Multiscale Model. Simul.
Vol. 9, No. 2, pp. 686–710

SWEEPING PRECONDITIONER FOR THE HELMHOLTZ EQUATION: MOVING PERFECTLY MATCHED LAYERS*

BJÖRN ENGQUIST† AND LEXING YING†

Abstract. This paper introduces a new sweeping preconditioner for the iterative solution of the variable coefficient Helmholtz equation in two and three dimensions. The algorithms follow the general structure of constructing an approximate LDL^t factorization by eliminating the unknowns layer by layer starting from an absorbing layer or boundary condition. The central idea of this paper is to approximate the Schur complement matrices of the factorization using moving perfectly matched layers (PMLs) introduced in the interior of the domain. Applying each Schur complement matrix is equivalent to solving a quasi-1D problem with a banded LU factorization in the 2D case and to solving a quasi-2D problem with a multifrontal method in the 3D case. The resulting preconditioner has linear application cost, and the preconditioned iterative solver converges in a number of iterations that is essentially independent of the number of unknowns or the frequency. Numerical results are presented in both two and three dimensions to demonstrate the efficiency of this new preconditioner.

Key words. Helmholtz equation, perfectly matched layers, high frequency waves, preconditioners, LDL^t factorization, Green's function, multifrontal methods, optimal ordering

AMS subject classifications. 65F08, 65N22, 65N80

DOI. 10.1137/100804644

1. Introduction.

This is the second of a series of papers on developing efficient preconditioners for the numerical solutions of the Helmholtz equation in two and three dimensions. Let the domain of interest be the unit box $D = (0,1)^d$ with $d = 2, 3$. The time-independent wave field $u(x)$ for $x \in D$ satisfies the following Helmholtz equation:

$$\Delta u(x) + \frac{\omega^2}{c^2(x)}\, u(x) = f(x),$$

where ω is the angular frequency, $c(x)$ is the velocity field, and $f(x)$ is the external force. Commonly used boundary conditions are the approximations of the Sommerfeld condition which guarantees that the wave field generated by $f(x)$ propagates out of the domain if $c(x)$ is constant outside a sufficiently large ball and other boundary conditions for part of the boundary can also be considered. By appropriately rescaling the system, it is convenient to assume that the mean of $c(x)$ is equal to 1. Then $\frac{\omega}{2\pi}$ is the (average) wave number of this problem and $\lambda = \frac{2\pi}{\omega}$ is the (typical) wavelength.

Equations of the Helmholtz type appear commonly in acoustics, elasticity, electromagnetics, geophysics, and quantum mechanics. Efficient and accurate numerical solution of the Helmholtz equation is a very important problem in current numerical mathematics. This is, however, a very difficult computational task due to two main reasons. First, in a typical setting, the Helmholtz equation is discretized with at least a constant number of points per wavelength. Therefore, the number of samples n in each dimension is proportional to ω, the total number of samples N is $n^d = O(\omega^d)$, and the approximating discrete

*Received by the editors August 6, 2010; accepted for publication (in revised form) April 7, 2011; published electronically June 28, 2011.

http://www.siam.org/journals/mms/9-2/80464.html

†Department of Mathematics and ICES, The University of Texas at Austin, Austin, TX 78712. The first author (engquist@ices.utexas.edu) is partially supported by the National Science Foundation. The second author (lexing@math.utexas.edu) is partially supported by the National Science Foundation and the Alfred P. Sloan Foundation.

system of the Helmholtz equation is an $O(\omega^d) \times O(\omega^d)$ linear system, which is extremely large in many practical high frequency simulations. Second, since the discrete system is highly indefinite and has a very oscillatory Green's function due to the wave nature of the Helmholtz equation, most direct and iterative solvers based on the multiscale paradigm are no longer efficient. For further remarks, see the discussion in [12].

1.1. Approach and contribution. In the previous paper [12], we introduced a sweeping preconditioner that constructs an approximate LDL^t factorization layer by layer starting from an absorbing layer. An important observation regarding the sweeping preconditioner is that the intermediate Schur complement matrices of the LDL^t factorization correspond to the restriction of the half-space Green's function of the Helmholtz equation to a single layer. In [12], we represented the intermediate Schur complement matrices of the factorization efficiently in the hierarchical matrix framework [16]. In 2D, the efficiency of this preconditioner is supported by analysis, has linear complexity, and results in very small number of iterations when combined with the GMRES solver. In 3D, however, the theoretical justification is lacking and constructing the preconditioner can be more costly.

In this paper, we propose a new sweeping preconditioner that works well in both two and three dimensions. The central idea of this new approach is to represent these Schur complement matrices in terms of *moving perfectly matched layers* (PMLs) introduced in the interior of the domain. Applying these Schur complement matrices then corresponds to inverting a discrete Helmholtz system with a moving PML. Since each moving PML is typically only a few (8 to 12) grids wide, fast direct algorithms can be leveraged for this task. In 2D, this discrete system with the moving PML layer is a quasi-1D problem and can be solved efficiently using a banded LU factorization in an appropriate ordering. The construction and application costs of the preconditioner are $O(n^2) = O(N)$ and $O(n^2) = O(N)$, respectively. In 3D, the discrete Helmholtz system with the moving PML is a quasi-2D problem and can be solved efficiently using the multifrontal methods. The construction and application costs of the preconditioner are $O(n^4) = O(N^{4/3})$ and $O(n^3 \log n) = O(N \log N)$, respectively. The construction complexity can be further improved to linear scaling by applying either the hierarchical matrix approximation or the moving PML approximation recursively to each discrete Helmholtz system of the moving PML. Numerical results show that in both 2D and 3D this new sweeping preconditioner gives rise to iteration numbers that are essentially independent of N when combined with the GMRES solver. As a result, we have a linear solution method for the discrete Helmholtz system.

1.2. Related work. There has been vast literature on developing efficient algorithms for the Helmholtz equation. A partial list of previous results includes [3], [4], [6], [8], [10], [11], [14], [19], [23], [24], [27]. We refer to the review article [13] and our previous paper [12] for detailed discussions. The brief discussion below is restricted to those that are closely related to the approach proposed in this paper.

The most efficient direct methods for solving the discrete Helmholtz systems are the multifrontal methods or their pivoted versions [9], [15], [22]. The multifrontal methods exploit the locality of the discrete operator and construct an LDL^t factorization based on a hierarchical partitioning of the domain. The cost of a multifrontal method depends strongly on the number of dimensions. For a 2D problem with $N = n^2$ unknowns, a multifrontal method takes $O(N^{3/2})$ flops and $O(N \log N)$ storage space. The prefactor is usually rather small, making the multifrontal methods effectively the default choice for most 2D Helmholtz problems. However, for a 3D problem with $N = n^3$ unknowns, a

multifrontal method requires $O(N^2)$ flops and $O(N^{4/3})$ storage space, which can be very costly for large scale 3D problems.

The approach proposed here essentially reduces the dimensions of the problem by working with n subproblems with one dimension lower. In the 3D case, for each subproblem, it leverages the effectiveness of the 2D multifrontal methods by solving a quasi-2D problem. The price of this reduction is that we end up with only an approximate inverse. However, this approximate inverse is reasonably accurate and works very well as a preconditioner when combined with standard iterative solvers in all our variable coefficient test cases.

1.3. Contents. The rest of this paper is organized as follows. Section 2 presents the new sweeping preconditioner in the 2D case, and section 3 reports the 2D numerical results. We extend this approach to the 3D case in section 4 and report the 3D numerical results in section 5. Finally, section 6 discusses some future directions of this work.

2. Preconditioner in 2D. We will first discuss the sweeping factorization in general and then introduce the moving PML.

2.1. Discretization and sweeping factorization. Recall that our computational domain in 2D is $D = (0,1)^2$. In order to simplify the discussion, we assume that the Dirichlet zero boundary condition is used on the side $x_2 = 1$ while an approximation to the Sommerfeld boundary condition is enforced on the other three sides. One standard way of incorporating the Sommerfeld boundary condition is to use the PML [5], [7], [17]. Introduce

$$\sigma_1(t) = \begin{cases} \frac{C}{\eta} \cdot \left(\frac{t-\eta}{\eta}\right)^2 & t \in [0,\eta], \\ 0 & t \in [\eta, 1-\eta], \\ \frac{C}{\eta} \cdot \left(\frac{t-1+\eta}{\eta}\right)^2 & t \in [1-\eta, 1], \end{cases} \qquad \sigma_2(t) = \begin{cases} \frac{C}{\eta} \cdot \left(\frac{t-\eta}{\eta}\right)^2 & t \in [0,\eta], \\ 0 & t \in [\eta, 1], \end{cases} \tag{2.1}$$

and

$$s_1(x_1) = \left(1 + i\frac{\sigma_1(x_1)}{\omega}\right)^{-1}, \qquad s_2(x_2) = \left(1 + i\frac{\sigma_2(x_2)}{\omega}\right)^{-1}.$$

Here η is typically around one wavelength, and C is an appropriate positive constant independent of ω. The PML method replaces ∂_1 with $s_1(x_1)\partial_1$ and ∂_2 with $s_2(x_2)\partial_2$, respectively. This effectively provides a damping layer of width η near the three sides with the Sommerfeld boundary condition. The resulting equation becomes

$$\left((s_1\partial_1)(s_1\partial_1) + (s_2\partial_2)(s_2\partial_2) + \frac{\omega^2}{c^2(x)}\right) u = fx \in D = (0,1)^2,$$
$$u = 0x \in \partial D.$$

We assume that $f(x)$ is supported inside $[\eta, 1-\eta] \times [\eta, 1]$ (away from the PML). Dividing the above equation by $s_1(x_1)s_2(x_2)$ brings the result

$$\left(\partial_1\left(\frac{s_1}{s_2}\partial_1\right) + \partial_2\left(\frac{s_2}{s_1}\partial_2\right) + \frac{\omega^2}{s_1 s_2 c^2(x)}\right) u = f.$$

The main advantage of this equation is its symmetry. We discretize the domain $[0,1]^2$ with a Cartesian grid with spacing $h = 1/(n+1)$. The number of points n in each dimension is

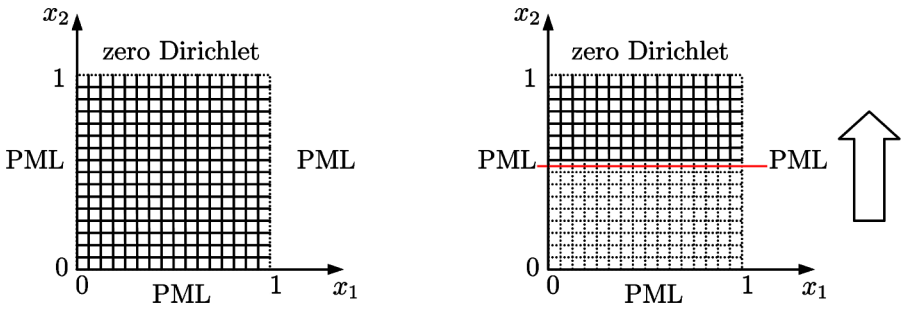


FIG. 2.1. *Left: Discretization grid in 2D. Right: Sweeping order in 2D. The dotted grid indicates the part that has already been eliminated.*

proportional to the wave number ω since a constant number of points is required for each wavelength. The set of all interior points of this grid is denoted by

$$\mathcal{P} = \{p_{i,j} = (ih, jh) \colon 1 \le i, j \le n\}$$

(see Figure 2.1(left)), and the total number of grid points is $N = n^2$.

