FAST FACTORIZATION UPDATE FOR GENERAL ELLIPTIC EQUATIONS UNDER MULTIPLE COEFFICIENT UPDATES

XIAO LIU*, JIANLIN XIA†, AND MAARTEN V. DE HOOP‡

Abstract. For discretized elliptic equations, we develop a new factorization update algorithm that is suitable for incorporating coefficient updates with large support and large magnitude in subdomains. When a large number of local updates are involved, in addition to the standard factors in various (interior) subdomains, we precompute some factors in the corresponding exterior subdomains. Exterior boundary maps are constructed hierarchically. The data dependencies among tree-based interior and exterior factors are exploited to enable extensive information reuse. For coefficient updates in a subdomain, only the interior problem in that subdomain needs to be re-factorized and there is no need to propagate updates to other tree nodes. The combination of the new interior factors with a chain of existing factors quickly provides the new global factor and thus an effective solution algorithm. The introduction of exterior factors avoids updating higher-level subdomains with large system sizes, and makes the idea suitable for handling multiple occurrences of updates. The method can also accommodate the case when the support of updates moves.

Key words. elliptic equations, coefficient update, fast factorization update, exterior boundary map, exterior factor, Schur complement domain decomposition

AMS subject classifications. 15A23, 65F05, 65N22, 65Y20

1. Introduction. In the solution of elliptic partial differential equations (PDEs) in practical fields such as inverse problems and computational biology, it often needs to update the coefficients associated with subdomains. For example, one key application in inverse problems is the iterative reconstruction of the wavespeed governed by the Helmholtz equation, which needs to incorporate modified coefficients into the following reference problem:

$$Lu = f \text{ in } D, \quad L = -\nabla \cdot p_2(x)\nabla + p_1(x) \cdot \nabla + p_0(x),$$

where $D$ is the domain of interest, $p_0(x)$, $p_1(x)$, and $p_2(x)$ are coefficient functions of the partial differential operator $L$. After discretizations with continuous Galerkin or finite difference approaches, we get a system of linear equations with a sparse coefficient matrix.

1.1. Coefficient update problem. Given the reference problem (1.1), the coefficient update problem is written as

$$\tilde{L}u = f \text{ in } D, \quad \tilde{L} = -\nabla \cdot \tilde{p}_2(x)\nabla + \tilde{p}_1(x) \cdot \nabla + \tilde{p}_0(x),$$

where $\tilde{p}_0(x)$, $\tilde{p}_1(x)$, and $\tilde{p}_2(x)$ are the modified coefficients and $\tilde{u}$ is the new solution. The modification is localized if the coefficient update $(\tilde{L} - L)$ has small support. Assuming that we know the reference solution $u$ of (1.1), then (1.2) is equivalent to

$$\tilde{L}(\tilde{u} - u) = f - \tilde{L}u = (L - \tilde{L})u.$$

*Department of Computational and Applied Mathematics, Rice University, Houston, TX 77005 (xiao.liu@rice.edu)
†Department of Mathematics and Department of Computer Science, Purdue University, West Lafayette, IN 47907 (xiaj@purdue.edu). The research of Jianlin Xia was supported in part by NSF CAREER Award DMS-1255416 and NSF Grant DMS-1819166.
‡Department of Computational and Applied mathematics, Rice University, Houston, TX 77005 (mdehoop@rice.edu). Maarten V. de Hoop gratefully acknowledges support from the Simons Foundation under the MATH + X program, NSF under grant DMS-1559587, and the corporate members of the Geo-Mathematical Group at Rice University and Total.

This manuscript is for review purposes only.
Note that the right-hand side of (1.3) has the same local support as the coefficient update.

There are several strategies for solving either (1.2) or (1.3). For iterative solution, one can either reuse the preconditioner for $L$ or perform additional changes for better convergence. For direct solution, if there is only a small amount of local updates, then the Sherman-Morrison-Woodbury (SMW) formula may be used. However, if there is a sequence of many local updates, then a factorization update from $L$ to $\hat{L}$ is preferred. The primary focus of this paper is to develop a fast factorization update algorithm in direct solution. Our algorithm has nearly optimal complexity for the update of the factorization, and is effective for handling modifications ($\hat{L} - L$) supported at various different locations.

Note that (1.2) can also be formulated as integral equations. Applying the solution operator $G$ of (1.1) to both sides of (1.2), we get

\[(I + G(\hat{L} - L))\hat{u} = u.\]

Restricting to the support of ($\hat{L} - L$), we get the Lippmann-Schwinger integral equation. For direct solutions, (1.4) is not suitable since dense factorization in subdomains can be expensive. Boundary integral formulations may be more suitable because of the reduced system size, and are in fact related to our approach.

1.2. Existing work. Sparse direct solvers provide robust solutions to the fixed reference problem (1.1). After nested dissection reordering [10], the factorization of an $n \times n$ sparse discretized matrix generally costs $O(n^{3/2})$ in 2D, and $O(n^2)$ in 3D. Recent software packages provide the option of solving sparse right-hand sides, for example MUMPS [24, 27] and PARDISO [28, 25]. A similar factorization process can be derived from Schur-complement domain decomposition strategies [5, 13, 16, 22, 26].

In the recent years, rank-structured representations are developed to effectively compress fill-in and obtain fast factorizations of elliptic problems. Several such representations are $H$ matrices [14], $H^2$ matrices [15], and hierarchically semiseparable (HSS) matrices [3, 33]. Sparse factorization with HSS operations is proposed in [12, 30, 31, 32].

Updating LU factorizations of general matrices has been studied in [2, 4, 7, 11]. For sparse factorizations, these methods propagate updates from child nodes to ancestors in elimination trees. For integral operators, updates to local geometries and kernels are studied in [8, 23, 34]. In [8], the update of the structures and the values of hierarchical matrices under adaptive refinement is discussed. In [23], the changes are propagated bottom-up in a quadtree. The SMW formula is used in [34] to compute the action of the inverse. For all of these methods, the updates are typically restricted to a few entries or low-rank updates. If the updates have large support or move locations, these methods may become inefficient.

For updating the coefficients in the PDE problem (1.2), the amount of modifications can be large due to the volumetric change in the support of ($\hat{L} - L$). For such a situation, it is beneficial to decompose the problem into a modified interior problem and a fixed exterior problem. This idea traces back to [18, 19], where boundary integral equations are formulated for piecewise constant media. For inhomogeneous reference problems, related formulations are developed in [17, 29], where the fundamental solution is replaced by the inverse matrix of some finite difference stencil. In order to efficiently precompute selected parts of the inverse, the location of the updates usually needs to be fixed.
1.3. Overview of the proposed method. In this work, we design a fast factorization update algorithm that is suitable for handling multiple volumetric updates. The method has a precomputation step that factorizes the reference problem in various interior and exterior subdomains. When the problem changes, re-factorizations are done only for those subdomains containing the changes, and the solution is updated by solving (1.3) using the locality of the right-hand side.

The method starts from a domain partitioning governed by a binary tree (denoted by $T$), similarly to related direct solvers. In the factorization of the reference problem, interior boundary value problems for adjacent subdomains are combined by eliminating their shared interface. The work is bottom-up in $T$. That is, child nodes pass data to parents.

For solving coefficient update problems with a relatively large amount of updates, we precompute additional factors following a top-down traversal of $T$ before knowing the specific region or value of perturbations. This top-down process constructs factors for exterior boundary value problems, which helps to bypass existing data dependencies. Then for the solution of (1.3), we only re-factorize the smallest subdomain containing the updates, and select existing factors of exterior problems which remain unchanged. For each subtree $\tilde{T} \subset T$ corresponding to the updates, the solution update algorithm treats the nodes inside and outside $\tilde{T}$ separately. Inside $\tilde{T}$, the solution algorithm is similar to the traditional one, but requires the factors of the updated system. Outside $\tilde{T}$, a boundary value problem is solved using the factorization of the exterior problems.

The advantages of our method include:

- For the factorization update, the use of tree-based interior and exterior factors enables us to change only the factors inside the region of coefficient updates, namely, only the nodes in $\tilde{T}$. There is no propagation of updates to other nodes. Thus, the factorization update cost only depends on the size of the updates instead of the total number of unknowns.

- The method is suitable for incorporating coefficient updates with large support and large magnitude in subdomains.

- Because the precomputation prepares for coefficient updates in any subtree of $T$, the supports of updates are allowed to move.

- Regarding the discretized Green’s function, the explicit precomputation and storage of relevant dense matrices are replaced by fast and flexible matrix-vector products. The matrix-vector products support local applications inside certain subdomains.

