ELSEVIER

Applied Numerical Mathematics 27 (1998) 443–464

APPLIED NUMERICAL MATHEMATICS

On the implementation of the Dirichlet-to-Neumann radiation condition for iterative solution of the Helmholtz equation

Assad A. Oberai *, Manish Malhotra, Peter M. Pinsky

Department of Mechanical Engineering, Stanford University, Stanford, CA 94305, USA

Abstract

The Helmholtz equation posed on an unbounded domain with the Sommerfeld condition prescribed at infinity is considered. The unbounded domain is eliminated by imposing a Dirichlet-to-Neumann (DtN) map or a modified DtN map on a truncating surface and the resulting bounded domain problem is modeled using the finite element method. The resulting system of linear equations is then solved using a Krylov subspace iterative method. New, efficient algorithms to compute matrix–vector products that are based on the structure of the DtN and the modified DtN map are presented. Connections between the DtN map and the discrete Fourier transform in two dimensions and discrete spherical transform in three dimensions are established, and are utilized to develop fast implementations of matrix–vector product algorithms. Also, an SSOR-type preconditioner that is based on a local radiation condition is considered for the modified DtN formulation. An efficient implementation is proposed by extending Eisenstat's trick for the standard SSOR preconditioner. Finally, numerical examples which illustrate the efficacy of the proposed algorithms are presented. © 1998 Elsevier Science B.V. and IMACS. All rights reserved.

1. Introduction

In this study we consider the solution of the scalar Helmholtz equation in unbounded domains using the finite element method. Such problems arise in structural acoustics, where we wish to model the time-harmonic response of a structure surrounded by an isotropic fluid in an unbounded domain. If we ignore the effects of the viscosity of the fluid, then the first-order, time-harmonic, fluctuations of the pressure field in the fluid are governed by the Helmholtz equation with the Sommerfeld condition imposed at infinity. The Sommerfeld condition ensures the uniqueness of the solution to the problem by requiring that all the waves at infinity be outgoing. One approach for modeling such problems using the finite element method involves the truncation of the infinite fluid domain at an artificial surface, where a boundary condition is prescribed on the normal derivative of the pressure. A variety of such "non-reflecting boundary conditions" (NRBCs) have been proposed; see, e.g., [10] for a review.

* Corresponding author. E-mail: oberai@am-sun2.stanford.edu.

In this study we consider NRBCs based on an exact Dirichlet-to-Neumann (DtN) map for the exterior Helmholtz equation—the so-called DtN [6,14] and the modified DtN [11] radiation conditions. These boundary conditions have the advantage that they can be made arbitrarily accurate; however, they have a drawback in that they are non-local and couple all degrees of freedom on the truncating surface. Consequently, they are generally considered to be prohibitively expensive and are used less frequently in practice, than other methods like infinite elements and local NRBCs. The main focus of this paper is to describe algorithms for efficient implementation of these boundary conditions, with particular emphasis on large-scale problems.

For a given formulation, implementation costs essentially comprise of storage and computational costs, both of which are closely linked to the solution method employed for solving the system of linear equations. This choice of the solution method, in turn, is dictated by the size of the problem, and for large-scale problems the driving algorithm is almost always iterative. Amongst iterative procedures, the most popular algorithms fall under the category of Krylov-subspace methods. The two key steps in a Krylov-subspace method, which constitute the main part of implementation costs, are:

(1) the multiplication of a vector with the coefficient matrix at each iteration, and
(2) preconditioning the system of equations to accelerate convergence.

The purpose of this paper is twofold: to develop

(1) storage- and computationally efficient algorithms for performing matrix–vector products with DtN-based finite element matrices, and
(2) efficient preconditioners that are suitable for the resulting computational procedures.

To accomplish the first objective, efficient algorithms are proposed that utilize the special structure of kernels in the DtN map for computing matrix–vector products. These algorithms, in particular, provide an interpretation of the DtN-based maps as discrete Fourier transforms in two dimensions, and as discrete spherical transforms in three dimensions. This connection is exploited in special variants of the proposed algorithms. Secondly, a preconditioning approach that is based on an SSOR-type splitting of a part of the coefficient matrix is proposed. An extension of Eisenstat's trick [5] is proposed, whereby the preconditioner can be used very efficiently in conjunction with the matrix–vector product algorithms developed in this study.