We denote by $u_{i,j}$, $f_{i,j}$, and $c_{i,j}$ the values of $u(x)$, $f(x)$, and $c(x)$ at point $p_{i,j} = (ih, jh)$. The 5-point stencil finite difference method writes down the equation at points in $\mathcal{P}$ using central difference. The resulting equation at $x_{i,j} = (ih, jh)$ is

$$\begin{aligned} &\frac{1}{h^2}\left(\frac{s_1}{s_2}\right)_{i-\frac{1}{2},j} u_{i-1,j} + \frac{1}{h^2}\left(\frac{s_1}{s_2}\right)_{i+\frac{1}{2},j} u_{i+1,j} + \frac{1}{h^2}\left(\frac{s_2}{s_1}\right)_{i,j-\frac{1}{2}} u_{i,j-1} \\ &\qquad + \frac{1}{h^2}\left(\frac{s_2}{s_1}\right)_{i,j+\frac{1}{2}} u_{i,j+1} + \left(\frac{\omega^2}{(s_1 s_2)_{i,j} \cdot c_{i,j}^2} - (\cdots)\right) u_{i,j} = f_{i,j} \end{aligned} \tag{2.2}$$

with $u_{i',j'}$ equal to zero for (i', j') that violates $1 \le i', j' \le n$. Here $(\cdots)$ stands for the sum of the first four coefficients. We order both $u_{i,j}$ and $f_{i,j}$ row by row starting from the first row $j = 1$ and define the vectors

$$\begin{aligned} u &= (u_{1,1}, u_{2,1}, \ldots, u_{n,1}, \ldots, u_{1,n}, u_{2,n}, \ldots, u_{n,n})^t, \\ f &= (f_{1,1}, f_{2,1}, \ldots, f_{n,1}, \ldots, f_{1,n}, f_{2,n}, \ldots, f_{n,n})^t. \end{aligned}$$

Denote the discrete system of (2.2) by $Au = f$. We further introduce a block version of it by defining $\mathcal{P}_m$ to be the set of the indices in the mth row

$$\mathcal{P}_m = \{p_{1,m}, p_{2,m,} \ldots, p_{n,m}\}$$

and introducing

$$u_m = (u_{1,m}, u_{2,m}, \ldots, u_{n,m})^t \quad \text{and} \quad f_m = (f_{1,m}, f_{2,m}, \ldots, f_{n,m})^t.$$

Then

$$u = (u_1^t, u_2^t, \ldots, u_n^t)^t, \qquad f = (f_1^t, f_2^t, \ldots, f_n^t)^t.$$

Using these notations, the system $Au = f$ takes the following block tridiagonal form:

$$\begin{pmatrix} A_{1,1} & A_{1,2} & & \\ A_{2,1} & A_{2,2} & \ddots & \\ & \ddots & \ddots & A_{n-1,n} \\ & & A_{n,n-1} & A_{n,n} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{pmatrix}, \tag{2.3}$$

where $A_{m,m}$ are tridiagonal and $A_{m,m-1} = A_{m-1,m}^t$ are diagonal matrices.

The sweeping factorization of the matrix A is essentially a block LDL^t factorization that eliminates the unknowns layer by layer, starting from the absorbing layer near $x_2 = 0$ (see [12] for details). The result of this process is a factorization

$$A = L_1 \cdots L_{n-1} \begin{pmatrix} S_1 & & & \\ & S_2 & & \\ & & \ddots & \\ & & & S_n \end{pmatrix} L_{n-1}^t \cdots L_1^t, \tag{2.4}$$

where $S_1 = A_{1,1}$, $S_m = A_{m,m} - A_{m,m-1} S_{m-1}^{-1} A_{m-1,m}$ for $m = 2, \ldots, n$, and L_k is given by

$$L_k(\mathcal{P}_{k+1}, \mathcal{P}_k) = A_{k+1,k} S_k^{-1}, \qquad L_k(\mathcal{P}_i, \mathcal{P}_i) = I(1 \le i \le n), \quad \text{and zero otherwise.}$$

This process is illustrated graphically in Figure 2.1(right). Inverting this factorization for A gives the following formula for u:

$$u = (L_1^t)^{-1} \cdots (L_{n-1}^t)^{-1} \begin{pmatrix} S_1^{-1} & & & \\ & S_2^{-1} & & \\ & & \ddots & \\ & & & S_n^{-1} \end{pmatrix} L_{n-1}^{-1} \cdots L_1^{-1} f.$$

Algorithmically, the construction of the sweeping factorization of A can be summarized as follows by introducing $T_m = S_m^{-1}$.

Algorithm 2.1.
Construction of the sweeping factorization of A.

1: $S_1 = A_{1,1}$ and $T_1 = S_1^{-1}$
2: **for** $m = 2, \ldots, n$ **do**
3: $S_m = A_{m,m} - A_{m,m-1} T_{m-1} A_{m-1,m}$ and $T_m = S_m^{-1}$
4: **end for**

Since S_m and T_m are in general dense matrices of size $n \times n$, the cost of the construction algorithm is of order $O(n^4) = O(N^2)$. The computation of $u = A^{-1} f$ is carried out in the following algorithm once the factorization is ready.

Algorithm 2.2.
Computation of $u = A^{-1} f$ using the sweeping factorization of A.

1: **for** $m = 1, \ldots, n$ **do**
2: $u_m = f_m$

3: **end for**
4: **for** $m = 1, \ldots, n-1$ **do**
5: $u_{m+1} = u_{m+1} - A_{m+1,m}(T_m u_m)$
6: **end for**
7: **for** $m = 1, \ldots, n$ **do**
8: $u_m = T_m u_m$
9: **end for**
10: **for** $m = n-1, \ldots, 1$ **do**
11: $u_m = u_m - T_m(A_{m,m+1} u_{m+1})$
12: **end for**

Obviously the computations of $T_m u_m$ in the second and the third loops need only be carried out once for each m. We prefer to write the algorithm this way for simplicity of presentation. The cost of computing u with Algorithm 2.2 is of order $O(n^3) = O(N^{3/2})$, which is about $O(N^{1/2})$ times more expensive compared to the multifrontal method. Therefore, these two algorithms themselves are not very useful.

2.2. Moving PML. In Algorithms 2.1 and 2.2, the dominant cost is the construction and the application of the matrices T_m. Following [12], we consider the physical meaning of the Schur complement matrices T_m of the sweeping factorization. Let us restrict to the top-left $m \times m$ block of the above factorization.

$$\begin{pmatrix} A_{1,1} & A_{1,2} & & \\ A_{2,1} & A_{2,2} & \ddots & \\ & \ddots & \ddots & A_{m-1,m} \\ & & A_{m,m-1} & A_{m,m} \end{pmatrix} = L_1 \cdots L_{m-1} \begin{pmatrix} S_1 & & & \\ & S_2 & & \\ & & \ddots & \\ & & & S_m \end{pmatrix} L_{m-1}^t \cdots L_1^t, \tag{2.5}$$

where the L_k matrices are redefined to their restriction to the top-left $m \times m$ blocks. The matrix on the left is in fact the discrete Helmholtz equation restricted to the half-space below $x_2 = (m+1)h$ and with zero boundary condition on this line. Inverting the factorization (2.5) gives

$$\begin{pmatrix} A_{1,1} & A_{1,2} & & \\ A_{2,1} & A_{2,2} & \ddots & \\ & \ddots & \ddots & A_{m-1,m} \\ & & A_{m,m-1} & A_{m,m} \end{pmatrix}^{-1}$$
$$= (L_1^t)^{-1} \cdots (L_{m-1}^t)^{-1} \begin{pmatrix} S_1^{-1} & & & \\ & S_2^{-1} & & \\ & & \ddots & \\ & & & S_m^{-1} \end{pmatrix} L_{m-1}^{-1} \cdots L_1^{-1}.$$

The matrix on the left side is an approximation of the discrete half-space Green's function of the Helmholtz operator with zero boundary condition. On the right side, due to

the definition of the matrices $L_1, \ldots, L_{m-1}$, the (m,m)th block of the product is exactly equal to S_m^{-1}. Therefore, we reach the central observation: $T_m = S_m^{-1}$ *approximates the discrete half-space Green's function of the Helmholtz operator with zero boundary at* $x_2 = (m+1)h$, *restricted to the points on* $x_2 = mh$.

In the previous paper [12], T_m is approximated using the hierarchical matrix framework. Since the 3D Green's function propagates oscillations in all directions even when restricted to a plane, the numerical ranks of large off-diagonal blocks can be quite significant, and the theoretical justification of that method is lacking in 3D. Here, we try to approximate the matrix T_m in a different way.

As an operator, $T_m \colon \mathbb{C}^n \to \mathbb{C}^n$ maps an external force $g_m \in \mathbb{C}^n$ loaded only on the mth layer to the solution $v_m \in \mathbb{C}^n$ restricted to the same layer. Therefore, the domain of interest of T_m is a neighborhood of $x_2 = mh$. Let us recall that the main idea of the PML approximation is to approximate an unbounded domain problem with a bounded one that surrounds the domain of interest with PMLs. Therefore, it is natural to replace the half-space Helmholtz problem associated with T_m with an approximate local subproblem with a PML close to $x_2 = mh$. Therefore, *the central idea is to push the PML from* $x_2 = 0$ *to a location that is a few buffer layers away from* $x_2 = mh$ *when approximating* T_m. We call this approach the *moving PML* method, since these new PMLs do not exist in the original problem as they are only introduced in order to approximate T_m efficiently. The purpose of keeping a few extra buffer layers is that the resulting approximation is more accurate. On the other hand, these extra layers increase the size and solution cost of the local subproblem. However, since the goal is to construct a sufficiently accurate preconditioner, it is reasonable to even move the PML right next to $x_2 = mh$. As we will see in the numerical tests, the extra buffer layers provide little improvement on the approximation accuracy, and hence the moving PML is indeed placed right next to $x_2 = mh$ in all numerical examples.

