumption on \(\Omega\) is needed:
\[A1\] Assume that \(\Omega\) is regular so that the unique solution \((v, q) \in (X, Q)\) of the generalized Stokes problem
\[
(2.5) \quad -\Delta v + \nabla q = g \quad \text{in } \Omega,

\nabla \cdot v = \rho \quad \text{in } \Omega,
\]
\[v = 0 \quad \text{on } \partial \Omega,
\]
for any prescribed \((g, \rho) \in (L^2(\Omega)^2, L^2_0(\Omega) \cap H^1(\Omega))\) exists and satisfies
\[
(2.6) \quad ||v||_{2, \Omega} + ||q||_{1, \Omega} \leq C(||g||_{0, \Omega} + ||\rho||_{1, \Omega}).
\]

Let \(\tau_H\) and \(\tau_h\) be two regular triangulations of the domain \(\Omega\), with \(H, h\) being the maximal diameters of the elements in \(\tau_H\) and \(\tau_h\), respectively, and \(h < H\). We also require that \(\tau_H\) and \(\tau_h\) are nested. The finite element spaces \((X^H, Q^H)\) and \((X^h, Q^h)\) are chosen as the Taylor-Hood element spaces, i.e., \(X^H\) (or \(X^h\)) and \(Q^H\) (or \(Q^h\)) contain piecewise polynomials of degrees \(k\) and \(k - 1\) respectively.

The standard mixed finite element methods for the Stokes problem (2.1) is: find \((u_h, p_h) \in (X^h_0, Q^h_0) = (X^h, Q^h) \cap (X, Q)\) such that
\[
(2.7) \quad \mathcal{C}(u_h, p_h; v^h, q^h) = (f, v^h) \quad \forall (v^h, q^h) \in (X^h_0, Q^h_0).
\]
The approximation system (2.7) possesses a unique solution \((u_h, p_h)\), which also satisfies the following error estimates \([1]\)
\[
(2.8) \quad ||u - u_h||_{1, \Omega} + ||p - p_h||_{0, \Omega} \leq C h^k \{ ||u||_{k+1, \Omega} + ||p||_{k, \Omega} \},

||u - u_h||_{0, \Omega} + ||p - p_h||_{-1, \Omega} \leq C h^{k+1} \{ ||u||_{k+1, \Omega} + ||p||_{k, \Omega} \}
\]
provided \((u, p) \in (H^{k+1}(\Omega), H^k(\Omega)).\)
3. Local and Parallel Finite Element Algorithms Based on the Partition of Unity. In this section, we present a framework for domain decomposition based on the partition of unity as follows.

First, we recall some definitions on the partition of unity (cf. [17]). Let \( \{ \Omega_i \}_{i=1}^N \) be an open cover of \( \Omega \) and let \( \{ \varphi_i \}_{i=1}^N \) be a partition of unity subordinate to the cover \( \{ \Omega_i \}_{i=1}^N \) satisfying

\[
\begin{align*}
\text{supp } \varphi_i &\subset \Omega_i, \forall i, \\
\sum_i \varphi_i &\equiv 1 \text{ on } \Omega, \\
\|\varphi_i\|_{L^\infty(\mathbb{R}^n)} &\leq C_{\infty}, \\
\|\nabla \varphi_i\|_{L^\infty(\mathbb{R}^n)} &\leq \frac{C_G}{\text{diam}\Omega_i},
\end{align*}
\]

where \( C_{\infty}, C_G \) are two constants.

For implementation, we give a practical and controllable way to accomplish partition of unity, by which a given triangulation can deduce a class of partition of unity. We also need a regular conforming triangulation \( \tau_{H_p}(\text{with mesh size } H_p) \) of \( \Omega \) for partition of unity. It is worthy noting that, for convenience of our present theoretical analysis, we assume that \( H_p \) is fixed and independent of \( H \) and \( h \), and \( \tau_{H_p}, \tau_H, \) and \( \tau_h \) are nested and satisfy \( h < H \leq H_p \).

Then we can define the partition of unity as follows. For each vertex \( x_i \in \tau_{H_p} \), let \( D_i \subset \Omega \) denote the union of triangles in triangulation \( \tau_{H_p} \) that own \( x_i \) as a common vertex.

Then for each \( i \), define by \( \phi_i \) a continuous, piecewise linear Lagrange basis function such that \( \phi_i(x_m) = \delta_{i,m} \). It’s easy to see that \( D_i = \text{supp } \phi_i \cap \Omega \).

In the following, we denote by \( D^{1,s} \) the subdomain with \( s \) layer(s) of oversampling of \( D_i \), which will serve as the local computational domain. More precisely, for \( s = 0, D^{1,0} = D^i \) denotes the subdomain without oversampling. \( D^{1,1} \) is the subdomain with one layer of oversampling, and is defined by the union of the support of \( \phi_i \) and one layer of its neighbors:

\[
D^{1,1} = \bigcup_{x_m \in D^{1,0}} D^m.
\]

Then the subdomain with two layers of oversampling is denoted by

\[
D^{1,2} = \bigcup_{x_m \in D^{1,1}} D^m.
\]

In a similar way, the subdomains with multiple layers oversampling can also be constructed. In the extreme case, we can enlarge each local domain \( D^i \) to be the whole domain \( \Omega \).

Figure 3.1 helps us to understand the above notations, and shows a subdomain with no oversampling and its one and two layers of oversampling. In each subfigure, a vertex \( x_i \) is marked by red color. The corresponding local domain \( D^{i,0} = D^i \) (in the left one) is painted by blue color, local computational domain \( D^{i,1} \) (in the middle one) is the union of blue and red subdomains, and \( D^{i,2} \) (in the right one) is the union of blue, red and green subdomains.

It’s easy to check that for any \( s, \{ D^{1,s} \}_{i=1}^N \) is an open cover of \( \Omega \) and \( \{ \phi_i \}_{i=1}^N \) is a partition of unity subordinate to the cover \( \{ D^{1,s} \}_{i=1}^N \).

Based on the above partition of unity, we develop a new local and parallel technique as follows.

**ALGORITHM 1:**

Step 1. Find a global coarse grid solution \((u_H, p_H) \in (X_0^H, Q_0^H)\) such that

\[
\mathcal{C}(u_H, p_H; v^H, q^H) = (f, v^H) \quad \forall (v^H, q^H) \in (X_0^H, Q_0^H).
\]
Step 2. Correct the residual \((\epsilon^i, \epsilon^i) \in (X^h_0(D^{i,s}), Q^h_0(D^{i,s})) (i = 1, 2, \cdots, N)\) on a fine grid of the overlapping subdomain \(D^{i,s}\), such that

\[
\mathcal{C}(\epsilon^i, \epsilon^i; \nu^h, q^h) = (f, \nu^h) - \mathcal{C}(u^H, p^H; \nu^h, q^h) \quad \forall (\nu^h, q^h) \in (X^h_0(D^{i,s}), Q^h_0(D^{i,s})),
\]

where

\[
X^h_0(D^{i,s}) := \{ v^h \in X^h(\Omega) : \text{supp} v^h \subset \subset D^{i,s} \},
\]

\[
Q^h_0(D^{i,s}) := \{ q^h \in Q^h(\Omega) : \text{supp} q^h \subset \subset D^{i,s} \text{and} \int_{D^{i,s}} q^h \mathrm{d}x = 0 \}.
\]

Step 3. Update: \((u^i, p^i) = (u^H, p^H) + (\epsilon^i, \epsilon^i) \) in \(D^{i,s}\).