Previously, a matrix-free implementation of the non-local DtN condition was proposed by Malhotra and Pinsky [17] that avoids evaluation of the dense finite element matrix associated with it and is also suitable for Krylov-subspace methods. However, in the absence of an explicitly available coefficient matrix, effective preconditioners—especially those based on algebraic approaches—are difficult to construct. In such a situation the use of a hierarchical basis preconditioner with QMR, in conjunction with the matrix-free computational approach, was studied in [16]. In this paper, we extend the work in [16, 17] on efficient matrix–vector product implementations, as well as consider a more general, SSOR-type preconditioning approach.

The remaining part of the paper is organized as follows. In Section 2, we review the DtN and modified DtN formulations, including the matrix equations resulting from their finite element discretization. In Section 3, we describe the matrix–vector product algorithms and the SSOR-type preconditioner. In Section 4, we present numerical examples to illustrate the performance of the proposed algorithms, and make some concluding remarks in Section 5.

2. The exterior Helmholtz problem

The exterior Helmholtz problem in d dimensions posed on an unbounded domain $\Omega \subset \mathbb{R}^d$ is given by: Find u such that

$$\mathcal{L}u \equiv -\nabla^2 u - k^2 u = f \quad \text{in } \Omega, \tag{1}$$

$$u = g \quad \text{on } \Gamma_g, \tag{2}$$

$$\nabla u \cdot \boldsymbol{n} = u_{,n} = h \quad \text{on } \Gamma_h, \tag{3}$$

$$\lim_{r\to\infty} r^{(d-1)/2}\left(\frac{\partial}{\partial r}u - iku\right) = 0. \tag{4}$$

In Eq. (1), k is the wavenumber associated with the problem and f is the prescribed forcing function. In Eq. (2), g is the prescribed Dirichlet data on Γ_g. In Eq. (3), h is the prescribed Neumann data on Γ_h and $\boldsymbol{n}$ is the unit outward normal on Γ_h. Eq. (4) is the Sommerfeld radiation condition which requires all waves at infinity to be outgoing.

In the following subsection we pose the original problem defined by Eqs. (1)–(4) on a bounded domain. In this formulation the effect of the unbounded domain is incorporated through a boundary condition imposed at the truncating surface.

2.1. DtN-based bounded domain formulation

In the unbounded domain Ω consider a hypersurface denoted by Γ_R that divides Ω into a bounded domain Ω_R and an unbounded domain Ω_∞ (see Fig. 1). Γ_R is chosen such that $\Gamma_g \cup \Gamma_h$ is completely contained in Ω_R, and $f|_{\Omega_\infty} \equiv 0$. Further, in this study we assume that Γ_R is a "d"-dimensional ball of radius R, though more general shapes are possible. The strong form of the problem associated with the

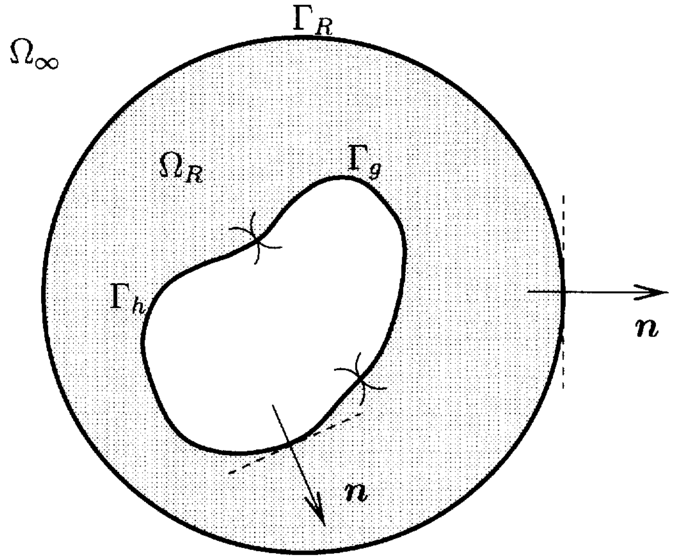

Fig. 1. Computational domain for the exterior acoustics problem.

bounded-domain formulation is given by: Find u such that

$$\mathcal{L}u = f \quad \text{in } \Omega_R, \tag{5}$$

$$u = g \quad \text{on } \Gamma_g, \tag{6}$$

$$u_{,n} = h \quad \text{on } \Gamma_h, \tag{7}$$

$$u_{,n} = M(u) \quad \text{on } \Gamma_R, \tag{8}$$

where $M(u)$ is the so-called non-reflecting boundary condition imposed on $u_{,n}$ at Γ_R.

