ISSN 2706-655X Vol.3, Issue 5: 84-90, DOI: 10.25236/IJFET.2021.030510

A Parallel Preconditioned Solver for the 3D Helmholtz Equation

Dongsheng Cheng[a*], Jiaxi Lin[b], Kaicheng Lan[c] and Yuexin Zheng[d]

School of Software Engineering, Shenzhen Institute of Information Technology, Shenzhen, China
[a]chengds@sziit.edu.cn, [b]1468567729@qq.com, [c]2415570005@qq.com, [d]619826473@qq.com
**corresponding author*

Abstract: *Discretization of the 3D Helmholtz equation leads to a linear system of a large size. The resulting linear system is difficult to be solved by a sequential method. In this paper, we propose a parallel preconditioned Bi-CGSTAB method for solving the system based on MPI(Message Passing Interface). We precondition the 3D Helmholtz equation by the complex shifted-Laplacian preconditioner, and employ the Krylov subspace method Bi-CGSTAB combined with the multi-grid to solve the resulting system. Numerical experiments are presented to illustrate the efficiency of the parallel Preconditioned solver.*

Keywords: *Helmholtz equation, Preconditioner, Parallel, Bi-CGSTAB, Multigrid.*

1. Introduction

The Helmholtz equation governs wave propagations and scattering phenomena, and it finds important applications in many areas of science and engineering such as geophysics, aeronautics and marine technology. Most popular numerical methods for solving the Helmholtz equation are the finite difference method and the finite element method. It is known that the numerical solution of the Helmholtz equation suffers from the "pollution effect" (cf. [7]), which deteriorates severely as the wavenumber increases. The pollution effect may be reduced in some situations. However, it can never be eliminated for the 2D and 3D problems. In practice, the wavenumber k and discretization step size h should satisfy at least the "rules of thumb" that kh remains a small constant. Consequently, small step size h results in a linear system of a large size. For the 2D problem, the linear system can be solved by the direct method such as the LU decomposition. For the 3D problem, the matrix size of the resulting linear system rapidly becomes extremely large, especially when the wavenumbers is large. The cost of direct methods for solving such a linear system becomes prohibitively expensive, and we have to resort to iterative methods.

Solving the 3D Helmholtz equation requires considerable memories and computing time and this makes it not feasible to use the traditional serial method. For this reason, parallel algorithms are highly desirable for solving such problems. In [10], a parallel preconditioned iterative solver was designed, based on a parallel partition in two spacial dimensions and it was tested for small wavenumbers in a computer having 24 processors. In this paper, we develop a MPI-based parallel preconditioned iterative method for the 3D Helmholtz equation by using the Krylov subspace method Bi-CGSTAB combined with the complex shifted-Laplacian preconditioner. The preconditioned linear system is obtained from discretization of a dispersion minimizing finite difference scheme proposed in [4]. To approximately invert the preconditioner, we employ the 3D full-coarsening multigrid. The multigrid-based preconditioned Bi-CGSTAB solver can be well parallelized. The mail difficulty is the parallelization of the multigrid. We manage to solve successfully the problems arising in developing the parallel multigrid method, which include the parallel partitioning of the 3D computational domain, the information transferring between two neighboring levels as well as two neighboring sub-regions. Numerical experiments are presented to show that the parallel preconditioned iterative solver has a high parallel speedup.

2. Multigrid-based Preconditioned Iterative Solver

In this section, we review the multigrid-based preconditioned Bi-CGSTAB solver for solving the 3D Helmholtz equation for the wave problem. The Helmholtz equation is preconditioned with the

ISSN 2706-655X Vol.3, Issue 5: 84-90, DOI: 10.25236/IJFET.2021.030510

complex shifted-Laplacian preconditioner.