Step 4. Obtain the global finite element solution \(u^h = \sum_{i=1}^{N} \phi_i u^i, \ p^h = \sum_{i=1}^{N} \phi_i p^i\).

Remark 1: Note that, the subproblems in Step 2 only depend on the global coarse grid solution, and are independent of each other. Thus they can be carried out in parallel naturally.

Remark 2: According to the following analysis, we always assume the index \(s \geq 1\) and is fixed and not large.

Remark 3: The local domains \(D^{i,s} \subset \Omega\) are purely constructed by the triangulation \(\tau_{H_{\epsilon}}\). Due to the assumption that the triangulations \(\tau_{H_{\epsilon}}, \tau_H, \tau_h\) are nested, in the constructions of finite element spaces \((X^h_0(D^{i,s}), Q^h_0(D^{i,s}))\), the partition of the domain \(D^{i,s}\) is actually deduced by the limitation of the triangulation \(\tau_h\) on domain \(D^{i,s}\), that is \(\{K_h \in \tau_h : K_h \subset D^{i,s}\}\).

Moreover, based on \(\tau_{H_{\epsilon}}\) and the corresponding basis functions \(\{\phi_i\}_{i=1}^{N}\) defined above, we can construct a variant partition of unity.

Actually, we can divide all \(N\) vertices of \(\tau_{H_{\epsilon}}\) into several groups \(G_j, j = 1, 2, \cdots, M, \) with \(\bigcup_{j=1}^{M} G_j = \{x_1, x_2, \cdots, x_N\}\) and \(G_l \cap G_i = 0, l \neq j, \) usually, \(M < N\). Meanwhile, the basis functions shall be classified into several classes \(\Phi_j = \sum_{x_i \in G_j} \phi_i, j = 1, 2, \cdots, M\). Then, each local domain becomes \(\Omega_j = \text{supp} \Phi_j \cap \Omega = \bigcup_{x_i \in G_j} \text{supp} \phi_i \cap \Omega = \bigcup_{x_i \in G_j} D_i \cap \Omega\). Obviously, we shall also define local computational domain of oversampling with zero, one, two, and more layers, as follows.

\[
\Omega^0_j = \Omega_j,
\]

\[
\Omega^1_j = \bigcup_{x_m \in \Omega_j} D^{m,0},
\]

\[
\Omega^2_j = \bigcup_{x_m \in \Omega_j} D^{m,1},
\]

\[
\Omega^{M-1}_j = \bigcup_{x_m \in \Omega_j} D^{m,M-1}.
\]
\[ \Omega_j^2 = \bigcup_{x_m \in \Omega_j} D_{m,1}^j = \bigcup_{x_m \in \Omega_j^1} D_{m,0}^j, \]

... It’s trivial to check that \( \{\Omega_j^2\}_{j=1}^M \) is an open cover of \( \Omega \) and \( \{\Phi_j\}_{j=1}^M \) is a partition of unity subordinate to the cover \( \{\Omega_j^2\}_{j=1}^M \). In this case, we call the variant of partition of unity as "M-PU" according to the number of groups \( M \). Specially, when \( M = N \), it turn back to the standard partition of unity. Since there is no difficulty to derive the same approximation results for our algorithms based on the standard partition of unity or its variant, in the following analysis, we only discuss our algorithms based on the standard partition of unity. However, in numerical simulations, we will show the computational results for both cases.

To get a local a priori estimates which will play a crucial role in our analysis, we need to borrow the following lemma (Lemma 3.2 in [11]).

**Lemma 3.1.** Suppose that \( f \in H^{-1}(\Omega) \) and \( D \subset \subset \Omega_0 \subset \Omega \). Assume that \((w, r) \in (X_h^1(\Omega_0), Q_h^1(\Omega_0))\) satisfies
\[
(\text{3.7}) \\
\mathcal{A}(w, r; v, q) = (f, v) \quad \forall (v, q) \in (X_0^1(\Omega_0), Q_0^1(\Omega_0)),
\]
then
\[
(\text{3.8}) \\
\|w\|_{1,D} + \|r\|_{0,D} \leq C (\|w\|_{0,\Omega_0} + \|r\|_{-1,\Omega_0} + \|f\|_{-1,\Omega_0}).
\]

To derive our local a priori error estimate, based on Lemma 3.1, we present the following local estimate. It worthy noting that the constant \( C \) in (3.8) depends on the diameter of \( \Omega_0 \) (see e.g., [24]). This requires that the scale of \( \Omega \) and hence \( N \) should be \( O(1) \). Therefore, hereafter in this section, we always assume that the triangulation \( \tau_{H_p} \) for partition of unity is fixed, and with scale \( H_p \geq c_0 \) for some constant \( c_0 > 0 \).

**Theorem 3.2.** Assume that each \((u^i, p^i) \in (X_h^1(D^{i,s}), Q_h^1(D^{i,s}))\) with \( s \geq 1 \), is obtained by Step 1 to Step 3 of **ALGORITHM 1**, then
\[
(\text{3.9}) \\
\|u_h - u^i\|_{1,D}^2 + \|p_h - p^i\|_{0,D}^2 \leq C (\|u_h - u^i\|_{0,D}^2 + \|p_h - p^i\|_{0,D}^2 + \|u_h - u_H\|_{1,D^{i,s}}^2 + \|p_h - p_H\|_{1,D^{i,s}}^2) + H^2(\|u_h - u_H\|_{1,D}^2 + \|p_h - p_H\|_{1,D}^2).
\]

**Proof.** The proof is similar as that in [11]. For simplicity, let \( D = D^i, \Omega_0 = D^{i,s}, \) since \( s \geq 1 \), we have \( D \subset \subset \Omega_0 \). By Step 1 to Step 3 of **ALGORITHM 1**, 
\[
(\text{3.10}) \\
\mathcal{A}(u_h - u^i, p_h - p^i; v^h, q^h) = 0 \quad \forall (v^h, q^h) \in (X_0^1(\Omega_0), Q_0^1(\Omega_0)).
\]

By Lemma 3.1, we obtain
\[
\|u_h - u^i\|_{1,D} + \|p_h - p^i\|_{0,D} \leq C (\|u_h - u^i\|_{0,\Omega_0} + \|p_h - p^i\|_{-1,\Omega_0})
\leq C (\|u_h - u_H\|_{0,\Omega_0} + \|p_h - p_H\|_{-1,\Omega_0} + \|e^i\|_{0,\Omega_0} + \|\epsilon^i\|_{-1,\Omega_0}).
\]