The method is tested on the transmission problem for the Helmholtz equation. The precomputation has the same scaling as related direct factorizations. The method is especially suitable for large number of changes (e.g. $10^5$ nodals), because the re-factorization cost is independent of the total number of unknowns.

The remaining sections are organized as follows. We formulate the interior and exterior problems in Section 2. Hierarchical factorization algorithms are developed in Section 3 for the coefficient update problems. The algorithm complexity is estimated in Section 4 and is supported by the performance tests in Section 5. Some conclusions are drawn in Section 6.

2. Interior and exterior problems and basic solution update methods.

Factorization update problems can be complicated in general because there are many different scenarios regarding the locations and sizes of the updates. We first present our method for the simplest case and then generalize it to more advanced forms. In

This manuscript is for review purposes only.
Section 2.1, updates in fixed locations are solved by a one-level relation between an interior and an exterior problem. In Section 2.2, a two-level method gives additional flexibility to change the locations and sizes of the updates.

The problem of changing the coefficient in the interior of a subdomain is originally formulated and solved using potential theories, see for example [19, Theorem 4.1]. Note that the fundamental solution (free-space Green’s function) is challenging to compute or to store in inhomogeneous media. We choose instead a Schur-complement domain decomposition formulation, which focuses on solving sub-problems on the boundaries of subdomains.

For a certain subdomain \( \Omega \subset D \), we start by introducing unknowns on the boundary \( \partial \Omega \) and in the interior \( \Omega \). Consider an auxiliary local PDE problem

\[ \begin{cases}
    Lu^{(\Omega)} = f^{(\Omega)} \quad \text{in } \Omega, \\
    \alpha u^{(\Omega)} + \beta \nu \cdot \left(p_2 \nabla u^{(\Omega)}\right) = g^{(\Omega)} \quad \text{on } \partial \Omega,
\end{cases} \tag{2.1} \]

where \( L \) is defined in (1.1) with leading-order coefficient function \( p_2(x) \), \( f^{(\Omega)} \) is the interior source, \( g^{(\Omega)} \) is the boundary source, \( \nu \) is the outward unit normal vector with respect to \( \partial \Omega \), and \( \alpha, \beta \) are two scalar coefficients. The solution \( u^{(\Omega)} \) generates the boundary data \( \tilde{g}^{(\Omega)} \) on \( \partial \Omega \) defined as

\[ \tilde{g}^{(\Omega)} = \hat{\alpha} u^{(\Omega)} + \hat{\beta} \nu \cdot \left(p_2 \nabla u^{(\Omega)}\right) \quad \text{on } \partial \Omega, \tag{2.2} \]

where \( \hat{\alpha}, \hat{\beta} \) are scalar coefficients such that \( \tilde{g}^{(\Omega)} \) is not a scalar multiple of \( g^{(\Omega)} \).

Next, we introduce solution operators of the local problem (2.1), and they involve the boundary-boundary, interior-boundary, boundary-interior, and interior-interior interactions for the subdomain \( \Omega \). For given \( f^{(\Omega)} \) and \( g^{(\Omega)} \), the solution of (2.1) is expressed as

\[ u^{(\Omega)} = G^{(\Omega)} f^{(\Omega)} + K^{(\Omega)} g^{(\Omega)}, \tag{2.3} \]

where \( G^{(\Omega)} \) is the interior solution operator, the kernel of which is the Green’s function, and \( K^{(\Omega)} \) is the solution operator of the corresponding boundary value problem. \( \tilde{g}^{(\Omega)} \) also has a linear relation with \( f^{(\Omega)} \) and \( g^{(\Omega)} \)

\[ \tilde{g}^{(\Omega)} = T^{(\Omega)} g^{(\Omega)} + S^{(\Omega)} f^{(\Omega)}, \tag{2.4} \]

where \( T^{(\Omega)} \) is the boundary map between the boundary source \( g^{(\Omega)} \) and the boundary data \( \tilde{g}^{(\Omega)} \), and \( S^{(\Omega)} \) is the linear map from the interior source \( f^{(\Omega)} \) to \( \tilde{g}^{(\Omega)} \).

After discretizations, (2.3)–(2.4) become matrix-vector multiplications that can be combined as

\[ \begin{pmatrix} \tilde{g}^{(\Omega)} \\ u^{(\Omega)} \end{pmatrix} = \begin{pmatrix} T^{(\Omega)} & S^{(\Omega)} \\ K^{(\Omega)} & G^{(\Omega)} \end{pmatrix} \begin{pmatrix} g^{(\Omega)} \\ f^{(\Omega)} \end{pmatrix}. \tag{2.5} \]

The size of \( T^{(\Omega)} \) is usually smaller than the other blocks \( (S^{(\Omega)}, K^{(\Omega)}, \text{ and } G^{(\Omega)}) \), because \( \partial \Omega \) is one dimension lower than \( \Omega \). In Section 2.1, we show how (2.5) is used to solve the coefficient update problem. Starting from Section 2.2, we improve the efficiency by considering the factorizations inside \( \Omega \) and avoiding forming large matrices explicitly. For the rest of the paper, we use linear algebra notation for ease of exposition.
2.1. One-level method and interior and exterior problems. We show the basic idea of solving the coefficient update problem (1.3) by combining the information of interior and exterior subdomains. For coefficient updates supported in $\Omega$, (2.5) is insufficient because $g^{(\Omega)}$ is unknown. To get the unknowns on $\partial \Omega$, we need to consider the exterior subdomain $\Omega^c := D \setminus \Omega$, which is the relative complement of $\Omega$’s closure in $D$. There is one level of domain partitioning, where $\Omega$ and $\Omega^c$ are level-one subdomains of $D$.

Similar to (2.5), for the exterior subdomain $\Omega^c$, we have

$$
(2.6) \quad \begin{pmatrix} g^{(\Omega^c)} \\ u^{(\Omega^c)} \end{pmatrix} = \begin{pmatrix} T^{(\Omega^c)} & S^{(\Omega^c)} \\ K^{(\Omega^c)} & G^{(\Omega^c)} \end{pmatrix} \begin{pmatrix} g^{(\Omega')} \\ f^{(\Omega')} \end{pmatrix},
$$

which contains the solution operators to the problem (2.1) with $\Omega$ replaced by $\Omega^c$.

Choosing a special case of Robin-to-Robin map such that $\alpha \beta \neq 0$ in (2.1) and $(\hat{\alpha}, \hat{\beta}) = (\alpha, -\beta)$ in (2.2), then the transmission condition on $\partial \Omega$ is

$$
(2.7) \quad g^{(\Omega)} = \hat{g}^{(\Omega^c)}, \quad g^{(\Omega^c)} = \hat{g}^{(\Omega)},
$$

because the outward normal changes sign across $\partial \Omega$. By eliminating $\hat{g}^{(\Omega)}$ and $\hat{g}^{(\Omega^c)}$ in (2.5)–(2.6), we get

$$
(2.8) \quad \begin{pmatrix} T^{(\Omega)} & -I & S^{(\Omega)} & 0 \\ -I & T^{(\Omega^c)} & 0 & S^{(\Omega^c)} \\ K^{(\Omega)} & 0 & G^{(\Omega)} & 0 \\ 0 & K^{(\Omega^c)} & 0 & G^{(\Omega^c)} \end{pmatrix} \begin{pmatrix} g^{(\Omega)} \\ g^{(\Omega^c)} \\ f^{(\Omega)} \\ f^{(\Omega^c)} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ u^{(\Omega)} \\ u^{(\Omega^c)} \end{pmatrix}.
$$

Let

$$
(2.9) \quad M^{(\partial \Omega)} = \begin{pmatrix} T^{(\Omega)} & -I \\ -I & T^{(\Omega^c)} \end{pmatrix}.
$$

The solution operator in $D$ is the Schur complement of $M^{(\partial \Omega)}$ in (2.8) as follows:

$$
(2.10) \quad G^{(\Omega)} = \begin{pmatrix} G^{(\Omega)} \\ G^{(\Omega^c)} \end{pmatrix} - \begin{pmatrix} K^{(\Omega)} \\ K^{(\Omega^c)} \end{pmatrix} (M^{(\partial \Omega)})^{-1} \begin{pmatrix} S^{(\Omega)} \\ S^{(\Omega^c)} \end{pmatrix}.
$$

The coefficient update problem (1.3) can be solved by computing matrix-vector products of $G^{(\Omega)}$ using (2.10). The boundary map matrices need to be formed explicitly in order to factorize $M^{(\partial \Omega)}$, but the remaining ones can be implicit as long as matrix-vector products can be performed.