Let $M_{\mathrm{dtn}}(u)$ be the boundary operator that makes the problems (1)–(4) and (5)–(8) equivalent. This exact non-reflecting boundary condition is called the Dirichlet-to-Neumann (DtN) map. In two dimensions, in polar coordinates (r, θ), which relate to the Cartesian coordinates (x, y) through $x = r\cos\theta$, $y = r\sin\theta$, this map is given by

$$M_{\mathrm{dtn}}(u) = \sum_{n=-\infty}^{\infty} \frac{1}{2\pi R} \frac{kH_{|n|}^{(1)'}(kR)}{H_{|n|}^{(1)}(kR)} \int_{\Gamma_R} \exp(\mathrm{i}n\theta)\exp(-\mathrm{i}n\theta_0)u(R, \theta_0)\,\mathrm{d}\Gamma_0. \tag{9}$$

In the above equation, $H_n^{(1)}$ is the Hankel function of the first kind of order n, $\mathrm{d}\Gamma_0 = R\,\mathrm{d}\theta_0$ is the differential length element on the circle of radius R, and the prime on a function is used to indicate a derivative with respect to the argument.

In three dimensions, in spherical coordinates (r, θ, ϕ) which relate to the Cartesian coordinates (x, y, z) through $x = r\sin\theta\cos\phi$, $y = r\sin\theta\sin\phi$, $z = r\cos\theta$, the DtN map is given by

$$M_{\mathrm{dtn}}(u) = \sum_{n=0}^{\infty}\sum_{j=-n}^{n} \frac{(2n+1)(n-|j|)!}{4\pi R^2(n+|j|)!} \frac{kh_n^{(1)'}(kR)}{h_n^{(1)}(kR)}$$

$$\times \int_{\Gamma_R} P_n^{|j|}(\cos\theta)\exp(\mathrm{i}j\phi)P_n^{|j|}(\cos\theta_0)\exp(-\mathrm{i}j\phi_0)u(R, \theta_0, \phi_0)\,\mathrm{d}\Gamma_0. \tag{10}$$

In the above equation, $h_n^{(1)}$ is the spherical Hankel function of the first kind of order n, P_n^j is the associated Legendre polynomial of degree n and order j and $\mathrm{d}\Gamma_0 = R^2\sin\theta_0\,\mathrm{d}\theta_0\,\mathrm{d}\phi_0$ is the differential area element on the surface of a sphere of radius R. For the derivation of Eqs. (9) and (10) see [14]. In this study, we will work with the following definition of the DtN map that takes into account both Eqs. (9) and (10):

$$M_{\mathrm{dtn}}(u) = \sum_{n=\underline{N}(d)}^{\infty} \sum_{j=-J(d)}^{J(d)} z_{|n|}(k, R; d)\psi_{nj}(\boldsymbol{x}; d) \int_{\Gamma_R} \psi_{nj}^*(\boldsymbol{x}_0; d)u(\boldsymbol{x}_0)\,\mathrm{d}\Gamma_0, \tag{11}$$

where

$$\underline{N}(d) = \begin{cases} -\infty, & d = 2, \\ 0, & d = 3, \end{cases} \tag{12}$$

$$J(d) = \begin{cases} 0, & d = 2, \\ n, & d = 3, \end{cases} \tag{13}$$

$$z_n(k,R;d)=\begin{cases}\dfrac{kH_n^{(1)\prime}(kR)}{H_n^{(1)}(kR)}, & d=2,\\[2mm] \dfrac{kh_n^{(1)\prime}(kR)}{h_n^{(1)}(kR)}, & d=3,\end{cases} \tag{14}$$

$$\psi_{nj}(\boldsymbol{x};d)=\begin{cases}\sqrt{\dfrac{1}{2\pi R}}\exp(\mathrm{i}n\theta), & d=2,\\[2mm] \sqrt{\dfrac{(2n+1)(n-|j|)!}{4\pi R^2(n+|j|)!}}P_n^{|j|}(\cos\theta)\exp(\mathrm{i}j\phi), & d=3.\end{cases} \tag{15}$$

The superscript $*$ in Eq. (11) is used to denote the complex-conjugate of a quantity.