To estimate \( |e^i|_{0,\Omega_0} + |\epsilon^i|_{-1,\Omega_0} \), we use the Aubin-Nitsch duality argument. Given any \((g, \rho) \in (L^2(\Omega_0), L^2_0(\Omega_0) \cap H^1(\Omega_0))\), there exists \((w, r) \in (H^1_0(\Omega_0) \cap H^2(\Omega_0), L^2_0(\Omega_0) \cap H^1(\Omega_0))\), such that
\[
(\text{3.11}) \\
\mathcal{A}(v, q; w, r) = (g, v) + (\rho, q) \quad \forall (v, q) \in (H^1_0(\Omega_0), L^2_0(\Omega_0)).
\]
Thanks to the assumption A1, we have \( \|w\|_{2,\Omega_0} + \|r\|_{1,\Omega_0} \leq C(\|g\|_{0,\Omega_0} + \|\rho\|_{1,\Omega_0}) \).
Moreover, assume that \((\mathbf{w}, r_h) \in (X_0^H(\Omega_0), Q_0^H(\Omega_0)) \) and \((\mathbf{w}_H, r_H) \in (X_0^H(\Omega_0), Q_0^H(\Omega_0)) \) and solve
\[
\mathcal{C}(\mathbf{v}, \mathbf{q}; \mathbf{w} - \mathbf{w}_h, r - r_h) = 0 \quad \forall (\mathbf{v}, \mathbf{q}) \in (X_0^H(\Omega_0), Q_0^H(\Omega_0)),
\mathcal{C}(\mathbf{v}, \mathbf{q}; \mathbf{w} - \mathbf{w}_H, r - r_H) = 0 \quad \forall (\mathbf{v}, \mathbf{q}) \in (X_0^H(\Omega_0), Q_0^H(\Omega_0)),
\]
respectively. Then, by the standard mixed finite element error estimates (see [1, 25]), we have
\[
\|\mathbf{w} - \mathbf{w}_h\|_{1,\Omega_0} + \|r - r_h\|_{0,\Omega_0} \leq C h (\|\mathbf{w}\|_{2,\Omega_0} + \|r\|_{1,\Omega_0}) \leq C h (\|g\|_{0,\Omega_0} + \|\rho\|_{1,\Omega_0}),
\|\mathbf{w} - \mathbf{w}_H\|_{1,\Omega_0} + \|r - r_H\|_{0,\Omega_0} \leq C H (\|\mathbf{w}\|_{2,\Omega_0} + \|r\|_{1,\Omega_0}) \leq C H (\|g\|_{0,\Omega_0} + \|\rho\|_{1,\Omega_0}).
\]
Letting \((\mathbf{v}, \mathbf{q}) = (\mathbf{e}, \mathbf{e}')\) in (3.11), and noticing (3.10), we get
\[
(g, e_i) + (r, e_i) = \mathcal{C}(e, e'; w, r) = \mathcal{C}(e, e'; w_h, r_h)
= \mathcal{C}(u_h - u, p_h - p; w_h, r_h)
= \mathcal{C}(u_h - u, p_h - p; w_h, r_h) + \mathcal{C}(u_h - u, p_h - p; w_h, r_h)
= \mathcal{C}(u_h - u, p_h - p; w - w_h, r - r_h) + \mathcal{C}(u_h - u, p_h - p; w - w_h, r - r_h)
\leq C (\|u_h - u\|_{1,\Omega_0} + \|p_h - p\|_{0,\Omega_0}) (\|w - w_h\|_{1,\Omega_0} + \|r - r_h\|_{0,\Omega_0})
\leq C H (\|u_h - u\|_{1,\Omega_0} + \|p_h - p\|_{0,\Omega_0}) (\|g\|_{0,\Omega_0} + \|\rho\|_{1,\Omega_0}),
\]
which yields
\[
\|\mathbf{e}\|_{0,\Omega_0} + \|\mathbf{e}'\|_{-1,\Omega_0} \leq C H (\|u_h - u\|_{1,\Omega_0} + \|p_h - p\|_{0,\Omega_0}).
\]
Finally, for each \(i\), we have
\[
\|u_h - u^i\|_{1,D^\star} + \|p_h - p^i\|_{0,D^\star} \leq C (\|u_h - u\|_{0,D^\star} + \|p_h - p\|_{-1,D^\star} + \|e^i\|_{0,D^\star} + \|e^i\|_{-1,D^\star})
\leq C (\|u_h - u\|_{0,D^\star} + \|p_h - p\|_{-1,D^\star} + H (\|u_h - u\|_{1,D^\star} + \|p_h - p\|_{0,D^\star})).
\]
Squaring both sides of the above inequality and using Hölder inequality, we can derive (3.9).

\[\Box\]

By employing the above local estimate, we have the following theorem concerning the global estimate.

**Theorem 3.3.** Given a regular triangulation \(\tau_h\), its partition of unity \(\{\phi_i\}_{i=1}^N\), and the associated subdomains \(\{D^\star\}_{i=1}^N\), let \((\mathbf{u}, p) \in (H^{k+1}(\Omega), H^k(\Omega))\), applying **Algorithm 1**, we have the following result,
\[
\|\mathbf{u} - u^h\|_{1,\Omega} + \|p - p^h\|_{0,\Omega} \leq C (h^k + H^{k+1}) (\|\mathbf{u}\|_{k+1,\Omega} + \|p\|_{k,\Omega}).
\]

**Proof.** Since \(\{\phi_i\}_{i=1}^N\) form a partition of unity, we have
\[
\mathbf{u}_h = \sum_{i=1}^N \phi_i \mathbf{u}_h, \quad p_h = \sum_{i=1}^N \phi_i p_h.
\]
For any vertex \( x_i \in \tau_{H_p} \), introduce
\[
\omega_i = \{ \text{vertex } x_j \in \tau_{H_p} : x_j \in \text{supp } \phi_i \}
\]
to denote all the vertices sharing same element with \( x_i \) on the triangulation \( \tau_{H_p} \). Then define
\[
M_{\text{max}} = \max_{i=1, \ldots, N} \# \omega_i,
\]
which is independent of \( H_p \) and \( N \) due to the regular triangulation \( \tau_{H_p} \). Then,
\[
\| u_h - u^h \|_{1, \Omega}^2 + \| p_h - p^h \|_{0, \Omega}^2
\]
\[
= \sum_{i=1}^N \| \phi_i (u_h - u^i) \|_{1, D_i}^2 + \sum_{i=1}^N \| \phi_i (p_h - p^i) \|_{1, D_i}^2
\]
\[
\leq \sum_{i=1}^N \left( \| \phi_i (u_h - u^i) \|_{1, D_i}^2 + \| \phi_i (p_h - p^i) \|_{1, D_i}^2 \right)
\]
\[
= \sum_{i=1}^N \left( \# \omega_i \sum_{j \in \omega_i} \| \phi_j (u_h - u^i) \|_{1, D_i}^2 + \# \omega_i \sum_{j \in \omega_i} \| \phi_j (p_h - p^i) \|_{1, D_i}^2 \right)
\]
\[
\leq M_{\text{max}} \sum_{i=1}^N \left( \sum_{j \in \omega_i} \| \phi_j (u_h - u^i) \|_{1, D_i}^2 + \sum_{j \in \omega_i} \| \phi_j (p_h - p^i) \|_{1, D_i}^2 \right)
\]
\[
\leq M_{\text{max}}^2 \sum_{i=1}^N \left( \| \phi_i (u_h - u^i) \|_{1, D_i}^2 + \| \phi_i (p_h - p^i) \|_{1, D_i}^2 \right).
\]
Noticing that \( \text{supp } \phi_i \subseteq D^i \), it satisfies
\[
\| \phi_i \|_{L^\infty (D^i)} \leq C_{\infty}, \quad \| \nabla \phi_i \|_{L^\infty (D^i)} \leq C_G H_p^{-1} \quad \text{with } \text{meas}(D^i) \leq CH_p^d.
\]
Since \( H_p \geq H \) is fixed and independent of \( H \), we get
\[
\| \phi_i (u_h - u^i) \|_{1, \Omega}^2 + \| \phi_i (p_h - p^i) \|_{0, \Omega}^2
\]
\[
\leq \| \phi_i (u_h - u^i) \|_{1, D_i}^2 + \| \phi_i (p_h - p^i) \|_{0, D_i}^2
\]
\[
\leq C \left( \| u_h - u^i \|_{1, D_i}^2 + \| p_h - p^i \|_{0, D_i}^2 \right).
\]
Therefore, by Theorem 3.2, we can deduce that
\[
\| u_h - u^h \|_{1, \Omega}^2 + \| p_h - p^h \|_{0, \Omega}^2
\]
\[
\leq CM_{\text{max}}^2 \sum_{i=1}^N \left( \| u_h - u^i \|_{1, D_i}^2 + \| p_h - p^i \|_{0, D_i}^2 \right)
\]
\[
\leq CM_{\text{max}}^2 \sum_{i=1}^N \left( \| u_h - u_H \|_{0, D_i}^2 + \| p_h - p_H \|_{0, D_i}^2 \right)
\]
\[
+ H^2 \left( \| u_h - u_H \|_{1, D_i}^2 + \| p_h - p_H \|_{0, D_i}^2 \right).
\]
Finally, by using the triangle inequality, we arrive at

\[ \text{and the minimum angle of the regular triangulation } C \]

Here, \( C_{ov}(s, d) \) is a finite integer bounded by the maximal number of elements \( E_i \) contained in each subdomain \( D^{i,s} \) of \( \tau_{H_{i,s}} \), and is actually determined by the layer index \( s \), the dimension \( d \), and the minimum angle of the regular triangulation \( \tau_{H_{i,s}} \), but doesn’t depend on \( N \).

