Received: 28 May 2019 | Revised: 21 October 2019 | Accepted: 9 December 2019

DOI: 10.1002/nla.2283

RESEARCH ARTICLE

WILEY

A fast Fourier transform based direct solver for the Helmholtz problem

Jari Toivanen | **Monika Wolfmayr***

Faculty of Information Technology, University of Jyväskylä, Jyväskylä, Finland

Correspondence
Monika Wolfmayr, Faculty of Information Technology, University of Jyväskylä, P.O. Box 35 (Agora), FI-40014 Jyväskylä, Finland.
Email: monika.k.wolfmayr@jyu.fi

Funding information
Academy of Finland, 295897

Summary

This article is devoted to the efficient numerical solution of the Helmholtz equation in a two- or three-dimensional (2D or 3D) rectangular domain with an absorbing boundary condition (ABC). The Helmholtz problem is discretized by standard bilinear and trilinear finite elements on an orthogonal mesh yielding a separable system of linear equations. The main key to high performance is to employ the fast Fourier transform (FFT) within a fast direct solver to solve the large separable systems. The computational complexity of the proposed FFT-based direct solver is $\mathcal{O}(N \log N)$ operations. Numerical results for both 2D and 3D problems are presented confirming the efficiency of the method discussed.

KEYWORDS

absorbing boundary conditions, fast direct solver, finite-element discretization, Fourier transform, Helmholtz equation

MOS SUBJECT CLASSIFICATION

35J05; 42A38; 65F05; 65N22

1 | INTRODUCTION

This work presents an efficient fast direct solver employing fast Fourier transform (FFT) for the Helmholtz equation

$$-\triangle u - \omega^2 u = f, \tag{1}$$

in a rectangular domain with absorbing boundary conditions (ABCs). The method can be applied for problems with the constant zeroth-order term coefficient ω being positive, negative, or zero. Herein, we focus on large indefinite Helmholtz problems as they are a common result from acoustic scattering problems. The zeroth-order term coefficient ω denotes the wave number, which is assumed to be a constant. This condition is valid for homogeneous media. For example, the iterative domain decomposition solution techniques for acoustic scattering in layered media considered in References 1,2 leads a sequence of such problems. In this work, we derive the fast solver for the case of having a first-order ABC imposed. However, the same method can be used for the second-order ABCs employed in Reference 3. These ABCs are the time-harmonic counterpart of the ABCs for the wave equation considered in Reference 4. Moreover, the solver can be used also for problems with Robin boundary conditions, which essentially broadens its applicability. Other separable Robin boundary conditions can be treated the same way as the ABCs, which are just one possible type of boundary conditions for the Helmholtz equation being solved by the method discussed in this work. Solving the Helmholtz equation is in

general difficult or impossible to solve efficiently with most numerical methods. Difficulties related with the numerical solution of time-harmonic Helmholtz equations are discussed, for instance, in Reference 5.

The numerical method proposed in this work is a fast direct solver, which is applicable for problems posed in rectangular domains and having suitable tensor product form matrices. This kind of diagonalization technique has already been proposed in Reference 6. However, we use FFT and sparsity to implement it efficiently. The method is applicable for any discretization leading to separable 9-point stencil for two-dimensional (2D) problems and 27-point stencil for three-dimensional (3D) problems. For example, bilinear or trilinear finite-element discretizations employed in this article lead to matrices with such suitable tensor product form. In addition, the fourth-order accurate modified versions of these elements[7] are applicable with the solver. When using fourth-order accurate bilinear or trilinear finite elements, the reduced pollution error is obtained. To improve the efficiency this method employs FFT instead of cyclic reduction. Several other fast direct solvers for elliptic problems in rectangular domains are discussed, for example, in Reference 8.

The fast direct solvers can be used as efficient preconditioners in the iterative solution of problems in more general domains, see References 9,10. Other methods, which are based on an equivalent formulation of the original problem to enable preconditioning with fast direct methods, are referred to as fictitious domain, domain imbedding, or capacitance matrix methods, and they have been successfully applied also to acoustic scattering problems, for instance, in References 10–13 as well as to scattering problems with multiple right-hand sides, where a specific method for such problems is considered, for example, in Reference 14.

A different fast direct solver for the Helmholtz equation (1) has already been presented in Reference 3, where cyclic reduction techniques are used in the solution procedure. In Reference 3, the application of a specific cyclic reduction-type fast direct method is presented, called the partial solution variant of the cyclic reduction (PSCR) method, to the solution of the Helmholtz equation[15–17] and its computational efficiency is demonstrated. For 3D problems, a general fast direct solver like the PSCR has excellent parallel scalability, see Reference 16. However, the use of FFT over cyclic reduction techniques is preferable, as FFT is generally faster. That is, partly due to very fast implementations of FFT.

The reason for not applying FFT previously was that the ABCs prevent the diagonalization using FFT. The following three steps describe the basic idea to solve this problem and are discussed in more detail in this work:

1. Some of the boundary conditions are changed on the ABC part of the boundary to be of periodic type. This modified problem can be solved now with an FFT solver. This step has the computational complexity of the FFT method, which is $\mathcal{O}(N \log N)$ operations.
2. As the boundary conditions have been changed to periodic ones, the residual vector is computed due to these incorrect boundary conditions. This has nonzero components only on the boundary parts, where we have changed the boundary conditions. The correction is computed by solving a problem with the original matrix, where the right-hand side vector is the residual. Only the component of the correction, which lie on the changed boundaries are computed. For this the so-called partial solution problem a special technique exists, see References 3,16,18,19, for example. Due to sparsity of the vectors the solution requires only $\mathcal{O}(N)$ or $\mathcal{O}(N \log N)$ operations in case of two or three dimensions, respectively, where N is the total number of unknowns. After this step, the correct boundary values are known.
3. A similar problem to the one in the first step is solved, but now the right-hand side vector is adjusted so that the solution has correct values on the boundaries. From this, it follows that the solution is also the solution of the original problem. The computational complexity is again $\mathcal{O}(N \log N)$ operations as in the first step.

A similar method to the above one was already proposed in Reference 20, but with the major difference that the problem in the second step was solved iteratively. The new solver employing the FFT method is efficient in terms of computational time and memory usage, especially in the 3D case. For 3D problems, a general fast direct solver like the PSCR method[16] requires $\mathcal{O}(N(\log N)^2)$ operations, whereas the FFT-based solver reduces the complexity to $\mathcal{O}(N \log N)$ operations. The FFT method combined with the direct solver as proposed in this article leads to the same, nearly optimal complexity of $\mathcal{O}(N \log N)$ operations for both 2D and 3D problems.

The article is organized as follows: In Section 2, we present the classic and variational formulation of the Helmholtz problem as well as its discretization by bilinear and trilinear finite elements leading to a system of linear equations. The main idea of the solver for this problem and some preliminaries, which appear in the initialization process of the implemented fast solver for the 2D and 3D case, are discussed in Section 3. Sections 4 and 5 are devoted to the 2D and 3D problems, respectively. Herein, some preliminaries as well as the fast solver steps are discussed. Finally, numerical results for both 2D and 3D problems are presented in Section 6, and conclusions are drawn in Section 7.

2 | PROBLEM FORMULATION AND DISCRETIZATION

Let $\Omega \subset \mathbb{R}^d$, $d = 2, 3$, be a d-dimensional rectangular domain and $\Gamma = \partial\Omega$ its boundary. We consider the Helmholtz equation describing the linear propagation of time-harmonic acoustic waves given by

$$-\triangle u - \omega^2 u = f \qquad \text{in } \Omega, \tag{2}$$

$$\mathcal{B}u = 0 \qquad \text{on } \Gamma, \tag{3}$$

where ω denotes the wave number. Herein, Equation (3) is an approximation for the Sommerfeld radiation condition

$$\lim_{r\to\infty} r^{(d-1)/2}\,(\partial_r u - i\omega u) = 0, \tag{4}$$

appearing for the general model problem in $\mathbb{R}^d$. The approximation is performed by truncating the unbounded domain at a finite distance, which provides the boundary condition at the truncation boundary. To reduce spurious reflections caused by this artificial boundary, we use local ABCs. In general, the boundary Γ can be decomposed into parts where different boundary conditions are imposed. Let now $\Gamma = \Gamma_B \cup \Gamma_N$ be decomposed into an ABC part Γ_B and a Neumann boundary condition part Γ_N. The Neumann boundary conditions are given by

$$\mathcal{B}u = \nabla u \cdot \boldsymbol{n} = 0 \qquad \text{on } \Gamma_N, \tag{5}$$

whereas in case of ABCs we have that

$$\mathcal{B}u = \nabla u \cdot \boldsymbol{n} - i\omega u = 0 \qquad \text{on } \Gamma_B, \tag{6}$$

where $\boldsymbol{n}$ denotes the outward normal to the boundary. This type of ABCs (6) are called first-order ABCs.