The weak form associated with the DtN formulation is obtained from Eqs. (5)–(8). It is given by: Find $u\in\mathcal{S}$ such that $\forall w\in\mathcal{V}$

$$a(w,u)-\big(w,M_{\mathrm{dtn}}(u)\big)_{\Gamma_R}=(w,f)+(w,h)_{\Gamma_h}, \tag{16}$$

where

$$a(w,u):=(\nabla w,\nabla u)-k^2(w,u). \tag{17}$$

In Eqs. (16) and (17) the inner products $(\cdot,\cdot)$, $(\cdot,\cdot)_{\Gamma_R}$ and $(\cdot,\cdot)_{\Gamma_h}$ are defined by

$$(w,u):=\int_{\Omega_R} w^*u\,\mathrm{d}\Omega, \tag{18}$$

$$(w,u)_{\Gamma_R}:=\int_{\Gamma_R} w^*u\,\mathrm{d}\Gamma, \tag{19}$$

$$(w,u)_{\Gamma_h}:=\int_{\Gamma_h} w^*u\,\mathrm{d}\Gamma. \tag{20}$$

The function spaces $\mathcal{S}$ and $\mathcal{V}$ are defined as follows:

$$\mathcal{S}:=\{v\mid v\in H^1(\Omega_R),\ v(\boldsymbol{x})=g(\boldsymbol{x}),\ \boldsymbol{x}\in\Gamma_g\}, \tag{21}$$

$$\mathcal{V}:=\{v\mid v\in H^1(\Omega_R),\ v(\boldsymbol{x})=0,\ \boldsymbol{x}\in\Gamma_g\}. \tag{22}$$

Remark. In two dimensions, the functions ψ_{nj} defined in Eq. (15) are the normalized exponential functions and in three dimension they are the normalized spherical harmonics. These functions satisfy the following orthonormality condition on Γ_R:

$$(\psi_{nj},\psi_{mk})_{\Gamma_R}=\delta_{nm}\delta_{jk}. \tag{23}$$

Writing (11) with $u=\psi_{mk}$, and using the above equation, we get

$$M_{\mathrm{dtn}}(\psi_{mk})=z_{|m|}\psi_{mk}. \tag{24}$$

That is, $z_{|m|}$ is the exact impedance at Γ_R for the harmonic ψ_{mk}.

2.2. *Modified DtN-based bounded domain formulation*

The DtN map as defined in Eq. (11) is a sum of an infinite number of terms. When implementing the DtN boundary condition as part of a numerical approximation to the continuous problem, we truncate the summation after a finite number of terms. From Eq. (24) it is obvious that for the harmonics not included in the summation, the DtN map imposes a homogeneous Neumann condition at Γ_R. This not only affects the accuracy of the boundary condition but has implications regarding the uniqueness of the problem as well. In [12], Harari and Hughes have shown that for a given problem there is a minimum number of terms ($N_{\min}$) required to ensure uniqueness. They have derived the following bound for $N_{\min}$.

$$N_{\min} \geqslant kR. \tag{25}$$

This requirement on the truncated DtN map may be overcome by using the modified DtN map proposed by Grote and Keller [11]. This map is obtained from the standard DtN map by adding any local differential operator, and then subtracting from the DtN map the contribution of this operator to each harmonic by modifying the impedance (z_n) associated with that harmonic. This leads to a boundary condition that is the sum of a local differential operator and an integral operator which is of the same form as the DtN map. The local operator can then be chosen to ensure uniqueness of the solution for any number of terms included in the integral operator. The modified DtN condition constructed in this way has the following properties.

(1) This boundary condition is exact for all the harmonics included in the kernel of its integral operator.
(2) For the harmonics not included in the kernel, the local operator induces an approximate non-reflecting boundary condition.
(3) The solution of the resulting problem is unique for any number of terms included in the integral operator.