Finally, by using the triangle inequality, we arrive at

\[
\begin{align*}
\|u - u^h\|_{1, \Omega} &+ \|p - p^h\|_{0, \Omega} \\
&\leq \|u - u_h\|_{1, \Omega} + \|p - p_h\|_{0, \Omega} + \|u - u^h\|_{1, \Omega} + \|p - p^h\|_{0, \Omega} \\
&\leq \|u - u_h\|_{1, \Omega} + \|p - p_h\|_{0, \Omega} + C(||u - u^h||_{1, \Omega} + ||p - p^h||_{0, \Omega})^{1/2} \\
&\leq C(h^{k+1}) \{||u||_{k+1, \Omega} + ||p||_{k, \Omega}\}. 
\end{align*}
\]

\( \square \)

Since the approximation \((u^h, p^h)\) is obtained by solving a series of local subproblems which are imposed with artificial homogeneous boundary conditions of the first kind, some local and non-physical oscillations may occur. This will certainly bring some bad influence to the global accuracy of the approximation. To diminish such influences, we improve the approximation \((u^h, p^h)\) by furthermore adding a coarse grid correction procedure to \textsc{Algorithm 1} as follows.

\textbf{Algorithm 2:}

Step 1. Find a global coarse grid solution \((u_H, p_H) \in (X^H_0, Q^H_0)\) such that

\[
(3.14) \quad \mathcal{E}(u_H, p_H; v^H, q^H) = (f, v^H) \quad \forall (v^H, q^H) \in (X^H_0, Q^H_0).
\]

Step 2. Correct the residue \((e^i, \epsilon^i) \in (X^H_0(D^{i,s}), Q^H_0(D^{i,s}))\) on a fine grid of the overlapping subdomain \(D^{i,s}, i \in 1, 2, \ldots, N\), such that

\[
(3.15) \quad \mathcal{E}(e^i, \epsilon^i; v^h, q^h) = (f, v^h) - \mathcal{E}(u_H, p_H; v^h, q^h) \quad \forall (v^h, q^h) \in (X^H_0(D^{i,s}), Q^H_0(D^{i,s})).
\]

Step 3. Update: \((u^i, p^i) = (u_H, p_H) + (e^i, \epsilon^i)\) in \(D^{i,s}\).

Step 4. Construct finite element solution \(u^h = \sum_{i=1}^{N} \phi^i u^i\), \(p^h = \sum_{i=1}^{N} \phi^i p^i\).

Step 5. Find a coarse grid correction \((e^H, \epsilon^H) \in (X^H_0, Q^H_0)\) such that

\[
(3.16) \quad \mathcal{E}(e^H, \epsilon^H; v^H, q^H) = (f, v^H) - \mathcal{E}(u^h, p^h; v^h, q^h) \quad \forall (v^H, q^H) \in (X^H_0, Q^H_0).
\]

Step 6. Obtain the final post-processing approximate solution \((\hat{u}^h, \hat{p}^h) = (u^h + e^H, p^h + \epsilon^H)\).

To derive error estimates for the finite element solution \((\hat{u}^h, \hat{p}^h)\) by \textsc{Algorithm 2} above, we define the projection operator \((R_H, L_H) : (X, Q) \to (X^H_0, Q^H_0)\) by

\[
(3.17) \quad \mathcal{E}(v^H, q^H; u - R_H u, p - L_H p) = 0 \quad \forall (v^H, q^H) \in (X^H_0, Q^H_0),
\]

which is well defined and satisfies the following approximation properties (the similar result can be found in [10]):

\[
(3.18) \quad \|u - R_H u\|_{1, \Omega} + \|p - L_H p\|_{0, \Omega} \leq CH \{||u||_{2, \Omega} + ||p||_{1, \Omega}\}.
\]
Let the partition of unity \( \{ \phi_i \}_{i=1}^N \) and its associated overlapping subdomains \( \{ D_i \}_{i=1}^N \) be given on a regular triangulation \( \tau_{H_k} \). Assume \((u,p) \in (H^{k+1}(\Omega), H^k(\Omega))\), when applying \textbf{Algorithm 2}, we have the following results,

\[ (3.22) \quad \|u - \hat{u}\|_{1,\Omega} + \|p - \hat{p}\|_{0,\Omega} \leq C(h^k + H^{k+1}) \{ \|u\|_{k+1,\Omega} + \|p\|_{k,\Omega} \}, \]

\[ (3.21) \quad \|u - \hat{u}\|_{0,\Omega} \leq C(h^{k+1} + H^{k+2}) \{ \|u\|_{k+1,\Omega} + \|p\|_{k,\Omega} \}. \]

**Proof.** From the coarse grid correction equation (3.16), we can derive

\[ (3.23) \quad \forall (v, q) \in (X^h_0, Q^h_0). \]

Using (3.17), we have

\[ (3.22) \quad \mathcal{E}(e^H, e^H; R_{H}, \mathbb{L}q) = (f, R_{H}v) - \mathcal{E}(u^h, p^h; R_{H}v, \mathbb{L}q) \quad \forall (v, q) \in (X^h_0, Q^h_0). \]

Choosing \((v^h, q^h) = (R_{H}v, \mathbb{L}q)\) in (2.7), we obtain

\[ (3.23) \quad \mathcal{E}(u_h, p_h; R_{H}v, \mathbb{L}q) = (f, R_{H}v) \quad \forall (v, q) \in (X^h_0, Q^h_0). \]

Therefore, combining (3.22) and (3.23) yields

\[ (3.24) \quad \mathcal{E}(e^H, e^H; v, q) = \mathcal{E}(u_h - u^h, p_h - p^h; R_{H}v, \mathbb{L}q) \quad \forall (v, q) \in (X^h_0, Q^h_0). \]

Thanks to (2.4) and the continuous properties of \( \mathcal{E}(\cdot, \cdot) \), we have

\[
\|e^H\|_{1,\Omega} + \|e^H\|_{0,\Omega} \leq \beta^{-1}_1 \sup_{(v,q) \in (X^h_0, Q^h_0)} \frac{\mathcal{E}(e^H, e^H; v, q)}{\|v\|_{1,\Omega} + \|q\|_{0,\Omega}}
\]

\[
= \beta^{-1}_1 \sup_{(v,q) \in (X^h_0, Q^h_0)} \frac{\mathcal{E}(u_h - u^h, p_h - p^h; R_{H}v, \mathbb{L}q)}{\|v\|_{1,\Omega} + \|q\|_{0,\Omega}}
\]

\[
\leq C(\|u_h - u^h\|_{1,\Omega} + \|p_h - p^h\|_{0,\Omega}).
\]