To make this precise, let us assume that the width η of the PML is an integer multiple of h, and let $b = \eta/h$ be the number of grid points in the PML in the transversal direction. Define

$$s_2^m(x_2) = \left(1 + i\frac{\sigma_2(x_2 - (m-b)h)}{\omega}\right)^{-1},$$

and introduce an auxiliary problem on the domain $D_m = [0,1] \times [(m-b)h, (m+1)h]$:

$$\begin{aligned} \left((s_1\partial_1)(s_1\partial_1) + (s_2^m\partial_2)(s_2^m\partial_2) + \frac{\omega^2}{c^2(x)}\right)u &= fx \in D_m, \\ u &= 0x \in \partial D_m. \end{aligned} \tag{2.6}$$

This equation is discretized with the subgrid

$$\mathcal{G}_m = \{p_{i,j}, 1 \le i \le n, m-b+1 \le j \le m\}$$

of the original grid $\mathcal{P}$, and the resulting $bn \times bn$ discrete Helmholtz operator is denoted by H_m. Following the main idea mentioned above, the operator $\tilde{T}_m \colon g_m \to v_m$ defined by

$$\begin{pmatrix} * \\ \vdots \\ * \\ v_m \end{pmatrix} = H_m^{-1} \begin{pmatrix} 0 \\ \vdots \\ 0 \\ g_m \end{pmatrix} \tag{2.7}$$

is an approximation to the matrix T_m. Notice that applying $\tilde{T}_m$ to an arbitrary vector g_m involves solving a linear system of matrix H_m, which comes from the *local* 5-point stencil on the narrow grid $\mathcal{G}_m$ that contains only b layers. Let us introduce a new ordering for $\mathcal{G}_m$,

$$p_{1,m-b+1}, p_{1,m-b+2}, \ldots, p_{1,m} \ldots p_{n,m-b+1}, p_{n,m-b+2}, \ldots, p_{n,m},$$

that iterates through the x_2 direction first, and let us denote the permutation matrix induced from this new ordering by P_m. Now the matrix $P_m H_m P_m^t$ is a banded matrix with only $b-1$ lower diagonals and $b-1$ upper diagonals. Since the LU factorization $L_m U_m = P_m H_m P_m^t$ can be constructed efficiently, the application of $\tilde{T}_m$ can be done rapidly.

The application of a PML right next to the layer to be eliminated corresponds to a PML or absorbing boundary condition next to a Dirichlet boundary condition. This has been used as an asymptotic technique for high frequency scattering under the name of on-surface radiation boundary condition (OSRBC) [2], [18]. The OSRBC is an approximation that is more accurate than physical optics but, of course, not as accurate as a full boundary integral formulation.

2.3. Approximate inversion and preconditioner. Let us incorporate the moving PML technique into Algorithms 2.1 and 2.2. The computation at the first $(b+1)$ layers needs to be handled differently, since it does not make sense to introduce moving PMLs for these initial layers. Let us call the first b layers the *front* part and define

$$u_F = (u_1^t, \ldots, u_b^t)^t \quad \text{and} \quad f_F = (f_1^t, \ldots, f_b^t)^t.$$

Then we can rewrite $Au = f$ as

$$\begin{pmatrix} A_{F,F} & A_{F,b+1} & & \\ A_{b+1,F} & A_{b+1,b+1} & \ddots & \\ & \ddots & \ddots & A_{n-1,n} \\ & & A_{n,n-1} & A_{n,n} \end{pmatrix} \begin{pmatrix} u_F \\ u_{b+1} \\ \vdots \\ u_n \end{pmatrix} = \begin{pmatrix} f_F \\ f_{b+1} \\ \vdots \\ f_n \end{pmatrix}.$$

The construction of the approximate sweeping factorization of A takes the following steps. Notice that since T_m are approximated directly there is no need to compute S_m anymore.

Algorithm 2.3.
Construction of the approximate sweeping factorization of A with moving PML.

1: Let $\mathcal{G}_F$ be the subgrid of the first b layers, $H_F = A_{F,F}$, and let P_F be the permutation matrix induced by the new ordering (x_2 first) of $\mathcal{G}_F$. Construct the LU factorization $L_F U_F = P_F H_F P_F^t$. This factorization implicitly defines $\tilde{T}_F \colon \mathbb{C}^{bn} \to \mathbb{C}^{bn}$.

2: **for** $m = b+1, \ldots, n$ **do**
3: Let $\mathcal{G}_m = \{p_{i,j}, 1 \le i \le n, m-b+1 \le j \le m\}$, let H_m be the discrete system of (2.6) on $\mathcal{G}_m$, and let P_m be the permutation induced by the new ordering of $\mathcal{G}_m$. Construct the LU factorization $L_m U_m = P_m H_m P_m^t$. This factorization implicitly defines $\tilde{T}_m \colon \mathbb{C}^n \to \mathbb{C}^n$.
4: **end for**

The cost of Algorithm 2.3 is $O(b^3 n^2) = O(b^3 N)$. The computation of $u \approx A^{-1} f$ using the constructed sweeping factorization is summarized in the following algorithm.

Algorithm 2.4.
Computation of $u \approx A^{-1} f$ using the sweeping factorization of A with moving PML.
1: $u_F = f_F$ and $u_m = f_m$ for $m = b+1, \ldots, n$.
2: $u_{b+1} = u_{b+1} - A_{b+1,F}(\tilde{T}_F u_F)$. $\tilde{T}_F u_F$ is computed as $P_F^t U_F^{-1} L_F^{-1} P_F u_F$.
3: **for** $m = b+1, \ldots, n-1$ **do**
4: $u_{m+1} = u_{m+1} - A_{m+1,m}(\tilde{T}_m u_m)$. The application of $\tilde{T}_m u_m$ is done by forming the vector $(0, \ldots, 0, u_m^t)^t$, applying $P_m^t U_m^{-1} L_m^{-1} P_m$ to it, and extracting the value on the last layer.
5: **end for**
6: $u_F = \tilde{T}_F u_F$. See the previous steps for the application of $\tilde{T}_F$.
7: **for** $m = b+1, \ldots, n$ **do**
8: $u_m = \tilde{T}_m u_m$. See the previous steps for the application of $\tilde{T}_m$.
9: **end for**
10: **for** $m = n-1, \ldots, b+1$ **do**
11: $u_m = u_m - \tilde{T}_m(A_{m,m+1} u_{m+1})$. See the previous steps for the application of $\tilde{T}_m$.
12: **end for**
13: $u_F = u_F - \tilde{T}_F(A_{F,b+1} u_{b+1})$. See the previous steps for the application of $\tilde{T}_F$.

The cost of Algorithm 2.4 is $O(b^2 n^2) = O(b^2 N)$. Since b is a fixed constant, the cost is essentially linear. Algorithm 2.4 defines an operator

$$M \colon f = (f_F^t, f_{b+1}^t, \ldots, f_n^t)^t \to u = (u_F^t, u_{b+1}^t, \ldots, u_n^t)^t,$$

which is an approximate inverse of the discrete Helmholtz operator A. In practice, instead of generating the sweeping factorization of the original matrix A, it is more effective to generate the factorization for the matrix A_α associated with the modified Helmholtz equation

$$\Delta u(x) + \frac{(\omega + i\alpha)^2}{c^2(x)} u(x) = f(x), \tag{2.8}$$

where the damping parameter α is an $O(1)$ positive constant. We denote by $M_\alpha \colon f \to u$ the operator defined by Algorithm 2.4 with this modified equation. We would like to emphasize that (2.8) is very different from the equation used in the shifted Laplacian approach (for example, [14], [19]): in the shifted Laplacian formulation the damping parameter is $O(\omega)$ while here it is $O(1)$.

Since α is small, A_α and M_α are close to A and M, respectively. Therefore, we propose to solve the preconditioner system

$$M_\alpha A u = M_\alpha f$$

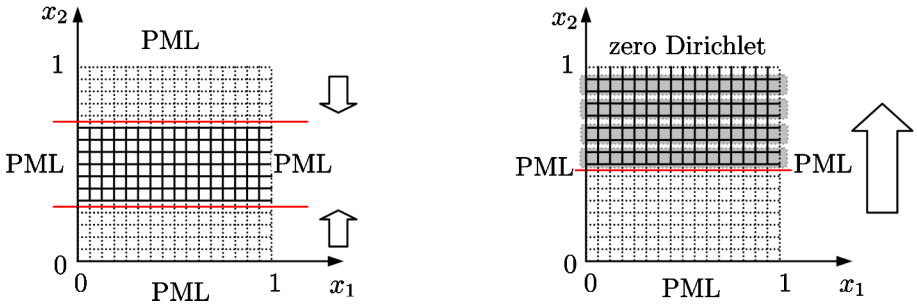


FIG. 2.2. *Different sweeping patterns. Left: For problems with PML at both $x_2 = 0$ and $x_2 = 1$, the algorithm sweeps from both ends towards the center. Right: Instead of one layer, multiple layers of unknowns can be eliminated within each iteration of the algorithm.*

using the GMRES solver [25], [26]. As the cost of applying M_α to any vector is $O(n^2) = O(N)$, the total cost of the iterative solver scales like $O(N_I N)$, where N_I is the number of iterations. As the numerical results in section 3 demonstrate, N_I depends at most logarithmically on N, thus resulting in a solver with almost linear complexity.

The problem considered so far has zero Dirichlet boundary condition on $x_2 = 1$. A common situation is to impose PML at all sides. In this case, the algorithms need a slight modification. Instead of sweeping upward from $x_2 = 0$, the algorithm sweeps with two fronts, one from $x_2 = 0$ upward and the other from $x_2 = 1$ downward (see Figure 2.2 (left)). Similar to u_F and f_F near $x_2 = 0$, we introduce near $x_2 = 1$,

$$u_E = (u^t_{n-b+1}, \ldots, u^t_n)^t \quad \text{and} \quad f_E = (f^t_{n-b+1}, \ldots, f^t_n)^t,$$

and we write $Au = f$ in the following block form:

$$\begin{pmatrix} A_{F,F} & A_{F,b+1} & & & & \\ A_{b+1,F} & A_{b+1,b+1} & \ddots & & & \\ & \ddots & \ddots & \ddots & & \\ & & \ddots & A_{n-b,n-b} & A_{n-b,E} \\ & & & A_{E,n-b} & A_{E,E} \end{pmatrix} \begin{pmatrix} u_F \\ u_{b+1} \\ \vdots \\ u_{n-b} \\ u_E \end{pmatrix} = \begin{pmatrix} f_F \\ f_{b+1} \\ \vdots \\ f_{n-b} \\ f_E \end{pmatrix}.$$

The upward sweep goes through $m = F, b+1, b+2, \ldots, (n-1)/2$, and the downward sweep visits $m = E, n-b, n-b-1, \ldots, (n+3)/2$. Finally, the algorithm visits the middle layer $m = (n+1)/2$ with moving PMLs on both sides.

