Recursive Sweeping Preconditioner for the 3D Helmholtz Equation

Fei Liu[♯] and Lexing Ying[†♯]
† Department of Mathematics, Stanford University
♯ Institute for Computational and Mathematical Engineering, Stanford University

Feb 2015

Abstract

This paper introduces the recursive sweeping preconditioner for the numerical solution of the Helmholtz equation in 3D. This is based on the earlier work of the sweeping preconditioner with the moving perfectly matched layers (PMLs). The key idea is to apply the sweeping preconditioner recursively to the quasi-2D auxiliary problems introduced in the 3D sweeping preconditioner. Compared to the non-recursive 3D sweeping preconditioner, the setup cost of this new approach drops from $O(N^{4/3})$ to $O(N)$, the application cost per iteration drops from $O(N \log N)$ to $O(N)$, and the iteration count only increases mildly when combined with the standard GMRES solver. Several numerical examples are tested and the results are compared with the non-recursive sweeping preconditioner to demonstrate the efficiency of the new approach.

Keyword. Helmholtz equation, perfectly matched layers, preconditioners, high frequency waves.

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

1 Introduction

Let the domain of interest be the unit cube $D = (0,1)^3$ for simplicity. The time-independent wave field $u(x)$ satisfies the Helmholtz equation

$$\Delta u(x) + \frac{\omega^2}{c^2(x)} u(x) = f(x), \quad \forall x \in D, \tag{1}$$

where ω is the angular frequency, $c(x)$ is the velocity field with a bound $c_{\min} \le c(x) \le c_{\max}$ where $c_{\min}$ and $c_{\max}$ are assumed to be of $\Theta(1)$, and $f(x)$ is the time-independent external force. The typical boundary conditions for this problem are approximations of the Sommerfeld radiation condition, which means that the wave is absorbed by the boundary and there is no reflection coming from it. Other boundary conditions, such as the Dirichlet boundary condition, can also be specified on part of the boundary depending on the modeling setup.

In this setting, $\omega/(2\pi)$ is the typical wave number of the problem and $\lambda = 2\pi/\omega$ is the typical wavelength. For most applications, the Helmholtz equation is discretized with at least a few number of points (typically 4 to 20) per wavelength. So the number of points n in each direction is at least proportional to ω. As a result, the total degree of freedom $N = n^3 = \Omega(\omega^3)$ can be very large for high frequency 3D problems. In addition, the corresponding discrete system is highly indefinite and the standard iterative solvers and/or preconditioners are no longer efficient for such problems. These together make the problem challenging for numerical solution. We refer to the review article [9] by Ernst and Gander for more details on this.

Recently in [7], Engquist and Ying developed a sweeping preconditioner using the moving perfectly matched layers (PMLs) and obtained essentially linear solve times for 3D high frequency Helmholtz equations. A key step of that approach is to approximate the 3D problem with a sequence of $O(n)$ PML-padded auxiliary quasi-2D problems, each of which can be solved efficiently with sparse direct method such as the nested dissection algorithm. As an extension, this paper applies the sweeping idea recursively to further reduce each auxiliary quasi-2D problem into a sequence of PML-padded quasi-1D problems, each of which can be solved easily with the sparse LDU factorization for banded systems. As a result, the setup cost of the preconditioner improves from $O(N^{4/3})$ to $O(N)$ and the application cost reduces from $O(N \log N)$ to $O(N)$.

There has been a vast literature on iterative methods and preconditioners for the Helmholtz equation and we refer to the review articles [8] by Erlangga and [9] by Ernst and Gander for a rather complete discussion. The discussion here only touches on the methods that share similarity with the sweeping preconditioners. The analytic ILU factorization (AILU) [10] is the first to use incomplete LDU factorizations for preconditioning the Helmholtz equation. Compared to the moving PML sweeping preconditioner, the method uses the absorbing boundary condition (ABC), which is less effective compared to the PML, and hence the iteration count grows much more rapidly.

Since the sweeping preconditioners [6, 7] were proposed, there have been a number of exciting developments for the numerical solutions of the high frequency Helmholtz equation, including but not limited to [15, 14, 18, 16, 17, 19, 2, 3, 20]. In [15], Stolk proposed a domain decomposition algorithm that utilizes suitable transmission conditions based on the PMLs between the subdomains to achieve a near-linear cost. In [14], Poulson et al discussed a parallel version of the moving PML sweeping preconditioner to deal with large scale problems from applications such as seismic inversion. In [18, 16, 17], Tsuji and co-authors extended the moving PML sweeping preconditioner method to other time-harmonic wave equations and more general numerical discretization schemes. In [19], Vion and Geuzaine proposed a double sweep algorithm, studied several implementations of the absorbing boundary conditions, and compared their numerical performance. Finally in [2, 3], Chen and Xiang introduced a sweeping-style domain decomposition method where the emphasis was on the source transferring between the adjacent subdomains. In [20], Zepeda-Núñez and Demanet developed a novel parallel domain decomposition method that uses transmission conditions to define explicitly the up- and down-going waves.

The rest of the paper is organized as follows. We first state the problem and the discretization used in Section 2. Section 3 reviews the non-recursive moving PML sweeping preconditioner proposed in [7]. Section 4 discusses in detail the recursive sweeping preconditioner. Numerical results are presented in Section 5. Finally, the conclusion and some future directions are provided in Section 6.

2 Problem Formulation

Following [7], we assume that the perfectly matched layer (PML) [1, 4, 12] is utilized at part of the boundary where the Sommerfeld radiation condition is specified. The sweeping preconditioner in

[7] requires that at least one of the six faces of the domain $D = (0,1)^3$ is specified with the PML boundary condition. As we shall see soon, the recursive sweeping preconditioner instead requires the PML condition to be specified at least at two non-parallel faces. Without loss of generality, we assume that it is specified at $x_2 = 0$ and $x_3 = 0$. There is no restriction on the type of boundary conditions specified on the other four faces. However, to simplify the discussion, we assume that the Dirichlet condition is used. The PML boundary condition introduces auxiliary functions

$$\sigma(x) = \begin{cases} \dfrac{C}{\eta}\left(\dfrac{x-\eta}{\eta}\right)^2, & x \in [0,\eta], \\ 0, & x \in (\eta, 1], \end{cases}$$

and

$$s(x) = \left(1 + i\frac{\sigma(x)}{\omega}\right)^{-1}, \quad s_1(x) \equiv 1, \quad s_2(x) = s(x_2), \quad s_3(x) = s(x_3),$$

where C is an appropriate positive constant independent of ω, and η is the PML width, which is typically around one wavelength. The Helmholtz equation with PML is

$$\begin{cases} \left((s_1\partial_1)(s_1\partial_1) + (s_2\partial_2)(s_2\partial_2) + (s_3\partial_3)(s_3\partial_3) + \dfrac{\omega^2}{c^2(x)}\right) u(x) = f(x), & \forall x \in D = (0,1)^3, \\ u(x) = 0, & \forall x \in \partial D. \end{cases} \tag{2}$$

It is typically assumed that that the support of $f(x)$ is in $(0,1) \times (\eta,1) \times (\eta,1)$, which means that the force is not located in the PML region. The cube $[0,1]^3$ is discretized with a Cartesian grid where the grid size is $h = \frac{1}{n+1}$ and n is proportional to ω. The set of all the interior points of the grid is given by

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

and the degree of freedom is $N = n^3$.