Then, by (3.13) and using the triangle inequality, we get

\[ (3.25) \quad \|u_h - u^h\|_{1,\Omega} + \|p_h - p^h\|_{0,\Omega} \leq \|u - \hat{u}\|_{1,\Omega} + \|p - \hat{p}\|_{0,\Omega} + \|e^H\|_{1,\Omega} + \|e^H\|_{1,\Omega} \leq C(h^k + H^{k+1}) \{ \|u\|_{k+1,\Omega} + \|p\|_{k,\Omega} \}. \]

Using the triangle inequality again, we finally derive

\[ (3.26) \quad \|u - \hat{u}\|_{1,\Omega} + \|p - \hat{p}\|_{0,\Omega} \leq C(h^k + H^{k+1}) \{ \|u\|_{k+1,\Omega} + \|p\|_{k,\Omega} \}. \]

For the \( L^2 \)-error estimate, we use again the Aubin-Nitsche duality argument. Thanks to the assumption A1, by a similar argument with Layton and Tobiska used in [26], for \( u_h - \hat{u}^h \in L^2(\Omega) \), the dual problem is to seek \((\Phi, \Psi) \in (X, Q)\) satisfying

\[ (3.27) \quad \mathcal{E}(v, q; \Phi, \Psi) = (v, u_h - \hat{u}^h) \quad \forall (v, q) \in (X, Q), \]

and

\[ \|\Phi\|_{2,\Omega} + \|\Psi\|_{1,\Omega} \leq C\|u_h - \hat{u}^h\|_{0,\Omega}. \]
Then, taking \((v, q) = (u_h - \hat{u}^h, p_h - \hat{p}^h)\) in (3.27) yields

\[
\|u_h - \hat{u}^h\|^2_{0, \Omega} = \mathcal{C}(u_h - \hat{u}^h, p_h - \hat{p}^h; \Phi, \Psi)
\]

\[
\leq \mathcal{C}(\|u_h - \hat{u}^h\|_{1, \Omega} + \|p_h - \hat{p}^h\|_{0, \Omega})(\|(I - R_H)\Phi\|_{1, \Omega} + \|(I - L_H)\Psi\|_{0, \Omega})
\]

\[
\leq \mathcal{C}H(\|u_h - \hat{u}^h\|_{1, \Omega} + \|p_h - \hat{p}^h\|_{0, \Omega})(\|\Phi\|_{2, \Omega} + \|\Psi\|_{1, \Omega})
\]

\[
\leq \mathcal{C}H(\|u_h - \hat{u}^h\|_{1, \Omega} + \|p_h - \hat{p}^h\|_{0, \Omega})\|u_h - \hat{u}^h\|_{0, \Omega}.
\]

By using the previous result in (3.25),

\[
\|u_h - \hat{u}^h\|_{0, \Omega} \leq \mathcal{C}H(\|u_h - \hat{u}^h\|_{1, \Omega} + \|p_h - \hat{p}^h\|_{0, \Omega}) \leq \mathcal{C}H^{k+2}(\|u\|_{k+1, \Omega} + \|p\|_{k, \Omega}).
\]

By using the properties of projector operator \((R_H, L_H)\), we have

\[
\mathcal{C}(u_h - \hat{u}^h, p_h - \hat{p}^h; (I - R_H)\Phi, (I - L_H)\Psi) = 0.
\]

Thus, with the properties of projector operator \((R_H, L_H)\), we have

\[
\|u_h - \hat{u}^h\|_{0, \Omega} \leq \mathcal{C}H(\|u_h - \hat{u}^h\|_{1, \Omega} + \|p_h - \hat{p}^h\|_{0, \Omega}) \leq \mathcal{C}H^{k+2}(\|u\|_{k+1, \Omega} + \|p\|_{k, \Omega}).
\]

Finally, using the error estimates of the classical mixed finite element method (2.8) and the triangle inequality, we arrive at

\[
\|u - \hat{u}\|_{0, \Omega} \leq \|u - u_h\|_{0, \Omega} + \|u_h - \hat{u}^h\|_{0, \Omega} \leq \mathcal{C}(h^{k+1} + H^{k+2})(\|u\|_{k+1, \Omega} + \|p\|_{k, \Omega}).
\]

\[\square\]

4. Numerical Tests. In all following experiments, the algorithms are implemented using public domain finite element software Freefem++. All simulations were performed on a Dawning parallel cluster composed of 32 nodes (each node consists of eight-core 2.0 GHz CPU, 8\times 2 GB DRAM, and all the nodes are connected together by 20Gbps InfiniBand). The message-passing interface is supported by MPICH.

To verify the analysis results, we consider the following 2D numerical examples. The domain is the unit square domain \(\Omega = [0, 1] \times [0, 1]\) with a uniform triangulation \(\tau_h\) (\(h\) could also be \(H_p, H\)). The stable Taylor-Hood mixed finite element pair \(P_2 - P_1\) is used to solve the Stokes problem.

For convenience of presentation, we introduce the following notations:
- SFEM means the standard finite element method.
- LPPUM means the local and parallel algorithms based on the partition of unity.
- S-LPPUM means LPPUM based on a standard partition of unity.
- M-LPPUM means LPPUM based on a variant partition of unity with \(M\) groups.
- \(P_1\)-PU on \(\tau_{H_p}\) means the partition of unity by the continuous, linear Lagrangian basis functions on the triangulation \(\tau_{H_p}\).
- \(M\)-PU means the variant of partition of unity according to the number of groups. Usually, \(M\) is defined by the number of the processors.
- \(N\) means the number of nodes of \(\tau_{H_p}\) as well as the number of sub-problems.

4.1. Convergence Test. Firstly we consider a smooth problem (referred as Problem 1) with exact solution

\[
u = (u_1, u_2),
\]

\[
u_1 = 10x^2(x - 1)^2y(y - 1)(2y - 1),
\]

\[
u_2 = -10x(x - 1)(2x - 1)y^2(y - 1)^2,
\]

\[
p = 10(2x - 1)(2y - 1),
\]
then, we can get $f$ by (2.1).

According to the results in Theorems 3.3 and 3.4, we should have

$$
\| u - u^k \|_{1, \Omega} + \| p - p^k \|_{0, \Omega} = O(h^2 + H^3),
$$

$$
\| u - u^k \|_{1, \Omega} + \| p - p^k \|_{0, \Omega} = O(h^2 + H^3),
$$

$$
\| u - u^k \|_{0, \Omega} = O(h^3 + H^4).
$$

To verify the optimal convergence orders for $H^1$-norm of velocity and $L^2$-norm of pressure, we shall choose $H$ and $h$ such that $h \sim H^{3/2}$. On the other hand, to reach the optimal convergence order for $L^2$-norm of velocity, we shall fix $H$ and $h$ such that $h \sim H^{4/3}$.

In this test, we always fix $P_1$-PU on $\tau_H$, with $H_p = 1/16$ and $N = 289$, and use $s = 1$, which means one layer oversampling, to test the convergence properties of the LPPUM (S-LPPUM in this subsection, related with ALGORITHM 1 or ALGORITHM 2).

For the $H^1$-error of velocity and $L^2$-error of pressure, we compute the finite element solutions by LPPUM in parallel using 32 processors with coarse mesh sizes $H=1/16$, 1/32, 1/48, 1/64 and corresponding fine mesh sizes $h = 1/64$, 1/192, 1/336, 1/512. SFEM are also solved with the same fine mesh size $h$ using a single processor. The results are presented in Tables 4.1 and 4.2, respectively. Here and thereafter, wall time in Table 4.2 denotes the maximal CPU time among all processors used for LPPUM, including the CPU time for solving the two global coarse grid problems and the subproblems.