Algorithm 2.3 eliminates one layer of unknowns in each iteration. We can also instead eliminate several layers of unknowns together (see Figure 2.2(right)). The resulting algorithm spends more computational time within each elimination step, since the discrete system H_m contains more layers in the x_2 dimension. On the other hand, the number of elimination steps goes down by a factor equal to the number of layers processed within each elimination step. In practice, the actual number d of layers processed within each step depends on the width of the moving PML and is chosen to minimize the overall computation time and storage.

3. Numerical results in 2D. In this section, we present several numerical results to illustrate the properties of the sweeping preconditioner described in section 2. The

algorithms are implemented sequentially in MATLAB and all tests are performed on a 16-core 2.6 GHz computer. We use GMRES as the iterative solver with relative residual equal to 10^{-3}.

3.1. PML. The examples in this section have the PML boundary condition specified at all sides. We consider three velocity fields in the domain $D = (0,1)^2$:

1. A smooth converging lens with a Gaussian profile at the center $(r_1, r_2) = (1/2, 1/2)$ of the domain (see Figure 3.1(a))

$$c(x_1, x_2) = \frac{4}{3}\left(1 - \frac{1}{2}\exp\left(-32((x_1 - r_1)^2 + (x_2 - r_2)^2)\right)\right).$$

2. A vertical waveguide with centralized Gaussian cross section (see Figure 3.1(b))

$$c(x_1, x_2) = \frac{4}{3}\left(1 - \frac{1}{2}\exp\left(-32\left(x_1 - \frac{1}{2}\right)^2\right)\right).$$

3. A randomly generated velocity field with values in (0.7,1.3) and correlation length equal to $1/16$ (see Figure 3.1(c)).

For each velocity field, we test with two external forces $f(x)$.

1. $f(x)$ is a narrow Gaussian point source located at $(r_1, r_2) = (1/2, 1/8)$:

$$f(x_1, x_2) = \exp\left(-\left(\frac{4\omega}{\pi}\right)^2((x_1 - r_1)^2 + (x_2 - r_2)^2)\right).$$

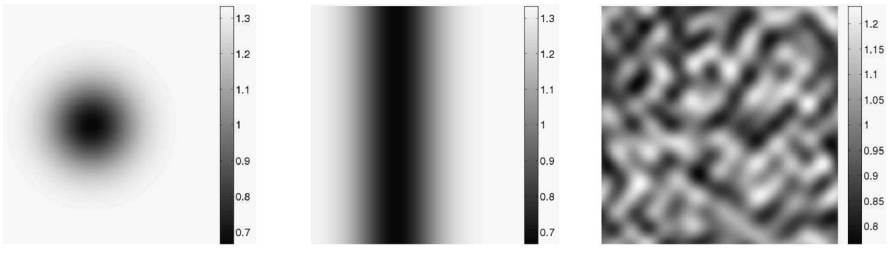
   The response of this forcing term generates circular waves propagating at all directions.

2. $f(x)$ is a Gaussian wave packet whose wavelength is comparable to the typical wavelength of the domain. This packet centers at $(r_1, r_2) = (1/8, 1/8)$ and points to direction $(d_1, d_2) = \frac{1}{\sqrt{2}}(1,1)$:

$$f(x_1, x_2) = \exp\left(-4\omega((x_1 - r_1)^2 + (x_2 - r_2)^2)\right) \cdot \exp\left(i\omega(x_1 d_1 + x_2 d_2)\right).$$

   The response of this forcing term generates a Gaussian beam initially pointing towards the (1,1) direction.

First, we study how the sweeping preconditioner behaves when ω varies. For each velocity field, we perform tests for $\frac{\omega}{2\pi} = 16, 32, \ldots, 256$. In these tests, we discretize each wavelength with $q = 8$ points and $n = 127, 255, \ldots, 2047$. The damping parameter α of the modified system (2.8) is set to be 2. The width of the moving PML is equal to $12h$



(a) (b) (c)

FIG. 3.1. *Test velocity fields.*

(i.e., $b = 12$) and the number d of layers processed within each iteration of Algorithms 2.3 and 2.4 is equal to 12. The sweeping pattern indicated in Figure 2.2(left) is used in these tests.

The results of the first velocity field are summarized in Table 3.1. T_{setup} denotes the time used to construct the preconditioner in seconds. For each external force, N_{iter} is the number of iterations of the preconditioned GMRES iteration, and T_{solve} is the overall solution time. When ω and n double, N increases by a factor of 4 and the setup cost in Table 3.1 increases roughly by a factor of 4 as well, which is consistent with the $O(N)$ complexity of Algorithm 2.3. At the same time, the number of iterations is essentially independent of n. As a result, the overall solution time increases by a factor of 4 or 5 when N quadruples, exhibiting the linear complexity of Algorithm 2.4.

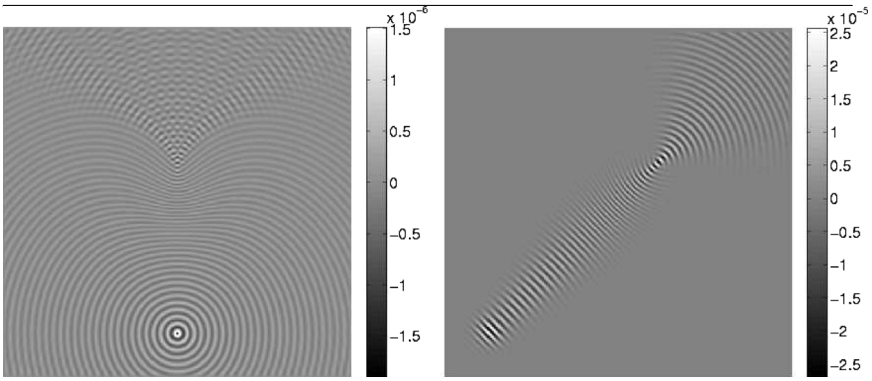
The results of the second and third velocity fields are summarized in Tables 3.2 and 3.3, respectively. The quantitative behavior of these tests is similar to the one of the first velocity field. In all cases, the GMRES iteration converges in about 20 iterations with the sweeping preconditioner.

Second, we study how the sweeping preconditioner behaves when q (the number of discretization points per wavelength) varies. We fix $\frac{\omega}{2\pi}$ to be 32 and let q be 8, 16, 32, 64. The test results for the three velocity fields are summarized in Tables 3.4, 3.5, and 3.6. These results show that the number of iterations remains to scale at most logarithmically, and the running time of the solution algorithm scales roughly linearly with respect to the number of unknowns.

Let us compare these numerical results with the ones from the previous paper [12]. The algorithms proposed in this paper are implemented in MATLAB, while the ones in [12] are implemented in C++ with compiler optimization. Hence, direct comparison of the running time is clearly in favor of the algorithms in the previous paper. We would

TABLE 3.1

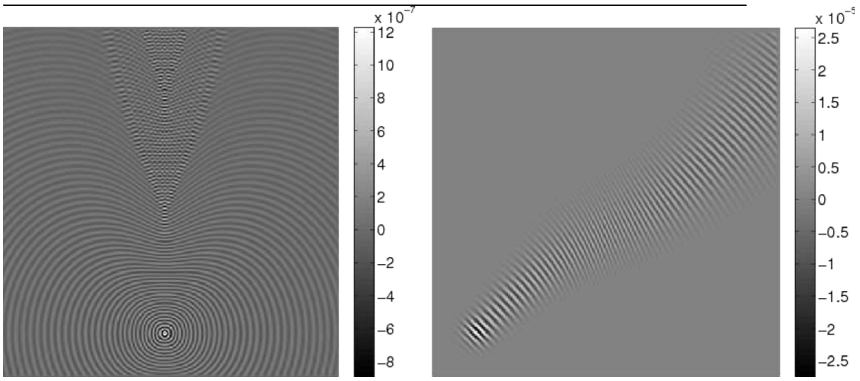
Results of velocity field 1 *with varying* ω. *Top: Solutions for two external forces with* $\omega/(2\pi) = 64$. *Bottom: Results for different* ω.



				Test 1		Test 2	
$\omega/(2\pi)$	q	$N = n^2$	T_{setup}	N_{iter}	T_{solve}	N_{iter}	T_{solve}
16	8	127^2	$2.86e-01$	14	$4.73e-01$	15	$3.81e-01$
32	8	255^2	$8.95e-01$	15	$1.59e+00$	15	$1.57e+00$
64	8	511^2	$3.78e+00$	15	$7.14e+00$	15	$7.12e+00$
128	8	1023^2	$1.61e+01$	15	$2.90e+01$	13	$2.54e+01$
256	8	2047^2	$6.85e+01$	16	$1.44e+02$	11	$9.42e+01$

TABLE 3.2

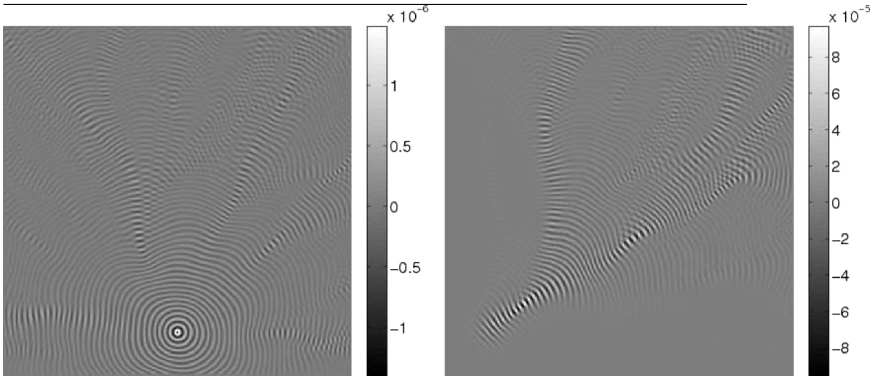
Results of velocity field 2 *with varying* ω. *Top: Solutions for two external forces with* $\omega/(2\pi) = 64$. *Bottom: Results for different* ω.