Applying the standard 7-point finite difference stencil results in the discretized system

$$\begin{aligned} &\frac{(s_1)_{i,j,k}}{h}\left(\frac{(s_1)_{i+1/2,j,k}}{h}(u_{i+1,j,k} - u_{i,j,k}) - \frac{(s_1)_{i-1/2,j,k}}{h}(u_{i,j,k} - u_{i-1,j,k})\right) \\ &+\frac{(s_2)_{i,j,k}}{h}\left(\frac{(s_2)_{i,j+1/2,k}}{h}(u_{i,j+1,k} - u_{i,j,k}) - \frac{(s_2)_{i,j-1/2,k}}{h}(u_{i,j,k} - u_{i,j-1,k})\right) \\ &+\frac{(s_3)_{i,j,k}}{h}\left(\frac{(s_3)_{i,j,k+1/2}}{h}(u_{i,j,k+1} - u_{i,j,k}) - \frac{(s_3)_{i,j,k-1/2}}{h}(u_{i,j,k} - u_{i,j,k-1})\right) \\ &+\left(\frac{\omega^2}{c^2}\right)_{i,j,k} u_{i,j,k} = f_{i,j,k}, \quad \forall 1 \le i,j,k \le n, \end{aligned} \tag{3}$$

where the subscript (i,j,k) means that the corresponding function is evaluated at the point $p_{i,j,k} = (ih, jh, kh)$ and the definition of the points here extends to half integers as well. The computational task is to solve (3) efficiently. We note that, unlike the symmetric version adopted in [6, 7], here the nonsymmetric version of the equation is used. Figure 1 provides an illustration of the computational domain and the discretization grid.

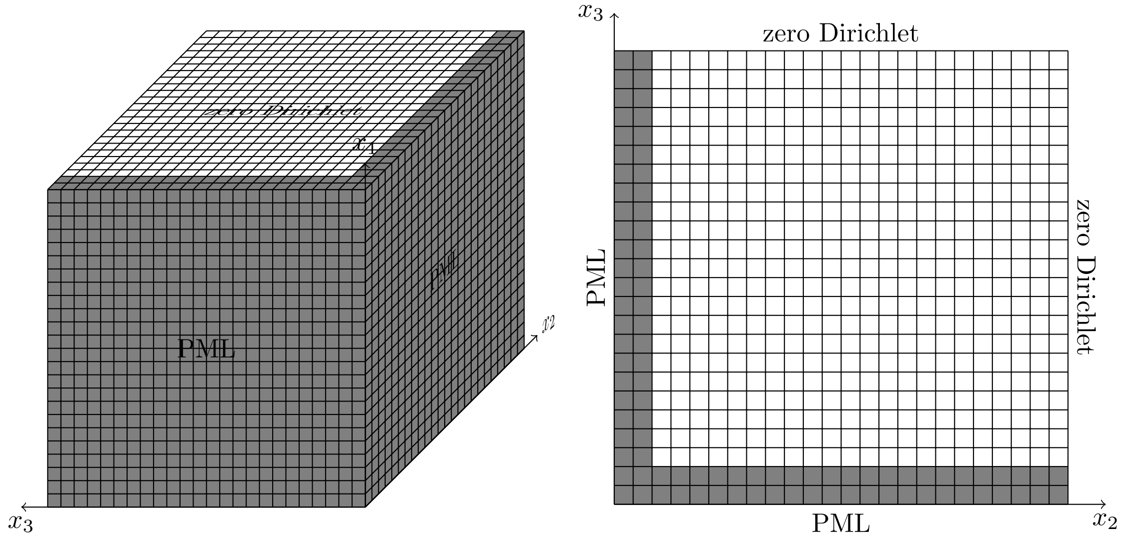


Figure 1: The domain of interest. Left is a 3D view of the domain. Right is an x_2-x_3 cross section view, where each cell stands for a 1D column. The gray area stands for the PML region.

3 Review of the Sweeping Preconditioner with Moving PML

This section gives a brief review of the non-recursive moving PML sweeping preconditioner proposed in [7] for completeness. More details can be found in the original paper [7]. The starting point of the sweeping preconditioner is a block LDU factorization called the sweeping factorization. To build this factorization, the algorithm sweeps along the x_3 direction starting from the face $x_3 = 0$. The unknowns with subscript index (i, j, k) are ordered with column-major order, i.e., first dimension 1, then dimension 2, and finally dimension 3. We define the vectors

$$u = [u_{1,1,1}, \dots, u_{n,1,1}, \dots, u_{n,n,1}, \dots, u_{n,n,n}]^T,$$
$$f = [f_{1,1,1}, \dots, f_{n,1,1}, \dots, f_{n,n,1}, \dots, f_{n,n,n}]^T.$$

By introducing

$$P_m = \{p_{1,1,m}, \dots, p_{n,1,m}, \dots, p_{n,n,m}\}$$

as the points on the m-th plane and also

$$u_{:,:,m} = [u_{1,1,m}, \dots, u_{n,1,m}, \dots, u_{n,n,m}]^T,$$
$$f_{:,:,m} = [f_{1,1,m}, \dots, f_{n,1,m}, \dots, f_{n,n,m}]^T,$$

one can write the system (3) compactly as $Au = f$ with the following block form

$$\begin{bmatrix} A_{1,1} & A_{1,2} & & \\ A_{2,1} & A_{2,2} & \ddots & \\ & \ddots & \ddots & A_{n-1,n} \\ & & A_{n,n-1} & A_{n,n} \end{bmatrix} \begin{bmatrix} u_{:,:,1} \\ u_{:,:,2} \\ \vdots \\ u_{:,:,n} \end{bmatrix} = \begin{bmatrix} f_{:,:,1} \\ f_{:,:,2} \\ \vdots \\ f_{:,:,n} \end{bmatrix}. \tag{4}$$

By defining S_k and T_k recursively via

$$S_1 = A_{1,1}, \quad T_1 = S_1^{-1},$$
$$S_m = A_{m,m} - A_{m,m-1} T_{m-1} A_{m-1,m}, \quad T_m = S_m^{-1}, \quad m = 2, \ldots, n,$$

the standard block LDU factorization of the block tridiagonal matrix A is

$$A = L_1 \ldots L_{n-1} \begin{bmatrix} S_1 & & \\ & \ddots & \\ & & S_n \end{bmatrix} U_{n-1} \ldots U_1,$$

where L_m and U_m are the corresponding unit lower and upper triangular matrices with the only non-zero off-diagonal blocks

$$L_m(P_{m+1}, P_m) = A_{m+1,m} T_m, \quad U_m(P_m, P_{m+1}) = T_m A_{m,m+1}, \quad m = 1, \ldots, n-1.$$

It is not difficult to see that computing this factorization takes $O(N^{7/3})$ steps. Once it is available, u can be computed in $O(N^{5/3})$ steps by

$$u = \begin{bmatrix} u_{:,:,1} \\ \vdots \\ u_{:,:,n} \end{bmatrix} = A^{-1} f = U_1^{-1} \ldots U_{n-1}^{-1} \begin{bmatrix} T_1 & & \\ & \ddots & \\ & & T_n \end{bmatrix} L_{n-1}^{-1} \ldots L_1^{-1} f$$