In the following section we will observe that these properties of the modified DtN boundary condition are crucial for its implementation in conjunction with preconditioned Krylov-subspace iterative methods.

The strong form of the problem associated with the modified DtN map is given by Eqs. (5)–(8), where the operator $M(u)$ is now given by

$$M(u) = B(u) + M_{\mathrm{mdtn}}(u). \tag{26}$$

In the above equation, B is the local differential operator and M_{mdtn} is the integral operator. Based on the definition of B, there is a family of modified DtN boundary conditions that can be constructed. The simplest choice for B is

$$B(u) = \alpha(k, R; d)\, u(R, x), \tag{27}$$

where α given by

$$\alpha(k, R; d) = \begin{cases} \mathrm{i}k - \dfrac{1}{2R}, & d = 2, \\ \mathrm{i}k - \dfrac{1}{R}, & d = 3. \end{cases} \tag{28}$$

With this choice of α, $B(u)$ is the first order Bayliss–Gunzburger–Turkel [1] operator and M_{mdtn} is given by

$$M_{\mathrm{mdtn}}(u) = \sum_{n=\underline{N}(d)}^{\infty} \sum_{j=-J(d)}^{J(d)} \tilde{z}_{|n|}(k, R; d)\psi_{nj}(\boldsymbol{x}; d) \int_{\Gamma_R} \psi^{*}_{nj}(\boldsymbol{x}_0; d)u(\boldsymbol{x}_0)\,\mathrm{d}\Gamma_0, \tag{29}$$

where

$$\tilde{z}_n(k,R;d) = z_n(k,R;d) - \alpha(k,R;d). \tag{30}$$

The weak form of the problem associated with the modified DtN boundary condition is given by: Find $u \in \mathcal{S}$ such that $\forall w \in \mathcal{V}$

$$a(w,u) - \alpha(w,u)_{\Gamma_R} - \big(w, M_{\mathrm{mdtn}}(u)\big)_{\Gamma_R} = (w,f) + (w,h)_{\Gamma_h}. \tag{31}$$

2.3. *Finite element discretization*

To obtain a finite element approximation to the solution of Eq. (16), we apply the Galerkin method to Eq. (16) and arrive at the following finite-dimensional problem: Find $u^h \in \mathcal{S}^h \subset \mathcal{S}$ such that $\forall w^h \in \mathcal{V}^h \subset \mathcal{V}$,

$$a(w^h,u^h) - \big(w^h, M^h_{\mathrm{dtn}}(u^h)\big)_{\Gamma_R} = (w^h,f) + (w^h,h)_{\Gamma_h}, \tag{32}$$

where the superscript h on the functions and the function spaces denotes a finite-dimensional basis. The discretized DtN map $M^h_{\mathrm{dtn}}(u^h)$ expressed in this finite-dimensional basis is given by

$$M^h_{\mathrm{dtn}}(u^h) = \sum_{n=\underline{N}^h(d)}^{N} \sum_{j=-J(d)}^{J(d)} z_{|n|}(k,R;d)\psi^h_{nj}(\boldsymbol{x};d) \int_{\Gamma_R} \psi^{h*}_{nj}(\boldsymbol{x}_0;d)u^h(\boldsymbol{x}_0)\,\mathrm{d}\Gamma_0, \tag{33}$$

where

$$\underline{N}^h(d) = \begin{cases} -N, & d=2, \\ 0, & d=3, \end{cases} \tag{34}$$

and $\psi^h_{nj}(\boldsymbol{x};d)$ is the projection of $\psi_{nj}(\boldsymbol{x};d)$ onto the finite-dimensional basis. In particular, we choose $\psi^h_{nj}(\boldsymbol{x};d)$ to be the interpolant of $\psi_{nj}(\boldsymbol{x};d)$.

