SWEEPING PRECONDITIONER WITH MOVING PERFECTLY MATCHED LAYER FOR THE HELMHOLTZ EQUATION

Alex Bocchieri
abocchieri@wisc.edu
Math/CS 714 Final Project

December 19, 2021

1 Introduction

The Helmholtz equation defined as $\Delta u(x) + \frac{\omega^2}{c^2(x)} u(x) = f(x)$ is a time-independent factorization of the wave equation, where w is angular frequency, $c(x)$ is the velocity field, and $f(x)$ is the external force. This equation arises often across many fields of physics. Many different approaches exist for numerically solving the Helmholtz equation in a discretized system.

Boundary conditions must be handled properly when numerically solving a wave equation on a finite grid. The Sommerfeld condition for waves that propagate to an infinite distance is not satisfied when computing on a finite domain. This leads to non-physical reflections at the boundaries [1]. The typical correction is to impose absorbing boundary conditions. The perfectly matched layer (PML) [2] is one common method for absorbing waves at the boundaries. The PML method adds a damping region around the original domain (or within it) and accordingly modifies the equation we are solving. The PML reduces non-physical reflections and provides a more favorable system to solve for, but it also adds computational cost.

Certain iterative methods can be applied for solving the Helmholtz equation. They require preconditioning due to the Helmholtz system's indefiniteness. The methods in [3] and [4] apply a sweeping preconditioner and iteratively solve the system using GMRES [5]. Both methods incorporate a PML when building the system. Their key observation is that the system can be factorized into a form where block matrices correspond to the Helmholtz Green's function and are highly compressible. In [4], these matrices are approximated using the hierarchical matrices. In [3], these matrices are applied by solving a smaller Helmholtz system in a sub-grid with a moving PML.

In this project, I implement the sweeping preconditioner with moving PML for solving the 2D Helmholtz equation using the method in [3]. The paper first describes the LDL^T factorization of the A matrix for computing the solution in $O(N^2)$ with a *stationary* PML in Algorithms 2.1 and 2.2. Then, the paper describes the sweeping factorization of the A matrix with a *moving* PML, where smaller layers of the grid can be solved in linear time. A preconditioner can then be applied efficiently to the full Helmholtz system. This is detailed in Algorithms 2.3 and 2.4. Ultimately, applying the preconditioner and solving the system take about $O(N)$ time.

2 Methods

In general, the moving PML preconditioner is as follows. Work in the 2D domain $(x_1, x_2) \in (0, 1)^2$ discretized on an n x n grid where the number of interior grid points $N = n^2$. Assume Dirichlet zero boundary conditions at $x_2 = 1$. Approximations to the Sommerfeld boundary conditions are imposed on the other three sides (absorbing layers) using a PML. The paper provides the 5-point stencil for building the A matrix along with the PML boundary conditions. The Helmholtz system can be represented by $Au = f$.

The A matrix can be block-LDL^T factorized and inverted, providing a starting point to build M (the preconditioner of A). S_m is the mth block (of size n x n) in D that corresponds to the mth layer (row) in the grid. An operator $T_m = S_m^{-1}$ is introduced and approximated by $\tilde{T}_m$. $\tilde{T}_m$ is efficiently applied by solving the Helmholtz system on a b x

n subgrid below the $(m+1)$th layer of the grid, where b is the height of the PML in grid points. This subgrid contains the PML that has been moved up from the bottom of the grid to the mth layer. The system in this subgrid can be solved efficiently with LU factorization. Algorithm 2.3 implicitly defines $\tilde{T}_m$ for each layer, and Algorithm 2.4 describes how to apply M (including how to apply $\tilde{T}_m$) in about $O(N)$ time.

Now that M can be applied efficiently, GMRES is used to solve $MAu = Mf$. The preconditioned system converges to the solution extremely fast compared to the non-preconditioned system.

3 Experiments

I reproduced experiments in Section 3 of the paper using:

- Velocity Field 1 (c1): smooth converging lens with a Gaussian profile centered at (0.5, 0.5)
- Velocity Field 2 (c2): vertical waveguide with centralized Gaussian cross section
- External Force 1 (f1): Gaussian point source at (0.5, 0.125)
- External Force 2 (f2): Gaussian wave packet centered at (0.125, 0.125) directed to $(1/\sqrt{2}, 1/\sqrt{2})$

The same values for parameters such as $\frac{\omega}{2\pi}$, n, b, etc. as in the paper were used. $b = 12$ is the height of the PML in grid points. I tuned the "appropriate positive constant" C for the PML boundary equations for each experiment to obtain proper solutions.