Based on the current formulation, we propose an algorithm for directly solving the simplest coefficient update problem in which the region of modifications $\Omega$ is known. The factorization operations related to the reference operator $L$ include:

1. Factorize $L$ in $\Omega$ so that the matrix-vector product (2.5) can be computed by direct solutions.
2. Factorize $L$ in $\Omega^c$ similarly for (2.6).
3. Factorize $M^{(\partial \Omega)}$ in (2.9).

Then for each new problem $\hat{L}u = f$, the solution process is:

1. Solve $Lu = f$ by multiplying (2.10) with $f$.
2. Update the factors of $L$ to get those of $\hat{L}$ in $\Omega$.
3. Solve $\hat{L} (\hat{u} - u) = (L - \hat{L})u$ by multiplying (2.10) with $(L - \hat{L})u$. 

This manuscript is for review purposes only.
If $\Omega$ is much smaller than $D$, the method is very effective because the factorization in $\Omega$ is much cheaper than that in $D$. The last step of solution does not involve $G^{(\Omega^c)}$, $S^{(\Omega^c)}$ because the right-hand side is supported in $\Omega$.

**Remark 2.1.** Before describing more sophisticated generalizations, we show that this method can already be beneficial for coefficient updates in disjoint locations. If the problem can be modified in at most $J$ subdomains denoted by $\{\Omega_j : j = 1, 2, \ldots, J\}$ with disjoint closure, then we choose $\Omega = \bigcup_j \Omega_j$ as their union. The solution update method can be described as:

1. Factorize $L$ in $\Omega^c$, and $\hat{L}$ in each $\Omega_j$.
2. Compute $\hat{u} - u$ by multiplying (2.10) with $(L - \hat{L})u$. Note that each operator for $\Omega$ is decoupled, for example,

$$T^{(\Omega)} = \text{diag}(T^{(\Omega_1)}, T^{(\Omega_2)}, \ldots, T^{(\Omega_J)}),$$

where $\text{diag}()$ is used to denote a block diagonal matrix.

Because of the decoupled forms, the method is essentially still a one-level method and the level-one subdomains are $\Omega_1, \Omega_2, \ldots, \Omega_J$, and $\Omega^c$.

**2.2. Two-level method.** If a level-one subdomain $\Omega$ is partitioned further into two non-overlapping subdomains $\Omega_1, \Omega_2$, and coefficient updates may be restricted to one of the subdomains, then based on (2.10), there are three equivalent representations of the solution kernel:

$$\begin{align*}
(2.11) \quad G^{(D)} &= \begin{pmatrix} G^{(\Omega)} \\ G^{(\Omega^c)} \end{pmatrix} - \begin{pmatrix} K^{(\Omega)} \\ K^{(\Omega^c)} \end{pmatrix} (M^{(\partial \Omega)})^{-1} \begin{pmatrix} S^{(\Omega)} \\ S^{(\Omega^c)} \end{pmatrix} \\
&= \begin{pmatrix} G^{(\Omega_1)} \\ G^{(\Omega_2)} \end{pmatrix} - \begin{pmatrix} K^{(\Omega_1)} \\ K^{(\Omega_2)} \end{pmatrix} (M^{(\partial \Omega_1)})^{-1} \begin{pmatrix} S^{(\Omega_1)} \\ S^{(\Omega^c)} \end{pmatrix} \\
&= \begin{pmatrix} G^{(\Omega_2)} \\ G^{(\Omega_2)} \end{pmatrix} - \begin{pmatrix} K^{(\Omega_2)} \\ K^{(\Omega_2)} \end{pmatrix} (M^{(\partial \Omega_2)})^{-1} \begin{pmatrix} S^{(\Omega_2)} \\ S^{(\Omega^c)} \end{pmatrix}.
\end{align*}$$

One can observe that these three representations select the interior subdomain as $\Omega$, $\Omega_1$, and $\Omega_2$ respectively. Here, we discuss the procedure to generate all the components in (2.11), and how to solve the problem by fast matrix-vector products of (2.11).

The direct method is based on the inherent dependencies among different subdomains. The set of subdomains has a partial order governed by the subset relation “$\subseteq$”. The graph in Figure 2.1 visualizes the partial order, each edge of which starts from a subset and points to a superset. Three tree structures can be extracted from the graph in Figure 2.1, which are illustrated separately in Figure 2.2. According to the support of coefficient modifications, one of the tree structure can be selected to solve the problem:

- For modifications in $\Omega$, the interior subdomain is $\Omega$ which contains $\Omega_1$ and $\Omega_2$, and the exterior subdomain is $\Omega^c$;
- For modifications in $\Omega_1$, the interior subdomain is $\Omega_1$, and the exterior subdomain is $\Omega^c_1$ which contains $\Omega_2$ and $\Omega^c$;
- For modifications in $\Omega_2$, the interior subdomain is $\Omega_2$, and the exterior subdomain is $\Omega^c_2$ which contains $\Omega_1$ and $\Omega^c$.

For $\Omega$, $\Omega^c_1$, and $\Omega^c_2$, each one contains two subdomains. Here, it is important to effectively combine the results from smaller subdomains.
We construct each component of (2.11) by factorizing the related interior and exterior problems. The three cases in (2.11) share a similar relation, but the formulas become more sophisticated because now \( \Omega_1 \), \( \Omega_2 \), and \( \Omega^c \) have different shared boundaries. We define them as

\[
\Gamma_0 = \partial \Omega_1 \cap \partial \Omega_2, \quad \Gamma_1 = \partial \Omega_1 \cap \partial \Omega, \quad \Gamma_2 = \partial \Omega_2 \cap \partial \Omega.
\]

Similar to the derivation from (2.7) to (2.8), solution operators for \( \Omega \) can be obtained from merging \( \Omega_1 \) and \( \Omega_2 \). The same transmission condition (2.7) is imposed on \( \Gamma_0 \), and we get

\[
(2.12) \quad \begin{pmatrix}
T^{(\Omega_1)}_{0,0} & -I & T^{(\Omega_1)}_{0,1} & 0 & S^{(\Omega_1)}_{0,1} & 0 \\
-I & T^{(\Omega_2)}_{0,0} & 0 & T^{(\Omega_2)}_{0,1} & 0 & S^{(\Omega_2)}_{0,2} \\
T^{(\Omega_1)}_{1,0} & 0 & T^{(\Omega_1)}_{1,1} & 0 & S^{(\Omega_1)}_{1,1} & 0 \\
0 & T^{(\Omega_2)}_{2,0} & 0 & T^{(\Omega_2)}_{2,2} & 0 & S^{(\Omega_2)}_{2,2} \\
K^{(\Omega_1)}_{0,0} & 0 & K^{(\Omega_1)}_{1,1} & 0 & G^{(\Omega_1)}_{0,1} & 0 \\
0 & K^{(\Omega_2)}_{0,0} & 0 & K^{(\Omega_2)}_{1,2} & 0 & G^{(\Omega_2)}_{1,2}
\end{pmatrix}
\begin{pmatrix}
g^{(\Omega_1)}_0 \\
g^{(\Omega_2)}_0 \\
g^{(\Omega_1)}_1 \\
g^{(\Omega_2)}_1 \\
g^{(\Omega_1)}_2 \\
g^{(\Omega_2)}_2
\end{pmatrix} =
\begin{pmatrix}
0 \\
0 \\
0 \\
0 \\
f^{(\Omega_1)}_0 \\
f^{(\Omega_2)}_0
\end{pmatrix},
\]

where \( g^{(\Omega_0)}_k \) denotes the restriction of \( g^{(\Omega_0)} \) on \( \Gamma_k \), \( T^{(\Omega_0)}_{0,1} \) denotes the restriction of \( T^{(\Omega_0)} \) on \( \Gamma_0 \times \Gamma_1 \), the colon in the subscript means taking no restriction in the corresponding column or row set, and the other notation can be similarly understood. The first four block rows are rewritten from (2.4), and the transmission condition is substituted in the first two block rows. The last two block rows are from (2.3). The coupling between subdomains lies in the leading 2 \( \times \) 2 block

\[
(2.13) \quad M^{(\Gamma_0)} = \begin{pmatrix}
T^{(\Omega_1)}_{0,0} & -I \\
-I & T^{(\Omega_2)}_{0,0}
\end{pmatrix},
\]

This manuscript is for review purposes only.
The Schur complement of $M^{(\Gamma_0)}$ in (2.12) contains solution operators (2.5) for $\Omega$, where

\begin{align}
(2.14) \quad T^{(\Omega)} &= \begin{pmatrix} T^{(\Omega_1)}_{1,1} & T^{(\Omega_1)}_{1,2} \\ T^{(\Omega_2)}_{2,1} & T^{(\Omega_2)}_{2,2} \end{pmatrix}, \\
(2.15) \quad S^{(\Omega)} &= \begin{pmatrix} S^{(\Omega_1)}_{1,1} & S^{(\Omega_1)}_{1,2} \\ S^{(\Omega_2)}_{2,1} & S^{(\Omega_2)}_{2,2} \end{pmatrix}, \\
(2.16) \quad K^{(\Omega)} &= \begin{pmatrix} K^{(\Omega_1)}_{1,1} & K^{(\Omega_1)}_{1,2} \\ K^{(\Omega_2)}_{2,1} & K^{(\Omega_2)}_{2,2} \end{pmatrix}, \\
(2.17) \quad G^{(\Omega)} &= \begin{pmatrix} G^{(\Omega_1)}_{1,1} & G^{(\Omega_1)}_{1,2} \\ G^{(\Omega_2)}_{2,1} & G^{(\Omega_2)}_{2,2} \end{pmatrix},
\end{align}

Again, we do not form $S^{(\Omega)}$, $K^{(\Omega)}$, and $G^{(\Omega)}$ explicitly because they can be much larger than the boundary map $T^{(\Omega)}$. (2.15)–(2.17) can be used to compute fast matrix-vector products instead.