We then expand the functions w^h and u^h as follows:

$$w^h(\boldsymbol{x}) = \boldsymbol{w}\cdot\boldsymbol{N}(\boldsymbol{x}), \tag{35}$$

$$u^h(\boldsymbol{x}) = \boldsymbol{u}\cdot\boldsymbol{N}(\boldsymbol{x}). \tag{36}$$

In the above equation, $\boldsymbol{N}(\boldsymbol{x})$ is a vector of the standard finite element basis functions and $\boldsymbol{w}$ and $\boldsymbol{u}$ are the vectors containing the nodal values of w^h and u^h, respectively. Using these expressions in Eq. (32) and evaluating the integrals, we arrive at the following system of linear equations:

$$\boldsymbol{K}_{\mathrm{dtn}}\boldsymbol{u} = \boldsymbol{f}, \tag{37}$$

where

$$\boldsymbol{K}_{\mathrm{dtn}} := \boldsymbol{K}_{\Omega} - \boldsymbol{B}_{\mathrm{dtn}}. \tag{38}$$

Here $\boldsymbol{K}_{\Omega}$ is the matrix associated with the bounded domain Ω_R. Its elements are given by

$$[\boldsymbol{K}_{\Omega}]_{ik} = (\nabla N_i, \nabla N_k) - k^2(N_i,N_k). \tag{39}$$

$\boldsymbol{B}_{\mathrm{dtn}}$ is the matrix associated with the DtN boundary condition. The elements of this matrix are given by

$$[\boldsymbol{B}_{\mathrm{dtn}}]_{ik} = \big(N_i, M^h_{\mathrm{dtn}}(N_k)\big)_{\Gamma_R}. \tag{40}$$

Remarks.

(1) By comparing Eq. (11) with Eq. (33) it is seen that the finite-dimensional DtN map is obtained by truncating the infinite series in the original DtN map and by projecting the harmonics that comprise the DtN kernel onto a finite-dimensional basis. Also from the summation in (33) we conclude that the total number of terms in the truncated DtN map, denoted by N_T is given by

$$N_T = \begin{cases} 2N+1, & d=2, \\ (N+1)^2, & d=3. \end{cases} \tag{41}$$

(2) Due to the compact nature of the finite element shape functions, $\boldsymbol{K}_\Omega$ is a sparse matrix.

(3) Due to the non-local nature of the DtN map, $\boldsymbol{B}_{\text{dtn}}$ is a dense matrix. To be precise, let N_{dof} be the total number of degrees of freedom in the finite element model and out of these let N_{Γ_R} be the number of degrees of freedom on Γ_R. Then $\boldsymbol{B}_{\text{dtn}} \in \mathbb{C}^{N_{\text{dof}} \times N_{\text{dof}}}$ with $N_{\Gamma_R} \times N_{\Gamma_R}$ non-zero entries. The structure of $\boldsymbol{B}_{\text{dtn}}$ becomes clearer if we assume that all the degrees of freedom on Γ_R are numbered last. Then $\boldsymbol{B}_{\text{dtn}}$ has zeros everywhere except on the lower $N_{\Gamma_R} \times N_{\Gamma_R}$ block, which is completely full.

(4) From Eqs. (38)–(40) we conclude that $\boldsymbol{K}_{\text{dtn}}$ is an indefinite complex-symmetric matrix.

When applied to the weak form associated with the modified DtN boundary condition (31), the Galerkin method yields: Find $u^h \in \mathcal{S}^h \subset \mathcal{S}$ such that $\forall w^h \in \mathcal{V}^h \subset \mathcal{V}$,

$$a(w^h, u^h) - \alpha (w^h, u^h)_{\Gamma_R} - \left(w^h, M^h_{\text{mdtn}}(u^h)\right)_{\Gamma_R} = (w^h, f) + (w^h, h)_{\Gamma_h}, \tag{42}$$

where

$$M^h_{\text{mdtn}}(u^h) = \sum_{n=\underline{N}^h(d)}^{N} \sum_{j=-J(d)}^{J(d)} \tilde{z}_{|n|}(k, R; d)\psi^h_{nj}(\boldsymbol{x}; d) \int_{\Gamma_R} \psi^{h*}_{nj}(\boldsymbol{x}_0; d) u^h(\boldsymbol{x}_0)\,\mathrm{d}\Gamma_0. \tag{43}$$

Using Eqs. (35) and (36) in Eq. (42) we get the following system of linear equations for the modified DtN formulation:

$$\boldsymbol{K}_{\text{mdtn}}\boldsymbol{u} = \boldsymbol{f}, \tag{44}$$

where

$$\boldsymbol{K}_{\text{mdtn}} := \boldsymbol{K}_\Omega - \alpha \boldsymbol{M}_\Gamma - \boldsymbol{B}_{\text{mdtn}}. \tag{45}$$

In the above equation

$$[\boldsymbol{M}_\Gamma]_{ik} := (N_i, N_k)_{\Gamma_R}, \tag{46}$$

$$[\boldsymbol{B}_{\text{mdtn}}]_{ik} := \left(N_i, M^h_{\text{mdtn}}(N_k)\right)_{\Gamma_R}. \tag{47}$$

Remarks.