Solutions and runtimes are reported in the Results section for each combination of velocity field and external force (c1-f1, c1-f2, c2-f1, c2-f2). Other experiments using c1-f1 but placing the point source in different locations were also conducted.

4 Results

The solutions and runtimes for c1-f1, c1-f1 moved, c1-f2, c2-f1, and c2-f2 are displayed in the figures below. In the runtime plots, "Solve Time" refers to how long GMRES took to converge on the solution, "Init Time" refers to the time spent building the A matrix and LU factorizations before starting GMRES, and "Total Time" is their sum. Solutions with larger n and $\frac{\omega}{2\pi}$ have very fine patterns and may require zooming in on the pdf to see them.

5 Discussion

The solutions for c1-f1, c1-f2, c2-f1, and c2-f2 match those shown in Tables 3.1 and 3.2 in the paper. Runtimes are linear with respect to N, as expected. GMRES with the preconditioner took 1-3 iterations to converge for all these experiments from $n = 127$ to 1023. GMRES without the preconditioner took about 10,000 iterations to converge for $n = 63$, and it did not reach convergence for $n = 127$. Indeed, the preconditioner is needed to obtain a solution for a reasonably large n.

My implementation does not have a PML at the top of the grid. Reflections at the top are visible, while the other three sides of the grid have no reflections. This phenomenon is particularly visible in Figure 5.

The constant value C in the PML boundary conditions has a significant effect on the computed solution. I would try many values for C, and most would result in a visually incorrect solution. When I found a C that provided a visually correct solution, the solution values were either very large (e.g. values around $1e13$), or they were close to 0. A solution close to 0 is expected, and I tried to find C values accordingly. Furthermore, visually correct solutions with very large numbers took GMRES significantly longer to solve for than solutions with small numbers. For example, when I run the solver with $N = 255^2$, Velocity Field 1, Forcing Function 2, and $C = 61$, GMRES takes about 0.75 seconds to compute a small-numbered solution. When I use $C = 62$, GMRES takes about 1.5 seconds to compute a large-numbered solution (that still looks visually correct). Similarly for $N = 1023^2$, the solve times were about 13.0 seconds and 25.6 seconds for $C = 100.6$ and $C = 100$, respectively. I also noticed examples where the same C value produced a large-numbered solution on my MacBook Pro and a small-numbered solution on my Linux desktop. This suggests the method is very sensitive to choosing an appropriate value for C.

The memory used for when $n = 1023$ was close to 15 GB on my machine with 16 GB of RAM and makes it more likely for the operating system to kill the solver when it is running (which happened in c1-f2).