We consider solving the 3D Helmholtz equation

$$Au: = -\Delta u - k^2 u = g \tag{1}$$

where Δ is the 3D Laplacian, unknown u usually represents a pressure field in the frequency domain, k := 2 π f/v is the wavenumber with f and v indicating respectively the frequency and the wave velocity, g denotes the source term. In this paper, the wavenumber is referred to the dimensionless wavenumber (cf. [8]), which is equal to 2 π f ι /v with ι being the size of the square domain. To model the wave propagation in a finite domain, artificial absorbing boundary conditions such as the perfectly matched layer (PML, cf. [3, 11, 12]) are often used to eliminate non-physical reflections of the solution at the boundary. We let ξ_x, ξ_y, ξ_z denote the 1D damping functions (cf. [4]) of the x, y, z directions, respectively, and define

$$C_1 = \frac{\xi_y \xi_z}{\xi_x},\ C_2 = \frac{\xi_x \xi_z}{\xi_y},\ C_3 = \frac{\xi_x \xi_y}{\xi_z},\ C_4 = \xi_x \xi_y \xi_z \tag{2}$$

Applying the PML technique to (1), we obtain the 3D Helmholtz equation with PML

$$Au: = -\frac{\partial}{\partial x}\left(C_1 \frac{\partial u}{\partial x}\right) - \frac{\partial}{\partial y}\left(C_2 \frac{\partial u}{\partial y}\right) - \frac{\partial}{\partial z}\left(C_3 \frac{\partial u}{\partial z}\right) - C_4 k^2 u = g \tag{3}$$

which is equipped with a zero Dirichlet boundary condition. Solutions of equation (3) will have no non-physical reflections at the boundary and thus, we shall use it to replace equation (1) in our further numerical consideration.

It is known that the standard iterative method converges slowly and even diverges when solving the linear system that results from the discretization of (3). To make the iterative method efficient, we need preconditioning. Many authors (cf. [1, 2, 5, 6, 9]) developed shifted-Laplacian preconditioners for the Helmholtz equation. Shifted-Laplacian preconditioners, especially the complex shifted-Laplacian preconditioner, perform efficiently. In this paper, we employ the complex shifted-Laplacian preconditioner.

$$Mu: = -\frac{\partial}{\partial x}\left(C_1 \frac{\partial u}{\partial x}\right) - \frac{\partial}{\partial y}\left(C_2 \frac{\partial u}{\partial y}\right) - \frac{\partial}{\partial z}\left(C_3 \frac{\partial u}{\partial z}\right) - (1 - \beta i) C_4 k^2 u \tag{4}$$

where β is a positive number, and i is the imaginary unit. Using (4) to precondition the 3D Helmholtz equation (3) leads to the preconditioned equation in the operator form

$$AM^{-1} v = g, \quad \text{with} \quad v = Mu. \tag{5}$$

Discretization of (5) shall lead to the discrete preconditioned system, with the discretization of M being the preconditioner. We shall employ the Krylov subspace method Bi-CGSTAB combined with the multigrid method to solve the preconditioned linear system and call this method the multigrid-based preconditioned Bi-CGSTAB solver. We choose the Bi-CGSTAB method due to its fast convergence. The multigrid is used to invert the preconditioner approximately. Though the preconditioner does not have to be inverted exactly, it would be beneficial if a more accurate approximation can be obtained at a low cost. For this reason, the multigrid method is a good choice. In this paper, the original purpose is to develop an MPI-based parallelization of Bi-CGSTAB method to solve the discrete preconditioned system, which contains a process of inverting approximately the preconditioner M by using the matrix-based multigrid. The Bi-CGSTAB method may be described as matrix-vector multiplications and it can be well parallelized by using the standard MPI-subroutines.

3. Parallel Partition of the Computational Domain

In this section, we consider partitioning the computational domain with grids, which is the foundation of the parallel multigrid method.