For the exterior subdomain $\Omega_1^c$, we merge $\Omega_2$ and $\Omega^c$ with similar procedures. Using the transmission condition (2.7) on $\Gamma_2$, we have

\begin{align}
(2.18) \quad T^{(\Omega_1^c)} &= \begin{pmatrix} T^{(\Omega_2)}_{1,0} & T^{(\Omega_2)}_{1,1} \\ T^{(\Omega_2)}_{2,0} & T^{(\Omega_2)}_{2,1} \end{pmatrix}, \\
(2.19) \quad K^{(\Omega_1^c)} &= \begin{pmatrix} K^{(\Omega_2)}_{1,0} & K^{(\Omega_2)}_{1,1} \\ K^{(\Omega_2)}_{2,0} & K^{(\Omega_2)}_{2,1} \end{pmatrix}, \\
\end{align}

where

\begin{equation}
M^{(\Gamma_2)} = \begin{pmatrix} T^{(\Omega_2)}_{1,1} & -I \\ -I & T^{(\Omega_2)}_{2,2} \end{pmatrix}.
\end{equation}

Clearly, we can also merge $\Omega_1$ and $\Omega^c$ by exchanging the role of $\Omega_1$ and $\Omega_2$ in (2.18)–(2.20).

Finally, for computing the solution, we develop tree-based algorithms built upon the leaf subdomains $\Omega_1$, $\Omega_2$, and $\Omega^c$ by substituting (2.13)–(2.20) into (2.11). For example, if the coefficient updates and the right-hand sides are supported in $\Omega_1$, based on the second case of (2.11) the solution process is as follows.

1. Factorize the updated operator $L$ in $\Omega_1$ for forming $T^{(\Omega_1)}$ and for computing matrix-vector products of $S^{(\Omega_1)}$, $K^{(\Omega_1)}$, and $G^{(\Omega_1)}$.
2. Solve the coupling system for $\partial \Omega_1$ using the second case of (2.11):
   \begin{equation}
   \begin{pmatrix} T^{(\Omega_1)} \\ -I \end{pmatrix} \begin{pmatrix} g^{(\Omega_1)} \\ f^{(\Omega_1)} \end{pmatrix} = \begin{pmatrix} -S^{(\Omega_1)} f^{(\Omega_1)} \\ 0 \end{pmatrix}.
   \end{equation}
3. Compute the solution in $\Omega_1$ using (2.3):
   \begin{equation}
   u^{(\Omega_1)} = G^{(\Omega_1)} f^{(\Omega_1)} + K^{(\Omega_1)} g^{(\Omega_1)}.
   \end{equation}
4. Solve the coupling system for $\Gamma_2$:
   \begin{equation}
   M^{(\Gamma_2)} \begin{pmatrix} g^{(\Omega_2)} \\ f^{(\Omega_2)} \end{pmatrix} = \begin{pmatrix} -T^{(\Omega_2)} g^{(\Omega_2)} \\ 0 \end{pmatrix}.
   \end{equation}
5. Compute the solution in $\Omega_2$ and $\Omega^c$:
\[
\begin{align*}
 u^{(\Omega_2)} &= K^{(\Omega_2)} g_0^{(\Omega_2)} + K^{(\Omega_2)} g_2^{(\Omega_2)}, \\
 u^{(\Omega^c)} &= K^{(\Omega^c)} g_1^{(\Omega^c)} + K^{(\Omega^c)} g_2^{(\Omega^c)}.
\end{align*}
\]

In steps 4 and 5, $K^{(\Omega_2)} g^{(\Omega_2)}$ is computed using (2.19). This two-level process illustrates the capability of dealing with coefficient updates of different volumes. The results of this section provide key components of the general hierarchical algorithms in Section 3.

3. General hierarchical algorithms. In this section, we write the complete hierarchical algorithms for solving coefficient update problems. In particular, we focus on generalizing the two-level method in Section 2.2 to a constructive multi-level method. The multi-level method involves the tree-based domain partitioning. Comparing with simpler alternatives in Section 2, the multi-level method is more flexible because it supports updates in any subdomain used in the domain partitioning, and is more efficient because the computational cost is minimized by isolating the smallest subdomains containing the coefficient updates. Besides a factorization update in subdomains, the major steps include: introduction of exterior subdomains in the domain partitioning, factorization of interior and exterior problems, and solution update with localized right-hand sides.

3.1. Transformation of binary domain partitioning. First, we describe the structures of the domain partitioning when exterior subdomains are introduced. The computational domain $D$ is partitioned hierarchically following a tree denoted by $T$. For notational simplicity, we restrict the discussion to binary trees. If $i$ is the parent node of $c_1$ and $c_2$ in the tree $T$, then the open subdomain $\Omega_i \subset D$ is partitioned into two open subdomains $\Omega_{c_1}$ and $\Omega_{c_2}$ such that

\[
\Omega_{c_1} \cap \Omega_{c_2} = \emptyset, \quad \overline{\Omega_i} = \overline{\Omega_{c_1}} \cup \overline{\Omega_{c_2}}.
\]

According to Figure 2.1, for the interior problems, each parent $i$ depends on the children $c_1$ and $c_2$; for the exterior domains, $\Omega_{c_1}$ can be partitioned into $\Omega_i$ and $\Omega_{c_2}$, and $\Omega_{c_2}$ can be partitioned into $\Omega_i$ and $\Omega_{c_1}$. The partitioning of exterior subdomains is well defined in the sense of (3.1) because of the following lemma.

**Lemma 3.1.** If $\Omega_i$, $\Omega_{c_1}$, and $\Omega_{c_2}$ are open subdomains of $D$ satisfying (3.1), then

\[
\Omega_i^c \cap \Omega_{c_2} = \emptyset, \quad \overline{\Omega_i} = \overline{\Omega_i^c} \cup \overline{\Omega_{c_2}},
\]

where $\Omega_j^c$ represents $D \setminus \overline{\Omega_j}$ for each $j \in \{i, c_1, c_2\}$.

**Proof.** $\overline{\Omega_i} \supset \overline{\Omega_{c_2}}$ from (3.1), so

\[
\Omega_i^c \cap \Omega_{c_2} = (D \setminus \overline{\Omega_i}) \cap \Omega_{c_2} \subset (D \setminus \overline{\Omega_{c_2}}) \cap \Omega_{c_2} = \emptyset.
\]

The open sets $\Omega_{c_1}$ and $\Omega_{c_2}$ have empty intersection, so

\[
\overline{\Omega_{c_1}} \cap \overline{\Omega_{c_2}} = \emptyset, \quad \overline{\Omega_{c_1}} \subset D \setminus \overline{\Omega_{c_2}} = \Omega_{c_1}.
\]

Because $\Omega_{c_1} \subset \Omega_{c_1}^c$ and $\Omega_{c_2} \subset \Omega_{c_2}^c$, $\Omega_{c_1}^c \subset \Omega_{c_1} \cup \Omega_{c_2}$ because

\[
\Omega_{c_1}^c = D \setminus \overline{\Omega_{c_1}} \subset D \setminus (\overline{\Omega_i} \setminus \overline{\Omega_{c_2}}) \subset (D \setminus \overline{\Omega_i}) \cup \overline{\Omega_{c_2}} = \Omega_i^c \cup \overline{\Omega_{c_2}}.
\]
Suppose the problem is modified in $\Omega_p$ for a level-$l$ node $p$. Write the path from the root $i_0$ to $p$ as $i_0 \to i_1 \to \cdots \to i_l = p$, so $\Omega_{i_0} \supset \Omega_{i_1} \supset \cdots \supset \Omega_{i_l} = \Omega_p$. Therefore, modifications in $\Omega_i$ not only lead to changes in the subtree generated by $p$, but also propagate along the path to the root. The goal here is to reorganize the domain partitioning such that $p$ is a child of the root, then changes in $\Omega_p$ do not propagate to multiple larger subdomains. Denote $i_k$’s sibling by $j_k$ for $1 \leq k \leq l$. See the left panel of Figure 3.1 for the illustration of $i_k, j_k$ in $T$. Denote $i_k$ the new node associated with the exterior subdomain

$$\Omega_{i_k} = \Omega_{i_k}^c, \quad 1 < k \leq l. \quad (3.3)$$

We construct the new binary domain partitioning step by step:

1. For the root node $i_0$, let $i_1, i_l$ be its children. From (3.3), one can check that $\Omega_{i_1} \cap \Omega_{i_l}^c = \emptyset, \quad \Omega_{i_0} = \Omega_{i_1} \cup \Omega_{i_l}^c$.