				Test 1		Test 2	
$\omega/(2\pi)$	q	$N = n^2$	T_{setup}	N_{iter}	T_{solve}	N_{iter}	T_{solve}
16	8	127^2	$3.03e-01$	18	$5.13e-01$	16	$3.94e-01$
32	8	255^2	$9.30e-01$	19	$2.58e+00$	16	$2.17e+00$
64	8	511^2	$3.76e+00$	19	$1.02e+01$	15	$7.61e+00$
128	8	1023^2	$1.61e+01$	19	$4.18e+01$	13	$2.75e+01$
256	8	2047^2	$6.78e+01$	19	$1.86e+02$	12	$1.10e+02$

TABLE 3.3

Results of velocity field 3 *with varying* ω. *Top: Solutions for two external forces with* $\omega/(2\pi) = 64$. *Bottom: Results for different* ω.



				Test 1		Test 2	
$\omega/(2\pi)$	q	$N = n^2$	T_{setup}	N_{iter}	T_{solve}	N_{iter}	T_{solve}
16	8	127^2	$3.56e-01$	18	$4.91e-01$	19	$5.03e-01$
32	8	255^2	$9.31e-01$	18	$2.42e+00$	19	$2.61e+00$
64	8	511^2	$3.76e+00$	17	$8.66e+00$	23	$1.24e+01$
128	8	1023^2	$1.60e+01$	19	$3.90e+01$	22	$4.80e+01$
256	8	2047^2	$6.82e+01$	17	$1.54e+02$	17	$1.48e+02$

TABLE 3.4
Results of velocity field 1 *with varying* q.

				Test 1		Test 2	
$\omega/(2\pi)$	q	$N = n^2$	T_{setup}	N_{iter}	T_{solve}	N_{iter}	T_{solve}
32	8	255^2	$9.19e-01$	15	$1.65e+00$	15	$1.61e+00$
32	16	511^2	$3.91e+00$	14	$6.94e+00$	15	$7.22e+00$
32	32	1023^2	$1.59e+01$	17	$8.87e+01$	17	$9.39e+01$
32	64	2047^2	$6.68e+01$	19	$3.74e+02$	20	$4.15e+02$

TABLE 3.5
Results of velocity field 2 *with varying* q.

				Test 1		Test 2	
$\omega/(2\pi)$	q	$N = n^2$	T_{setup}	N_{iter}	T_{solve}	N_{iter}	T_{solve}
32	8	255^2	$9.28e-01$	19	$2.14e+00$	16	$1.73e+00$
32	16	511^2	$3.69e+00$	17	$1.29e+01$	15	$1.13e+01$
32	32	1023^2	$1.58e+01$	24	$1.13e+02$	15	$7.16e+01$
32	64	2047^2	$6.63e+01$	26	$5.29e+02$	17	$3.47e+02$

TABLE 3.6
Results of velocity field 3 *with varying* q.

				Test 1		Test 2	
$\omega/(2\pi)$	q	$N = n^2$	T_{setup}	N_{iter}	T_{solve}	N_{iter}	T_{solve}
32	8	255^2	$1.00e+00$	16	$1.73e+00$	16	$1.81e+00$
32	16	511^2	$3.66e+00$	14	$1.34e+01$	18	$1.87e+01$
32	32	1023^2	$1.52e+01$	18	$8.16e+01$	19	$9.22e+01$
32	64	2047^2	$6.57e+01$	19	$3.99e+02$	21	$4.62e+02$

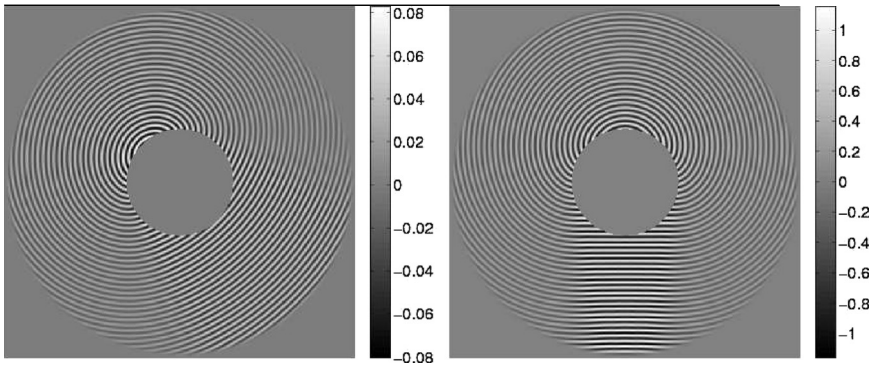
expect the running time of the algorithms in this paper to improve when implemented in optimized C++ code. Even with this disadvantage, the setup time T_{setup} of the current approach is about 20 times faster. This is mainly due to the fact that the implementation of the LU factorization is much more efficient compared to our implementation of the hierarchical matrix algebra in [12]. On the other hand, the number of iterations N_{iter} and solution time T_{solve} of the current algorithms are higher mainly due to the fact that the preconditioner is constructed based on the damped equation in (2.8).

3.2. Scattering problem. The sweeping preconditioner proposed in this paper can also be extended to scattering problems. Let us consider a simple case where the scatterer is a sound soft disk centered at the origin with radius r_0. In polar coordinates, the scattered field satisfies the following equations:

$$\frac{1}{r}(ru_r)_r + \frac{1}{r^2}u_{\theta\theta} + \frac{w^2}{c^2(r,\theta)}u = f,$$
$$u(r_0,\theta) = -u_{\text{inc}}(r_0,\theta),$$

TABLE 3.7

Results of the scattering problem. Top: Scattered fields for two incident waves with $\omega/(2\pi) = 64$. Bottom: Results for different ω.



				Incident field 1		Incident field 2	
$\omega/(2\pi)$	q	N	T_{setup}	N_{iter}	T_{solve}	N_{iter}	T_{solve}
16	8	45×403	$6.11e-01$	7	$3.61e-01$	7	$2.25e-01$
32	8	90×805	$2.61e+00$	7	$1.11e+00$	7	$1.11e+00$
64	8	180×1609	$1.17e+01$	7	$4.92e+00$	7	$4.90e+00$
128	8	359×3217	$4.95e+01$	7	$2.10e+01$	7	$2.06e+01$
256	8	717×6434	$1.99e+02$	7	$9.01e+01$	7	$9.00e+01$

where u_{inc} is the incident field and the Sommerfeld boundary condition is specified for r goes to infinity. One way to solve this scattering problem is to truncate the domain at $r = r_1$ for some $r_1 > r_0$ and apply the PML condition at $r = r_1$. We can then apply the sweeping preconditioner in the radial direction from $r = r_1$ to $r = r_0$. In the following example, $c(r, \theta) = 1$, $r_0 = 0.15$, and $r_1 = 0.5$. The polar grid is determined so that each wavelength is discretized with at least $q = 8$ points in the Cartesian (x_1, x_2) coordinates. For each fixed ω, two incident fields are used: one is the Green's function centered at $(-0.2, 0.2)$, and the other is the plane wave $\exp(-i\omega x_2)$ traveling towards the negative x_2 direction. We perform tests for $\frac{\omega}{2\pi} = 16, 32, 64, 128, 256$, and the numerical results are reported in Table 3.7.

4. Preconditioner in 3D. The presentation of the 3D preconditioner follows the layout of the 2D case.

4.1. Discretization and sweeping factorization. The computational domain is $D = (0, 1)^3$. Similar to the 2D case, assume that the Dirichlet boundary condition is used on the side $x_3 = 1$ and the PML boundary condition is enforced on other sides. Define

$$\sigma_1(t) = \sigma_2(t) = \begin{cases} \frac{C}{\eta} \cdot \left(\frac{t-\eta}{\eta}\right)^2 & t \in [0, \eta], \\ 0 & t \in [\eta, 1-\eta], \\ \frac{C}{\eta} \cdot \left(\frac{t-1+\eta}{\eta}\right)^2 & t \in [1-\eta, 1], \end{cases} \qquad \sigma_3(t) = \begin{cases} \frac{C}{\eta} \cdot \left(\frac{t-\eta}{\eta}\right)^2 & t \in [0, \eta], \\ 0 & t \in [\eta, 1], \end{cases}$$

and

$$s_1(x_1) = \left(1 + i\frac{\sigma(x_1)}{\omega}\right)^{-1}, \qquad s_2(x_2) = \left(1 + i\frac{\sigma(x_2)}{\omega}\right)^{-1}, \qquad s_3(x_3) = \left(1 + i\frac{\sigma(x_3)}{\omega}\right)^{-1}.$$

The PML approach replaces ∂_1, ∂_2, and ∂_2 with $s_1(x_1)\partial_1$, $s_2(x_2)\partial_2$, and $s_3(x_3)\partial_3$, respectively. This effectively provides a damping layer of width η near the sides with Sommerfeld condition. The resulting equation takes the form

$$\left((s_1\partial_1)(s_1\partial_1) + (s_2\partial_2)(s_2\partial_2) + (s_3\partial_3)(s_3\partial_3) + \frac{\omega^2}{c^2(x)}\right)u = fx \in (0,1)^3,$$
$$u = 0x \in \partial([0,1]^3).$$

We assume that $f(x)$ is supported inside $[\eta, 1-\eta] \times [\eta, 1-\eta] \times [\eta, 1]$ (away from the PML). Dividing the above equation by $s_1(x_1)s_2(x_2)s_3(x_3)$ brings the result

$$\left(\partial_1\left(\frac{s_1}{s_2 s_3}\partial_1\right) + \partial_2\left(\frac{s_2}{s_1 s_3}\partial_2\right) + \partial_3\left(\frac{s_3}{s_1 s_2}\partial_3\right) + \frac{\omega^2}{s_1 s_2 s_3 c^2(x)}\right)u = f.$$

The domain $[0,1]^3$ is discretized with a Cartesian grid with spacing $h = 1/(n+1)$, where the number of points n in each dimension is proportional to ω. The interior points of this grid are

$$\mathcal{P} = \{p_{i,j,k} = (ih, jh, kh) \colon 1 \le i, j, k \le n\}$$

(see Figure 4.1(left)), and the total number of grid points is $N = n^3$.