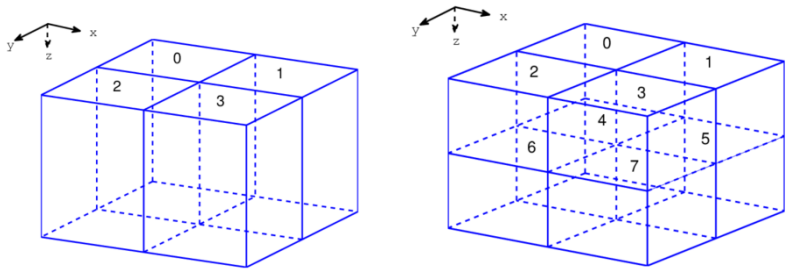
It is known that the parallel computing refers to making computation with more than one processor at the same time. Hence, the first thing we have to do is to partition the computing task, that is, to divide the task into many small sub-tasks, each of which is done by a processor. In this paper, it has an obvious geometrical background (the 3D computational domain) for the multigrid method. Thus, to partition the computing task is equivalent to partition the 3D computational domain. However, the 3D

ISSN 2706-655X Vol.3, Issue 5: 84-90, DOI: 10.25236/IJFET.2021.030510

computational domain with the grids it contains, generates all the basic data for our computation, that is, the matrix and vector data. Consequently, the partition of the 3D computational domain leads to the partition of the data, which is the specific aim for our practical parallel computations involving operations of matrices and vectors. Since it has a clear geometrical background, we here choose to partition the 3D computational domain instead of partitioning directly the data. A parallel partition will divide the computational domain into many sub-regions, each of which contains a part of the grids.

It is known that the parallel partition divides the computational domain into many sub-regions, which are disjoint with each other. It means that two adjacent sub-regions have no overlapping part, which indicates there is no overlap between the data corresponding to a sub-region and that corresponding to its neighboring sub-region. This is not beneficial to the parallel computing. Since there is no overlap, two adjacent sub-regions are not able to obtain information from each other. To deal with this problem, we shall extend the sub-regions, that is, to extend a sub-region into his neighboring sub-region so that the two adjacent sub-regions have overlap. Taking into account the multi-level grids it contains, the sub-regions shall be extended hieratically according to the level of the grids. We first extend the sub-region with the finest (original) grid, and then extend similarly the sub-regions with coarse grids of each level. It is pointed out that the extension is not arbitrary, and it should satisfies two principles. Firstly, we should guarantee that in each extended sub-region, the data information can flow freely among grids of different levels via restriction operator and prolongation operator. That is, in each extended sub-region, we should keep the multi-level grids are nested and matched level by level in the sense that each of the fine grid point has at least two coarse grid point neighbors. Secondly, a sub-region should be extended as less as possible so that the overlap between it and its neighboring sub-region has less grid points, which means a less total amount of computations. In a word, we extend the sub-region as less as possible, meanwhile, make sure that in the extended sub-region, each fine grid point has at least two coarse grid point neighbors. After extension, the extended sub-regions forms a set of coverings for the computational domains, and two adjacent coverings can communicate information freely since they have overlapping part.

We next shall describe the parallel partition of the 3D computational domain and the extension of the resulted sub-region with the original(finest) grid. We now begin with partitioning the 3D computational domain. There are two ways to partition the computational domain: the two- directional partition and the three-directional partition illustrated in Figure 1 (a) and (b), respectively.



(a) *(b)*

Figure 1: Illustration of two partitions: (a) only in x- and y-directions. (b) in all the three directions.

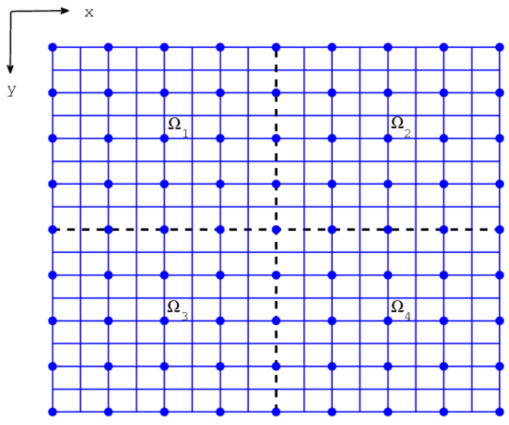


Figure 2: The partitions for two neighboring level grids: the fine and coarse grids.