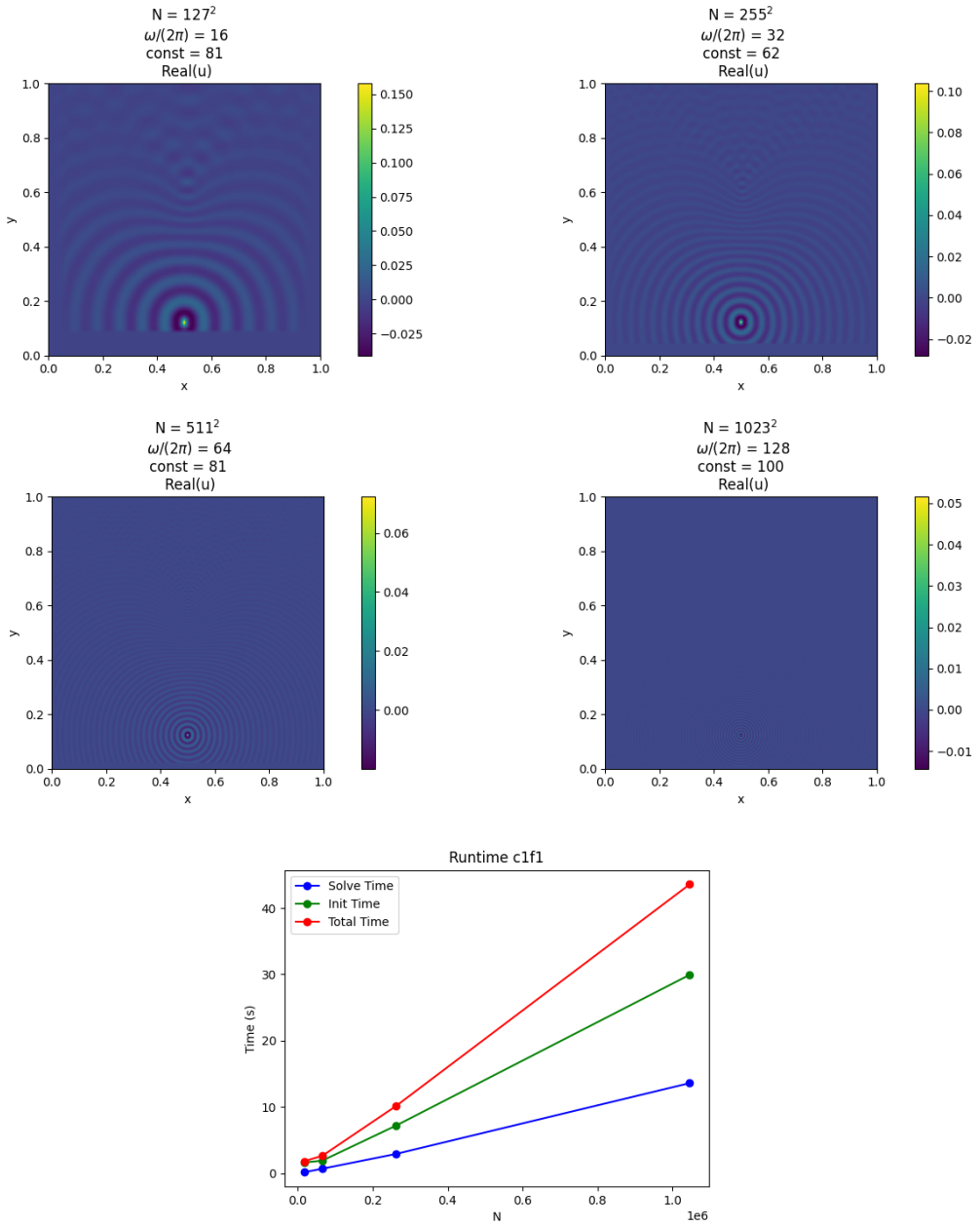


Figure 1: Solutions for Velocity Field 1 and External Force 1 with varying n and $\frac{\omega}{2\pi}$.