The main disadvantage of the above algorithm is, S_m and T_m are in general dense matrices of size $n^2 \times n^2$ so the corresponding dense linear algebra operations are expensive. The sweeping preconditioner overcomes this difficulty by approximating T_m efficiently for P_m with $mh \in (\eta, 1]$, i.e., for P_m not in the PML region at the face $x_3 = 0$. The key point is to consider the physical meaning of T_m. From now on let us assume $\eta = bh$ which implies that there are b layers in the PML region at $x_3 = 0$. Restricting the factorization to the upper-left $m \times m$ block of A where $m = b+1, \ldots, n$ gives

$$\begin{bmatrix} A_{1,1} & A_{1,2} & & \\ A_{2,1} & A_{2,2} & \ddots & \\ & \ddots & \ddots & A_{m-1,m} \\ & & A_{m,m-1} & A_{m,m} \end{bmatrix} = L_1 \ldots L_{m-1} \begin{bmatrix} S_1 & & & \\ & S_2 & & \\ & & \ddots & \\ & & & S_m \end{bmatrix} U_{m-1} \ldots U_1,$$

where L_t and U_t are redefined by restricting to their upper left $m \times m$ blocks. Inverting both sides leads to

$$\begin{bmatrix} A_{1,1} & A_{1,2} & & \\ A_{2,1} & A_{2,2} & \ddots & \\ & \ddots & \ddots & A_{m-1,m} \\ & & A_{m,m-1} & A_{m,m} \end{bmatrix}^{-1} = U_1^{-1} \ldots U_{m-1}^{-1} \begin{bmatrix} T_1 & & & \\ & T_2 & & \\ & & \ddots & \\ & & & T_m \end{bmatrix} L_{m-1}^{-1} \ldots L_1^{-1}.$$

The left-hand side is the discrete half-space Green's function with Dirichlet zero boundary condition at $x_3 = (m+1)h$ and a straightforward calculation shows that the lower-right block of the right-hand side is T_m. Therefore, T_m is the discrete half-space Green's function restricted to the m-th layer. Note that, the PML at $x_3 = 0$ is used to simulate an absorbing boundary condition. If we assume that there is little reflection during the transmission of the wave, we can approximate T_m by placing the PML right next to the m-th layer since the domain of interest is only the m-th layer (see Figure 2). This is the key idea of the moving PML sweeping preconditioner, where the operator T_m is numerically approximated by putting the PML right next to the domain of interest and solving a much smaller system to save the computational cost.

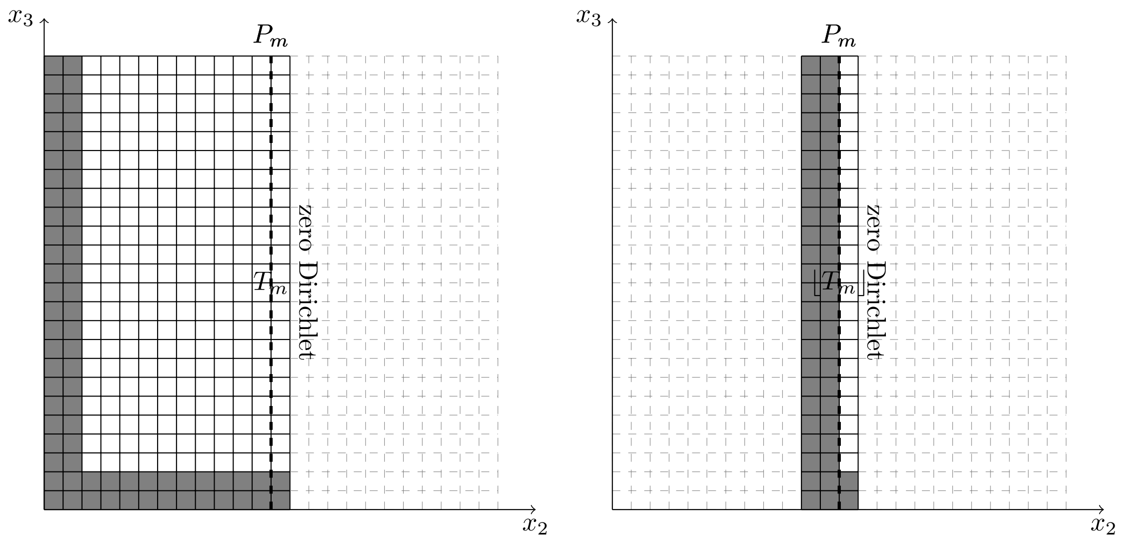


Figure 2: Left: T_m is the restriction to P_m (the dashed grid) of the half space Green's function on the solid grid. Right: By moving the PML right next to the layer P_m, the operator T_m is approximated by solving the equation on a much smaller grid.

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

$$\begin{cases} \left((s_1\partial_1)(s_1\partial_1) + (s_2\partial_2)(s_2\partial_2) + (s_3^m\partial_3)(s_3^m\partial_3) + \dfrac{\omega^2}{c^2(x)}\right) v(x) = g(x), & \forall x \in D_m, \\ v(x) = 0, & \forall x \in \partial D_m, \end{cases}$$

where $s_3^m(x) = s(x_3 - (m - b)h)$. The domain D_m is discretized with the partial grid

$$P_{(m-b+1):m} := \{P_t : m - b + 1 \leq t \leq m\}.$$

Applying the same central finite difference scheme gives rise to the corresponding discretized system, denoted as

$$H_m v = g, \quad m = b + 1, \ldots, n.$$

To approximate T_m, we numerically define operator $\lfloor T_m \rfloor : \alpha \in \mathbb{C}^{n^2} \to \beta \in \mathbb{C}^{n^2}$ by the following procedure:

1. Introduce a vector g defined on $P_{(m-b+1):m}$ by setting α to the layer P_m and zero everywhere else.
2. Solve the discretized auxiliary problem $H_m v = g$ on $P_{(m-b+1):m}$ with g from step 1.
3. Set β as the restriction on P_m of the solution v from step 2.

The discretized system is a quasi-2D system as b is typically a small constant, so the system can be solved efficiently by the nested dissection method [11, 5, 13].

The first b layers, which are in the PML region of the original problem (2), need to be handled with a slight difference. Define

$$\begin{aligned} u_{:,:,1:b} &= [u_{:,:,1}^T, \ldots, u_{:,:,b}^T]^T, \\ f_{:,:,1:b} &= [f_{:,:,1}^T, \ldots, f_{:,:,b}^T]^T. \end{aligned}$$

Then the system $Au = f$ can be written as

$$\begin{bmatrix} A_{1:b,1:b} & A_{1:b,b+1} & & \\ A_{b+1,1:b} & A_{b+1,b+1} & \ddots & \\ & \ddots & \ddots & A_{n-1,n} \\ & & A_{n,n-1} & A_{n,n} \end{bmatrix} \begin{bmatrix} u_{:,:,1:b} \\ u_{:,:,b+1} \\ \vdots \\ u_{:,:,n} \end{bmatrix} = \begin{bmatrix} f_{:,:,1:b} \\ f_{:,:,b+1} \\ \vdots \\ f_{:,:,n} \end{bmatrix}.$$

For the first b layers, we simply define $\lfloor T_{1:b} \rfloor$ as the inverse operator of $H_b := A_{1:b,1:b}$. However, it is essential that $\lfloor T_{1:b} \rfloor$ is stored in a factorized form by applying the nested dissection method to H_b, since $H_b v = g$ is also a quasi-2D problem.

Based on the above discussion, the setup algorithm of the moving PML sweeping preconditioner is given in Algorithm 1.

Algorithm 1 Construction of the moving PML sweeping preconditioner of the system (3). Complexity $= O(b^3 n^4) = O(b^3 N^{4/3})$.

```
Construct the nested dissection factorization of $H_b$, which defines $\lfloor T_{1:b} \rfloor$.
for $m = b+1, \ldots, n$ do
    Construct the nested dissection factorization of $H_m$, which defines $\lfloor T_m \rfloor$.
end for
```

Once the factorization is completed, $\lfloor T_{1:b} \rfloor$ and $\lfloor T_m \rfloor$ can be applied using the nested dissection factorization. The application process of the sweeping preconditioner is given in Algorithm 2.