ISSN 2706-655X Vol.3, Issue 5: 84-90, DOI: 10.25236/IJFET.2021.030510

After parallel partition, we next describe the concrete extension of the sub-region with the original (finest) grid. The extension of the sub-region with other level grids can be done in a fairly similar manner. We first illustrate in Figure 2 the 2D projection onto the x-y plane for the 3D domain with the original (finest) grids, which is assumed to be the 0th-level grid. To make our idea clearly, we also present the 1th-level grid, the grid point of which is plotted in blue dot. The 0th-level grid and 1th-level grid form a two-level grid. It is seen that after parallel partition, the 2D projection of the computational domain on the x-y plane is divided into four disjoint sub-regions Ω_j, j = 1, 2, 3, 4. Each sub-region contains a part of the two-level grids, the 0th-level and the 1th-level grids. For each sub-region, the black dash line represents the boundary that is generated by the parallel partition.

It is also observed that two adjacent sub-regions are disjoint, having no overlap, which disables the information exchanging between them. Hence, we need to extend the sub-regions into their adjacent sub-regions so as to keep two adjacent sub-regions having overlapping part. We next demonstrate the extension in Figure 3, which presents the extension of the sub-region with the 0th-level grid in Figure 2. As it can be seen that in Figure 3, the region $\Omega'_1, \Omega'_2,$ and Ω'_3 are extended, respectively, from the corresponding sub-regions $\Omega_1, \Omega_2,$ and Ω_3 in Figure 2, while with $\Omega'_4 = \Omega_4$. We now focus on region Ω'_1, which is obtained by extending the sub-region Ω_1 into its adjacent sub-regions Ω_2, Ω_3, Ω_4. The overlapping parts between Ω'_1 and Ω_2, Ω_3, Ω_4 are respectively the upper part, left part, central part of the shadow ares. The upper part of the red dash-dot lines is the left boundary of Ω'_1 and left part of the red dash-dot lines is the lower boundary of Ω'_1. We can see that the two-level grids contained in Ω'_1 are well nested and matched, since in Ω'_1 each fine (0th-level) grid point has at least two coarse (1th-level) grid point neighbors. In each direction, the overlapping parts between Ω'_1 and Ω_2, Ω_3, Ω_4 have three fine grid points.

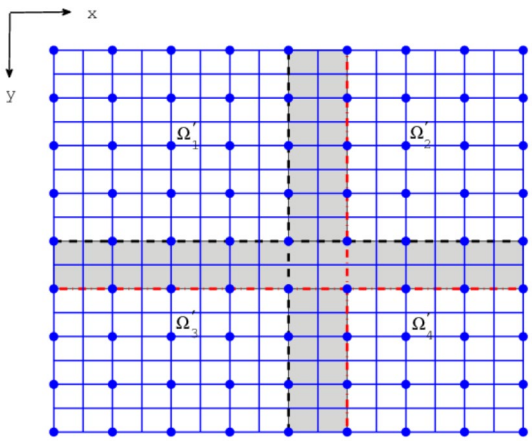


Figure 3: Extension of the coarse grids.