We preserve the partitioning in $\Omega_{i_1}$, and continue with the new node $i_1$.

2. For the node $i_k$ with $k \in \{l - 1, \ldots, 3\}$, let $j_k, i_{k-1}$ be $i_k$’s children. Since $i_{k-1}$ is the parent of $i_k, j_k$ in $T$, we have from (3.2)–(3.3) that $\Omega_{i_{k-1}} \cap \Omega_{j_k} = \emptyset, \quad \Omega_{i_k} = \Omega_{i_{k-1}}^c \cup \Omega_{j_k}$,

which means the partitioning from $i_k$ to $j_k, i_{k-1}$ is well defined. We preserve the partitioning in $\Omega_{j_k}$ and continue with the new node $i_{k-1}$.

3. For the node $i_2$, let $j_1, j_2$ be its children. From (3.2) and noticing that $\Omega_{j_1} = \Omega_{i_1}^c$, we have $\Omega_{j_1} \cap \Omega_{j_2} = \emptyset, \quad \Omega_{i_2} = \Omega_{j_1} \cup \Omega_{j_2}$.

The new binary tree is visualized in the right panel of Figure 3.1. The new tree can be constructed in $O(l)$ operations, because $l - 1$ nodes are removed and $l - 1$ nodes are introduced. From the construction process, we see that the new elements $\{i_k\}$ are not leaf nodes. That is to say, every exterior subdomain introduced here is a union of existing interior subdomains. The key results are summarized into the following theorem.

![Diagram](image-url)
Theorem 3.2. Given a binary tree \( T \), let \( \{\Omega_i : i \in T\} \) be a binary domain partitioning satisfying (3.1). For a non-root level-1 node \( p \in T \), there exists a well-defined binary domain partitioning such that

1. \( \Omega_p \) is a child subdomain of \( D \),
2. the elements of \( \{\Omega_i : i \text{ is an ancestor of } p \in T, 1 \leq \text{level}(i) < l\} \) are removed,
3. the elements of \( \{\Omega_i^c : i \text{ is an ancestor of } p \in T, 1 < \text{level}(i) \leq l\} \) are inserted,
4. every new element cannot be a leaf in the new binary partitioning.

The new domain partitioning is used to isolate the perturbations in \( \Omega_p \), because the level-one subdomains are precisely \( \Omega_p \) and \( \Omega_p^c \). Then, according to the solution operator (2.10), the interior problem in \( \Omega_p \) needs to be re-factorized, but the exterior problem in \( \Omega_p^c \) remains the same.

3.2. Hierarchical factorization and solution update. Inspired by the two-level example in Section 2.2, we describe the family of hierarchical algorithms needed for solving coefficient update problems, including the factorization and solution of interior and exterior problems. The major novelties are the hierarchical algorithms of exterior problems.

The factorization of interior problems follows a bottom-up (postordered) traversal of the tree \( T \). If the node \( i \) is a leaf, we factorize the discretized PDE (2.1) in \( \Omega_i \) to obtain the matrices defined in (2.3)–(2.4). If \( i \) has children, then the boundary map \( T(\Omega_i) \) can be constructed from those at its children using (2.14). The construction of interior boundary maps has been developed in [13, 21]. Since the process is the foundation of exterior problems and factorization update, we review this result in Algorithm 3.1, FACINT, using the notation in this paper.

The construction of exterior boundary maps follows a top-down (reverse postordered) traversal of \( T \). The major difference from computing interior boundary maps is that the data dependency is reversed. For the node \( i \) with children \( c_1, c_2 \), we have \( \Omega_{c_1}, \Omega_{c_2} \subset \Omega_i \) for the interior problems, but \( \Omega_{c_1}, \Omega_{c_2} \supset \Omega_i^c \) for the exterior ones. Based on (2.18), we construct \( T(\Omega_{c_1}) \) from \( T(\Omega_i), T(\Omega_{c_2}) \) and construct \( T(\Omega_i^c) \) from \( T(\Omega_i), T(\Omega_{c_1}), T(\Omega_{c_2}) \). This process is described in Algorithm 3.2, FACEXT.

For the coefficient update problem (1.3), recall that the coefficient update and the right-hand side are supported in the same subdomain \( \Omega_p \) for some \( p \in T \). According to the solution process at the end of Section 2.2, the major steps include: re-factorization in \( \Omega_p \), computing boundary sources on the boundary \( \partial \Omega_p \), and extracting the solution inside and outside \( \Omega_p \). This is Algorithm 3.4, SOLINT–SOLEXT.

In SOLINT, the modified operator \( L \) in \( \Omega_p \) is factorized and the solution in \( \Omega_p \) is computed via (2.3). It is essentially a local version of the solution algorithm presented in [21]. The matrix-vector products governed by (2.15)–(2.17) are carefully combined based on the superposition principle. Inside \( \Omega_p \), each subdomain is visited twice by a postordered and a reverse postordered traversal.

SOLEXT extends the solution to the exterior subdomain \( \Omega_p^c \) by solving a boundary value problem using \( K^{(\Omega_p^c)}g^{(\Omega_p^c)} \). It has a top-down traversal of the new domain partitioning inside \( \Omega_p^c \) defined in Theorem 3.2. For the matrix-vector product of \( K^{(\Omega_p^c)} \), (2.19) replaces (2.16) if there are exterior subdomains involved. At each step, we get the solution of a subdomain along the path from \( p \) to the root of \( T \), and the cost increases for high-level problems. The algorithm can be terminated in the middle once the desired part of the solution is computed.

In general, it does not need to know which subdomain is going to be changed.
in FACEXT, and its output can handle coefficient updates in any subdomain of the
domain partitioning. If we have additional information about \( p \), the cost and storage
can be further reduced by only calculating the exterior factors related to \( p \). As can
be seen in Theorem 3.2 and SOLEXT, the related nodes correspond to the ancestors
of \( p \). Table 3.1 lists the roles and properties of the routines.

**Table 3.1**

Major properties of the hierarchical factorization and solution algorithms.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type of traversal</th>
<th>Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>FACINT</td>
<td>Factorize interior problems</td>
<td>Postorder</td>
<td>(2.14)</td>
</tr>
<tr>
<td>FACEXT</td>
<td>Factorize exterior problems</td>
<td>Reverse postorder</td>
<td>(2.18)</td>
</tr>
<tr>
<td>SOLINT</td>
<td>Solve interior problems</td>
<td>Postorder, reverse postorder</td>
<td>(2.15)–(2.17)</td>
</tr>
<tr>
<td>SOLEXT</td>
<td>Solve exterior problems</td>
<td>Reverse postorder of new tree</td>
<td>(2.19)</td>
</tr>
</tbody>
</table>