Remark 1. In practical applications, so-called second-order ABCs are also important and were considered in[3] as well. However, the choice of the ABCs does not have any impact on the performance of the proposed solver. Thus, the same method can be used with given second-order ABCs, see Reference 3.

To obtain the discrete version of the Helmholtz problem (2) and (3), let us first state its weak formulation. For that, we introduce the Hilbert space $V = H^1(\Omega)$. Let the source term $f \in L_2(\Omega)$ be given. Multiplying Equation (2) with a test function as well as applying integration by parts and the boundary condition (3), yields the weak problem: Find $u \in V$ such that

$$a(u, v) = \int_\Omega fv \, d\boldsymbol{x} \qquad \forall v \in V, \tag{7}$$

where

$$a(u, v) = \int_\Omega \left(\nabla u \cdot \nabla v - \omega^2 uv\right) d\boldsymbol{x} - i\omega \int_{\partial\Omega} uv \, ds. \tag{8}$$

Let us denote the mesh points by $x_{j,l}$, $l = 1, \dots, n_j$ for every x_j-direction with $j \in \{1, \dots, d\}$ and the corresponding mesh size by $h_j = 1/(n_j - 1)$. Thus, the mesh is equidistant in each direction x_j. Moreover, we denote the full mesh size by N, which is given by $N = n_1 \times \cdots \times n_d$. Discretizing problem (7) by bilinear or trilinear finite elements on an orthogonal mesh leads to a system of linear equations given by

$$\boldsymbol{A}\boldsymbol{u} = \boldsymbol{f}, \tag{9}$$

where the matrix $\boldsymbol{A}$ has a separable tensor product form and $\boldsymbol{f} = (f_1, \dots, f_N)$ denotes the discrete right-hand side. For the 2D case, the matrix $\boldsymbol{A}$ is given by

$$\boldsymbol{A} = (\boldsymbol{K}_1 - \omega^2 \boldsymbol{M}_1) \otimes \boldsymbol{M}_2 + \boldsymbol{M}_1 \otimes \boldsymbol{K}_2, \tag{10}$$

whereas in three dimensions it is given by

$$\boldsymbol{A} = (\boldsymbol{K}_1 - \omega^2 \boldsymbol{M}_1) \otimes \boldsymbol{M}_2 \otimes \boldsymbol{M}_3 + \boldsymbol{M}_1 \otimes (\boldsymbol{K}_2 \otimes \boldsymbol{M}_3 + \boldsymbol{M}_2 \otimes \boldsymbol{K}_3). \tag{11}$$

Herein, the $n_j \times n_j$-matrices $\boldsymbol{K}_j$ and $\boldsymbol{M}_j$ are one-dimensional (1D) stiffness and mass matrices, respectively, in the x_j-direction with possible modifications on the boundaries due to the ABCs. They are computed by 1D numerical quadrature on the unit interval and are given as follows

$$\boldsymbol{K}_j = \frac{1}{h_j} \begin{pmatrix} k_{1,1} & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & k_{n_j,n_j} \end{pmatrix}, \tag{12}$$

and

$$\boldsymbol{M}_j = \frac{h_j}{6} \begin{pmatrix} 2 & 1 & & & \\ 1 & 4 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & 4 & 1 \\ & & & 1 & 2 \end{pmatrix}, \tag{13}$$

where the first and last entries are including the corresponding boundary conditions. ABCs (6) yield the entries $k_{1,1} = k_{n_j,n_j} = 1 - i\omega h_j$, whereas Neumann boundary conditions lead to $k_{1,1} = k_{n_j,n_j} = 1$. The matrix $\boldsymbol{M}_j$ is the same for both Neumann and (first-order) ABCs.

Remark 2. Without loss of generality let us assume that the ABCs are posed in direction of x_1 for both (opposite) sides.

In the following two sections, an efficient fast solver for solving the large linear systems $\boldsymbol{Au} = \boldsymbol{f}$ is proposed. We split the discussion into two sections corresponding to the 2D and 3D problems with the matrix $\boldsymbol{A}$ given by Equations (10) and (11), respectively.

3 | MAIN IDEA AND PRELIMINARIES

3.1 | Basic steps

The idea for solving the problem $\boldsymbol{Au} = \boldsymbol{f}$ is to consider an auxiliary problem

$$\boldsymbol{Bv} = \boldsymbol{f}, \tag{14}$$

where the system matrix $\boldsymbol{B}$ is derived by changing some ABCs to periodic ones. The key is that we can solve the modified (periodic) problem $\boldsymbol{Bv} = \boldsymbol{f}$ now using the FFT method, which is not possible for the original problem $\boldsymbol{Au} = \boldsymbol{f}$.

Before going into more details later, let us discuss the main steps of the solver.

Step 1: Solve problem (14) $\boldsymbol{Bv} = \boldsymbol{f}$.
Step 2: Let $\boldsymbol{w} = \boldsymbol{u} - \boldsymbol{v}$. Solve

$$\boldsymbol{Aw} = \boldsymbol{f} - \boldsymbol{Av} = \boldsymbol{Bv} - \boldsymbol{Av} = (\boldsymbol{B} - \boldsymbol{A})\boldsymbol{v}. \tag{15}$$

Step 3: Solve

$$\boldsymbol{Bu} = \boldsymbol{f} + (\boldsymbol{B} - \boldsymbol{A})(\boldsymbol{v} + \boldsymbol{w}). \tag{16}$$

Note that after Step 2, we would have already had the identity $\boldsymbol{u} = \boldsymbol{v} + \boldsymbol{w}$, but we do not use this identity directly to compute $\boldsymbol{u}$, which will be explained in Subsections 4.2 and 5.2.

3.2 | Preliminaries

We have assumed that the ABCs are given in x_1-direction on both opposite boundaries, see Remark 2. As, we have to change the boundary conditions on the two opposite boundaries to be of periodic type for the auxiliary problem (14), we consider the 1D stiffness and mass matrices $\boldsymbol{K}_1$ and $\boldsymbol{M}_1$ and change the entries corresponding to the boundary parts into periodic conditions for the auxiliary problem (14). We denote these matrices by $\boldsymbol{K}_1^B$ and $\boldsymbol{M}_1^B$. The $n_j \times n_j$ circulant matrices $\boldsymbol{K}_j^B$ and $\boldsymbol{M}_j^B$ are given as follows

$$\boldsymbol{K}_j^B = \frac{1}{h_j}\begin{pmatrix} 2 & -1 & & & -1 \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ -1 & & & -1 & 2 \end{pmatrix}, \tag{17}$$

and

$$\boldsymbol{M}_j^B = \frac{h_j}{6}\begin{pmatrix} 4 & 1 & & & 1 \\ 1 & 4 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & 4 & 1 \\ 1 & & & 1 & 4 \end{pmatrix}, \tag{18}$$

which means that we have changed the boundary conditions on two opposite boundaries to be of periodic type. Hence,

$$\begin{aligned} \boldsymbol{K}_{j,(1,2)}^B &= \boldsymbol{K}_{j,(1,n_j)}^B = \boldsymbol{K}_{j,(2,1)}^B = \boldsymbol{K}_{j,(n_j,1)}^B, \\ \boldsymbol{M}_{j,(1,2)}^B &= \boldsymbol{M}_{j,(1,n_j)}^B = \boldsymbol{M}_{j,(2,1)}^B = \boldsymbol{M}_{j,(n_j,1)}^B. \end{aligned} \tag{19}$$

Let us consider now the original problem (9) and the auxiliary problem (14) in more detail. After a suitable permutation $\boldsymbol{A}$ and $\boldsymbol{B}$ have the block forms