4 Recursive Sweeping Preconditioner

Recall that the PML is also applied to the face $x_2 = 0$. Therefore, each quasi-2D auxiliary problem is itself a discretization of the Helmholtz equation with the PML specified on one side. Following the treatment in [7] for the 2D Helmholtz equation, it is natural to apply the same sweeping idea once again along the x_2 direction, instead of the nested dissection algorithm used in the previous section.

Algorithm 2 Computation of $u \approx A^{-1}f$ using the factorization from Algorithm 1. Complexity $= O(b^2 n^3 \log n) = O(b^2 N \log N)$.

```
$u_{:,:,1:b} = \lfloor T_{1:b} \rfloor f_{:,:,1:b}$
$u_{:,:,b+1} = \lfloor T_{b+1} \rfloor (f_{:,:,b+1} - A_{b+1,1:b} u_{:,:,1:b})$
for $m = b+1, \dots, n-1$ do
  $u_{:,:,m+1} = \lfloor T_{m+1} \rfloor (f_{:,:,m+1} - A_{m+1,m} u_{:,:,m})$
end for
for $m = n-1, \dots, b+1$ do
  $u_{:,:,m} = u_{:,:,m} - \lfloor T_m \rfloor (A_{m,m+1} u_{:,:,m+1})$
end for
$u_{:,:,1:b} = u_{:,:,1:b} - \lfloor T_{1:b} \rfloor (A_{1:b,b+1} u_{:,:,b+1})$
```

4.1 Inner sweeping

Recall that the quasi-2D subproblems of the non-recursive sweeping preconditioners are $H_m v = g, m = b, \dots, n$. Since they have essentially the same structure, it is sufficient to consider a single system $\widetilde{A} v = g$ where $\widetilde{A}$ can be anyone of H_m. Here the accent mark is to emphasize that the problem under consideration is quasi-2D. To formalize the sweeping preconditioner along the x_2 direction, we define, up to a translation,

$$\widetilde{P} = \{p_{i,j,k} = (ih, jh, kh) : 1 \le i, j \le n, 1 \le k \le b\},$$

to be the discretization grid. For each $m = 1, \dots, n$, let

$$\begin{aligned}
\widetilde{P}_m &= \{p_{1,m,1}, \dots, p_{1,m,b}, \dots, p_{n,m,b}\}, \\
v_{:,m,:} &= [v_{1,m,1}, \dots, v_{1,m,b}, \dots, v_{n,m,b}]^T, \\
g_{:,m,:} &= [g_{1,m,1}, \dots, g_{1,m,b}, \dots, g_{n,m,b}]^T.
\end{aligned}$$

For the first b layers in the x_2 direction, we also define

$$\begin{aligned}
\widetilde{P}_{1:b} &= \{\widetilde{P}_1, \dots, \widetilde{P}_b\}, \\
v_{:,1:b,:} &= [v_{:,1,:}^T, \dots, v_{:,b,:}^T]^T, \\
g_{:,1:b,:} &= [g_{:,1,:}^T, \dots, g_{:,b,:}^T]^T.
\end{aligned}$$

In this section, we reorder the vectors v, g by grouping the 3rd dimension first and applying the column-major ordering to dimensions 1 and 2:

$$\begin{aligned}
v &= [v_{:,1,:}^T, \dots, v_{:,n,:}^T]^T, \\
g &= [g_{:,1,:}^T, \dots, g_{:,n,:}^T]^T.
\end{aligned}$$

With this ordering, the corresponding system $\widetilde{A} v = g$ is written as

$$\begin{bmatrix}
\widetilde{A}_{1:b,1:b} & \widetilde{A}_{1:b,b+1} & & \\
\widetilde{A}_{b+1,1:b} & \widetilde{A}_{b+1,b+1} & \ddots & \\
 & \ddots & \ddots & \widetilde{A}_{n-1,n} \\
 & & \widetilde{A}_{n,n-1} & \widetilde{A}_{n,n}
\end{bmatrix}
\begin{bmatrix} v_{:,1:b,:} \\ v_{:,b+1,:} \\ \vdots \\ v_{:,n,:} \end{bmatrix}
=
\begin{bmatrix} g_{:,1:b,:} \\ g_{:,b+1,:} \\ \vdots \\ g_{:,n,:} \end{bmatrix}.$$

For the block LDU factorization of $\widetilde{A}$, we define

$$\widetilde{S}_{1:b} = \widetilde{A}_{1:b,1:b}, \quad \widetilde{T}_{1:b} = \widetilde{S}_{1:b}^{-1},$$
$$\widetilde{S}_{b+1} = \widetilde{A}_{b+1,b+1} - \widetilde{A}_{b+1,1:b}\widetilde{T}_{1:b}\widetilde{A}_{1:b,b+1}, \quad \widetilde{T}_{b+1} = \widetilde{S}_{b+1}^{-1},$$
$$\widetilde{S}_m = \widetilde{A}_{m,m} - \widetilde{A}_{m,m-1}\widetilde{T}_{m-1}\widetilde{A}_{m-1,m}, \quad \widetilde{T}_m = \widetilde{S}_m^{-1}, \quad m = b+2,\dots,n,$$

then $\widetilde{A}$ can be factorized as

$$\widetilde{A} = \widetilde{L}_{1:b}\widetilde{L}_{b+1}\dots\widetilde{L}_{n-1} \begin{bmatrix} \widetilde{S}_{1:b} & & & \\ & \widetilde{S}_{b+1} & & \\ & & \ddots & \\ & & & \widetilde{S}_n \end{bmatrix} \widetilde{U}_{n-1}\dots\widetilde{U}_{b+1}\widetilde{U}_{1:b},$$

where the non-zero off-diagonal blocks of the unit lower and upper triangular matrices $\widetilde{L}_m$ and $\widetilde{U}_m$ are given by

$$\widetilde{L}_{1:b}(\widetilde{P}_{b+1},\widetilde{P}_{1:b}) = \widetilde{A}_{b+1,1:b}\widetilde{T}_{1:b}, \quad \widetilde{U}_{1:b}(\widetilde{P}_{1:b},\widetilde{P}_{b+1}) = \widetilde{T}_{1:b}\widetilde{A}_{1:b,b+1},$$
$$\widetilde{L}_m(\widetilde{P}_{m+1},\widetilde{P}_m) = \widetilde{A}_{m+1,m}\widetilde{T}_m, \quad \widetilde{U}_m(\widetilde{P}_m,\widetilde{P}_{m+1}) = \widetilde{T}_m\widetilde{A}_{m,m+1}, \quad m = b+1,\dots,n-1.$$

Then the solution v can be computed by

$$v = \begin{bmatrix} v_{:,1:b,:} \\ v_{:,b+1,:} \\ \vdots \\ v_{:,n,:} \end{bmatrix} = \widetilde{A}^{-1}g = \widetilde{U}_{1:b}^{-1}\widetilde{U}_{b+1}^{-1}\dots\widetilde{U}_{n-1}^{-1} \begin{bmatrix} \widetilde{T}_{1:b} & & & \\ & \widetilde{T}_{b+1} & & \\ & & \ddots & \\ & & & \widetilde{T}_n \end{bmatrix} \widetilde{L}_{n-1}^{-1}\dots\widetilde{L}_{b+1}^{-1}\widetilde{L}_{1:b}^{-1}g.$$