For region Ω'_2, it is obtained by extending Ω_2 into Ω_4 only, while it is not necessary to extend Ω_2 into Ω_1 and Ω_3. Similarly, Ω'_3 is obtained by extending Ω_3 into Ω_4 only, while it is not necessary to extend Ω_3 into Ω_1 and Ω_2. This is because both Ω'_2 and Ω'_3 already have had overlapping parts with Ω'_1(the upper part and left part of the shadow area). Furthermore, when regions Ω_2 and Ω_3 are extended into Ω_4, the resulted Ω'_2 and Ω'_3 will naturally have overlapping part, that is, the central part of the shadow area. As for Ω'_4, it is exactly the sub-region Ω_4, which does not need extension. This is because when Ω'_1, Ω'_2 and Ω'_3 are obtained, Ω_4 naturally have overlapping parts with them, which are respectively the central part, left part and lower part of the shadow area. It is easily seen that sub-regions Ω_j, j = 1, 2, 3, 4 form a set of coverings for the projection of the 3D domain on the x-y plane, and any two adjacent coverings have overlapping part, which makes them be able to communicate data information with each other. Up to this point, we have described the extension of the sub-region with the finest (0th-level) grid. Accordingly, the resulted coverings is called the coverings corresponding to the finest (0th-level). Similarly, we have the coverings corresponding to the 1th-level, 2th-level, and so on. Finally, we point out that for coverings Ω_j, they have some boundaries (plotted in black dash lines and red dash-dot lines) in the interior of the original computational domain. In order to distinguish them from the boundaries of the original computational domain (original boundaries), we call them the artificial boundaries, since they are generated by the parallel partition artificially. The data information communication between a covering and its neighboring coverings usually occurs on the corresponding artificial boundaries.

For parallel computing, we next discretize the preconditioned Helmholtz equation (5) in the

ISSN 2706-655X Vol.3, Issue 5: 84-90, DOI: 10.25236/IJFET.2021.030510

resulting sub-region after extension corresponding to the original (finest or 0th-level) grid. For instance, in Fig. 3, we discretize (5) separately in each of sub-regions Ω_j, j = 1, 2, 3, 4. We assume that there exists another virtual boundaries adjacent to the artificial boundaries, and the Dirichlet zero boundary condition is imposed on both the virtual boundaries and original boundaries. In each sub-region, we arrange the unknowns in the order of z-, y- and x-directions. To discretize (5), we here adopt the 27-point finite difference scheme based on minimizing the numerical dispersion, proposed in [4]. This scheme is simple in construction and can suppress the "pollution effect" of the large wavenumbers to a certain degree. After discretization, in each sub-region, we obtain the discrete preconditioned system

$$AM^{-1}v = g, \tag{6}$$

$$Mu = v, \tag{7}$$

where matrices A, M are discrete form of the operator A and M, respectively. M is the preconditioner of the coeffcient matrix A and it can be obtained easily. Corresponding to each sub-region, Bi-CGSTAB combined with the multigrid method is employed to solve separately the linear systems (6)-(7). The Bi-CGSTAB method is used as the outer iteration which solves (6) while the multigrid is used as the inner iteration which solves (7).

The parallel partition described above divides the computational domain into many small sub-regions, to which we assign a processor. For instance, in Fig. 1 (b), there are total 8 sub-regions, which correspond to 8 processors labeled as 0, 1, ... , 7. After extension, the 8 sub-regions become 8 coverings, and each of which has the same processor label as the original sub-region. This is the basis for us to develop parallel algorithms.

Since a sub-region corresponds to a processor with local memory, we call the above matrix and vector the local matrix and local vector. The local matrix and vector have clear geometrical structures corresponding to the grid points in the sub-region, and these structures are very important for the efficient implementation of the parallel multigrid method. We store all the local vectors separately in the form of 1D array in the local memory, which can be considered as the distributed parallel storage. Local A and local M are sparse 27-diagonal, and also stored in the local memory by using the technique of sparse compressed storage, which possesses a relative less memory.