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{A}_{bb} & \boldsymbol{A}_{br} \\ \boldsymbol{A}_{rb} & \boldsymbol{A}_{rr} \end{pmatrix} \qquad \text{and} \qquad \boldsymbol{B} = \begin{pmatrix} \boldsymbol{B}_{bb} & \boldsymbol{A}_{br} \\ \boldsymbol{A}_{rb} & \boldsymbol{A}_{rr} \end{pmatrix}, \tag{20}$$

the subscripts b and r correspond to the nodes on the Γ_B boundary and to the rest of the nodes, respectively. We denote by $|b|$ and $|r|$ the corresponding sizes of the set of nodes on the Γ_B boundary and of the rest of the nodes such that $N = |b| + |r|$. Note that the matrix $\boldsymbol{B} - \boldsymbol{A}$ has the structure

$$\boldsymbol{B} - \boldsymbol{A} = \begin{pmatrix} \boldsymbol{B}_{bb} - \boldsymbol{A}_{bb} & 0 \\ 0 & 0 \end{pmatrix}, \tag{21}$$

and only the matrix

$$\boldsymbol{C}_{bb} = \boldsymbol{B}_{bb} - \boldsymbol{A}_{bb}, \tag{22}$$

has to be saved for the application of the fast solver. Steps of the fast solver for $\boldsymbol{Au} = \boldsymbol{f}$ include also the application of the so-called partial solution method, which is a special implementation of the method of separation of variables. This method involves the solution of the generalized eigenvalue problems

$$\boldsymbol{K}_1\boldsymbol{V}_1 = \boldsymbol{M}_1\boldsymbol{V}_1\boldsymbol{\Lambda}_1^A, \tag{23}$$

and

$$\boldsymbol{K}_1^B\boldsymbol{W}_1 = \boldsymbol{M}_1^B\boldsymbol{W}_1\boldsymbol{\Lambda}_1^B, \tag{24}$$

where the matrices $\boldsymbol{\Lambda}_1^A$ and $\boldsymbol{\Lambda}_1^B$ contain the eigenvalues as diagonal entries and the matrices $\boldsymbol{V}_1$ and $\boldsymbol{W}_1$ contain the corresponding eigenvectors as their columns. Let us define the more general problem

$$\boldsymbol{K}_j^B\boldsymbol{W}_j = \boldsymbol{M}_j^B\boldsymbol{W}_j\boldsymbol{\Lambda}_j^B, \tag{25}$$

with circulant matrices $\boldsymbol{K}_j^B$ and $\boldsymbol{M}_j^B$ for the problem size n_j corresponding to any x_j-direction. The $n_j \times n_j$ eigenvector matrix $\boldsymbol{W}_j$ for the generalized eigenvalue problem (25) is given by

$$\boldsymbol{W}_j = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 & 1 \\ 1 & e^{-\frac{2\pi i}{n_j}} & e^{-2\frac{2\pi i}{n_j}} & \dots & e^{-(n_j-2)\frac{2\pi i}{n_j}} & e^{-(n_j-1)\frac{2\pi i}{n_j}} \\ 1 & e^{-2\frac{2\pi i}{n_j}} & e^{-4\frac{2\pi i}{n_j}} & \dots & e^{-2(n_j-2)\frac{2\pi i}{n_j}} & e^{-2(n_j-1)\frac{2\pi i}{n_j}} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & e^{-(n_j-2)\frac{2\pi i}{n_j}} & e^{-(n_j-2)2\frac{2\pi i}{n_j}} & \dots & e^{-(n_j-2)^2\frac{2\pi i}{n_j}} & e^{-(n_j-1)(n_j-2)\frac{2\pi i}{n_j}} \\ 1 & e^{-(n_j-1)\frac{2\pi i}{n_j}} & e^{-2(n_j-1)\frac{2\pi i}{n_j}} & \dots & e^{-(n_j-2)(n_j-1)\frac{2\pi i}{n_j}} & e^{-(n_j-1)^2\frac{2\pi i}{n_j}} \end{pmatrix}, \tag{26}$$

and the diagonal entries of the corresponding diagonal eigenvalue matrix $\boldsymbol{\Lambda}_j^B$ are

$$\boldsymbol{\Lambda}_{j,l}^B = \frac{\boldsymbol{K}_{j,(1,1)}^B + \boldsymbol{K}_{j,(1,n_j)}^B e^{-(l-1)\frac{2\pi i}{n_j}} + \boldsymbol{K}_{j,(1,2)}^B e^{-(l-1)(n_j-1)\frac{2\pi i}{n_j}}}{\boldsymbol{M}_{j,(1,1)}^B + \boldsymbol{M}_{j,(1,n_j)}^B e^{-(l-1)\frac{2\pi i}{n_j}} + \boldsymbol{M}_{j,(1,2)}^B e^{-(l-1)(n_j-1)\frac{2\pi i}{n_j}}}, \tag{27}$$

for $l = 1, \dots, n_j$. To apply the partial solution method the eigenvectors are normalized so that they satisfy the conditions:

$$\boldsymbol{V}_1^T \boldsymbol{M}_1 \boldsymbol{V}_1 = \boldsymbol{I}_1 \qquad \text{and} \qquad \boldsymbol{V}_1^T \boldsymbol{K}_1 \boldsymbol{V}_1 = \boldsymbol{\Lambda}_1^A, \tag{28}$$

$$\boldsymbol{W}_1^T \boldsymbol{M}_1^B \boldsymbol{W}_1 = \boldsymbol{I}_1 \qquad \text{and} \qquad \boldsymbol{W}_1^T \boldsymbol{K}_1^B \boldsymbol{W}_1 = \boldsymbol{\Lambda}_1^B. \tag{29}$$

This can be achieved by multiplying $\boldsymbol{V}_1$ and $\boldsymbol{W}_1$ with the vectors $\boldsymbol{s}_1^A$ and $\boldsymbol{s}_1^B$ of length n_1, respectively. We denote the general vectors by $\boldsymbol{s}_j^A$ and $\boldsymbol{s}_j^B$ of length n_j. The entries of $\boldsymbol{s}_j^A$ are given by

$$\boldsymbol{s}_{j,l}^A = 1/\|\boldsymbol{M}_j\|_{\boldsymbol{V}_{j,l}}, \tag{30}$$

with the componentwise matrix norms

$$\|\boldsymbol{M}_j\|_{\boldsymbol{V}_{j,l}} = \left(\boldsymbol{V}_{j,l}^T \boldsymbol{M}_j \boldsymbol{V}_{j,l}\right)^{1/2}, \tag{31}$$

which are weighted by the lth eigenvectors of $\boldsymbol{V}_j$ for $l = 1, \dots, n_j$. Similarly, the entries of $\boldsymbol{s}_j^B$ are given by

$$\boldsymbol{s}_{j,l}^B = 1/\|\boldsymbol{M}_j^B\|_{\boldsymbol{W}_{j,l}}, \tag{32}$$

with the componentwise matrix norms

$$\|\boldsymbol{M}_j^B\|_{\boldsymbol{W}_{j,l}} = \left(\boldsymbol{W}_{j,l}^T \boldsymbol{M}_j^B \boldsymbol{W}_{j,l}\right)^{1/2}, \tag{33}$$

which are weighted by the lth eigenvectors of $\boldsymbol{W}_j$ for $l = 1, \dots, n_j$. In Equations (28) and (29), $\boldsymbol{I}_1$ denotes the identity matrix of size $n_1 \times n_1$. In the following, $\boldsymbol{I}_j$ and $\boldsymbol{I}_{jk}$ denote the identity matrices of sizes $n_j \times n_j$ and $(n_j \times n_k) \times (n_j \times n_k)$, respectively. The eigenvector matrices $\boldsymbol{V}_1$ and $\boldsymbol{W}_1$ are used for solving the partial solution problems when needed in the steps of the solver. However, the generalized eigenvalue problems (23) and (24) have to be solved only once during the solution process—in the initialization. The conditions (28) and (29) also lead to a convenient representation for the inverses of the system matrices $\boldsymbol{A}$ and $\boldsymbol{B}$, which is discussed in Subsections 4.1 and 5.1 for the respective 2D and 3D case. In the following two sections, we discuss the 2D and 3D problems in more detail separately, as the efficient implementation of the initialization process and the steps of the fast solver differs in both cases.