By comparing the factorization of the upper-left $(m-b+1)\times(m-b+1)$ block of $\widetilde{A}$, where $m = b+1,\dots,n$, we have

$$\begin{bmatrix} \widetilde{A}_{1:b,1:b} & \widetilde{A}_{1:b,b+1} & & \\ \widetilde{A}_{b+1,1:b} & \widetilde{A}_{b+1,b+1} & \ddots & \\ & \ddots & \ddots & \widetilde{A}_{m-1,m} \\ & & \widetilde{A}_{m,m-1} & \widetilde{A}_{m,m} \end{bmatrix} = \widetilde{L}_{1:b}\widetilde{L}_{b+1}\dots\widetilde{L}_{m-1} \begin{bmatrix} \widetilde{S}_{1:b} & & & \\ & \widetilde{S}_{b+1} & & \\ & & \ddots & \\ & & & \widetilde{S}_m \end{bmatrix} \widetilde{U}_{m-1}\dots\widetilde{U}_{b+1}\widetilde{U}_{1:b},$$

where $\widetilde{L}_t$ and $\widetilde{U}_t$ are redefined as their restrictions to their top-left $(m-b+1)\times(m-b+1)$ blocks. Inverting both sides gives

$$\begin{bmatrix} \widetilde{A}_{1:b,1:b} & \widetilde{A}_{1:b,b+1} & & \\ \widetilde{A}_{b+1,1:b} & \widetilde{A}_{b+1,b+1} & \ddots & \\ & \ddots & \ddots & \widetilde{A}_{m-1,m} \\ & & \widetilde{A}_{m,m-1} & \widetilde{A}_{m,m} \end{bmatrix}^{-1} = \widetilde{U}_{1:b}^{-1}\widetilde{U}_{b+1}^{-1}\dots\widetilde{U}_{m-1}^{-1} \begin{bmatrix} \widetilde{T}_{1:b} & & & \\ & \widetilde{T}_{b+1} & & \\ & & \ddots & \\ & & & \widetilde{T}_m \end{bmatrix} \widetilde{L}_{m-1}^{-1}\dots\widetilde{L}_{b+1}^{-1}\widetilde{L}_{1:b}^{-1}.$$

Thus, by repeating the argument in Section 3, the matrix $\widetilde{T}_m$ is the restriction to the layer $\widetilde{P}_m$ of the discrete half-space Green's function. It can be approximated by $\lfloor\widetilde{T}_m\rfloor$, which is defined by solving a quasi-1D problem obtained by placing a moving PML right next to $x_2 = mh$ (see Figure 3). Each auxiliary quasi-1D problem in this inner sweeping step can be solved by the sparse block LDU factorization efficiently, with ordering the system by grouping dimension 3 and 2 first and dimension 1 last.

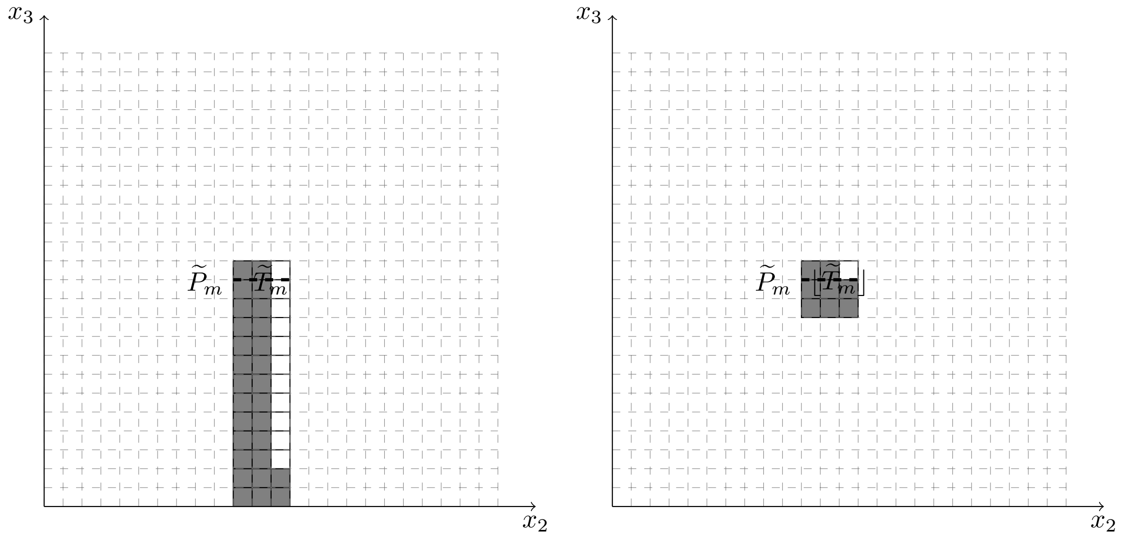


Figure 3: Left: $\widetilde{T}_m$ is the restriction to $\widetilde{P}_m$ (the dashed grid) of the Green's function on the quasi-2D solid grid. Right: By moving the PML right next to $\widetilde{P}_m$, the operator $\widetilde{T}_m$ is approximated by solving the problem on a quasi-1D grid.

More specifically, for each m, we introduce the auxiliary problem on the domain $\widetilde{D}_m = [0,1] \times [(m-b)h, (m+1)h] \times [0, (b+1)h]$:

$$\begin{cases} \left((s_1\partial_1)(s_1\partial_1) + (s_2^m\partial_2)(s_2^m\partial_2) + (s_3\partial_3)(s_3\partial_3) + \dfrac{\omega^2}{c^2(x)}\right) w(x) = q(x), & \forall x \in \widetilde{D}_m, \\ w(x) = 0, & \forall x \in \partial\widetilde{D}_m, \end{cases}$$

where $s_2^m(x) = s(x_2 - (m-b)h)$. The domain $\widetilde{D}_m$ is discretized with the grid

$$\widetilde{P}_{(m-b+1):m} := \{\widetilde{P}_t : m-b+1 \le t \le m\},$$

and the same central difference numerical scheme is used here. We denote the corresponding discretized system as $\widetilde{H}_m w = q$. Similar to the process described in Section 3, we define the operator $\lfloor\widetilde{T}_m\rfloor : \alpha \in \mathbb{C}^{nb} \to \beta \in \mathbb{C}^{nb}$ by the following procedure:

1. Introduce a vector q defined on the grid $\widetilde{P}_{(m-b+1):m}$ by setting α to the layer $\widetilde{P}_m$ and zero everywhere else.
2. Solve the auxiliary quasi-1D problem $\widetilde{H}_m w = q$ on $\widetilde{P}_{(m-b+1):m}$ with q from step 1.
3. Set β as the restriction on $\widetilde{P}_m$ of the solution w from step 2.

For the first b layers, $\lfloor \widetilde{T}_{1:b} \rfloor$ is simply defined as the inverse operator of $\widetilde{H}_b := \widetilde{A}_{1:b,1:b}$, which is essentially the same as $\widetilde{T}_{1:b}$, but implemented by using the sparse block LDU factorization of $\widetilde{H}_b$. Summarizing all this, the setup and application algorithm of the inner moving PML sweeping preconditioner are given in Algorithms 3 and 4, respectively.