For the $L^2$-error of velocity, we execute LPPUM with the same coarse mesh sizes and the fine mesh sizes $h = 1/48$, 1/96, 1/192, 1/256, and SFEM only on corresponding fine meshes. The results are shown in Table 4.3.

### Table 4.1
The errors of SFEM for Problem 1.

| $h$   | $||u - u^k||_{1, \Omega}$ | Order | $||p - p^k||_{0, \Omega}$ | Order | CPU |
|-------|---------------------------|-------|---------------------------|-------|-----|
| 1/64  | 2.05741e-04               |       | 6.30369e-04               |       | 9.54|
| 1/192 | 2.28735e-05               | 1.99947| 7.00409e-05               | 2.00000| 270.08|
| 1/336 | -                         | -     | -                         | -     | -   |
| 1/512 | -                         | -     | -                         | -     | -   |

### Table 4.2
The errors of LPPUM for Problem 1, $h \sim H^{3/2}$.

| $H$   | $h$   | $||u - u^k||_{1, \Omega}$ | Order | $||p - p^k||_{0, \Omega}$ | Order | wall time |
|-------|-------|---------------------------|-------|---------------------------|-------|-----------|
| 1/16  | 1/64  | 5.27119e-04               |       | 6.53205e-04               |       |           |
| 1/32  | 1/192 | 5.43798e-05               | 2.06755| 7.23288e-05               | 2.00313|           |
| 1/48  | 1/336 | 1.54570e-05               | 2.24785| 2.34217e-05               | 2.01488|           |
| 1/64  | 1/512 | 6.14200e-06               | 2.19108| 1.00519e-05               | 2.00825|           |

The observations and conclusions of this experiment are presented as follows:

- From Table 4.1, 4.2, the $H^1$-error of velocity and $L^2$-error of pressure by either ALGORITHM 1 or ALGORITHM 2 can achieve the optimal orders as shown in the theory analysis, and both accuracies are acceptable compared with those of SFEM. And by a coarse correction step, the
approximation solution of ALGORITHM 2 is a little better than that of ALGORITHM 1, especially for $H^1$-norm of velocity. Note that, in our current computational environment, SFEM can’t work with $h$ less than $1/336$, however, our LPPUM could run with small mesh size $h = 1/512$. Meanwhile, LPPUM saves a large amount of computational time.

<table>
<thead>
<tr>
<th>$H$</th>
<th>$h$</th>
<th>$|\mathbf{u} - \mathbf{u}<em>h|</em>{0,\Omega}$ Order</th>
<th>$|\mathbf{u} - \mathbf{u}<em>h^a|</em>{0,\Omega}$ Order</th>
<th>$|\mathbf{u} - \mathbf{u}<em>h^b|</em>{0,\Omega}$ Order</th>
<th>Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/16</td>
<td>1/48</td>
<td>8.66943e-07</td>
<td>1.62467e-05</td>
<td>2.35518e-06</td>
<td></td>
</tr>
<tr>
<td>1/32</td>
<td>1/96</td>
<td>1.08363e-07</td>
<td>3.00006</td>
<td>1.17434e-07</td>
<td>4.32592</td>
</tr>
<tr>
<td>1/48</td>
<td>1/192</td>
<td>1.35455e-08</td>
<td>5.94740e-07</td>
<td>1.94850</td>
<td>3.06221</td>
</tr>
<tr>
<td>1/64</td>
<td>1/256</td>
<td>-</td>
<td>-</td>
<td>2.58413e-07</td>
<td>1.17434</td>
</tr>
</tbody>
</table>

Table 4.3
The $L^2$-errors of SFEM and LPPUM for Problem 1, $h \sim H^{1/2}$.

- From Table 4.3, we can see that, the $L^2$-error of the velocity by LPPUM with ALGORITHM 1 is worse than that of SFEM. The error couldn’t keep the optimal order, and appears to be only $1.94850$ somewhere. Thus, a coarse grid correction step becomes necessary in this aspect. Actually, the solution $\hat{\mathbf{u}}_h$ by LPPUM with ALGORITHM 2 makes significant improvement on $\mathbf{u}_h$, and achieve the optimal order consequently.

To further test our LPPUM, we also consider another smooth problem (referred as Problem 2) with exact solution

$$\mathbf{u} = (u_1, u_2),$$

$$u_1 = \sin(\pi x)^2 \sin(2\pi y),$$

$$u_2 = -\sin(2\pi x) \sin(\pi y)^2,$$

$$p = \cos(\pi x) \cos(\pi y).$$

In present computations, the same parameters $h$, $H$ and numerical methods used in Tables 4.2 and 4.3 are chosen. The results are tabulated in Tables 4.4 and 4.5. From these two tables, we can observe similar phenomena and draw same conclusion as found from Tables 4.2 and 4.3.

<table>
<thead>
<tr>
<th>$H$</th>
<th>$h$</th>
<th>$|\mathbf{u} - \hat{\mathbf{u}}<em>h|</em>{1,\Omega}$ Order</th>
<th>$|p - \hat{p}<em>h|</em>{0,\Omega}$ Order</th>
<th>Wall time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/16</td>
<td>1/64</td>
<td>4.31410e-03</td>
<td>1.25330e-03</td>
<td>5.84</td>
</tr>
<tr>
<td>1/32</td>
<td>1/192</td>
<td>3.69535e-04</td>
<td>9.40079e-05</td>
<td>2.35766</td>
</tr>
<tr>
<td>1/48</td>
<td>1/336</td>
<td>1.7625e-05</td>
<td>2.04559</td>
<td>2.66703</td>
</tr>
<tr>
<td>1/64</td>
<td>1/512</td>
<td>5.03014e-05</td>
<td>2.01672</td>
<td>2.46686</td>
</tr>
</tbody>
</table>

Table 4.4
The errors of LPPUM for Problem 2, $h \sim H^{1/2}$.

4.2. Variant of LPPUM.
In this part, we will test M-LPPUM based on the variant of partition of unity to show the flexibility of LPPUM. Problem 1 is also selected for the present test. In computation, we fix $H = 1/32$, $h = 1/192$, and choose $P_1$-PU on $\tau_{H_p}$, $H_p = 1/16$ with $N = 289$, and the index $s = 1$. Obviously, we have a large number of basis functions.

For M-LPPUM, according to the use of the number of processors $J$, $J = 2$, $4$, $8$, $16$, $32$, we select the related variant of partition of unity $M$-PU with $M = J$. For illustration, the
Table 4.5
The $L^2$-errors of LPPUM for Problem 2, $h \sim H^\frac{3}{4}$.

| $H$  | $h$   | $||\mathbf{u} - \mathbf{u}^h||_{0, \Omega}$ | Order | $||\mathbf{u} - \hat{\mathbf{u}}^h||_{0, \Omega}$ | Order |
|------|-------|------------------------------------------|-------|------------------------------------------|-------|
| 1/16 | 1/48  | 7.47846e-05                              |       | 2.25297e-05                              |       |
| 1/32 | 1/96  | 4.68540e-06                              | 3.99650 | 1.73279e-06                              | 3.70066 |
| 1/48 | 1/192 | 8.97743e-07                              | 2.38380 | 2.16474e-07                              | 3.00083 |
| 1/64 | 1/256 | 2.95121e-07                              | 3.86711 | 9.13108e-08                              | 3.00054 |

| $\mathbf{u}_1^0$, $\mathbf{u}_1^1$ of each $M$-PU are shown in Figure 4.1, in which, the blue region is the original local computational domain $\mathbf{u}_1^0$, while the blue and red regions are the one layer oversampling domain $\mathbf{u}_1^1$. The computational results of M-LPPUM are presented in Table 4.6 with comparison to those of S-LPPUM.