4 | THE 2D CASE

4.1 | Preliminaries for the 2D problem

4.1.1 | Reformulation of problem matrices A and B

The matrix $\boldsymbol{B}$ for the auxiliary problem (14) in the 2D case is given by

$$\boldsymbol{B} = (\boldsymbol{K}_1^B - \omega^2 \boldsymbol{M}_1^B) \otimes \boldsymbol{M}_2 + \boldsymbol{M}_1^B \otimes \boldsymbol{K}_2. \tag{34}$$

Using the conditions (29) as follows

$$\begin{aligned} \boldsymbol{B} &= (\boldsymbol{W}_1^{-T} \boldsymbol{\Lambda}_1^B \boldsymbol{W}_1^{-1} - \omega^2 \boldsymbol{W}_1^{-T} \boldsymbol{I}_1 \boldsymbol{W}_1^{-1}) \otimes \boldsymbol{M}_2 + (\boldsymbol{W}_1^{-T} \boldsymbol{I}_1 \boldsymbol{W}_1^{-1}) \otimes \boldsymbol{K}_2 \\ &= (\boldsymbol{W}_1^{-T} (\boldsymbol{\Lambda}_1^B - \omega^2 \boldsymbol{I}_1) \boldsymbol{W}_1^{-1}) \otimes \boldsymbol{M}_2 + (\boldsymbol{W}_1^{-T} \boldsymbol{I}_1 \boldsymbol{W}_1^{-1}) \otimes \boldsymbol{K}_2 \\ &= (\boldsymbol{W}_1^{-T} \otimes \boldsymbol{I}_2)((\boldsymbol{\Lambda}_1^B - \omega^2 \boldsymbol{I}_1) \otimes \boldsymbol{M}_2 + \boldsymbol{I}_1 \otimes \boldsymbol{K}_2)(\boldsymbol{W}_1^{-1} \otimes \boldsymbol{I}_2), \end{aligned} \tag{35}$$

the inverse of $\boldsymbol{B}$ can be represented by

$$\boldsymbol{B}^{-1} = (\boldsymbol{W}_1 \otimes \boldsymbol{I}_2)\, \boldsymbol{H}_B^{-1} (\boldsymbol{W}_1^T \otimes \boldsymbol{I}_2), \tag{36}$$

where

$$\boldsymbol{H}_B = (\boldsymbol{\Lambda}_1^B - \omega^2 \boldsymbol{I}_1) \otimes \boldsymbol{M}_2 + \boldsymbol{I}_1 \otimes \boldsymbol{K}_2. \tag{37}$$

Similarly using the conditions (28), we obtain for the system matrix $\boldsymbol{A}$ the following representation:

$$\boldsymbol{A}^{-1} = (\boldsymbol{V}_1 \otimes \boldsymbol{I}_2)\, \boldsymbol{H}_A^{-1} (\boldsymbol{V}_1^T \otimes \boldsymbol{I}_2), \tag{38}$$

where

$$\boldsymbol{H}_A = (\boldsymbol{\Lambda}_1^A - \omega^2 \boldsymbol{I}_1) \otimes \boldsymbol{M}_2 + \boldsymbol{I}_1 \otimes \boldsymbol{K}_2. \tag{39}$$

4.1.2 | Lower–upper (LU) decomposition

Linear systems with the block diagonal matrices $\boldsymbol{H}_A$ and $\boldsymbol{H}_B$ are solved with a direct method. The matrices $\boldsymbol{H}_A$ and $\boldsymbol{H}_B$ consist of N diagonal blocks each of size N. In the following, let us discuss the method applied on the matrix $\boldsymbol{H}_B$ (given by Equation (37)), as it applies analogously for $\boldsymbol{H}_A$. First, the lower–upper (LU) decomposition of $\boldsymbol{H}_B$ is computed as follows

$$\boldsymbol{H}_B = \boldsymbol{L}_B \boldsymbol{U}_B. \tag{40}$$

Then the linear system

$$\boldsymbol{H}_B \boldsymbol{y} = \boldsymbol{L}_B \boldsymbol{U}_B \boldsymbol{y} = \boldsymbol{r}, \tag{41}$$

is solved by solving the respective two subproblems

$$\boldsymbol{L}_B \boldsymbol{z} = \boldsymbol{r} \qquad \text{and} \qquad \boldsymbol{U}_B \boldsymbol{y} = \boldsymbol{z}, \tag{42}$$

consecutively in the application of the fast solver. Note that $\boldsymbol{r}$ denotes now some right-hand side, which is in the different steps of the solver also different. However, we will describe that in more detail in the following subsection. We denote by

$$\boldsymbol{H}_A = \boldsymbol{L}_A \boldsymbol{U}_A, \tag{43}$$

the LU decomposition corresponding to $\boldsymbol{H}_A$. The structure of $\boldsymbol{H}_B$ and $\boldsymbol{H}_A$ is essential for the fast application of the direct solver. The diagonal blocks of these matrices are tridiagonal, which makes the LU decomposition fast for them. The computational complexity is optimal $\mathcal{O}(N)$.

4.2 | Fast solver in the 2D case

4.2.1 | Step 1

The auxiliary problem

$$\boldsymbol{B}\boldsymbol{v} = \boldsymbol{B}\begin{pmatrix}\boldsymbol{v}_b\\ \boldsymbol{v}_r\end{pmatrix} = \boldsymbol{f}, \tag{44}$$

is solved, but only $\boldsymbol{v}_b$ and not $\boldsymbol{v}_r$ is computed. For that, the FFT $\hat{\boldsymbol{f}}$ of the right-hand side $\boldsymbol{f}$ is computed first, where its coefficients $\hat{f}_k$ are given as follows

$$\hat{f}_k = \sum_{l=1}^{N} e^{-2\pi i \frac{(l-1)(k-1)}{N}} f_l, \tag{45}$$

for all $k = 1, \dots, N$ and normalization might be needed, which means that $\hat{\boldsymbol{f}}$ is multiplied by the vector $\boldsymbol{s}_1^B$, where its components are defined in (32) with $j = 1$. Note that computing the FFT corresponds to the multiplication $\hat{\boldsymbol{f}} = (\boldsymbol{W}_1^T \otimes \boldsymbol{I}_2)\,\boldsymbol{f}$. The vector $\hat{\boldsymbol{f}}$ is saved, as it will be needed in Step 3 as well. Next, we apply the LU decomposition (41) with the right-hand side $\hat{\boldsymbol{f}}$

$$\boldsymbol{L}_B\boldsymbol{z}_1 = \hat{\boldsymbol{f}} \qquad \text{and} \qquad \boldsymbol{U}_B\tilde{\boldsymbol{z}}_1 = \boldsymbol{z}_1. \tag{46}$$

Performing the inverse FFT on the resulting vector $\tilde{\boldsymbol{z}}_1$ would provide both $\boldsymbol{v}_b$ and $\boldsymbol{v}_r$, which would correspond to the multiplication $\boldsymbol{v} = (\boldsymbol{W}_1 \otimes \boldsymbol{I}_2)\,\tilde{\boldsymbol{z}}_1$. However, as we only need $\boldsymbol{v}_b$, instead of that, we multiply the vector $\tilde{\boldsymbol{z}}_1$ by the matrix $\boldsymbol{W}_1$ by taking advantage of the sparsity of the desired components $\boldsymbol{v}_b$. More precisely, we multiply $\tilde{\boldsymbol{z}}_1$ from the left side by the eigenvectors of $\boldsymbol{W}_1$, which correspond only to the boundary Γ_B denoted by the matrix $\boldsymbol{W}_1^b$ of size $|b| \times n_1$ leading to

$$\boldsymbol{v}_b = (\boldsymbol{W}_1^b \otimes \boldsymbol{I}_2)\,\tilde{\boldsymbol{z}}_1, \tag{47}$$

which resembles the representation (36) for $\boldsymbol{B}^{-1}$. The computational complexity of Step 1 is $\mathcal{O}(N \log N)$.