Algorithm 3 Construction of the inner moving PML sweeping preconditioner of the quasi-2D problem $\widetilde{A}v = g$. Complexity $= O(b^6 n^2)$.

```
Construct the sparse block LDU factorization of $\widetilde{H}_b$, which defines $\lfloor \widetilde{T}_{1:b} \rfloor$.
for $m = b+1, \dots, n$ do
    Construct the sparse block LDU factorization of $\widetilde{H}_m$, which defines $\lfloor \widetilde{T}_m \rfloor$.
end for
```

Algorithm 4 Computation of $v \approx \widetilde{A}^{-1} g$ using the factorization from Algorithm 3. Complexity $= O(b^4 n^2)$.

```
$v_{:,1:b,:} = \lfloor \widetilde{T}_{1:b} \rfloor g_{:,1:b,:}$
$v_{:,b+1,:} = \lfloor \widetilde{T}_{b+1} \rfloor (g_{:,b+1,:} - \widetilde{A}_{b+1,1:b} v_{:,1:b,:})$
for $m = b+1, \dots, n-1$ do
    $v_{:,m+1,:} = \lfloor \widetilde{T}_{m+1} \rfloor (g_{:,m+1,:} - \widetilde{A}_{m+1,m} v_{:,m,:})$
end for
for $m = n-1, \dots, b+1$ do
    $v_{:,m,:} = v_{:,m,:} - \lfloor \widetilde{T}_m \rfloor (\widetilde{A}_{m,m+1} v_{:,m+1,:})$
end for
$v_{:,1:b,:} = v_{:,1:b,:} - \lfloor \widetilde{T}_{1:b} \rfloor (\widetilde{A}_{1:b,b+1} v_{:,b+1,:})$
```

4.2 Putting together

As we pointed out earlier, the matrix $\widetilde{A}$ can be anyone of $H_m, m = b, \dots, n$, where Algorithms 3 and 4 can be applied. Notice that solving the subproblems exactly with the nested dissection algorithm results in the approximation $\lfloor T_m \rfloor$ to T_m. This extra-level of approximation defines a further approximation, which shall be denoted by $\llbracket T_m \rrbracket : \alpha \in \mathbb{C}^{n^2} \to \beta \in \mathbb{C}^{n^2}$ (to be precise, for the first b layers, it is $\llbracket T_{1:b} \rrbracket : \alpha \in \mathbb{C}^{n^2 b} \to \beta \in \mathbb{C}^{n^2 b}$). The steps for carrying out $\llbracket T_m \rrbracket$ are similar to the ones for $\lfloor T_m \rfloor$ except that one uses Algorithms 3 and 4 to solve the quasi-2D problems approximately (instead of the nested dissection method that solves them exactly).

Given all these preparations, the setup algorithm of the recursive sweeping preconditioner can be summarized compactly in Algorithm 5 and the application algorithm is given in Algorithm 6.

In the outer loop of Algorithm 6, the unknowns are eliminated layer by layer in the x_3 direction. In the application of $\llbracket T_m \rrbracket$, there is the inner loop in which the unknowns in each quasi-2D problem are eliminated in the x_2 direction. The whole algorithm serves as a preconditioner for the original linear system (3). Notice that, in the recursive sweeping preconditioner, the quasi-2D problems are solved only approximately. Therefore, the overall accuracy might not be as good as the non-recursive method. But as we will show in the next section, the performance of the preconditioner is only mildly affected.

Algorithm 5 Construction of the recursive moving PML sweeping preconditioner of the linear system (3). Complexity $= O(b^6 n^3) = O(b^6 N)$.

```
Construct the inner moving PML sweeping preconditioner of $H_b$ by Algorithm 3. This gives $\llbracket T_{1:b} \rrbracket$.
for $m = b+1, \ldots, n$ do
    Construct the inner moving PML sweeping preconditioner of $H_m$ by Algorithm 3. This gives $\llbracket T_m \rrbracket$.
end for
```

Algorithm 6 Computation of $u \approx A^{-1} f$ using the factorization from Algorithm 5. Complexity $= O(b^4 n^3) = O(b^4 N)$.

```
$u_{:,:,1:b} = \llbracket T_{1:b} \rrbracket f_{:,:,1:b}$
$u_{:,:,b+1} = \llbracket T_{b+1} \rrbracket (f_{:,:,b+1} - A_{b+1,1:b} u_{:,:,1:b})$
for $m = b+1, \ldots, n-1$ do
    $u_{:,:,m+1} = \llbracket T_{m+1} \rrbracket (f_{:,:,m+1} - A_{m+1,m} u_{:,:,m})$
end for
for $m = n-1, \ldots, b+1$ do
    $u_{:,:,m} = u_{:,:,m} - \llbracket T_m \rrbracket (A_{m,m+1} u_{:,:,m+1})$
end for
$u_{:,:,1:b} = u_{:,:,1:b} - \llbracket T_{1:b} \rrbracket (A_{1:b,b+1} u_{:,:,b+1})$
```

The above algorithms are described in a way to present the main ideas clearly. In the actual implementations, a couple of modifications are taken in order to maximize the efficiency:

1. For each auxiliary problem, both in the inner loop and the outer loop, several layers are processed together instead of one layer.

2. For the PML introduced in the auxiliary problems, the number of layers in the auxiliary PML region does not have to match the number of layers b used for the boundary PML at $x_2 = 0$ and $x_3 = 0$. In fact, the thickness of the auxiliary PML is typically thinner for the sake of efficiency.

3. The problem we described above has zero Dirichlet boundary conditions on the other four faces of the cube. If instead, the PMLs are put on all the faces, then the sweeping preconditioner sweeps with two fronts from two opposite faces respectively and they meet in the middle with a subproblem with PML on both sides instead of only one side, as described in [7].

5 Numerical Results

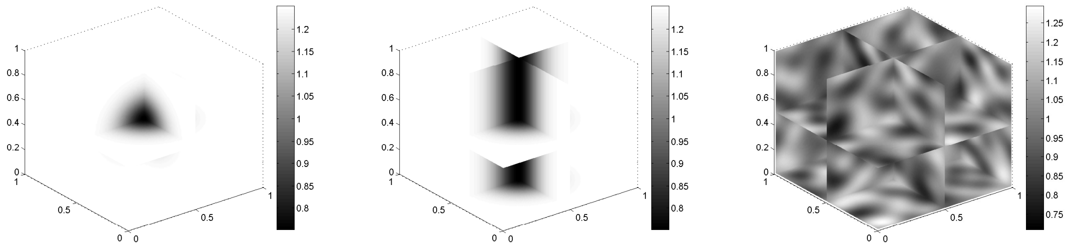
In this section we test several numerical examples to illustrate the performance of the recursive sweeping preconditioner. All algorithms are implemented in MATLAB and the tests are performed on a 2.0 GHz computer with 256 GB memory. We use the GMRES algorithm as the iterative solver with relative residual 10^{-3} and restart value 40 for the entire 3D system. The quasi-2D problems are solved approximately by applying the inner sweeping preconditioner only once for the sake of efficiency. The velocity fields and forces tested are kept the same with [7] so that the results can be compared easily. The PMLs are put on all six sides of the cube $[0,1]^3$ to simulate the Sommerfeld radiation condition.

We test three velocity fields (see Figure 4):

(a) A converging lens with a Gaussian profile at the center of the domain.

(b) A vertical waveguide with a Gaussian cross-section.

(c) A random velocity field.



(a) (b) (c)

Figure 4: The three velocity fields tested.

For each velocity field, the tests are performed for two external forces:

(a) A Gaussian point source centered at $(1/2, 1/2, 1/4)$.

(b) A Gaussian wave packet with wavelength comparable to the typical wavelength of the domain. The packet centers at $(1/2, 1/4, 1/4)$ and points to the direction $(0, 1/\sqrt{2}, 1/\sqrt{2})$.

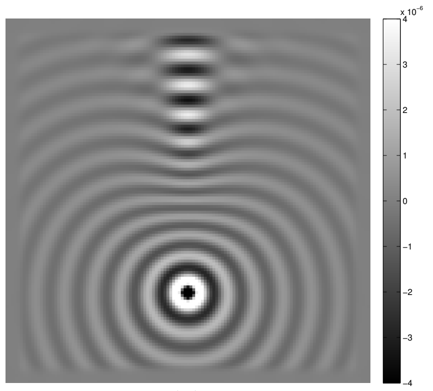
We vary the typical wave number $\omega/(2\pi)$, test the behavior of the recursive preconditioner, and compare the results with the non-recursive preconditioner.