**Algorithm 3.1** Factorization of interior problems (review of the result in [21])

1: procedure FACINT(\( \mathcal{T}, L \))
2: for each \( i \in \mathcal{T} \) following the postordered traversal do
3:   if \( i \) is a leaf then
4:     \((c_1, c_2) \leftarrow \text{'i's children}
5:     \Gamma_0 \leftarrow \partial \Omega_{c_1} \cap \partial \Omega_{c_2}, \Gamma_1 \leftarrow \partial \Omega_{c_1} \cap \partial \Omega_i, \Gamma_2 \leftarrow \partial \Omega_{c_2} \cap \partial \Omega_i
6:     \text{Factorize } M(\Gamma_0) = \begin{pmatrix} T_{0,0}^{(\Omega_{c_1})} & -I \\ -I & T_{0,0}^{(\Omega_{c_2})} \end{pmatrix} \text{ where } T_{j,k}^{(\Omega_m)} := T^{(\Omega_m)}|_{\Gamma_j \times \Gamma_k}
7:     \text{Based on (2.14), compute } T^{(\Omega_i)} \text{ via}
8:     \begin{pmatrix} T_{1,0}^{(\Omega_{c_1})} \\ T_{2,0}^{(\Omega_{c_2})} \end{pmatrix} - \begin{pmatrix} T_{1,0}^{(\Omega_{c_1})} \\ T_{2,0}^{(\Omega_{c_2})} \end{pmatrix} (M(\Gamma_0))^{-1} \begin{pmatrix} T_{0,1}^{(\Omega_{c_1})} \\ T_{0,2}^{(\Omega_{c_2})} \end{pmatrix}
9:   end if
10:  end for
11:  return \( T^{(\Omega_i)} \), factors of \( M^{(\Omega_i)} \), and for leaf nodes \( i \), \( S^{(\Omega_i)}, K^{(\Omega_i)}, G^{(\Omega_i)} \)
12: end procedure

In summary, we suggest the following calling sequence for solving coefficient up-
date problems:

1. SOLINT(\( \mathcal{T}, i_0, L, f, \ldots \)) for factorizing \( L \) and solving \( Lu = f \), where \( i_0 \) is the
   root of \( \mathcal{T} \);
2. FACEXT(\( \mathcal{T}, \ldots \)) for factorizing exterior problems;
3. SOLINT(\( \mathcal{T}, p, L - \hat{L}, (L - \hat{L})u, \ldots \)) for the solution update \( \hat{u} - u \) in \( \Omega_p \) and the
   exterior boundary source \( g^{(\Omega_p)} \);
4. SOLEXT(\( \mathcal{T}, p, g^{(\Omega_p)}, \ldots \)) for the solution update \( \hat{u} - u \) in \( \Omega_p^c \).

Note that the solution steps (1, 3, and 4) can be trivially extended for solving mul-
tiple right-hand sides. Before giving the complexity estimates in Section 4, there are
several qualitative arguments about the cost effectiveness of this family of algorithms.
The factorization of exterior problems does not increase the order of factorization.
Algorithm 3.2 Factorization of exterior problems

1: procedure FACEXT($T, T^{(\ast)}$)
2:     for each $i \in T$ following a reverse postordered traversal do
3:         if $i$ is not a leaf then
4:             $(c_1, c_2) \leftarrow i$’s children
5:             $\Gamma_0 \leftarrow \partial \Omega_{c_1} \cap \partial \Omega_{c_2}$, $\Gamma_1 \leftarrow \partial \Omega_{c_1} \cap \partial \Omega_i$, $\Gamma_2 \leftarrow \partial \Omega_{c_2} \cap \partial \Omega_i$
6:             Factorize $M^{(\Gamma_j)} = \begin{pmatrix} T_{j,0}^{(\Omega_{c_1})} & -I \\ -I & T_{j,2}^{(\Omega_{c_2})} \end{pmatrix}$, $j \in \{1, 2\}$
7:         Based on (2.18), compute $T^{(\Omega_{c_i})}$ via
8:             $\begin{pmatrix} T_{1,0}^{(\Omega_{c_1})} & T_{1,1}^{(\Omega_{c_2})} \\ T_{1,2}^{(\Omega_{c_2})} & T_{1,1}^{(\Omega_{c_1})} \end{pmatrix}^{-1} \begin{pmatrix} T_{2,0}^{(\Omega_{c_2})} \\ T_{2,1}^{(\Omega_{c_2})} \end{pmatrix}$
9:         Compute $T^{(\Omega_{c_i})}$ via
10:             $\begin{pmatrix} T_{1,0}^{(\Omega_{c_1})} & T_{1,1}^{(\Omega_{c_2})} \\ T_{1,2}^{(\Omega_{c_2})} & T_{1,1}^{(\Omega_{c_1})} \end{pmatrix}^{-1} \begin{pmatrix} T_{1,0}^{(\Omega_{c_1})} \\ T_{1,1}^{(\Omega_{c_1})} \end{pmatrix}$
11:     end if
12: end for
13: return $T^{(\ast)}$ and factors of $M^{(\ast)}$
14: end procedure

The complexity, because the cost depends on the sizes of boundaries $\{\partial \Omega_i\}$ in the same way as existing factorization of interior problems. The cost of the re-factorization step is low because it only depends on the local problem size in $\Omega_p$. The cost of solution is low if terminated early because Algorithm 3.4 visits smaller subdomains first.

4. Algorithmic complexity. In this section, we estimate the complexity of the algorithms presented in Section 3. The major components of our method includes:

- a precomputation step that constructs interior and exterior boundary maps of the reference problem,
- a factorization update step that modifies the factors of an interior problem, and
- a solution update step to get the final solution.

The complexity of the solution algorithms relies on the quality of the domain partitioning. For an $n \times n$ discretized linear system from a $d$-dimensional elliptic problem ($d = 2$ or 3). The following assumption is used to obtain an optimal complexity.

Assumption 4.1. Let $T$ be a complete binary tree containing $l$ levels. Each level-$k$ subdomain of the domain partitioning $\{\Omega_i : i \in T\}$ contains $O(n_k)$ interior unknowns and $O(m_k)$ boundary unknowns, where

$$n_k = 2^{-k}n, \quad m_k = n_k^{(d-1)/d}.$$ 

Furthermore, let $n_1 = O(1)$. Here, the constants in the big O notation are assumed to be uniformly bounded.

Remark 4.1. The condition on $n_k$ and $m_k$ requires that the domain partitioning is balanced. The fractional power in $m_k$ comes from the dimension reduction from a $d$-dimensional domain to a $(d-1)$-dimensional boundary.
Algorithm 3.3 Matrix-vector multiplications of $S^{(Ω)}$ and $K^{(Ω)}$ in (2.5)

1: procedure SMVINT($T, f, S^{(Ω)}, T^{(Ω)}, M^{(Ω)})$ ▷ Compute $\tilde{g}^{(Ω)} = S^{(Ω)} f^{(Ω)}$
2: for each $i ∈ T$ following the postordered traversal do
3:   if $i$ is a leaf then
4:       $\hat{g}^{(Ω_i)} ← S^{(Ω_i)} f^{(Ω_i)}$
5:   else
6:       $(c_1, c_2) ← i$’s children
7:       $\Gamma_0 ← \partial Ω_{c_1} \cap \partial Ω_{c_2}, \Gamma_1 ← \partial Ω_{c_1} \cap \partial Ω_i, \Gamma_2 ← \partial Ω_{c_2} \cap \partial Ω_i$
8:       Based on (2.15), compute
9:       $\hat{g}^{(Ω_i)} ← \left(\begin{array}{c} g_1^{(Ω_{c_1})} \\ g_2^{(Ω_{c_2})} \end{array}\right) = \left(\begin{array}{c} T_{1,0}^{(Ω_{c_1})} \\ T_{2,0}^{(Ω_{c_2})} \end{array}\right) (M^{(Ω_0)})^{-1} \left(\begin{array}{c} g_0^{(Ω_{c_1})} \\ g_0^{(Ω_{c_2})} \end{array}\right)$
10: end if
11: end for
12: return $\hat{g}^{(Ω)}$
13: end procedure

1: procedure KMVINT($T, g, K^{(Ω)}, T^{(Ω)}, M^{(Ω)})$ ▷ Compute $K^{(Ω_i)} g^{(Ω_i)}$
2: for each $i ∈ T$ following a reverse postordered traversal do
3:   if $i$ is a leaf then
4:       $u_{Ω_i} ← K^{(Ω_i)} g^{(Ω_i)}$
5:   else
6:       $(c_1, c_2) ← i$’s children
7:       $\Gamma_0 ← \partial Ω_{c_1} \cap \partial Ω_{c_2}, \Gamma_1 ← \partial Ω_{c_1} \cap \partial Ω_i, \Gamma_2 ← \partial Ω_{c_2} \cap \partial Ω_i$
8:       Based on (2.16), compute
9:       $\left(\begin{array}{c} g_0^{(Ω_{c_1})} \\ g_0^{(Ω_{c_2})} \end{array}\right) ← (M^{(Ω_0)})^{-1} \left(\begin{array}{c} -T_{0,1}^{(Ω_{c_1})} g_1^{(Ω_{c_1})} \\ -T_{0,2}^{(Ω_{c_2})} g_2^{(Ω_{c_2})} \end{array}\right)$
10: end if
11: end for
12: return $u$
13: end procedure