4.2.2 | Step 2

We introduce the additional vector $\boldsymbol{w}$ defined as $\boldsymbol{w} = \boldsymbol{u} - \boldsymbol{v}$. As

$$\boldsymbol{A}\boldsymbol{w} = \boldsymbol{A}\boldsymbol{u} - \boldsymbol{A}\boldsymbol{v} = \boldsymbol{f} - \boldsymbol{A}\boldsymbol{v} = \boldsymbol{B}\boldsymbol{v} - \boldsymbol{A}\boldsymbol{v}, \tag{48}$$

and (21), we obtain the following problem:

$$\boldsymbol{A}\boldsymbol{w} = \boldsymbol{A}\begin{pmatrix}\boldsymbol{w}_b\\ \boldsymbol{w}_r\end{pmatrix} = (\boldsymbol{B} - \boldsymbol{A})\boldsymbol{v} = \begin{pmatrix}\boldsymbol{C}_{bb}\boldsymbol{v}_b\\ 0\end{pmatrix}, \tag{49}$$

where $\boldsymbol{C}_{bb}$ is defined by Equation (22). We solve the problem (49) using the representation (38), but compute again only $\boldsymbol{w}_b$ and not $\boldsymbol{w}_r$, as the right-hand side as well as the desired components of the solution are both sparse. First, we compute

$$\boldsymbol{g}_b = (\boldsymbol{V}_1^{bT} \otimes \boldsymbol{I}_2)\,\boldsymbol{C}_{bb}\boldsymbol{v}_b, \tag{50}$$

where the matrix $\boldsymbol{V}_1^b$ of size $|b| \times n_1$ denotes the eigenvectors of $\boldsymbol{V}_1$, which correspond only to the boundary Γ_B. Then using the LU decomposition (43), we solve

$$\boldsymbol{L}_A \boldsymbol{z}_2 = \boldsymbol{g}_b \qquad \text{and} \qquad \boldsymbol{U}_A \tilde{\boldsymbol{z}}_2 = \boldsymbol{z}_2, \tag{51}$$

and, finally, we solve

$$\boldsymbol{w}_b = (\boldsymbol{V}_1^b \otimes \boldsymbol{I}_2)\, \tilde{\boldsymbol{z}}_2, \tag{52}$$

which resembles the representation (38) for $\boldsymbol{A}^{-1}$, but corresponds only to the boundary Γ_B. The computational complexity of Step 2 is of optimal order $\mathcal{O}(N)$.

4.2.3 | Step 3

Finally, to obtain the solution $\boldsymbol{u}$ of the original problem (9), we solve now the problem

$$\boldsymbol{B}\boldsymbol{u} = \boldsymbol{f} + (\boldsymbol{B} - \boldsymbol{A})(\boldsymbol{v} + \boldsymbol{w}) = \boldsymbol{f} + \begin{pmatrix} \boldsymbol{C}_{bb}(\boldsymbol{v}_b + \boldsymbol{w}_b) \\ 0 \end{pmatrix}, \tag{53}$$

due to $\boldsymbol{B}\boldsymbol{u} = \boldsymbol{A}\boldsymbol{u} + \boldsymbol{B}\boldsymbol{u} - \boldsymbol{A}\boldsymbol{u}$. As, we have already computed the Fourier transformation $\hat{\boldsymbol{f}}$ of $\boldsymbol{f}$ in Step 1 by Equation (45), we only need to compute the Fourier transformation of the second term of the right-hand side of Equation (53) and due to sparsity again only the part corresponding to the boundary Γ_B as follows

$$\boldsymbol{h}_b = (\boldsymbol{W}_1^{bT} \otimes \boldsymbol{I}_2)\, \boldsymbol{C}_{bb}(\boldsymbol{v}_b + \boldsymbol{w}_b), \tag{54}$$

leading to the Fourier transformation of the entire right-hand side of Equation (53) denoted by $\hat{\boldsymbol{f}} + \hat{\boldsymbol{h}}$. Next, we apply the LU decomposition (41) with this right-hand side yielding

$$\boldsymbol{L}_B \boldsymbol{z}_3 = \hat{\boldsymbol{f}} + \hat{\boldsymbol{h}} \qquad \text{and} \qquad \boldsymbol{U}_B \tilde{\boldsymbol{z}}_3 = \boldsymbol{z}_3. \tag{55}$$

In the last step all resulting components are needed (not only the ones corresponding to Γ_B) to obtain the solution $\boldsymbol{u}$ by applying the inverse Fourier transformation on $\tilde{\boldsymbol{z}}_3$, where its coefficients $\hat{u}_k$ are given as follows

$$\hat{u}_k = \frac{1}{N} \sum_{l=1}^{N} e^{2\pi i \frac{(l-1)(k-1)}{N}} \tilde{z}_{3,l}, \tag{56}$$

for all $k = 1, \dots, N$. Again the vector $\tilde{\boldsymbol{z}}_3$ may need to be multiplied by the vector $\boldsymbol{s}_1^B$ with entries defined in Equation (32) for n_1 before applying the inverse Fourier transformation on it. The computation of the inverse Fourier transformation corresponds to the multiplication

$$\boldsymbol{u} = (\boldsymbol{W}_1 \otimes \boldsymbol{I}_2)\, \tilde{\boldsymbol{z}}_3. \tag{57}$$

The computational complexity of Step 3 is $\mathcal{O}(N \log N)$.

Conclusion 1. Combing the results on the computational complexities of all three steps finally leads to the overall computational complexity of $\mathcal{O}(N \log N)$ operations for the 2D problem.

5 | THE 3D CASE

5.1 | Preliminaries for the 3D problem

5.1.1 | Matrices A and B

The matrix $\boldsymbol{B}$ in the three-dimensional case is given by

$$\boldsymbol{B} = (\boldsymbol{K}_1^B - \omega^2 \boldsymbol{M}_1^B) \otimes \boldsymbol{M}_2 \otimes \boldsymbol{M}_3 + \boldsymbol{M}_1^B \otimes (\boldsymbol{K}_2 \otimes \boldsymbol{M}_3 + \boldsymbol{M}_2 \otimes \boldsymbol{K}_3). \tag{58}$$

Using the Equations (28) and (29), the inverses of $\boldsymbol{A}$ and $\boldsymbol{B}$ can be represented analogously as in the 2D case as follows

$$\boldsymbol{A}^{-1} = (\boldsymbol{V}_1 \otimes \boldsymbol{I}_{23})\, \boldsymbol{H}_A^{-1} (\boldsymbol{V}_1^T \otimes \boldsymbol{I}_{23}), \tag{59}$$

and

$$\boldsymbol{B}^{-1} = (\boldsymbol{W}_1 \otimes \boldsymbol{I}_{23})\, \boldsymbol{H}_B^{-1} (\boldsymbol{W}_1^T \otimes \boldsymbol{I}_{23}), \tag{60}$$

where

$$\boldsymbol{H}_A = ((\boldsymbol{\Lambda}_1^A - \omega^2 \boldsymbol{I}_1) \otimes \boldsymbol{M}_2 + \boldsymbol{I}_1 \otimes \boldsymbol{K}_2) \otimes \boldsymbol{M}_3 + \boldsymbol{I}_1 \otimes \boldsymbol{M}_2 \otimes \boldsymbol{K}_3, \tag{61}$$

and

$$\boldsymbol{H}_B = ((\boldsymbol{\Lambda}_1^B - \omega^2 \boldsymbol{I}_1) \otimes \boldsymbol{M}_2 + \boldsymbol{I}_1 \otimes \boldsymbol{K}_2) \otimes \boldsymbol{M}_3 + \boldsymbol{I}_1 \otimes \boldsymbol{M}_2 \otimes \boldsymbol{K}_3, \tag{62}$$

respectively.