In these tests, each wavelength is discretized with $q = 8$ points. The width of the PML at the boundary of the cube is $9h$, and the width of the auxiliary PML for the middle layers is $5h$. The number of layers processed in each auxiliary problem is 4. The algorithm sweeps with two fronts from $x_3 = 0$ and $x_3 = 1$ in the outer loop, and with two fronts from $x_2 = 0$ and $x_2 = 1$ in the inner loop.

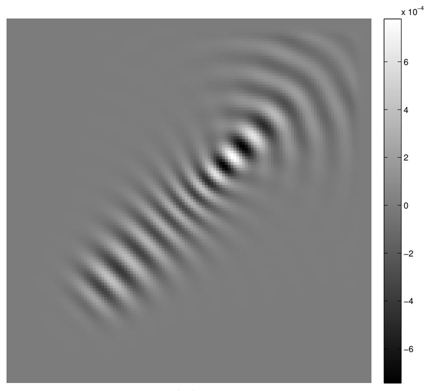
The results are reported in the following tables. T_{setup} is the time used to construct the preconditioner in seconds. T_{solve} is the time used to solve the system in the preconditioned GMRES solver in seconds and N_{iter} is the corresponding iteration number. "NR" stands for the original non-recursive method while "R" stands for the recursive method introduced in this paper. The "ratio" is the time cost of the recursive method over the non-recursive method. The numerical implementation of the non-recursive method is slightly improved as compared to [7], by incorporating a more accurate PML discretization. Therefore, the results here for the non-recursive method are better compared to the ones in [7].

From these tests we can make the following observations:

1. The setup time of the recursive preconditioner is significantly dropped compared to the non-recursive one. The advantage becomes more and more obvious when the problem size gets larger. This is because the setup cost of the recursive method scales like $O(N)$ and the non-recursive one scales like $O(N^{4/3})$.



(a)



(b)

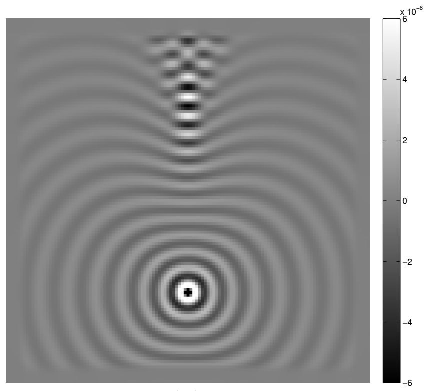
			T_{setup}				N_{iter}		T_{solve}		
$\omega/(2\pi)$	q	N	NR	R	ratio	$f(x)$	NR	R	NR	R	ratio
8	8	63^3	46.923	19.823	42%	(a)	3	4	12.313	16.355	133%
						(b)	4	4	14.973	16.862	113%
16	8	127^3	537.12	180.99	34%	(a)	3	4	116.44	169.34	145%
						(b)	4	4	150.67	168.65	112%
32	8	255^3	5927.0	1308.0	22%	(a)	4	5	1273.0	2039.8	160%
						(b)	4	5	1312.1	2070.4	158%

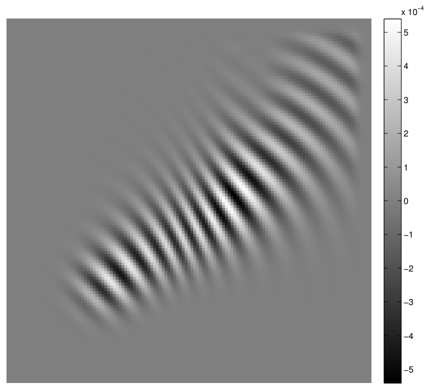
Table 1: Results for velocity field (a) in Figure 4. Solutions with $\omega/(2\pi) = 16$ at $x_1 = 0.5$ are presented.

2. The iteration number of the recursive preconditioner increases only slightly compared to the non-recursive one. Typically it needs about 1 more iteration. This is mainly because the recursive method solves the quasi-2D auxiliary problems approximately while the non-recursive one solves them accurately.

3. The application time of the recursive sweeping preconditioner is not as fast as the non-recursive method, due to a larger prefactor of the time complexity. However, we think this sacrifice is acceptable since a huge amount of time is saved in the setup process. One thing that needs to be pointed out is that the ratio of the solve time increases as the problem size increases, which seems to be unexpected since the solve time of the recursive method scales like $O(N)$ and the non-recursive one scales like $O(N \log N)$. The reason of this behavior is, when the problem size increases, the size of the dense linear algebra operations increases as well in the non-recursive method since the front size in the nested dissection method gets larger, while in the recursive method, the size of the dense linear algebra operations in the quasi-1D block LDU setup process and solve process is kept the same. Since MATLAB processes large scale dense linear algebra operations in a parallel way, the non-recursive method gains some advantages from this.

The memory cost of the recursive method is also advantageous. In our implementation, the recursive method costs only 30% memory compared to the non-recursive method in the $N = 255^3$


(a)


(b)

			T_{setup}				N_{iter}		T_{solve}		
$\omega/(2\pi)$	q	N	NR	R	ratio	$f(x)$	NR	R	NR	R	ratio
8	8	63^3	48.855	18.490	38%	(a)	3	4	11.249	15.996	142%
						(b)	3	5	11.048	19.581	177%
16	8	127^3	524.61	163.16	31%	(a)	4	5	152.32	212.59	140%
						(b)	3	5	111.71	213.24	191%
32	8	255^3	6038.8	1319.0	22%	(a)	5	6	1676.7	2471.9	147%
						(b)	4	5	1345.3	2084.6	155%

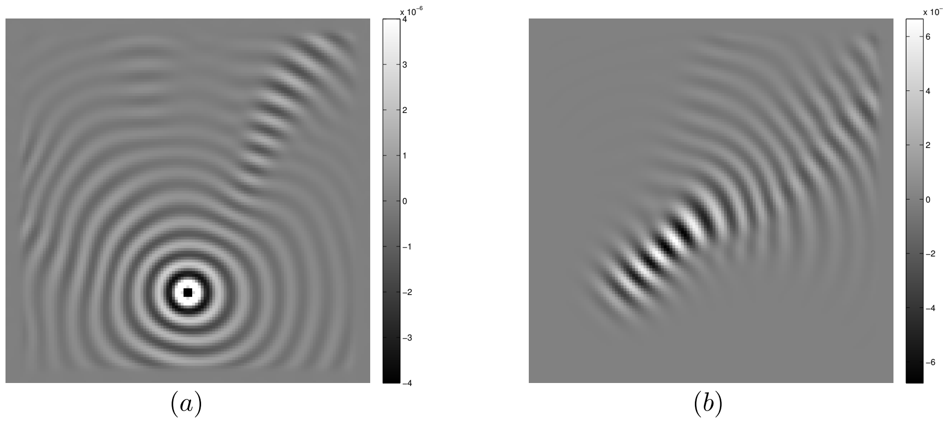
Table 2: Results for velocity field (b) in Figure 4. Solutions with $\omega/(2\pi) = 16$ at $x_1 = 0.5$ are presented.

case. Theoretically, the memory cost of the recursive method scales like $O(N)$ while the non-recursive one scales like $O(N \log(N))$. This is another main advantage of the recursive method.