We denote by $u_{i,j,k}$, $f_{i,j,k}$, and $c_{i,j,k}$ the values of $u(x)$, $f(x)$, and $c(x)$ at point $x_{i,j,k} = (ih, jh, kh)$. The standard 7-point stencil finite difference method writes down the equation at points in $\mathcal{P}$ using central difference. The resulting equation at (ih, jh, kh) is

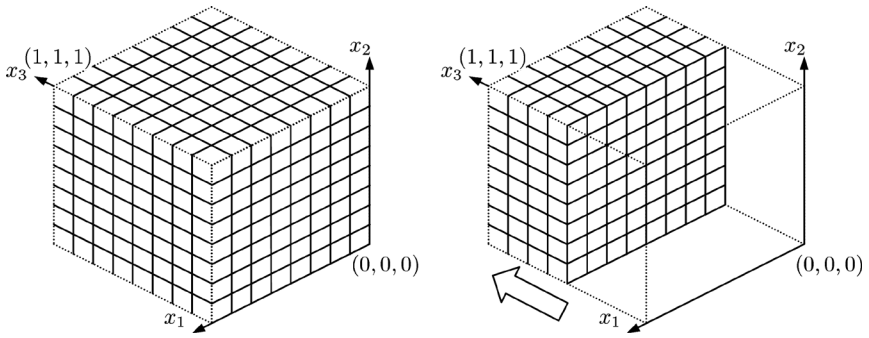


FIG. 4.1. *Left: Discretization grid in 3D. Right: Sweeping order in 3D, and the remaining grid shows the unknowns yet to be processed.*

$$\begin{aligned}
&\frac{1}{h^2}\left(\frac{s_1}{s_2 s_3}\right)_{i-\frac{1}{2},j,k} u_{i-1,j,k} + \frac{1}{h^2}\left(\frac{s_1}{s_2 s_3}\right)_{i+\frac{1}{2},j,k} u_{i+1,j,k} + \frac{1}{h^2}\left(\frac{s_2}{s_1 s_3}\right)_{i,j-\frac{1}{2},k} u_{i,j-1,k} \\
&+\frac{1}{h^2}\left(\frac{s_2}{s_1 s_3}\right)_{i,j+\frac{1}{2},k} u_{i,j+1,k} + \frac{1}{h^2}\left(\frac{s_3}{s_1 s_2}\right)_{i,j,k-\frac{1}{2}} u_{i,j,k-1} + \frac{1}{h^2}\left(\frac{s_3}{s_1 s_2}\right)_{i,j,k+\frac{1}{2}} u_{i,j,k+1} \\
&+\left(\frac{\omega^2}{(s_1 s_2 s_3)_{i,j,k}\cdot c^2_{i,j,k}} - (\cdots)\right) u_{i,j,k} = f_{i,j,k}
\end{aligned} \tag{4.1}$$

with $u_{i',j',k'}$ equal to zero for (i', j', k') that violates $1 \le i', j', k' \le n$. Here $(\cdots)$ stands for the sum of the six coefficients. We order $u_{i,j,k}$ and $f_{i,j,k}$ by going through the three dimensions in order and define the vectors

$$\begin{aligned}
u &= (u_{1,1,1}, u_{2,1,1}, \ldots, u_{n,1,1}, \ldots, u_{1,n,n}, u_{2,n,n}, \ldots, u_{n,n,n})^t, \\
f &= (f_{1,1,1}, f_{2,1,1}, \ldots, f_{n,1,1}, \ldots, f_{1,n,n}, f_{2,n,n}, \ldots, f_{n,n,n})^t.
\end{aligned}$$

The discrete system of (4.1) takes the form $Au = f$. Define $\mathcal{P}_m$ to be the indices in the mth row

$$\mathcal{P}_m = \{p_{1,1,m}, p_{2,1,m}, \ldots, p_{n,n,m}\},$$

and introduce

$$u_m = (u_{1,1,m}, u_{2,1,m}, \ldots, u_{n,n,m})^t \quad \text{and} \quad f_m = (f_{1,1,m}, f_{2,1,m}, \ldots, f_{n,n,m})^t.$$

Using these notations, we write

$$u = (u_1^t, u_2^t, \ldots, u_n^t)^t, \qquad f = (f_1^t, f_2^t, \ldots, f_n^t)^t,$$

and the system $Au = f$ takes the block tridiagonal form of (2.3), where each block is now of size $n^2 \times n^2$ and the off-diagonal blocks are diagonal matrices. The sweeping factorization takes the same form as the 2D one in (2.4). In order to design an efficient preconditioner, the main task is to construct an approximation to the Schur complement matrix $T_m\colon \mathbb{C}^{n^2} \to \mathbb{C}^{n^2}$, which maps an external force $g_m \in \mathbb{C}^{n^2}$ loaded only on the mth layer to the solution $v_m \in \mathbb{C}^{n^2}$ restricted to the same layer. Following the idea of pushing the PML near $x_3 = mh$, we define

$$s_3^m(x_3) = \left(1 + i\frac{\sigma_3(x_3 - (m-b)h)}{\omega}\right)^{-1},$$

and we introduce an auxiliary problem on the domain $D_m = [0,1] \times [0,1] \times [(m-b)h, (m+1)h]$:

$$\begin{aligned}
\left((s_1\partial_1)(s_1\partial_1) + (s_2\partial_2)(s_2\partial_2) + (s_3^m\partial_3)(s_3^m\partial_3) + \frac{\omega^2}{c^2(x)}\right) u &= fx \in D_m, \\
u &= 0x \in \partial D_m.
\end{aligned} \tag{4.2}$$

This equation is then discretized with the subgrid

$$\mathcal{G}_m = \{p_{i,j,k}, 1 \le i, j \le n, m - b + 1 \le k \le m\}$$

of the original grid $\mathcal{P}$. The resulting $bn^2 \times bn^2$ discrete Helmholtz operator is denoted by H_m. The operator $\tilde{T}_m \colon g_m \in \mathbb{C}^{n^2} \to v_m \in \mathbb{C}^{n^2}$ defined by (2.7) is an approximation of the Schur complement matrix T_m. Since H_m comes from the 7-point stencil with b layers, this can be viewed as a quasi-2D problem, which can be solved efficiently using a modified version of the multifrontal method [9], [15], [22].

The main idea of the multifrontal method is simple yet elegant. Take an $n \times n$ 2D grid as an example, and use M to denote the discrete operator resulting from a local stencil. The multifrontal method reorders the unknowns hierarchically in order to minimize the fill-ins of the LDL^t factorization of M. For the $n \times n$ Cartesian grid, one possible ordering is given in Figure 4.2 where the unknowns are clustered into groups and the groups are ordered hierarchically. The construction of the LDL^t factorization eliminates the unknowns group by group. The dominating cost of the algorithm is spent in inverting the unknowns of the last few groups, and the overall cost is $O(n^3)$, cubic in terms of the size of the last group. Moreover, the L matrix is never formed explicitly in the multifrontal method. Instead it is stored and applied as a sequence of (block) row operations for the sake of efficiency. Applying M^{-1} to an arbitrary vector using the result of the multifrontal algorithm takes $O(n^2 \log n)$ steps. In the current setting, we adopt the same hierarchical partitioning in the (x_1, x_2) plane, while keeping the unknowns with the same x_1 and x_2 but different x_3 indices in the same group. Since now the size of the last group is of order $O(bn)$, the construction phase of the multifrontal method takes $O(b^3n^3)$ steps and applying to an arbitrary vector takes $O(b^2n^2 \log n)$ steps.

4.2. Approximate inversion and preconditioner. Let us now combine the multifrontal method into Algorithms 2.1 and 2.2 to build the approximate inverse of H. Similar to the 2D case, we define

$$u_F = (u_1^t, \ldots, u_b^t)^t \qquad f_F = (f_1^t, \ldots, f_b^t)^t,$$

and we write

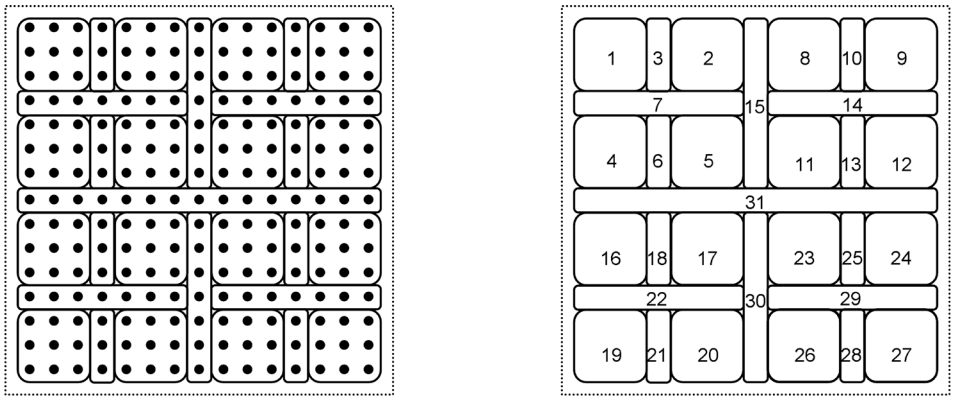


FIG. 4.2. *Multifrontal algorithm on a* 15×15 *2D Cartesian grid. Left: The unknowns are clustered into groups hierarchically to minimize the boundary between different groups. Right: Elimination order of different groups. The groups are eliminated in the increasing order of their indices.*

$$\begin{pmatrix} A_{F,F} & A_{F,b+1} & & \\ A_{b+1,F} & A_{b+1,b+1} & \ddots & \\ & \ddots & \ddots & A_{n-1,n} \\ & & A_{n,n-1} & A_{n,n} \end{pmatrix} \begin{pmatrix} u_F \\ u_{b+1} \\ \vdots \\ u_n \end{pmatrix} = \begin{pmatrix} f_F \\ f_{b+1} \\ \vdots \\ f_n \end{pmatrix}.$$

The goal of the construction of the approximate sweeping factorization of A is to compute the approximation $\tilde{T}_m$, and the algorithm consists of the following steps.

Algorithm 4.1.
Construction of the approximate sweeping factorization of A with moving PML.

1: Let $\mathcal{G}_F$ be the subgrid of the first b layers and let $H_F = A_{F,F}$. Construct the multifrontal factorization of H_F by partitioning $\mathcal{G}_F$ hierarchically in the (x_1, x_2) plane.
2: **for** $m = b+1, \ldots, n$ **do**
3: Let $\mathcal{G}_m = \{p_{i,j,k}, 1 \le i, j \le n, m-b+1 \le k \le m\}$ and let H_m be the system of (4.2) on $\mathcal{G}_m$. Construct the multifrontal factorization of H_m by partitioning $\mathcal{G}_m$ hierarchically in the (x_1, x_2) plane.
4: **end for**

The cost of Algorithm 4.1 is $O(b^3 n^4) = O(b^3 N^{4/3})$. Though the complexity is slightly higher than linear, it can be improved by building the inverses of H_m, applying the hierarchical matrix framework used in [12] or the moving PML idea once again to the solution of H_m. Either one of these two choices gives strictly linear complexity, and they are of significant theoretical interest. However, we observe that, for many practical problems that are not extremely large, the current version is at least equally competitive since the efficiency of the multifrontal implementation has been highly optimized due to its simple structure.

The computation of u from this sweeping factorization is summarized in the following algorithm.