(1) From Eq. (46) we conclude that $\boldsymbol{M}_\Gamma$ is the "mass" matrix associated with the degrees of freedom of Γ_R. Based on this, we expect the sparsity pattern of $\boldsymbol{K}_\Omega - \alpha \boldsymbol{M}_\Gamma$ to be the same as that of $\boldsymbol{K}_\Omega$.

(2) From Eqs. (47) and (43) it is obvious that $\boldsymbol{B}_{\text{mdtn}}$ has the same "dense" structure as $\boldsymbol{B}_{\text{dtn}}$.

(3) Also from the definition of $\boldsymbol{K}_{\text{mdtn}}$, we conclude that, like the DtN formulation, the system of linear equations obtained for the modified DtN formulation is indefinite and complex-symmetric.

Thus for both the DtN formulation and the modified DtN formulation the stiffness matrix is indefinite and complex symmetric and can be written as a sum of two matrices: one with the sparsity pattern associated with the standard finite element discretization and another with a "dense" structure that emanates from the non-locality of the DtN and the modified DtN boundary conditions.

3. Implementation for Krylov-subspace methods

In this section we consider using a Krylov-subspace iterative method for solving the system of Eqs. (37) and (44). In general, preconditioned Krylov-subspace methods are well suited to solve large sparse linear systems that arise from discretizing partial differential equations. For the special case of indefinite, complex symmetric systems, such as (37) and (44), the QMR algorithm [7,9] is amongst the most effective Krylov-subspace methods. The reader is referred to [15,16] for work on the application of this algorithm to problems in structural acoustics.

To solve a linear system of equations given by $\boldsymbol{Ax} = \boldsymbol{b}$, Krylov-subspace iterative methods require matrix–vector products of the kind $\boldsymbol{v} = \boldsymbol{Ap}$ at each iteration. Typically, these matrix–vector products constitute the major component of the computational effort for these methods. In the following subsection we demonstrate that the storage and computational costs for performing these operations when solving (37) and (44) can become prohibitively large. We then present efficient algorithms for performing matrix–vector products for these equations and also examine certain cases when these can be improved even further.

In practical applications, the convergence of Krylov-subspace iterative methods needs to be improved via the use of an effective preconditioner. In the last subsection, we propose a local NRBC-based preconditioning approach for (37) and (44) that can be used consistently with the matrix–vector algorithms proposed here.

3.1. Matrix–vector products

To solve Eq. (37) using a Krylov-subspace method, we need to calculate the following matrix–vector product at each iteration:

$$\boldsymbol{v} = \boldsymbol{K}_{\text{dtn}}\boldsymbol{p} = (\boldsymbol{K}_{\Omega} - \boldsymbol{B}_{\text{dtn}})\boldsymbol{p}. \tag{48}$$

First, we estimate the orders of storage requirements and computational costs incurred in computing $\boldsymbol{v}$ in a straightforward manner. In estimating these orders we do not make the distinction between the memory required for storing a real number and a complex number, or the cost of performing operations in real and complex arithmetic, since these cases differ only by a constant factor.

As pointed out in the previous section, $\boldsymbol{K}_{\Omega}$ is a sparse matrix. To store it, we require $O(N_{\text{dof}})$ real numbers. However, the matrix $\boldsymbol{B}_{\text{dtn}}$ is dense in the degrees of freedom on Γ_R. In order to store this matrix we require $O(N_{\Gamma_R}^2)$ complex numbers. So the total storage requirement for storing $\boldsymbol{K}_{\text{dtn}}$ is $O(N_{\text{dof}} + N_{\Gamma_R}^2)$ numbers. Since this estimate represents the order of the number of non-zero entries in $\boldsymbol{K}_{\text{dtn}}$, to calculate the matrix–vector product described in Eq. (48), we require $O(N_{\text{dof}} + N_{\Gamma_R}^2)$ operations. For matrix–vector products with $\boldsymbol{K}_{\text{mdtn}}$, we get exactly the same estimates.