5.1.2 | Applying the 2D fast solver

Linear systems with the block diagonal matrices $\boldsymbol{H}_B$ and $\boldsymbol{H}_A$ are again solved with a direct method. However, it is performed in a different way than for the 2D case, as the LU decomposition is slow for large block tridiagonal problems. More precisely, the efficient implementation in three dimensions contains the application of the 2D fast solver for n_1 subproblems of size $n_2 \times n_3$ including the computation of the partial solution method in x_2-direction. For that, one needs to solve the following generalized eigenvalue problems during the initialization process:

$$\boldsymbol{K}_2 \boldsymbol{V}_2 = \boldsymbol{M}_2 \boldsymbol{V}_2 \boldsymbol{\Lambda}_2^A \qquad \text{and} \qquad \boldsymbol{K}_2^B \boldsymbol{W}_2 = \boldsymbol{M}_2^B \boldsymbol{W}_2 \boldsymbol{\Lambda}_2^B, \tag{63}$$

with the matrices $\boldsymbol{K}_2^B$ and $\boldsymbol{M}_2^B$ as defined in Equations (17) and (18), but for the x_2-direction now. Moreover, the diagonal eigenvalue matrices $\boldsymbol{\Lambda}_2^A$ and $\boldsymbol{\Lambda}_2^B$ and the eigenvector matrices $\boldsymbol{V}_2$ and $\boldsymbol{W}_2$ are formed analogously as for the x_1-direction only the problem size changes to n_2. For that, the generalized eigenvalue problem with circulant matrices for a general problem size n_j was defined in Equation (25) and the corresponding eigenvector matrix is represented in Equation (26). The eigenvectors are normalized to satisfy the conditions

$$\boldsymbol{V}_2^T \boldsymbol{M}_2 \boldsymbol{V}_2 = \boldsymbol{I}_2 \qquad \text{and} \qquad \boldsymbol{V}_2^T \boldsymbol{K}_2 \boldsymbol{V}_2 = \boldsymbol{\Lambda}_2^A, \tag{64}$$

$$\boldsymbol{W}_2^T \boldsymbol{M}_2^B \boldsymbol{W}_2 = \boldsymbol{I}_2 \qquad \text{and} \qquad \boldsymbol{W}_2^T \boldsymbol{K}_2^B \boldsymbol{W}_2 = \boldsymbol{\Lambda}_2^B. \tag{65}$$

This can be achieved by multiplying $\boldsymbol{V}_2$ and $\boldsymbol{W}_2$ with the vectors $\boldsymbol{s}_2^A$ and $\boldsymbol{s}_2^B$ of length n_2, respectively, which are introduced in Equations (30)–(33). The fast solver includes the application of the partial solution method n_1 times for the following 2D $n_2 \times n_3$ subproblems in x_2-direction corresponding to the matrices (61) and (62):

$$\boldsymbol{A}_{A,l} = (\boldsymbol{K}_2 - \underbrace{(\omega^2 - \boldsymbol{\Lambda}_{1,l}^A)}_{=: p_A} \boldsymbol{M}_2) \otimes \boldsymbol{M}_3 + \boldsymbol{M}_2 \otimes \boldsymbol{K}_3, \tag{66}$$

and

$$\boldsymbol{A}_{B,l} = (\boldsymbol{K}_2 - \underbrace{(\omega^2 - \boldsymbol{\Lambda}_{1,l}^B)}_{=: p_B} \boldsymbol{M}_2) \otimes \boldsymbol{M}_3 + \boldsymbol{M}_2 \otimes \boldsymbol{K}_3, \tag{67}$$

where $l = 1, \dots, n_1$. This yields the direct solution for $\boldsymbol{H}_A^{-1}$ and $\boldsymbol{H}_B^{-1}$. Note that the subproblems (66) and (67) reflect the structure of Equations (61) and (62), respectively, in the framework of the 2D problem (10). Now the parameters

p_A and p_B have been introduced in Equations (66) and (67), respectively. The corresponding auxiliary problems are given by

$$\boldsymbol{B}_{A,l} = (\boldsymbol{K}_2^B - p_A \boldsymbol{M}_2^B) \otimes \boldsymbol{M}_3 + \boldsymbol{M}_2^B \otimes \boldsymbol{K}_3, \tag{68}$$

and

$$\boldsymbol{B}_{B,l} = (\boldsymbol{K}_2^B - p_B \boldsymbol{M}_2^B) \otimes \boldsymbol{M}_3 + \boldsymbol{M}_2^B \otimes \boldsymbol{K}_3, \tag{69}$$

which now reflect the structure of Equations (61) and (62), respectively, in the framework of the 2D problem (34). As defined in Equation (27), $\boldsymbol{\Lambda}_{1,l}^B$ is the lth-diagonal entry of the diagonal eigenvalue matrix $\boldsymbol{\Lambda}_1^B$. Analogously, we have denoted by $\boldsymbol{\Lambda}_{1,l}^A$ the lth-diagonal entry of the diagonal eigenvalue matrix $\boldsymbol{\Lambda}_1^A$.

In summary, the 2D problems (66) and (67) are solved for all $l = 1, \dots, n_1$ by applying n_1 times the 2D FFT-based fast solver from Subsection 4.2 using the auxiliary 2D problems (68) and (69). The computational complexity of this step is now $\mathcal{O}(N \log N)$, as the FFT method has to be applied now in this step as well.

5.2 | Fast solver in the 3D case

The 3D solver has the same three main steps, but now instead of applying LU decomposition we apply the 2D solver as described in the previous subsection. We present all the main steps of the solver similarly as for the 2D case in Subsection 4.2, but in less detail. Hence, we refer the reader also to Subsection 4.2 for more details.

5.2.1 | Step 1

The auxiliary problem (44)

$$\boldsymbol{B}\boldsymbol{v} = \boldsymbol{B} \begin{pmatrix} \boldsymbol{v}_b \\ \boldsymbol{v}_r \end{pmatrix} = \boldsymbol{f},$$

is solved, but only $\boldsymbol{v}_b$ and not $\boldsymbol{v}_r$ is computed. For that, we need to compute $\hat{\boldsymbol{f}} = (\boldsymbol{W}_1^T \otimes \boldsymbol{I}_{23})\, \boldsymbol{f}$ first, which is equivalent to the computation of the FFT $\hat{\boldsymbol{f}}$ for the right-hand side $\boldsymbol{f}$, where its coefficients $\hat{f}_k$ are given in Equation (45) for all $k = 1, \dots, N$. The vector $\hat{\boldsymbol{f}}$ can be normalized by multiplying it with the vector $\boldsymbol{s}_1^B$ of length n_1 with components defined in Equation (32). The vector $\hat{\boldsymbol{f}}$ is saved, as it will be needed in Step 3 as well. Next, we apply n_1 times the 2D solver for the $n_2 \times n_3$ problems (67) with the auxiliary problems (69) leading to the solution of the problem

$$\boldsymbol{H}_B \tilde{\boldsymbol{z}}_1 = \hat{\boldsymbol{f}}. \tag{70}$$

As, we only need $\boldsymbol{v}_b$, we now multiply the vector $\tilde{\boldsymbol{z}}_1$ by the matrix $\boldsymbol{W}_1$ by taking advantage of the sparsity of the desired components $\boldsymbol{v}_b$. More precisely, we multiply $\tilde{\boldsymbol{z}}_1$ from the left side by the components of the eigenvectors of $\boldsymbol{W}_1$, which correspond only to the boundary Γ_B denoted by $\boldsymbol{W}_1^b$ leading to

$$\boldsymbol{v}_b = (\boldsymbol{W}_1^b \otimes \boldsymbol{I}_{23})\, \tilde{\boldsymbol{z}}_1, \tag{71}$$

which resembles the representation (60) for $\boldsymbol{B}^{-1}$.

5.2.2 | Step 2

We introduce the additional vector $\boldsymbol{w}$ defined as $\boldsymbol{w} = \boldsymbol{u} - \boldsymbol{v}$. As Equations (48) and (21), we derive at problem (49)

$$\boldsymbol{A}\boldsymbol{w} = \boldsymbol{A} \begin{pmatrix} \boldsymbol{w}_b \\ \boldsymbol{w}_r \end{pmatrix} = (\boldsymbol{B} - \boldsymbol{A})\boldsymbol{v} = \begin{pmatrix} \boldsymbol{C}_{bb}\boldsymbol{v}_b \\ 0 \end{pmatrix}.$$

We solve this problem using the representation (59), but compute again only $\boldsymbol{w}_b$ and not $\boldsymbol{w}_r$, as the right-hand side as well as the desired components of the solution are both sparse. First, we solve

$$\boldsymbol{g}_b = (\boldsymbol{V}_1^{bT} \otimes \boldsymbol{I}_{23})\ \boldsymbol{C}_{bb}\boldsymbol{v}_b, \tag{72}$$

then by applying n_1 times the 2D solver for the $n_2 \times n_3$ subproblems (66) with the auxiliary problems (68) leading to the solution of the problem

$$\boldsymbol{H}_A\tilde{\boldsymbol{z}}_2 = \boldsymbol{g}_b. \tag{73}$$

Finally, we compute

$$\boldsymbol{w}_b = (\boldsymbol{V}_1^{b} \otimes \boldsymbol{I}_{23})\ \tilde{\boldsymbol{z}}_2, \tag{74}$$

which resembles the representation (59) for $\boldsymbol{A}^{-1}$, but corresponds only to the boundary Γ_B.