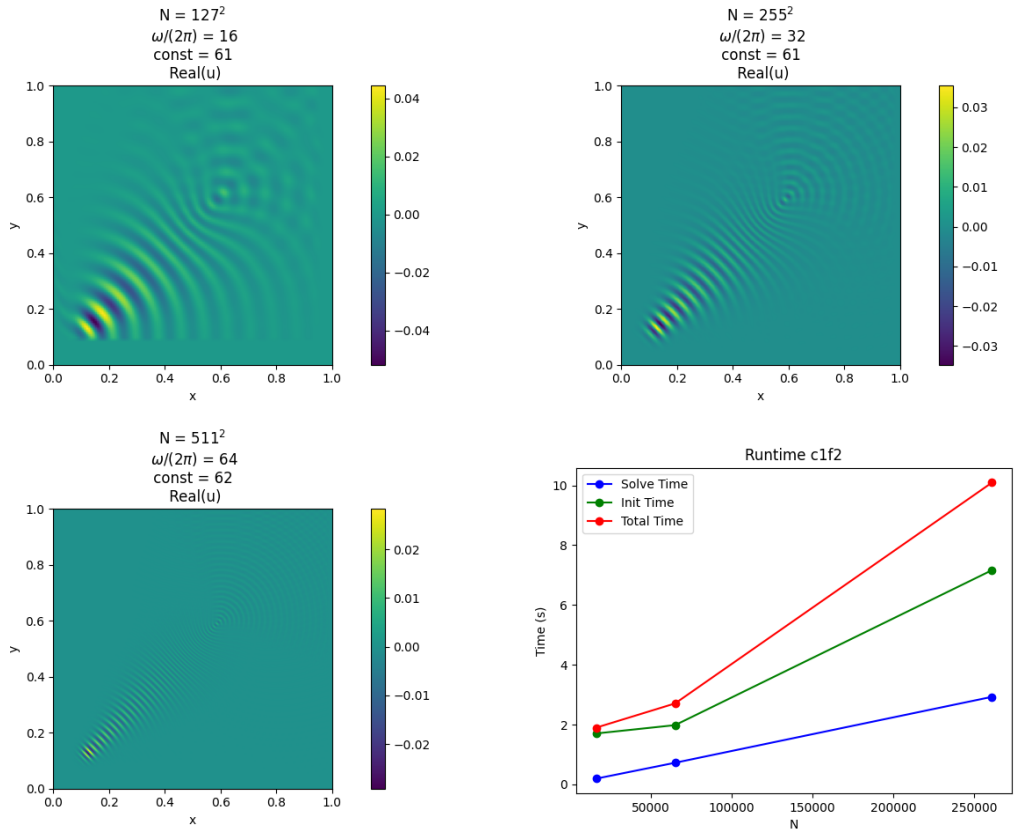


Figure 2: Solutions for Velocity Field 1 and External Force 2 with varying n and $\frac{\omega}{2\pi}$. $n = 1023$ (not plotted) gets killed on my machine for this example.