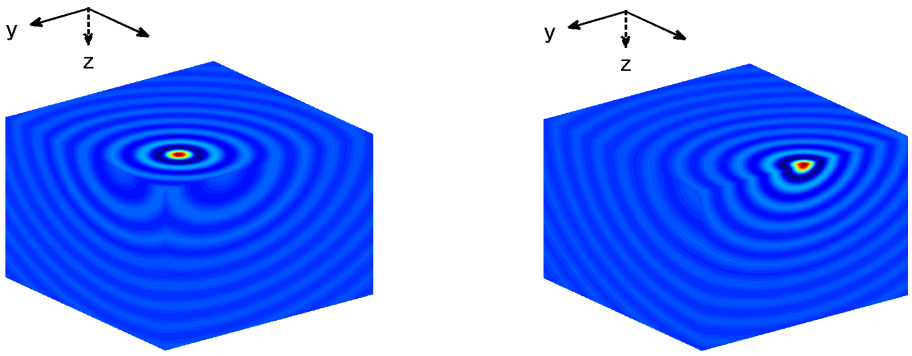
For the multigrid method, the full multigrid V-cycle (FMG) is employed, since it possesses both the efficiency of the standard V-cycle and robustness of the W-cycle. The pointwise Jacobi relaxation with underrelaxation (ω-JAC) is used as a smoother, where $0 \leq \omega \leq 1$ is a relaxation parameter. To implement the parallel multigrid method, we first partition the computational domain and extend the resulting sub-regions level by level, which is the foundation of the parallel computing. We then setup the parallel multigrid method, that is, to obtain local prolongation operators and local grid operators. Finally, we perform the parallel multigrid iteration, which involved a lot of data communication.

4. Numerical Experiments

In this section, we test the performance of the parallel multigrid-based preconditioned Bi-CGSTAB method for solving the 3D discrete Helmholtz equation (2.3) . All numerical experiments presented in this section are executed on a Dell server with 32 CPU cores and 64 threads.

In the computation, k is the wavenumber, and different values of k are tested. After k is given, and the discretization h is properly chosen. The thickness of the PML is 20, that is, the PML possesses 20 gridpoints in each direction. The 3D multigrid takes the form of FMG cycle with the full-coarsening strategy, matrix-based prolongation operator, full-weighting restriction operator, plus pointwise Jacobi relaxation with underrelaxation . The parallel preconditioned Bi-CGSTAB iteration terminates when the Euclidean norm of the relative residual error is less than 10-6.

ISSN 2706-655X Vol.3, Issue 5: 84-90, DOI: 10.25236/IJFET.2021.030510



(a) *(b)*

Figure 4: Visualization of numerical solutions for k = 60 with point sources placed at two different locations.

We first test the parallel speedup of the preconditioned iterative solver. We begin with computing a small wavenumber k=60 and visualize the numerical solutions, to verify the correctness of the parallel iterative solver. We use 4 processors, and the computing results are gathered on one processor. Figure 4 presents the numerical solutions for k=60 with point source placed at two different locations. We next solve the 3D Helmholtz equation with four large wavenumbers, that is, k=100, 150, 200, 250. The results are demonstrated in Table 1. Here, p indicates the number of processors that are used. The number of processors used ranges from 1 to 32. The number of computational grid points (unknowns) are 1833, 2753, 3653, and 4553 respectively. For k=250, the computing time for p=1 is 362.78 minutes, while the computing time for p=32 is only 17.43 minutes, that is, the time is saved by 95.2%. Denote by t(p) the execution time of the parallel preconditioned Bi-CGSTAB iterations with using p processors, we present in Tab. 2 the parallel speedup, which is given

$$S(p) := \frac{t(1)}{t(p)}, \quad p = 1, 4, 8, 16, 32. \tag{8}$$

As is observed, the parallel preconditioned Bi-CGSTAB method achieves a good speedup.

Table 1: Compute time (in minutes) of the parallel preconditioned Bi-CGSTAB method.

		p = 1	p = 4	p = 8	p = 16	p = 32
k=100	(1833)	25.96	7.08	3.81	2.06	1.12
k=150	(2753)	83.03	22.46	12.30	6.76	3.86
k=200	(3653)	193.45	53.31	28.82	15.91	9.07
k=250	(4553)	362.78	101.62	55.23	30.85	17.43

Table 2: Speedup of the parallel preconditioned Bi-CGSTAB method.