If boundary maps are stored as dense matrices, then according to (2.14) and (2.18), the precomputation of interior and exterior boundary maps has dense factorizations and multiplications at every node. The complexity $C_{\text{pre}}$ and the storage $S_{\text{pre}}$ are respectively

\[
C_{\text{pre}} = \sum_{k=0}^{\infty} 2^k O\left(m_k^3\right) = \begin{cases} O(n^{3/2}) & \text{in } 2D, \\ O(n^2) & \text{in } 3D, \end{cases}
\]

\[
S_{\text{pre}} = \sum_{k=0}^{\infty} 2^k O\left(m_k^2\right) = \begin{cases} O(n \log n) & \text{in } 2D, \\ O(n^{4/3}) & \text{in } 3D. \end{cases}
\]

The results are in the same orders as those in the direct factorization of sparse matrices.
Algorithm 3.4 Solution update with modified coefficients in $\Omega_p$

1: procedure SOLINT($T, p, \bar{L}, f, T^{(\Omega_p^c)})$ $\triangleright$ Solution in $\Omega_p$
2: $\bar{T} \leftarrow \text{subtree}(p)$ $\triangleright$ Subtree of $T$ with root $p$
3: FACINT($\bar{T}, \bar{L}$) for $\bar{T}(\bar{\cdot}, \bar{S}(\bar{\cdot}), \bar{K}(\bar{\cdot}), \bar{G}(\bar{\cdot}), \bar{M}(\bar{\cdot}))$ in $\Omega_p$
4: $\bar{g}(\bar{\cdot}) \leftarrow \text{SMVINT}(\bar{T}, \bar{f}, \bar{\bar{S}}(\bar{\cdot}), \bar{\bar{T}}(\bar{\cdot}), \bar{\bar{M}}(\bar{\cdot}))$ $\triangleright \bar{S}(\Omega_p) f(\Omega_p)$ via Algorithm 3.3
5: Based on (2.10), solve
   \[
   \begin{pmatrix}
   \bar{T}(\Omega_p) - I \\
   -I \\
   \end{pmatrix}
   \begin{pmatrix}
   g^{(\Omega_p)} \\
   0 \\
   \end{pmatrix}
   = \begin{pmatrix}
   -\bar{g}^{(\Omega_p)} \\
   0 \\
   \end{pmatrix}
   \]
6: for each $i \in \bar{T}$ following a reverse postordered traversal do
   $\triangleright u(\Omega_p) = \bar{G}(\Omega_p) f(\Omega_p) + \bar{K}(\Omega_p) g^{(\Omega_p)}$
7:     if $i$ is a leaf then
8:        $u(\Omega_p)|_{\Omega_i} \leftarrow \bar{G}(\Omega_i) f(\Omega_i) + \bar{K}(\Omega_i) g^{(\Omega_i)}$
9:     else
10:        $(c_1, c_2) \leftarrow i$'s children
11:        $\Gamma_0 \leftarrow \partial \Omega_{c_1} \cap \partial \Omega_{c_2}$, $\Gamma_1 \leftarrow \partial \Omega_{c_1} \cap \partial \Omega_i$, $\Gamma_2 \leftarrow \partial \Omega_{c_2} \cap \partial \Omega_i$
12:        Based on (2.16)--(2.17), compute
13:        \[
        \begin{pmatrix}
        g_{(\Omega_{c_1})} \\
        g_{(\Omega_{c_2})} \\
        \end{pmatrix}
        \leftarrow -(\bar{M}(\Gamma_0))^{-1}
        \begin{pmatrix}
        (\Omega_{c_1}) \\
        (\Omega_{c_2}) \\
        \end{pmatrix}
        \begin{pmatrix}
        \bar{g}_{(\Omega_{c_1})} \\
        \bar{g}_{(\Omega_{c_2})} \\
        \end{pmatrix}
        \]
14:        $g_{(\Omega_{c_1})} \leftarrow g_{(\Omega_1)}$, $g_{(\Omega_{c_2})} \leftarrow g_{(\Omega_2)}$
15:    end if
16: end for
17: return $u(\Omega_p^c), g^{(\Omega_p^c)}$
end procedure

1: procedure SOLEXT($T, p, g^{(\Omega_p^c)}, K(\cdot), T(\cdot), M(\cdot))$ $\triangleright$ Solution in $\Omega_p^c$ via $K^{(\Omega_p^c)} g^{(\Omega_p^c)}$
2: $c_1 \leftarrow p$
3: while $c_1$ is not the root do
4:     $c_2 \leftarrow c_1$'s sibling, $i \leftarrow c_1$'s parent
5:     $\Gamma_0 \leftarrow \partial \Omega_{c_1} \cap \partial \Omega_{c_2}$, $\Gamma_1 \leftarrow \partial \Omega_{c_1} \cap \partial \Omega_i$, $\Gamma_2 \leftarrow \partial \Omega_{c_2} \cap \partial \Omega_i$
6:     Based on (2.19), compute
7:        \[
    \begin{pmatrix}
    g_{(\Omega_{c_2})} \\
    g_{(\Omega_{c_1})} \\
    \end{pmatrix}
    \leftarrow -(M(\Gamma_2))^{-1}
    \begin{pmatrix}
    (\Omega_{c_2}) \\
    (\Omega_{c_1}) \\
    \end{pmatrix}
    \begin{pmatrix}
    T_{2,0}(\Omega_{c_2}) \\
    T_{2,1}(\Omega_{c_1}) \\
    \end{pmatrix}
    \]
8:     $u(\Omega_p^c)|_{\Omega_{c_2}} = \text{KMOVINT}(\bar{T}, g^{(\Omega_{c_2})}, K(\cdot), T(\cdot), M(\cdot))$ $\triangleright K^{(\Omega_{c_2})} g^{(\Omega_{c_2})}$ via Algorithm 3.3
9: $c_1 \leftarrow i$
10: end while
11: return $u(\Omega_p^c)$
12: end procedure

This manuscript is for review purposes only.
with nested dissection reordering.

Consider modifying the problem in some level-\(l\) subdomain \(\Omega_p\) containing \(O(n_l)\) interior unknowns. The subtree corresponding to \(\Omega_p\) has \((1-l)\) levels. The complexity \(C_{\text{upd}}\) and storage \(S_{\text{upd}}\) of local factorization update are respectively

\[
C_{\text{upd}} = \sum_{k=0}^{l-1} 2^k O(m_{k+l}^3) = \begin{cases} O(n_l^{3/2}) & \text{in 2D,} \\ O(n_l^2) & \text{in 3D,} \end{cases}
\]

\[
S_{\text{upd}} = \sum_{k=0}^{l-1} 2^k O(m_{k+l}^2) = \begin{cases} O(n_l \log n_l) & \text{in 2D,} \\ O(n_l^{4/3}) & \text{in 3D.} \end{cases}
\]

Observe that \(C_{\text{upd}}\) and \(S_{\text{upd}}\) only depend on the number of interior unknowns in \(\Omega_p\).

In comparison, we consider the naive factorization update method which changes the factors following the original data dependencies in \(T\). In addition to the re-factorization in \(\Omega_p\) that has complexity \(C_{\text{upd}}\) in (4.2), the naive method has an additional step which updates every ancestor of \(p\). This additional step costs

\[
C_{\text{anc}} = \sum_{k=0}^{l-1} O(m_k^2) = \begin{cases} O(n^{3/2}) & \text{in 2D,} \\ O(n^2) & \text{in 3D,} \end{cases}
\]

\[
S_{\text{anc}} = \sum_{k=0}^{l-1} O(m_k^2) = \begin{cases} O(n) & \text{in 2D,} \\ O(n^{4/3}) & \text{in 3D.} \end{cases}
\]

This additional cost, on the contrary, is primarily determined by \(n\) because the ancestors of \(p\) have larger and larger matrix sizes. The proposed new method reduces the cost from \(C_{\text{anc}} + C_{\text{upd}}\) to \(C_{\text{upd}}\). If \(n_l \ll n\), then the new method avoided the dominant cost (4.3) that is comparable to the cost (4.1) for re-factorizing the entire problem.

The solution update in Algorithm 3.4 has the solution in \(\Omega_p\) and \(\Omega_p^c\), and the computational cost is proportional to the memory access. The solution complexity is \(S_{\text{upd}}\) in \(\Omega_p\), and is \(S_{\text{pre}}\) in \(\Omega_p^c\). If the exterior solution is terminated early, then the total cost can be as low as \(S_{\text{upd}}\).

As a summary, the following theorem describes the complexity of the proposed algorithms.

**Theorem 4.1.** Let the domain partitioning satisfy Assumption 4.1. The cost of precomputation in Algorithm 3.1 (FACINT) and Algorithm 3.2 (FACEXT) is governed by the matrix size via (4.1). The cost of factorization update is (4.2), which only depends on the size of the updated subdomain.