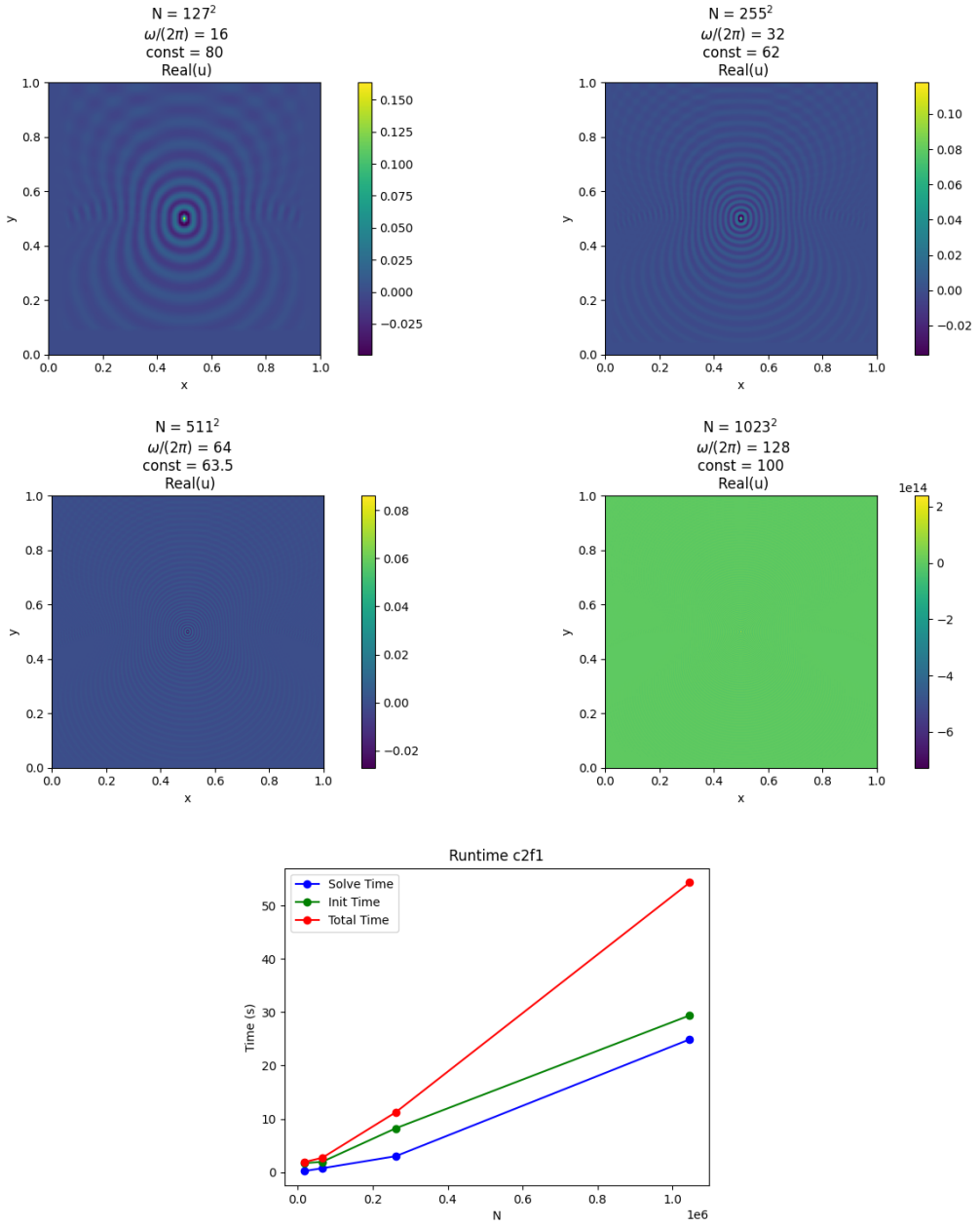


Figure 3: Solutions for Velocity Field 2 and External Force 1 with varying n and $\frac{\omega}{2\pi}$. I could not find a good C value for n=1023, so the solution is large-numbered. The pattern is still visible by zooming in on the pdf.

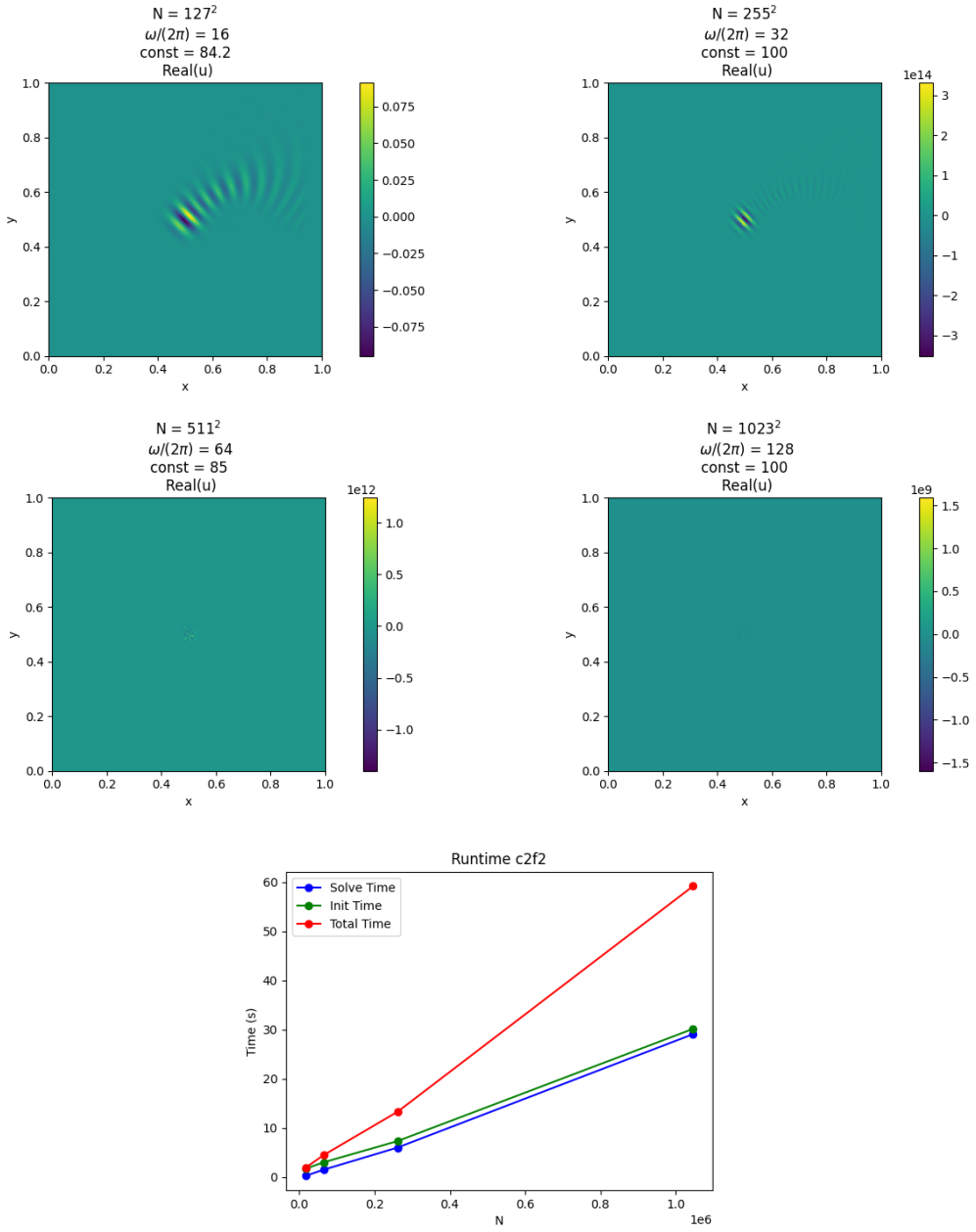
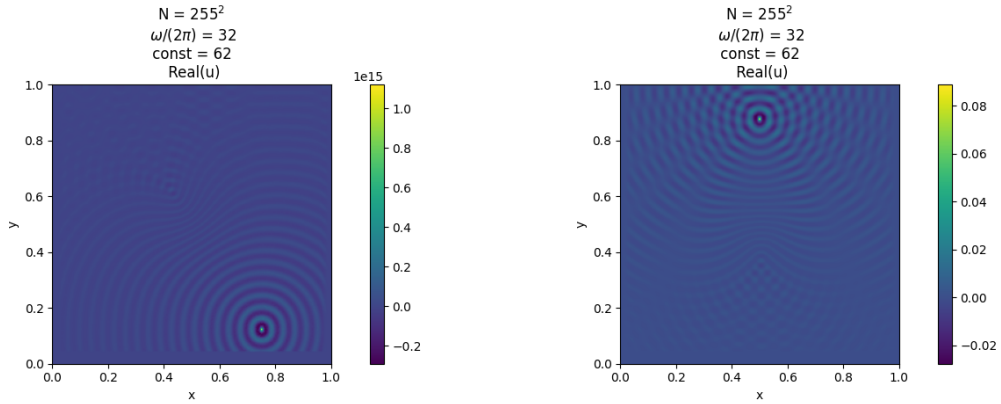