5.2.3 | Step 3

The last step yields the solution $\boldsymbol{u}$ of the original problem (9). First, we solve the problem (53)

$$\boldsymbol{B}\boldsymbol{u} = \boldsymbol{f} + (\boldsymbol{B}-\boldsymbol{A})(\boldsymbol{v}+\boldsymbol{w}) = \boldsymbol{f} + \begin{pmatrix} \boldsymbol{C}_{bb}(\boldsymbol{v}_b + \boldsymbol{w}_b) \\ 0 \end{pmatrix},$$

due to $\boldsymbol{B}\boldsymbol{u} = \boldsymbol{A}\boldsymbol{u} + \boldsymbol{B}\boldsymbol{u} - \boldsymbol{A}\boldsymbol{u}$. As, we have already computed the Fourier transformation $\hat{\boldsymbol{f}}$ of $\boldsymbol{f}$ in Step 1 by Equation (45), we only need to compute the Fourier transformation of the second term of the right-hand side of Equation (53) and due to sparsity again only the part corresponding to the boundary Γ_B as follows

$$\boldsymbol{h}_b = (\boldsymbol{W}_1^{bT} \otimes \boldsymbol{I}_{23})\ \boldsymbol{C}_{bb}(\boldsymbol{v}_b + \boldsymbol{w}_b), \tag{75}$$

leading to the Fourier transformation of the entire right-hand side of Equation (53) denoted by $\hat{\boldsymbol{f}} + \hat{\boldsymbol{h}}$. Next, we apply n_1 times the 2D solver for the $n_2 \times n_3$ problems (67) with the auxiliary problems (69) leading to the solution of the problem

$$\boldsymbol{H}_B\tilde{\boldsymbol{z}}_3 = \hat{\boldsymbol{f}} + \hat{\boldsymbol{h}}. \tag{76}$$

In the last step all resulting components are needed (not only the ones corresponding to Γ_B) to obtain the solution $\boldsymbol{u}$ by applying the inverse Fourier transformation on $\tilde{\boldsymbol{z}}_3$, where its coefficients $\hat{u}_k$ are defined as in Equation (56) for all $k = 1, \dots, N$. The vector $\tilde{\boldsymbol{z}}_3$ may be normalized by multiplying it with the vector $\boldsymbol{s}_1^B$ of length n_1 with the components defined in Equation (32) before applying the inverse Fourier transformation, which corresponds to the multiplication

$$\boldsymbol{u} = (\boldsymbol{W}_1 \otimes \boldsymbol{I}_{23})\ \tilde{\boldsymbol{z}}_3. \tag{77}$$

finally leading to the solution $\boldsymbol{u}$.

Conclusion 2. The computational complexities in all three steps for the 3D problem are $\mathcal{O}(N \log N)$ operations leading again to the overall complexity of $\mathcal{O}(N \log N)$ operations.

In the following section, we present numerical experiments for both the 2D and 3D case.

6 | NUMERICAL RESULTS

For $d = 2$ and $d = 3$, the computational domain is chose as $\Omega = [0,1]^d$ the unit square and unit cube, respectively. The wave number is set $\omega = 2\pi$ for all numerical experiments. The discretization meshes are uniform with respect to each x_j-direction, where the corresponding step sizes are denoted by $h_j = 1/(n_j - 1), j = 1, \dots, d$. The right-hand side is chosen

TABLE 1 CPU times in seconds for different values of $n = n_1 = n_2$ in the two-dimensional case

n	65	129	257	513	1,025	2,049
Initialization	0.07	0.09	0.31	1.74	13.38	118.99
Matlab's backslash	0.04	0.14	0.50	2.64	12.42	93.32
Fast solver	0.01	0.02	0.06	0.23	1.05	4.58

as 0.01 for the first n_1 entries and 1 for all the other entries. In this set of experiments, the efficiency of the fast direct solver is discussed. The numerical experiments have been computed using MATLAB 9.3, R2017b.

In the following, we compare the CPU times in seconds for computing the solution by applying Matlab's backslash and the fast solver presented in this work. In our case, Matlab's backslash uses the sparse direct solver UMFPACK for computing the solution of the sparse linear systems (10) and (11), see Reference 21. Besides the computational times of applying Matlab's backslash and the FFT-based direct solver also the computational times needed in the initialization process are presented. However, the initialization processes differ between the 2D and 3D problems, which has already been addressed in Sections 4 and 5. Summing up, in the 2D case the LU decompositions are computed for the matrices $\boldsymbol{H}_B$ and $\boldsymbol{H}_A$ defined in Equations (37) and (39), respectively, (see Subsection 4.1.2), whereas the 3D problem is not solved using LU decomposition in three dimensions. In this case, the action of the inverses of $\boldsymbol{H}_B$ and $\boldsymbol{H}_A$ defined in Equations (62) and (61), respectively, are computed by applying the fast solver in two dimensions n_1 times (see Subsection 5.1.2). This approach leads to a more efficient implementation in three dimensions.

The largest numerical experiments were computed in three dimensions with 135,005,697 ($= 513^3$) unknowns. Note that this size is already too large regarding the computational memory in Matlab for setting up the whole matrix matrix $\boldsymbol{A}$ as defined in Equation (11), which would be needed to apply Matlab's backslash. However, the fast solver presented in this work still computes the solution in a reasonable amount of time without using too much computational memory, as the fast solver does not need to form the whole matrix $\boldsymbol{A}$, but only of the left upper block (22) denoted by $\boldsymbol{C}_{bb}$ of the matrix $\boldsymbol{B} - \boldsymbol{A}$ defined in Equation (21).

Remark 3. All numerical experiments presented in this section were performed on a laptop with Intel(R) Core(TM) i5-6267U CPU @ 2.90GHz processor and 16 GB 2133 MHz LPDDR3 memory.

6.1 | Numerical results for the 2D case

For the first set of numerical experiments, we choose $n = n_1 = n_2$. The CPU times in seconds for the initialization process as well as the application of the fast solver and Matlab's backslash for comparison are shown in Table 1 for different values of n. The largest numerical experiments in two dimensions have been computed for $n = 2,049$ with 4,198,401 unknowns. As can be observed in Table 1, the CPU times applying Matlab's backslash are about the same size as for the initialization, whereas the CPU times of the FFT-based fast direct solver grow with $\mathcal{O}(N \log N)$. The observed CPU times are in good agreement with the nearly optimal computational complexity described by Conclusion 1 in Section 4.

Table 2 presents the CPU times in seconds for various combinations of n_1 and n_2 in the 2D case. More precisely, we compare here the computational times with respect to a large difference in the magnitude of n_1 and n_2. Both cases are discussed $n_1 \gg n_2$ and $n_2 \gg n_1$. As one can observe in Table 2 it does not influence the computational times of the solver whether $n_1 \gg n_2$ or $n_2 \gg n_1$. For $n_1 = 65$ and $n_2 = 2,049$, the fast solver's CPU time was 0.10 seconds, and for $n_1 = 2,049$ and $n_2 = 65$, it was 0.12 seconds. However, if n_1 is very large, it influences the computational times of the initialization

TABLE 2 CPU times in seconds for various combinations of n_1 and n_2 in the two-dimensional case

n_1	65	65	2,049	2,049
n_2	65	2,049	65	2,049
Initialization	0.07	0.17	110.83	118.99
Matlab's backslash	0.04	0.98	1.07	93.32
Fast solver	0.01	0.10	0.12	4.58

process, more precisely, of solving the generalized eigenvalue problem (23) performed by applying Matlab's function `eig`, which requires the full representations of the matrices $\boldsymbol{K}_1$ and $\boldsymbol{M}_1$, which are otherwise stored as sparse matrices. These increased computational times can be observed in the last two columns of Table 2. We note here that the solution method described in Reference 3 would be faster for solving this generalized eigenvalue problem.