From Table 4.6, in the case when fix $P_1$-PU on $\tau_{H^p}$, we can observe that the errors of M-LPPUM will increase as $M$ goes larger. Especially, when choosing $M = N$ as the largest possible number, M-LPPUM becomes S-LPPUM. The errors by ALGORITHM 2 support this trend, and the results of ALGORITHM 1 have a little perturbation. And the accuracies are also acceptable compared with those of SFEM, especially using M-LPPUM related with ALGORITHM 2.

Fig. 4.1. $\mathbf{u}_1^0$=blue region, $\mathbf{u}_1^1$=blue and red regions for M-PU. From left to right. 2-PU; 4-PU; 8-PU; 16-PU; 32-PU.

Table 4.6
The errors of LPPUM, $P_1$-PU on $\tau_{16}$, $N = 289$.

| LPPUM | $||\mathbf{u} - \mathbf{u}^h||_{1, \Omega}$ | $||p - p^h||_{0, \Omega}$ | $||\mathbf{u} - \hat{\mathbf{u}}^h||_{1, \Omega}$ | $||p - \hat{p}^h||_{0, \Omega}$ |
|-------|------------------------------------------|--------------------------|------------------------------------------|--------------------------|
| 2-LPPUM | 2.28795e-05                              | 7.00420e-05              | 2.28736e-05                              | 7.00409e-05              |
| 4-LPPUM | 2.29953e-05                              | 7.00767e-05              | 2.28736e-05                              | 7.00409e-05              |
| 8-LPPUM | 2.30538e-05                              | 7.05087e-05              | 2.28737e-05                              | 7.00410e-05              |
| 16-LPPUM | 2.31542e-05                             | 7.20808e-05              | 2.28739e-05                              | 7.00410e-05              |
| 32-LPPUM | 6.85250e-05                             | 1.67032e-04              | 2.29787e-05                              | 7.00537e-05              |
| S-LPPUM | 5.43798e-05                             | 7.23288e-05              | 2.36935e-05                              | 7.01097e-05              |
The performance of a parallel algorithm in a homogeneous parallel environment is measured by speedup and parallel efficiency which are commonly calculated by

\[ S_p = \frac{T(J_1)}{T(J_2)}, \quad E_p = \frac{J_1 \times T(J_1)}{J_2 \times T(J_2)}, \]

where \( T(J_1) \) and \( T(J_2) \) (\( J_1 \leq J_2 \)) are the wall time of the parallel program using \( J_1 \) and \( J_2 \) processors, respectively.

Table 4.7 reports the wall time of M-LPPUM in a parallel environment using \( J \) number of processors with \( J = 2, 4, 8, 16, 32 \), and presents the corresponding speedup and parallel efficiency which are computed by comparison with \( J_1 = 2 \). We also show the results of S-LPPUM in Table 4.8. Figure 4.2 intuitively describes the evolution of the speedup, parallel efficiency and wall time with the number of processors, respectively.

From Tables 4.7-4.8 and Figure 4.2, we can see that our LPPUM, especially, S-LPPUM, has a good parallel performance. We also observe that, although S-LPPUM has a better parallel performance than M-LPPUM, however, wall time of S-LPPUM is always larger than that of M-LPPUM.

In summary, our LPPUM is highly efficient. When using small number of processors, M-LPPUM is obviously a better choice, while S-LPPUM should be chosen for a large number of processors.

![Table 4.7](image)

Wall time \( T(J) \) in seconds, speedup \( S_p \) and parallel efficiency \( E_p \) of ALGORITHM 2 with M-LPPUM, \( M = J \), \( P_{1\cdot PU} \) on \( \tau_{H_p} \). \( H_p = 1/16, N = 289, H = 1/32, h = 1/192 \).

<table>
<thead>
<tr>
<th>( J )</th>
<th>2</th>
<th>4</th>
<th>8</th>
<th>16</th>
<th>32</th>
</tr>
</thead>
<tbody>
<tr>
<td>( T(J) )</td>
<td>216.88</td>
<td>135.38</td>
<td>86.73</td>
<td>62.68</td>
<td>48.96</td>
</tr>
<tr>
<td>( S_p = \frac{T(J_1)}{T(J_2)} )</td>
<td>1.00</td>
<td>1.60</td>
<td>2.50</td>
<td>3.46</td>
<td>4.43</td>
</tr>
<tr>
<td>( E_p = \frac{J_1 \times T(J_1)}{J_2 \times T(J_2)} )</td>
<td>1.00</td>
<td>0.80</td>
<td>0.63</td>
<td>0.43</td>
<td>0.28</td>
</tr>
</tbody>
</table>

![Table 4.8](image)

Wall time \( T(J) \) in seconds, speedup \( S_p \) and parallel efficiency \( E_p \) of ALGORITHM 2 with S-LPPUM, \( P_{1\cdot PU} \) on \( \tau_{H_p} \). \( H_p = 1/16, N = 289, H = 1/32, h = 1/192 \).

<table>
<thead>
<tr>
<th>( J )</th>
<th>2</th>
<th>4</th>
<th>8</th>
<th>16</th>
<th>32</th>
</tr>
</thead>
<tbody>
<tr>
<td>( T(J) )</td>
<td>1204.28</td>
<td>617.88</td>
<td>304.02</td>
<td>151.87</td>
<td>86.56</td>
</tr>
<tr>
<td>( S_p = \frac{T(J_1)}{T(J_2)} )</td>
<td>1.00</td>
<td>1.95</td>
<td>3.96</td>
<td>7.93</td>
<td>13.91</td>
</tr>
<tr>
<td>( E_p = \frac{J_1 \times T(J_1)}{J_2 \times T(J_2)} )</td>
<td>1.00</td>
<td>0.97</td>
<td>0.99</td>
<td>0.99</td>
<td>0.87</td>
</tr>
</tbody>
</table>

Then, we take some tests for the comparison of our algorithm with the local and parallel algorithm in [11] (referred as He-Xu-Zhou-LP for simplicity). Problem 1 is also used for testing example. Since it’s difficult to implement He-Xu-Zhou-LP at the element level, we divide the computational domain \( \Omega \) into four subdomains, \( S_0^1 = [0, 1] \times [0, 1/4] \), \( S_0^2 = [0, 1] \times [1/4, 1/2] \), \( S_0^3 = [0, 1] \times [1/2, 3/4] \), \( S_0^4 = [0, 1] \times [3/4, 1] \). Then we choose \( \tau_{H_p} = 1/16 \), and enlarge each \( S_0^j(j = 1, 2, 3, 4) \) by one layer of oversampling to obtain subdomain \( S_j^1 \). Hence, we have \( M = 4 \), and define ALGORITHM 1 with M-LPPUM by M-LPPUM A1 for simplicity. Moreover, since He-Xu-Zhou-LP doesn’t obtain a global finite element solution, we recall and use the piecewise norms to measure the approximate errors of velocity and pressure (cf. [11]):

\[ |||u - u^h|||_{1,\Omega} = \left( \sum_{j=1}^{4} |||u - u^h|||_{1,S_0^j}^2 \right)^{1/2}, \quad |||p - p^h|||_{0,\Omega} = \left( \sum_{j=1}^{4} |||p - p^h|||_{0,S_0^j}^2 \right)^{1/2}. \]
Now we shall compare our M-LPPUM A1 with one layer oversampling with the algorithm of He-Xu-Zhou-LP. Both algorithms are computed in the subdomains with similar sizes and oversampling. The computational results of both algorithms are listed in Table 4.9. From this table, we can see that both methods get the optimal convergence orders, and reach the similar accuracies. In Table 4.9, we also show the wall time (last column) of both algorithms, and the CPU time for data communication (in brackets of last column) of our M-LPPUM A1. The communication time is only for constructing a global solution (Step 4 in \textsc{Algorithm 1}). We can find that, excluding the communication time, both algorithms spend nearly the same time. It is worthy noting that, after this step of constructing the global solution, we can execute the coarse grid correction step (Step 5 in \textsc{Algorithm 2}) to improve the resulting finite element approximate solution. The effect of this correction step in improving the accuracy of approximate velocity in $L^2$ norm up to the optimal convergence order has been verified in previous subsection.