Algorithm 4.2.
Computation of $u \approx A^{-1} f$ using the sweeping factorization of A with moving PML.

1: $u_F = f_F$ and $u_m = f_m$ for $m = b+1, \ldots, n$.
2: $u_{b+1} = u_{b+1} - A_{b+1,F}(\tilde{T}_F u_F)$. $\tilde{T}_F u_F$ is computed using the multifrontal factorization of H_F.
3: **for** $m = b+1, \ldots, n-1$ **do**
4: $u_{m+1} = u_{m+1} - A_{m+1,m}(\tilde{T}_m u_m)$. The application of $\tilde{T}_m u_m$ is done by forming the vector $(0, \ldots, 0, u_m^t)^t$, applying H_m^{-1} to it using the multifrontal factorization of H_m, and extracting the value on the last layer.
5: **end for**
6: $u_F = \tilde{T}_F u_F$. See the previous steps for the application of $\tilde{T}_F$.
7: **for** $m = b+1, \ldots, n$ **do**
8: $u_m = \tilde{T}_m u_m$. See the previous steps for the application of $\tilde{T}_m$.
9: **end for**
10: **for** $m = n-1, \ldots, b+1$ **do**
11: $u_m = u_m - \tilde{T}_m(A_{m,m+1} u_{m+1})$. See the previous steps for the application of $\tilde{T}_m$.
12: **end for**

13: $u_F = u_F - \tilde{T}_F(A_{F,b+1}u_{b+1})$. See the previous steps for the application of $\tilde{T}_F$.

The cost of Algorithm 4.2 is $O(b^2n^3 \log n) = O(b^2N \log N)$.

For the reason mentioned in section 2, we apply Algorithms 4.1 and 4.2 to the discrete operator A_α of the modified system

$$\Delta u(x) + \frac{(\omega + i\alpha)^2}{c^2(x)} u(x) = f(x), \tag{4.3}$$

where α is an $O(1)$ positive constant. We denote by $M_\alpha \colon f \to u$ the operator defined by Algorithm 2.4 for this modified equation. Since A_α is close to A when α is small, we propose to solve the preconditioner system

$$M_\alpha A u = M_\alpha f$$

using the GMRES solver [25], [26]. Because the cost of applying M_α to any vector is $O(N \log N)$, the total cost of the GMRES solver is $O(N_I N \log N)$, where N_I is the number of iterations required. As the numerical results in section 5 demonstrate, N_I is essentially independent of the number of unknowns N, thus resulting in an algorithm with almost linear complexity.

5. Numerical results in 3D. In this section, we present several numerical results to illustrate the algorithms described in section 4. We use GMRES as the iterative solver with relative residual equal to 10^{-3}.

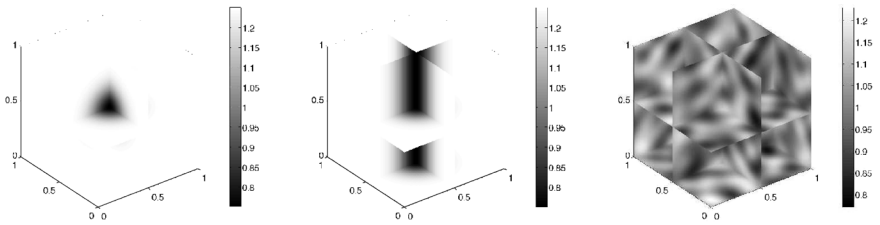
The examples in this section have the PML boundary condition specified at all sides. We consider three velocity fields in the domain $D = (0,1)^3$.

1. A converging lens with a Gaussian profile at the center of the domain (see Figure 5.1(a)).
2. A vertical waveguide with Gaussian cross section (see Figure 5.1(b)).
3. A random velocity field with values in (0.7,1.3) and correlation length equal to 1/8 (see Figure 5.1(c)).

For each velocity field, we test with two external forces $f(x)$.

1. $f(x)$ is a Gaussian point source located at $(r_1, r_2, r_3) = (1/2, 1/2, 1/4)$. The response of this forcing term generates circular waves propagating at all directions.
2. $f(x)$ is a Gaussian wave packet whose wavelength is comparable to the typical wavelength of the domain. This packet centers at $(r_1, r_2, r_3) = (1/2, 1/4, 1/4)$ and points to direction $(d_1, d_2, d_3) = \frac{1}{\sqrt{2}}(0, 1, 1)$.

First, we study how the sweeping preconditioner behaves when ω varies. For each velocity field, we perform tests for $\frac{\omega}{2\pi}$ equal to 5, 10, 20. In these tests, each wavelength is

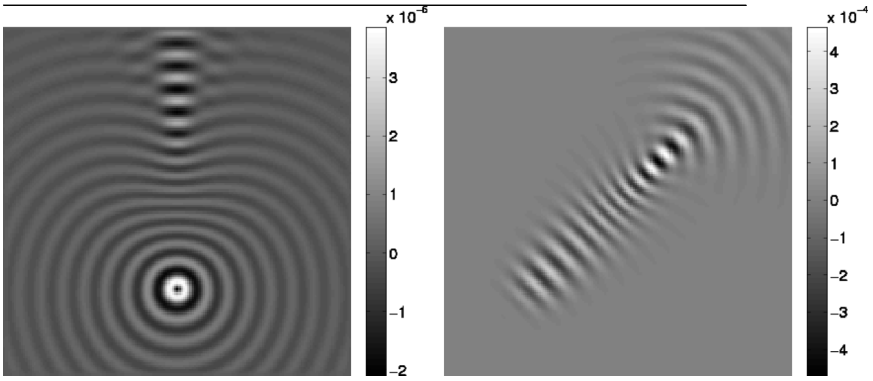


(a) (b) (c)

FIG. 5.1. *Test velocity fields.*

TABLE 5.1

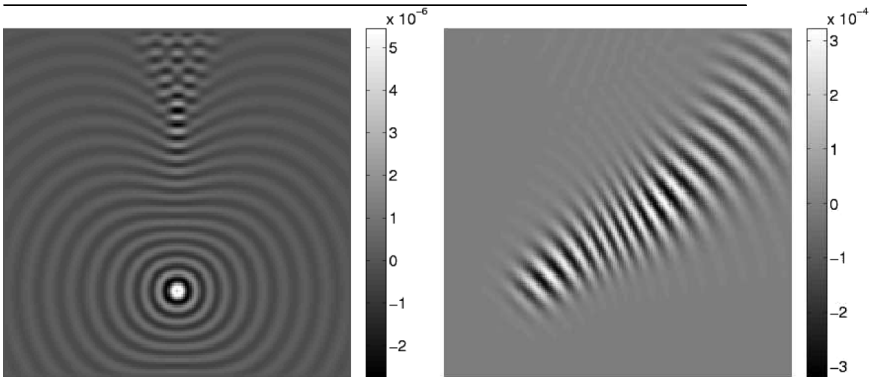
Results of velocity field 1 *with varying* ω. *Top: Solutions for two external forces with* $\omega/(2\pi) = 16$ *on a plane near* $x_1 = 0.5$. *Bottom: Results for different* ω.



				Test 1		Test 2	
$\omega/(2\pi)$	q	$N = n^3$	T_{setup}	N_{iter}	T_{solve}	N_{iter}	T_{solve}
5	8	39^3	$4.80e+00$	11	$4.53e+00$	11	$4.63e+00$
10	8	79^3	$6.37e+01$	11	$4.92e+01$	11	$4.93e+01$
20	8	159^3	$8.27e+02$	12	$5.53e+02$	12	$5.94e+02$

TABLE 5.2

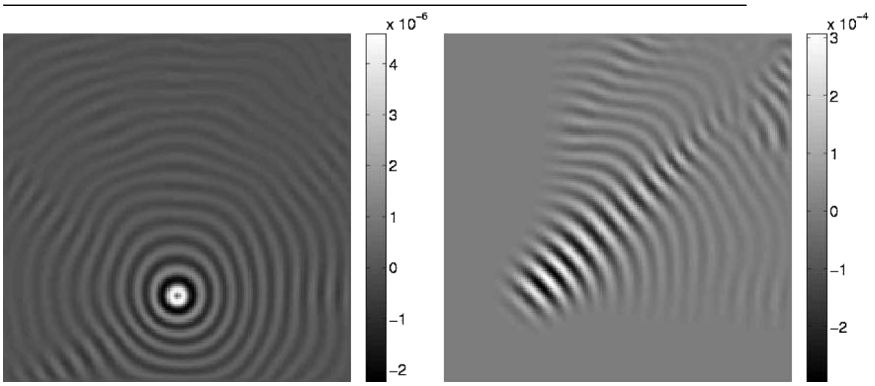
Results of velocity field 2 *with varying* ω. *Top: Solutions for two external forces with* $\omega/(2\pi) = 16$ *on a plane near* $x_1 = 0.5$. *Bottom: Results for different* ω.



				Test 1		Test 2	
$\omega/(2\pi)$	q	$N = n^3$	T_{setup}	N_{iter}	T_{solve}	N_{iter}	T_{solve}
5	8	39^3	$4.83e+00$	12	$5.14e+00$	12	$5.03e+00$
10	8	79^3	$6.76e+01$	13	$5.70e+01$	12	$5.64e+01$
20	8	159^3	$8.24e+02$	14	$6.32e+02$	11	$5.40e+02$

again discretized with $q = 8$ points, and the number of samples in each dimension is $n = 39$, 79, 159. The damping parameter α of (4.3) is set to 1. The width of the PML is equal to $6h$ (i.e., $b = 6$), and the number of layers processed within each iteration of Algorithms 4.1 and 4.2 is equal to 3 (i.e., $d = 3$). The preconditioner sweeps the domain with two fronts that start from $x_3 = 0$ and $x_3 = 1$.

TABLE 5.3
Results of velocity field 3 *with varying* ω. *Top: Solutions for two external forces with* $\omega/(2\pi) = 16$ *on a plane near* $x_1 = 0.5$. *Bottom: Results for different* ω.



				Test 1		Test 2	
$\omega/(2\pi)$	q	$N = n^3$	T_{setup}	N_{iter}	T_{solve}	N_{iter}	T_{solve}
5	8	39^3	$4.85e+00$	12	$5.26e+00$	12	$5.44e+00$
10	8	79^3	$6.69e+01$	11	$5.10e+01$	13	$5.99e+01$
20	8	159^3	$8.42e+02$	11	$5.58e+02$	13	$6.28e+02$

The results of the first velocity field are reported in Table 5.1. The two plots show the solutions of the two right sides on a plane near $x_1 = 0.5$. T_{setup} is the time used to construct the preconditioner in seconds. N_{iter} is the number of iterations of the preconditioned GMRES iteration, and T_{solve} is the solution time. The estimate in section 4 shows that the setup time scales like $O(N^{4/3})$. When ω doubles, N increases by a factor of 8, and T_{setup} should increase by a factor of 16. The numerical results show that the actual growth factor is even lower. A remarkable feature of the sweeping preconditioner is that in all cases the preconditioned GMRES solver converges in at most 12 iterations. Finally, we would like to point out that our algorithm is quite efficient: for the case with $\omega/(2\pi) = 20$ with more than four million unknowns, the solution time is less than 600 seconds. The results of the second and the third velocity fields are reported in Tables 5.2 and 5.3, respectively. In all tests, the GMRES iteration converges in at most 13 iterations when combined with the new sweeping preconditioner.