**5. Numerical tests.** In this section, we check how the cost of our direct method scales with respect to the size of the computational domain and the support of the coefficient update. The method is able to solve general elliptic problems with coefficient updates. A particular problem of interest is the variable-coefficient Helmholtz equation

\[-\Delta u(x) - k^2(x)u(x) = f(x),\]

where \(k(x)\) is the wavenumber that may be updated in various applications. The solution algorithms are implemented in MATLAB, and are run in serial on a Linux workstation with 3.5GHz CPU and 64GB RAM.

In two dimensional space, we discretize the Helmholtz equation by a continuous Galerkin method with fourth-order nodal bases. The performance of the direct
FAST FACTORIZATION UPDATE FOR ELLIPTIC EQUATIONS

method is mostly determined by the matrix size and sparsity pattern. The matrix size equals the number of nodals in the domain, and high-order schemes usually lead to more nonzeros. We update the wavenumber in a subdomain close to the center of the computational domain, and the magnitude of the update is as large as $1/2$ of the original wavenumber.

If we enlarge the computational domain and increase $n$ while fixing the size of the modified subdomain, the test results are listed in Table 5.1 and plotted in Figure 5.1(a). As estimated by (4.1), the factorizations of the interior problems (Algorithm 3.1) and the exterior problems (Algorithm 3.2) share the same order of complexity. Direct factorizations contribute to the major computational cost and storage of the method. Algorithm 3.4 (SOLINT) contains the re-factorization and solution in the modified subdomain, and the cost does not depend on the matrix $n$. Algorithm 3.4 (SOLEXT) is the solution in the exterior subdomain, and the cost depends approximately linearly on $n$. Such complexity is consistent with our estimate.

For the largest computational domain with $n$ fixed, we also vary the size $n_l$ of the modified subdomain. The results are listed in Table 5.2 and plotted in Figure 5.1(b). The cost of SOLINT is dominated by the direct factorization in the modified subdomain. The dependence on $n_l$ as illustrated in Figure 5.1(b) is a little better than the estimate in (4.2). The cost of SOLEXT does not increase because $n$ is fixed.

These test results demonstrate that the proposed algorithms are capable of solving the challenging cases where the coefficient updates have large magnitude and support. The algorithms can accommodate large amounts of modifications fairly easily. In addition, the solution update algorithms produce high accuracies as in stand-alone direct solvers and no approximation is made.

We would also like to mention that, the large magnitude and support of the updates make the modified problems no longer close to the reference problem. This situation is handled efficiently with our algorithms, but causes troubles to standard methods such as iterative solvers using the factorization of the reference problem as a preconditioner. To verify this, we reuse the factorization of the reference problem as a preconditioner to solve the four matrices considered in Table 5.2. This preconditioner quickly losses effectiveness when the modified subdomain increases its size. It takes 32, 180, 717, and 2585 preconditioned GMRES iterations respectively to reach the relative residual accuracy $10^{-5}$.
Table 5.1
Test for increasing matrix sizes \( n \) with a fixed modified subdomain size \( (n_1 = 160^2) \). The updated solution \( u \) is compared with a stand-alone direct solution \( v \), and the relative \( \ell_p \)-distance is \( \| u - v \|_p / \| v \|_p \).

(a) Problem setup

<table>
<thead>
<tr>
<th>#nodals</th>
<th>321(^2)</th>
<th>641(^2)</th>
<th>1281(^2)</th>
<th>2561(^2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matrix size</td>
<td>103,041</td>
<td>410,881</td>
<td>1,640,961</td>
<td>6,558,721</td>
</tr>
<tr>
<td>#non-zeros</td>
<td>2,437,184</td>
<td>9,748,736</td>
<td>38,994,944</td>
<td>155,979,776</td>
</tr>
</tbody>
</table>

(b) Factorization of interior problems

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>1.77s</td>
<td>7.70s</td>
<td>33.10s</td>
<td>156.30s</td>
</tr>
<tr>
<td>Flops</td>
<td>3.11E9</td>
<td>1.58E10</td>
<td>8.93E10</td>
<td>5.62E11</td>
</tr>
<tr>
<td>Factor storage</td>
<td>9.03E6</td>
<td>4.65E7</td>
<td>2.31E8</td>
<td>1.11E9</td>
</tr>
</tbody>
</table>

(c) Factorization of exterior problems

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>0.52s</td>
<td>3.75s</td>
<td>25.02s</td>
<td>170.29s</td>
</tr>
<tr>
<td>Flops</td>
<td>1.66E9</td>
<td>1.75E10</td>
<td>1.62E11</td>
<td>1.35E12</td>
</tr>
<tr>
<td>Factor storage</td>
<td>3.87E6</td>
<td>2.56E7</td>
<td>1.46E8</td>
<td>7.66E8</td>
</tr>
</tbody>
</table>

(d) Solution of the reference problem

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>0.08s</td>
<td>0.32s</td>
<td>1.39s</td>
<td>7.08s</td>
</tr>
<tr>
<td>Flops</td>
<td>2.52E7</td>
<td>1.11E8</td>
<td>4.83E8</td>
<td>2.10E9</td>
</tr>
</tbody>
</table>

(e) Solution update after modifying \( 160^2 \) nodals

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SOLINT time</td>
<td>0.46s</td>
<td>0.56s</td>
<td>0.58s</td>
<td>0.67s</td>
</tr>
<tr>
<td>SOLINT flops</td>
<td>7.90E8</td>
<td>1.19E9</td>
<td>1.19E9</td>
<td>1.19E9</td>
</tr>
<tr>
<td>SOLEXT time</td>
<td>0.03s</td>
<td>0.14s</td>
<td>0.63s</td>
<td>2.89s</td>
</tr>
<tr>
<td>SOLEXT flops</td>
<td>9.34E6</td>
<td>5.21E7</td>
<td>2.47E8</td>
<td>1.18E9</td>
</tr>
<tr>
<td>Relative ( \ell_2 )-distance</td>
<td>4.74E-16</td>
<td>5.95E-16</td>
<td>6.88E-16</td>
<td>6.75E-16</td>
</tr>
<tr>
<td>Relative ( \ell_\infty )-distance</td>
<td>1.20E-15</td>
<td>1.34E-15</td>
<td>9.89E-16</td>
<td>7.81E-16</td>
</tr>
</tbody>
</table>

Table 5.2
Test for a fixed matrix size \( (2561^2) \) and increasing modified subdomain sizes.

<table>
<thead>
<tr>
<th>Modified nodals</th>
<th>40(^2)</th>
<th>80(^2)</th>
<th>160(^2)</th>
<th>320(^2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOLINT time</td>
<td>0.12s</td>
<td>0.14s</td>
<td>0.47s</td>
<td>1.86s</td>
</tr>
<tr>
<td>SOLINT flops</td>
<td>4.73E7</td>
<td>2.16E8</td>
<td>9.94E8</td>
<td>4.85E9</td>
</tr>
<tr>
<td>SOLEXT time</td>
<td>2.93s</td>
<td>2.52s</td>
<td>2.50s</td>
<td>2.47s</td>
</tr>
<tr>
<td>SOLEXT flops</td>
<td>1.08E9</td>
<td>1.08E9</td>
<td>1.08E9</td>
<td>1.06E9</td>
</tr>
<tr>
<td>Relative ( \ell_2 )-distance</td>
<td>3.76E-16</td>
<td>5.06E-16</td>
<td>6.75E-16</td>
<td>8.02E-16</td>
</tr>
<tr>
<td>Relative ( \ell_\infty )-distance</td>
<td>7.31E-16</td>
<td>6.40E-16</td>
<td>7.81E-16</td>
<td>8.78E-16</td>
</tr>
</tbody>
</table>

6. Conclusions. We developed a new framework for updating the factorization of discretized elliptic operators. A major significance is the hierarchical construction of exterior boundary maps. For each modified operator, we only need to update the factorization for locations where the coefficients are updated, and the locations of coefficient update are allowed to change to different subdomains. Tree-based algorithms were given for solving the interior and exterior problems. The complexity estimates
and the scaling test based on the Helmholtz equation show that the cost of factorization update only depends on the size of the modified subdomain and that the solution update cost is much faster than the standard direct solution algorithms. The solution update algorithms produce high accuracies as in expensive stand-alone direct solvers. The method is suitable for solving the challenging cases where the updates have large magnitude and support.

Acknowledgement. We would like to thank Yuanzhe Xi and Christopher Wong for valuable discussions and comments.

REFERENCES


[22] P. G. Martinsson, A direct solver for variable coefficient elliptic PDEs discretized via a