4.3. The driven cavity flow. The last example is the ‘lid driven cavity’ based on Stokes equations, which is a popular benchmark problem for testing numerical schemes. In this test, fluid is enclosed in a square box, with an imposed velocity of unity in the horizontal direction on the top boundary, and a no slip condition on the remaining walls. And we set $f = 0$, $\nu = 1$.

Following the above discussions in subsection 4.1, 4.2, we only consider S-LPPUM, and denote \textsc{Algorithms 1 and 2} by S-LPPUM 1, S-LPPUM 2, respectively.

As discussed above, for simplicity, we also fix $P_1$-PU on $\tau_{H_p}$, $H_p = 1/16$, let $s = 1$, $H = 32$ and $h = 192$, and compare it with SFEM on the same fine mesh with size $h = 1/192$. We portion out these subproblems equally (almost) according to 2, 4, 8, 16 and 32 processors.

In particular, we draw the $x$- and $y$-components of velocity along the vertical and horizontal centerlines for SFEM and S-LPPUM 1, S-LPPUM 2 in Figure 4.3. The data of S-LPPUM are

\begin{table}[h]
\centering
\caption{The comparison of M-LPPUM A1 and He-Xu-Zhou-LP, $M = 4$, $h \sim H^2$.}
\begin{tabular}{llcccc}
\hline
He-Xu-Zhou-LP & $H$ & $h$ & $|||u - u_h|||_{1,\Omega}$ Order & $|||p - p_h|||_{0,\Omega}$ Order & wall time \\
\hline
1/16 & 1/64 & 2.06561e-04 & 6.31702e-04 & 2.76 \\
1/32 & 1/192 & 2.29079e-05 & 2.00172 & 36.16 \\
1/48 & 1/336 & 7.47369e-06 & 2.00417 & 223.04(45.51) \\
\hline
M-LPPUM A1 & $H$ & $h$ & $|||u - u_h|||_{1,\Omega}$ Order & $|||p - p_h|||_{0,\Omega}$ Order & wall time (comm.) \\
\hline
1/16 & 1/64 & 2.07553e-04 & 6.31042e-04 & 3.03(0.34) \\
1/32 & 1/192 & 2.29653e-05 & 2.00380 & 45.76(6.64) \\
1/48 & 1/336 & 7.48141e-06 & 2.00417 & 223.04(45.51) \\
\hline
\end{tabular}
\end{table}
all close to that of SFEM, and good agreements verify our S-LPPUM.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure1}
\caption{Left, x component of velocity along the vertical centerline; right, y component of velocity along the horizontal centerline.}
\end{figure}

We also draw the pressure contour lines for the driven cavity by using SFEM and S-LPPUM 1, S-LPPUM 2 in Figure 4.4, from which we can observe that the globally continuous pressure by S-LPPUM are obtained without any oscillation, and the agreement of their profiles with that of SFEM implies that LPPUM is stable.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure2}
\caption{The pressure contour lines. From left to right: SFEM; S-LPPUM 1; S-LPPUM 2.}
\end{figure}

Finally, Tables 4.10 and 4.11 present the wall time of S-LPPUM 1, S-LPPUM 2 and their corresponding speedup $S_p$ and parallel efficiency $E_p$, respectively. By a more intuitive way, Figure 4.5 shows their evolution of speedup $S_p$, parallel efficiency $E_p$ and wall time with respect to the number of processors.

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|}
\hline
$J$ & 2 & 4 & 8 & 16 \\
\hline
$T(J)$ & 1196.38 & 605.92 & 320.74 & 154.31 & 85.97 \\
$S_p = \frac{T(2)}{T(J)}$ & 1.00 & 1.97 & 3.73 & 7.75 & 13.92 \\
$E_p = \frac{2 \cdot T(2)}{T(J)}$ & 1.00 & 0.99 & 0.93 & 0.97 & 0.87 \\
\hline
\end{tabular}
\caption{Wall time $T(J)$ in seconds, speedup $S_p$ and parallel efficiency $E_p$ of S-LPPUM 1, $P_1$-PU on $\tau_{H_p}, H_p = 1/16, N = 289$.}
\end{table}

Both Tables 4.10-4.11 and Figure 4.5 demonstrate that our parallel algorithms retain good parallel performance. Nearly linear speedup and parallel efficiency can be observed by S-
Table 4.11
Wall time $T(J)$ in seconds, speedup $S_p$ and parallel efficiency $E_p$ of S-LPPUM 2, $P_1$-PU on $\tau_{H_p}, H_p = 1/16$, $N = 289$.

<table>
<thead>
<tr>
<th>$J$</th>
<th>2</th>
<th>4</th>
<th>8</th>
<th>16</th>
<th>32</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T(J)$</td>
<td>1203.49</td>
<td>613.08</td>
<td>327.44</td>
<td>160.52</td>
<td>91.83</td>
</tr>
<tr>
<td>$S_p = \frac{T(2)}{T(J)}$</td>
<td>1.00</td>
<td>1.96</td>
<td>3.68</td>
<td>7.50</td>
<td>13.11</td>
</tr>
<tr>
<td>$E_p = \frac{2 \times T(2)}{J \times T(J)}$</td>
<td>1.00</td>
<td>0.98</td>
<td>0.92</td>
<td>0.94</td>
<td>0.82</td>
</tr>
</tbody>
</table>

![Graphs showing speedup, parallel efficiency, and wall time](image)

**Fig. 4.5.** Left, evolution of speedup $\frac{T(2)}{T(J)}$ with number of processors; middle, evolution of parallel efficiency $\frac{2 \times T(2)}{J \times T(J)}$ with number of processors; right, wall time with number of processors.

LPPUM 1 and S-LPPUM 2. From these Tables and Figure, we also see that, both S-LPPUM 1 and S-LPPUM 2 spend almost the same wall time, which implies that the coarse correction step in S-LPPUM 2 requires very little computational time. However, as indicated in Table 4.3 in Subsection 4.1, S-LPPUM 2 can improve the velocity in the sense of $L^2$ norm, and requires very little extra time than S-LPPUM 1, thus, S-LPPUM 2 is a better choice in practice.

In summary, by above discussions, our local and parallel algorithms based on the partition of unity are highly efficient and flexible.

5. Conclusions. In this paper, we have designed two local and parallel finite element algorithms based on the partition of unity for the Stokes problem. The strategy for partition of unity makes the parallel algorithms more flexible and highly efficient. Moreover, a globally continuous finite element solution is constructed. These two algorithms can be extended to a variety of partial differential equations, such as nonlinear problems like Navier-Stokes equations. Besides, adaptive strategies for oversampling and refinements are also our new challenges and further researches.

REFERENCES