Second, we study how the sweeping preconditioner behaves when q (the number of discretization points per wavelength) varies. We fix $\frac{\omega}{2\pi}$ to be 5 and let q be 8, 16, 32. The test results for the three velocity fields are summarized in Tables 5.4, 5.5, and 5.6, respectively. These results show that the number of iterations remains roughly constant,

TABLE 5.4
Results of velocity field 1 *with varying* q.

				Test 1		Test 2	
$\omega/(2\pi)$	q	$N = n^3$	T_{setup}	N_{iter}	T_{solve}	N_{iter}	T_{solve}
5	8	39^3	$4.87e+00$	11	$4.91e+00$	11	$4.96e+00$
5	16	79^3	$6.59e+01$	11	$4.70e+01$	12	$5.55e+01$
5	32	159^3	$8.07e+02$	13	$5.91e+02$	13	$6.31e+02$

TABLE 5.5
Results of velocity field 2 *with varying* q.

				Test 1		Test 2	
$\omega/(2\pi)$	q	$N = n^3$	T_{setup}	N_{iter}	T_{solve}	N_{iter}	T_{solve}
5	8	39^3	$4.80e+00$	12	$5.36e+00$	12	$4.95e+00$
5	16	79^3	$6.74e+01$	13	$5.53e+01$	12	$5.51e+01$
5	32	159^3	$8.18e+02$	14	$6.48e+02$	14	$6.45e+02$

TABLE 5.6
Results of velocity field 3 *with varying* p.

				Test 1		Test 2	
$\omega/(2\pi)$	q	$N = n^3$	T_{setup}	N_{iter}	T_{solve}	N_{iter}	T_{solve}
5	8	39^3	$4.82e+00$	12	$4.92e+00$	12	$5.08e+00$
5	16	79^3	$6.77e+01$	12	$5.17e+01$	13	$6.04e+01$
5	32	159^3	$8.16e+02$	13	$6.26e+02$	15	$7.14e+02$

and the running time of the solution algorithm scales roughly linearly with respect to the number of unknowns.

Let us compare these numerical results with the ones from the 3D results from the previous paper [12]. The setup time T_{setup} of the current algorithms is much lower: for the problem of 20 wavelengths across, the current setup time is in the hundreds of seconds while the setup time in [12] is in the tens of thousands of seconds. This is mainly due to the fact that our implementation of the multifrontal algorithm in this paper is more efficient compared to our implementation of the 2D hierarchical matrix algebra in [12]. The number of iterations N_{iter} is about 5 times larger, again due to the introduction of the damping parameter α. Notice that the solution time T_{solve} is only about 3 times larger, and this is due to the efficiency of applying $\tilde{T}_m$ using the multifrontal factorization.

6. Conclusion and future work. In this paper, we proposed a new sweeping preconditioner for the Helmholtz equation in two and three dimensions. Similar to the previous paper [12], the preconditioner is based on an approximate block LDL^t factorization that eliminates the unknowns layer by layer starting from an absorbing layer or boundary condition. What is new is that the Schur complement matrices of the block LDL^t factorization are approximated by introducing moving PMLs in the interior of the domain. In the 2D case, applying these Schur complement matrices corresponds to solving quasi-1D problems by an LU factorization with optimal ordering. In the 3D case, applying these Schur complement matrices corresponds to solving quasi-2D problems with multifrontal methods. The resulting preconditioner has a linear application cost, and the number of iterations is essentially independent of the number of unknowns or the frequency when combined with the GMRES solver.

Some questions remain open. First, we tested the algorithms with the PML boundary condition as the numerical implementation of the Sommerfeld condition. Many other boundary conditions are available, and we believe that the current algorithms

should work for these boundary conditions. We presented the algorithms using the simplest central difference scheme (5-point stencil in 2D and 7-point stencil in 3D). The dispersion relationships of these schemes are rather poor approximations to the true one. One would like to investigate other more accurate stencils and other types of discretizations such as finite element, spectral element, and discontinuous Galerkin.

Parallel processing is necessary for large scale 3D problems. In Algorithms 2.3 and 4.1, the computations for different T_m are fully independent, and therefore, the setup stage is fully parallelizable. For the application stage (Algorithms 2.4 and 4.2), although the overall structure is sequential by itself, the calculation of the multifrontal method within each iteration can be well parallelized. Several efficient implementations are already available [1], [20] for this purpose. There is also an alternative to parallelize via a coarse scale domain decomposition and apply our technique within each subdomain.

The approach of the current paper is readily applicable to nonuniform and even adaptive grids. In fact, the restriction of the nonuniform or adaptive grid of the original problem can be used for the subproblems associated with the moving PMLs, as long as the grid can resolve the moving PML with sufficient accuracy. Since the multifrontal methods for nonuniform and adaptive grids are readily available [1], [21], it makes the current approach more flexible compared with the one based on the hierarchical matrix representation in the previous paper [12].

Acknowledgment. The authors thank the reviewers for their comments and suggestions.

REFERENCES

[1] P. R. Amestoy, I. S. Duff, J.-Y. L'Excellent, and J. Koster, *A fully asynchronous multifrontal solver using distributed dynamic scheduling*, SIAM J. Matrix Anal. Appl., 23 (2001), pp. 15–41.

[2] A. Atle and B. Engquist, *On surface radiation conditions for high-frequency wave scattering*, J. Comput. Appl. Math., 204 (2007), pp. 306–316.

[3] A. Bayliss, C. I. Goldstein, and E. Turkel, *An iterative method for the Helmholtz equation*, J. Comput. Phys., 49 (1983), pp. 443–457.

[4] J.-D. Benamou and B. Desprès, *A domain decomposition method for the Helmholtz equation and related optimal control problems*, J. Comput. Phys., 136 (1997), pp. 68–82.

[5] J.-P. Berenger, *A perfectly matched layer for the absorption of electromagnetic waves*, J. Comput. Phys., 114 (1994), pp. 185–200.

[6] A. Brandt and I. Livshits, *Wave-ray multigrid method for standing wave equations*, Electron. Trans. Numer. Anal., 6 (1997), pp. 162–181.

[7] W. C. Chew and W. H. Weedon, *A 3-d perfectly matched medium from modified Maxwell's equations with stretched coordinates*, Microw. Opt. Technol. Lett., 7 (1994), pp. 599–604.

[8] B. Després, *Domain decomposition method and the Helmholtz problem*, in Mathematical and Numerical Aspects of Wave Propagation Phenomena (Strasbourg, 1991), SIAM, Philadelphia, 1991, pp. 44–52.

[9] J. Duff and J. Reid, *The multifrontal solution of indefinite sparse symmetric linear equations*, ACM Trans. Math. Software, 9 (1983), pp. 302–325.

[10] H. C. Elman, O. G. Ernst, and D. P. O'Leary, *A multigrid method enhanced by Krylov subspace iteration for discrete Helmholtz equations*, SIAM J. Sci. Comput., 23 (2001), pp. 1291–1315.

[11] B. Engquist and L. Ying, *Fast directional multilevel algorithms for oscillatory kernels*, SIAM J. Sci. Comput., 29 (2007), pp. 1710–1737.

[12] B. Engquist and L. Ying, *Sweeping preconditioner for the Helmholtz equation: Hierarchical matrix representation*, Communications in Pure and Applied Mathematics, 64 (2011), pp. 697–795.

[13] Y. A. Erlangga, *Advances in iterative methods and preconditioners for the Helmholtz equation*, Arch. Comput. Methods Eng., 15 (2008), pp. 37–66.

[14] Y. A. Erlangga, C. W. Oosterlee, and C. Vuik, *A novel multigrid based preconditioner for heterogeneous Helmholtz problems*, SIAM J. Sci. Comput., 27 (2006), pp. 1471–1492.

[15] J. George, *Nested dissection of a regular finite element mesh*, SIAM J. Numer. Anal., 10 (1973), pp. 345–363.

[16] W. Hackbusch, *A sparse matrix arithmetic based on $\mathcal{H}$-matrices. Part* I*: Introduction to $\mathcal{H}$-matrices*, Computing, 62 (1999), pp. 89–108.

[17] S. Johnson, *Notes on Perfectly Matched Layers*, Technical report, Massachusetts Institute of Technology, Cambridge, MA, 2010.

[18] G. A. Kriegsmann, A. Taflove, and K. R. Umashankar, *A new formulation of electromagnetic wave scattering using an on-surface radiation boundary condition approach*, IEEE Trans. Antennas and Propagation, 35 (1987), pp. 153–161.

[19] A. Laird and M. Giles, *Preconditioned Iterative Solution of the* $2D$ *Helmholtz Equation*, Technical report, NA 02-12, Computing Lab, Oxford University, Oxford, UK, 2002.

[20] L. Lin, C. Yang, J. Lu, L. Ying, and W. E, *A fast parallel algorithm for selected inversion of structured sparse matrices with application to* $2D$ *electronic structure calculations*, SIAM J. Sci. Comput., 33 (2010) pp. 1329–1351.

[21] L. Lin, C. Yang, J. Meza, J. Lu, L. Ying, and W. E, *Selinv—An algorithm for selected inversion of a sparse symmetric matrix*, ACM Trans. Math. Software, 37 (2011).

[22] J. Liu, *The multifrontal method for sparse matrix solution: Theory and practice*, SIAM Rev., 34 (1992), pp. 82–109.

[23] D. Osei-Kuffuor and Y. Saad, *Preconditioning Helmholtz linear systems*. Appl. Numer. Math., 60 (2010).

[24] V. Rokhlin, *Diagonal forms of translation operators for the Helmholtz equation in three dimensions*, Appl. Comput. Harmon. Anal., 1 (1993), pp. 82–93.

[25] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed., Society for Industrial and Applied Mathematics, Philadelphia, 2003.

[26] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.

[27] A. Toselli, *Some Results on Overlapping Schwarz Methods for the Helmholtz Equation Employing Perfectly Matched Layers*. Technical report, Department of Computer Science, New York University, New York, NY, 1998.