		p = 1	p = 4	p = 8	p = 16	p = 32
k=100	(1833)	1.00	3.67	6.81	12.60	23.18
k=150	(2753)	1.00	3.70	6.75	12.28	21.51
k=200	(3653)	1.00	3.62	6.71	12.16	21.32
k=250	(4553)	1.00	3.57	6.57	11.76	20.81

5. Conclusion

In this paper, we develop a parallel preconditioned iterative solver for the 3D Helmholtz equation, which is preconditioned by the complex shifted-Laplacian. Krylov subspace method Bi-CGSTAB combined with the multigrid is employed to solve the discrete preconditioned system. Based on the MPI, we realize the parallelization of the preconditioned iterative solver, and achieve a high parallel performance. We manage to solve some important problems arising in the parallelization, including the parallel partition of the computational domain, extension of the resulting sub-regions with the multi-level grids, and information transferring between two neighboring levels and two sub-regions. Finally, we test the parallel preconditioned iterative solver. Numerical results are presented to illustrate

ISSN 2706-655X Vol.3, Issue 5: 84-90, DOI: 10.25236/IJFET.2021.030510

the efficiency of the parallel preconditioned iterative solver.

Acknowledgements

This research was supported by the College Student "Climbing Plan" Program of Guangdong Province under grant pdjh2020b1198, the Natural Science Foundation of Guangdong Province under grant 2020A1515010566, and the Science and Technology Innovation Commission of Shenzhen under grants JCYJ20170306095959113.

References

[1] Maganioti, A.E., Chrissanthi, H.D., Charalabos, P.C., Andreas, R.D., George, P.N. and Christos, C.N. (2010) Cointegration of Event-Related Potential (ERP) Signals in Experiments with Different Electromagnetic Field (EMF) Conditions. Health, 2, 400-406.

[2] Bootorabi, F., Haapasalo, J., Smith, E., Haapasalo, H. and Parkkila, S. (2011) Carbonic Anhydrase VII—A Potential Prognostic Marker in Gliomas. Health, 3, 6-1

[3] Bayliss A., Goldstein C. and Turkel E.(1983) An iterative method for Helmholtz equation. Journal of Computational Physics, 49, 631-644.

[4] Bayliss A., Goldstein C. and Turkel E.(1985) The Numerical Solution of the Helmholtz Equation for Wave Propagation Problems in Underwater Acoustics. Computers Mathematics with Applications, 11, 655-665.

[5] B´erenger J. (1994) A perfectly matched layer for the absorption of electromagnetic waves. Journal of Computational Physics, 114, 185-200.

[6] Chen Z., Cheng D. and Wu T(2012). An dispersion minimizing finite difference scheme and preconditioned solver for the 3D Helmholtz equation. Journal of Computational Physics, 231, 8152-8175.

[7] Erlangga Y., Oosterlee C. and Vuik C. (2006) A novel multigrid based preconditioner for heterogeneous Helmholtz problems. SIAM Journal on Scientific Computing, 27, 1471-1492.

[8] Erlangga Y., Vuik C. and Oosterlee C. (2004) On a class of preconditioners for the Helmholtz equation.

Applied Numerical Mathematics, 50, 629-651.

[9] Ihlenburg F. and Babu˘ska I. (1995) Finite element solution of the Helmholtz equation with high wave number, Part I: The h-version of the FEM. Computers Mathematics with Applications, 30, 9-37.

[10] Ihlenburg F. and Babus˘ka I. (1995) Dispersion analysis and error estimation of galerkin finite element methods for the Helmholtz equation. International Journal for Numerical Methods in Engineer, 38, 4207-4235.

[11] Laird A. and Giles M.(2002) Preconditioned Iterative Solution of the 2D Helmholtz equation, Report 02/12, Oxford Computer Laboratory, Oxford, UK.

[12] Riyanti C., Kononov A., Erlangga Y., Vuik C., Oosterlee C., Plessix R. and Mulder W. (2007) A parallel multigrid-based preconditioner for the 3D heterogeneous high-frequency Helmholtz equation, Journal of Computational Physics, 224, 431-448.