Figure 4: Solutions for Velocity Field 2 and External Force 2 with varying n and $\frac{\omega}{2\pi}$. I could not find a good C value for n=255, so the solution is large-numbered. I could not obtain a good solution for n=511, 1023.



(a) External Force: Gaussian point source at (0.75, 0.125) Velocity Field: smooth converging lens with a Gaussian profile centered at (0.5, 0.5)

(b) External Force: Gaussian point source at (0.5, 0.875) Velocity Field: smooth converging lens with a Gaussian profile centered at (0.5, 0.5)

Figure 5: Solutions for Velocity Field 1 and External Force 1 at other locations. The same c1-f1 equations are used, but the point source has been moved elsewhere in the grid.

6 Conclusion

I have implemented the sweeping preconditioner with moving PML for the 2D Helmholtz equation as described in [3]. I can reproduce the solutions shown in the paper in $O(N)$ runtime. Solving the plain Helmholtz system quickly becomes intractable as n grows. However, by using the preconditioner, large systems can be solved. Furthermore, the preconditioner is built with PMLs that minimize non-physical reflections at the grid's boundaries.

References

[1] Yogi A Erlangga. Advances in iterative methods and preconditioners for the helmholtz equation. *Archives of Computational Methods in Engineering*, 15(1):37–66, 2008.

[2] Jean-Pierre Berenger. A perfectly matched layer for the absorption of electromagnetic waves. *Journal of computational physics*, 114(2):185–200, 1994.

[3] Björn Engquist and Lexing Ying. Sweeping preconditioner for the helmholtz equation: moving perfectly matched layers. *Multiscale Modeling & Simulation*, 9(2):686–710, 2011.

[4] Björn Engquist and Lexing Ying. Sweeping preconditioner for the helmholtz equation: hierarchical matrix representation. *Communications on pure and applied mathematics*, 64(5):697–735, 2011.

[5] Youcef Saad and Martin H Schultz. Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on scientific and statistical computing*, 7(3):856–869, 1986.