6.2 | Numerical results for the 3D case

In the 3D case, we chose $n_j = n$ for all $j = 1, 2, 3$ for the first set of numerical experiments again. The CPU times in seconds for the initialization process as well as the application of the fast solver and Matlab's backslash for comparison are shown in Table 3 for different values of n. As can be observed from Table 3, the computational times for applying Matlab's backslash rise very quickly. For $n = 65$, its CPU time was 572.91 seconds, whereas the fast solver's CPU time was only 1.03 seconds. Then already for $n = 129$ with 2,146,689 unknowns, Matlab runs out of memory when the matrix $\boldsymbol{A}$ is formed, which makes it impossible to solve the problem (9) by applying Matlab's backslash. However, as the FFT-based fast solver only needs the setting up of the left upper block $\boldsymbol{C}_{bb}$ of the matrix $\boldsymbol{B} - \boldsymbol{A}$ and not of the entire matrix $\boldsymbol{A}$, the solver can be applied solving the problem (9) up to size $n = 513$ with 135,005,697 unknowns (Remark 3). Moreover, we can observe from Table 3 that the CPU times applying Matlab's backslash are already much larger for $n = 33$ than for the initialization process and the application of the fast solver. Table 3 shows that the CPU times of applying the FFT-based fast direct solver are nearly optimal order of complexity $\mathcal{O}(N \log N)$ and match well with the discussed results on the computational complexity in Section 5; see Conclusion 2.

Table 4 presents the CPU times in seconds for various combinations of $n_j, j = 1, 2, 3$, in the 3D case similar to Table 2 for two dimensions comparing computational times with respect to a large difference in the magnitudes of n_j. Again, we discuss all possible combinations, for example, $n_1 \gg n_2, n_3$ or $n_1, n_2 \gg n_3$. The third and fourth column of Table 4 comparing $n_2 \gg n_1, n_3$ and $n_3 \gg n_1, n_2$ show that the CPU times in applying Matlab's backslash are similar (1.06 and 0.96 in the third and fourth column, respectively) and also applying the fast solver (0.22 and 0.25 in the third and fourth column, respectively). However, note that the FFT-based direct solver is already four times faster than Matlab's backslash for this set of values. In addition, the sixth, seventh, and eighth column present similar numerical results, but Matlab's backslash cannot be applied anymore for these combinations, as Matlab runs out of memory when forming the matrix $\boldsymbol{A}$. Note that the computational times for the initialization process needed to apply the FFT-based direct solver take only around 2 seconds for these sets of values.

Remark 4. For a large n, applying Matlab's LU decomposition `lu` on the subproblems (66)–(69) as well as Matlab's `kron` function are the most time-consuming functions in the computational process. Based on this it can be expected that, for example, a `C++` implementation would be essentially faster than the Matlab implementation.

TABLE 3 CPU times in seconds for different values of $n = n_1 = n_2 = n_3$ in the three-dimensional case

n	9	17	33	65	129	257	513
Initialization	0.07	0.07	0.08	0.45	0.41	2.16	17.33
Matlab's backslash	0.02	0.28	6.39	572.91	–	–	–
Fast solver	0.09	0.12	0.21	1.03	8.43	69.55	672.27

TABLE 4 CPU times in seconds for various combinations of n_1 and n_2 and n_3 in the three-dimensional case

n_1	9	9	9	513	9	513	513	513
n_2	9	513	9	9	513	9	513	513
n_3	9	9	513	9	513	513	9	513
Initialization	0.07	0.36	0.12	1.91	2.17	2.20	2.40	17.33
Matlab's backslash	0.02	1.06	0.96	0.90	–	–	–	–
Fast solver	0.09	0.22	0.25	0.62	10.95	8.00	7.01	672.27

7 | CONCLUSIONS

In this work, we have derived an FFT-based direct solver for solving efficiently the Helmholtz equation in a rectangular domain with an ABC. The model problem and fast solver are discussed for 2D and 3D domains as well as numerical results for both cases are presented. We have shown that the method has the nearly optimal order of complexity $\mathcal{O}(N \log N)$ matching the efficiency of the FFT method. The numerical experiments illustrate that the computational complexity $\mathcal{O}(N \log N)$ is also achievable in practice. In particular, the numerical results in this work demonstrate the efficiency of the fast solver compared with Matlab's backslash and also with respect to the computational memory needed to solve the discretized Helmholtz problem for a large number of unknowns.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge the financial support by the Academy of Finland under the grant 295897. The authors like to thank Dr. Kazufumi Ito for many fruitful discussions on these fast direct solvers. Finally, the authors thank the anonymous referee as well as the editor for the valuable comments improving the article.

CONFLICTS OF INTEREST

This work does not have any conflicts of interest

ORCID

*Monika Wolfmayr** https://orcid.org/0000-0002-3548-8240

REFERENCES

1. Heikkola E, Ito K, Toivanen J. A parallel domain decomposition method for the Helmholtz equation in layered media. SIAM J Sci Comput. 2019;41(5):C505–C521.
2. Ito K, Qiao Z, Toivanen J. A domain decomposition solver for acoustic scattering by elastic objects in layered media. J Comput Phys. 2008;227(19):8685–8698.
3. Heikkola E, Rossi T, Toivanen J. Fast direct solution of the Helmholtz equation with a perfectly matched layer or an absorbing boundary condition. Int J Numer Methods Eng. 2003;57(14):2007–2025.
4. Bamberger A, Joly P, Roberts JE. Second-order absorbing boundary conditions for the wave equation: a solution for the corner problem. SIAM J Numer Anal. 1990;27(2):323–352.
5. Turkel E. Numerical difficulties solving time harmonic systems. Nato Sci S SS III Comput Systems Sci. 2001;177:319–337.
6. Lynch RE, Rice JR, Thomas DH. Direct solution of partial difference equations by tensor product methods. Numer Math. 1964;6(1):185–199.
7. Guddati MN, Yue B. Modified integration rules for reducing dispersion in finite element methods. Comput Methods Appl Mech Eng. 2004;193:275–287.
8. Swarztrauber PN. The methods of cyclic reduction, Fourier analysis and the FACR algorithm for the discrete solution of Poisson's equation on a rectangle. SIAM Rev. 1977;19(3):490–501.
9. Buzbee B, Dorr FW, George JA, Golub GH. The direct solution of the discrete Poisson equation on irregular regions. SIAM J Numer Anal. 1971;8(4):722–736.
10. Heikkola E, Rossi T, Toivanen J. A parallel fictitious domain method for the three-dimensional Helmholtz equation. SIAM J Sci Comput. 2003;24(5):1567–1588.
11. Ernst OG. A finite-element capacitance matrix method for exterior Helmholtz problems. Numer Math. 1996;75(2):175–204.
12. Heikkola E, Kuznetsov YA, Lipnikov KN. Fictitious domain methods for the numerical solution of three-dimensional acoustic scattering problems. J Comput Acoust. 1999;7(3):161–183.
13. Heikkola E, Kuznetsov YA, Neittaanmäki P, Toivanen J. Fictitious domain methods for the numerical solution of two-dimensional scattering problems. J Comput Phys. 1998;145(1):89–109.
14. Tezaur R, Macedo A, Farhat C. Iterative solution of large-scale acoustic scattering problems with multiple right hand-sides by a domain decomposition method with Lagrange multipliers. Int J Numer Methods Eng. 2001;51(10):1175–1193.
15. Rossi T, Toivanen J. A nonstandard cyclic reduction method, its variants and stability. SIAM J Matrix Anal Appl. 1999;20(3):628–645.
16. Rossi T, Toivanen J. A parallel fast direct solver for block tridiagonal systems with separable matrices of arbitrary dimension. SIAM J Sci Comput. 1999;20(5):1778–1793.
17. Vassilevski P. Fast algorithm for solving a linear algebraic problem with separable variables. Dokl Bolg Akad Nauk. 1984;37(3):305–308.
18. Banegas A. Fast poisson solvers for problems with sparsity. Math Comput. 1978;32(142):441–446.
19. Kuznetsov YA, Matsokin AM. On partial solution of systems of linear algebraic equations. Sov J Numer Anal Math Modelling. 1989;4:453–467.

20. Ernst O, Golub GH. A domain decomposition approach to solving the Helmholtz equation with a radiation boundary condition. Contemp Math. 1994;157:177–192.
21. Davis TA. Direct methods for sparse linear systems. Vol 2. Philadelphia, PA: Fundamentals of Algorithms, SIAM, 2006.

How to cite this article: Toivanen J, Wolfmayr* M. A fast Fourier transform based direct solver for the Helmholtz problem. *Numer Linear Algebra Appl*. 2020;e2283. https://doi.org/10.1002/nla.2283