6 Conclusion and Future Work

In this paper, we introduced a new recursive sweeping preconditioner for the 3D Helmholtz equation based on the moving PML sweeping preconditioner proposed in [7]. The idea of the sweeping preconditioner is used recursively for the auxiliary quasi-2D problems. Both the setup cost and application cost of the preconditioner are reduced to strict linear complexity. The iteration number remains essentially independent of the problem size when combined with the standard GMRES solver. Numerical results show that the setup time drops significantly compared to the non-recursive method, while the solve cost increases only slightly.

Several questions still remain open and some potential improvements can be made. First, we use the PML to simulate the Sommerfeld condition. Many other simulations of the absorbing boundary condition can be implemented and the recursive sweeping idea can be used as long as the stencil of the simulation is local. Second, the numerical scheme used in this paper is the standard central difference scheme, whose dispersion relationship is a poor approximation of the true one. More accurate numerical schemes can be implemented and the iteration number may be dropped



			T_{setup}				N_{iter}		T_{solve}		
$\omega/(2\pi)$	q	N	NR	R	ratio	$f(x)$	NR	R	NR	R	ratio
8	8	63^3	48.949	18.213	37%	(a)	4	4	16.487	16.252	99%
						(b)	4	4	15.747	16.513	105%
16	8	127^3	553.70	147.00	27%	(a)	4	4	158.41	170.59	108%
						(b)	5	5	196.91	215.84	110%
32	8	255^3	6024.1	1299.2	22%	(a)	5	5	1599.6	1929.1	121%
						(b)	5	6	1635.2	2314.5	142%

Table 3: Results for velocity field (c) in Figure 4. Solutions with $\omega/(2\pi) = 16$ at $x_1 = 0.5$ are presented.

potentially benefiting from the increment of the accuracy of the numerical scheme.

Parallel processing can also be introduced to the current recursive method. First, When sweeping from both sides of the domain, either in the outer loop of the algorithm or in the inner loop, the processing of the two fronts can be paralleled so in total it could be 4 times faster with parallelization theoretically. Second, the quasi-1D problems are solved by the block LDU factorization in the current setting. If instead, we use the 1D nested dissection algorithm for the quasi-1D problems, then it can be easily paralleled and the total cost will remain essentially the same. Last, one can notice that, the setup process of the algorithm is essentially $O(n^2)$ quasi-1D subproblems which are independent with each other so this process can be done in parallel, and compared to the original method, which contains only $O(n)$ quasi-2D independent subproblems, the potential advantages of parallelization in the setup stage is more obvious here.

There are also several other advantages of the recursive sweeping method that concern flexibility. First, as mentioned above, the setup process contains $O(n^2)$ quasi-1D independent subproblems. So if the velocity field is modified on a subdomain which involves only limited subproblems, then the factorization can be updated with only a slight modification on these involved subproblems. Compared to the original method, where the subproblems are $O(n)$ quasi-2D plates, the recursive method is more flexible on updating the factorization. This could be advantageous in seismic imaging where the velocity field is tested and modified frequently. Second, when the factorization for the $O(n^2)$ subproblems is done, there are naturally two ways of using the factorization. One is, as mentioned in this paper, sweeping along the x_3 direction in the outer loop, and sweeping along the x_2 direction in the inner loop. Another choice is to do the opposite, which is sweeping along the x_2 direction in the outer loop and the x_3 direction in the inner loop. Each of these two choices shows some "bias" since the residual of the system is accumulated in some "chosen" order. So one may ask that, is it possible to combine the two choices together to make the solve process more flexible such that the total solve time can be even less? This is another interesting question to be examined.

Acknowledgments

This work was partially supported by the National Science Foundation under award DMS-1328230 and the U.S. Department of Energy's Advanced Scientific Computing Research program under award DE-FC02-13ER26134/DE-SC0009409. We thank Lenya Ryzhik for providing computing resources and thank Laurent Demanet and Paul Childs for helpful discussions.

References

[1] J.-P. Berenger. A perfectly matched layer for the absorption of electromagnetic waves. *J. Comput. Phys.*, 114(2):185–200, 1994.

[2] Z. Chen and X. Xiang. A source transfer domain decomposition method for Helmholtz equations in unbounded domain. *SIAM J. Numer. Anal.*, 51(4):2331–2356, 2013.

[3] Z. Chen and X. Xiang. A source transfer domain decomposition method for Helmholtz equations in unbounded domain Part II: Extensions. *Numer. Math. Theory Methods Appl.*, 6(3):538–555, 2013.

[4] W. C. Chew and W. H. Weedon. A 3D perfectly matched medium from modified Maxwell's equations with stretched coordinates. *Microw. Opt. Techn. Let.*, 7(13):599–604, 1994.

[5] I. S. Duff and J. K. Reid. The multifrontal solution of indefinite sparse symmetric linear equations. *ACM Trans. Math. Software*, 9(3):302–325, 1983.

[6] B. Engquist and L. Ying. Sweeping preconditioner for the Helmholtz equation: hierarchical matrix representation. *Comm. Pure Appl. Math.*, 64(5):697–735, 2011.

[7] B. Engquist and L. Ying. Sweeping preconditioner for the Helmholtz equation: moving perfectly matched layers. *Multiscale Model. Simul.*, 9(2):686–710, 2011.

[8] Y. A. Erlangga. Advances in iterative methods and preconditioners for the Helmholtz equation. *Arch. Comput. Methods Eng.*, 15(1):37–66, 2008.

[9] O. G. Ernst and M. J. Gander. Why it is difficult to solve Helmholtz problems with classical iterative methods. In *Numerical analysis of multiscale problems*, volume 83 of *Lect. Notes Comput. Sci. Eng.*, pages 325–363. Springer, Heidelberg, 2012.

[10] M. J. Gander and F. Nataf. AILU for Helmholtz problems: a new preconditioner based on the analytic parabolic factorization. *J. Comput. Acoust.*, 9(4):1499–1506, 2001.

[11] A. George. Nested dissection of a regular finite element mesh. *SIAM J. Numer. Anal.*, 10:345–363, 1973. Collection of articles dedicated to the memory of George E. Forsythe.

[12] S. G. Johnson. Notes on perfectly matched layers (PMLs). *Lecture notes, Massachusetts Institute of Technology, Massachusetts*, 2008.

[13] J. W. H. Liu. The multifrontal method for sparse matrix solution: theory and practice. *SIAM Rev.*, 34(1):82–109, 1992.

[14] J. Poulson, B. Engquist, S. Li, and L. Ying. A parallel sweeping preconditioner for heterogeneous 3D Helmholtz equations. *SIAM J. Sci. Comput.*, 35(3):C194–C212, 2013.

[15] C. C. Stolk. A rapidly converging domain decomposition method for the Helmholtz equation. *J. Comput. Phys.*, 241(0):240 – 252, 2013.

[16] P. Tsuji, B. Engquist, and L. Ying. A sweeping preconditioner for time-harmonic Maxwell's equations with finite elements. *J. Comput. Phys.*, 231(9):3770–3783, 2012.

[17] P. Tsuji, J. Poulson, B. Engquist, and L. Ying. Sweeping preconditioners for elastic wave propagation with spectral element methods. *ESAIM Math. Model. Numer. Anal.*, 48(2):433–447, 2014.

[18] P. Tsuji and L. Ying. A sweeping preconditioner for Yee's finite difference approximation of time-harmonic Maxwell's equations. *Front. Math. China*, 7(2):347–363, 2012.

[19] A. Vion and C. Geuzaine. Double sweep preconditioner for optimized schwarz methods applied to the Helmholtz problem. *J. Comput. Phys.*, 266(0):171 – 190, 2014.

[20] L. Zepeda-Núñez and L. Demanet. The method of polarized traces for the 2D Helmholtz equation. *ArXiv e-prints*, Oct. 2014.