The $O(N_{\Gamma_R}^2)$ terms in the estimates above are a direct consequence of the non-local nature of the DtN and the modified DtN boundary conditions. For typical problems in structural acoustics, $N_{\Gamma_R}^2$ is much greater than N_{dof} and this disparity increases with the problem size. Thus the computational and storage costs associated with either the DtN or the modified DtN formulation can become large enough to limit their use in conjunction with Krylov subspace methods. Our goal in this subsection is to formulate algorithms for matrix–vector products that keep these costs in check.

To begin with, we examine the matrix $\boldsymbol{B}_{\text{dtn}}$ closely. From Eqs. (47) and (33) we have

$$\begin{aligned}
[\boldsymbol{B}_{\text{dtn}}]_{ik} &= \left(N_i, M^h_{\text{dtn}}(N_k)\right)_{\Gamma_R} \\
&= \int_{\Gamma_R} N_i(\boldsymbol{x}) \sum_{n=\underline{N}^h(d)}^{N} \sum_{j=-J(d)}^{J(d)} z_{|n|}(k,R;d)\psi^h_{nj}(\boldsymbol{x};d) \int_{\Gamma_R} \psi^{h*}_{nj}(\boldsymbol{x}_0;d) N_k(\boldsymbol{x}_0)\,\mathrm{d}\Gamma_0\,\mathrm{d}\Gamma \\
&= \int_{\Gamma_R} N_i(\boldsymbol{x}) \sum_{n=\underline{N}^h(d)}^{N} \sum_{j=-J(d)}^{J(d)} z_{|n|}(k,R;d)[\boldsymbol{\psi}_{nj}]_l N_l(\boldsymbol{x}) \int_{\Gamma_R} [\boldsymbol{\psi}^*_{nj}]_m N_m(\boldsymbol{x}_0) N_k(\boldsymbol{x}_0)\,\mathrm{d}\Gamma_0\,\mathrm{d}\Gamma \qquad (49) \\
&= \int_{\Gamma_R} N_i(\boldsymbol{x}) N_l(\boldsymbol{x})\,\mathrm{d}\Gamma \sum_{n=\underline{N}^h(d)}^{N} \sum_{j=-J(d)}^{J(d)} z_{|n|}(k,R;d)[\boldsymbol{\psi}_{nj}]_l [\boldsymbol{\psi}^*_{nj}]_m \int_{\Gamma_R} N_m(\boldsymbol{x}_0) N_k(\boldsymbol{x}_0)\,\mathrm{d}\Gamma_0 \\
&= [\boldsymbol{M}_\Gamma]_{il} \left(\sum_{n=\underline{N}^h(d)}^{N} \sum_{j=-J(d)}^{J(d)} [\boldsymbol{\psi}_{nj}]_l z_{|n|}(k,R;d) [\boldsymbol{\psi}^*_{nj}]_m \right) [\boldsymbol{M}_\Gamma]_{mk}, \qquad (50)
\end{aligned}$$

where $\boldsymbol{M}_\Gamma$ is defined in Eq. (46) and $\boldsymbol{\psi}_{nj}$, defined by

$$[\boldsymbol{\psi}_{nj}]_l := \psi_{nj}(\boldsymbol{x}_l;d), \quad \boldsymbol{x}_l \in \Gamma_R, \qquad (51)$$

is the vector containing the nodal values of the harmonic defined by (n,j), on Γ_R. Based on this alternative definition of $\boldsymbol{B}_{\text{dtn}}$, we propose the following algorithm for calculating $\boldsymbol{v} = \boldsymbol{K}_{\text{dtn}}\boldsymbol{p}$.

Algorithm 3.1.1. Matrix–vector product for DtN.

Input: A vector $\boldsymbol{p} \in \mathbb{C}^{N_{\text{dof}}}$.

Output: The vector $\boldsymbol{v} = \boldsymbol{K}_{\text{dtn}}\boldsymbol{p}$.

```
1. Set $\boldsymbol{v}_1 = \boldsymbol{K}_\Omega \boldsymbol{p}$
2. Set $\overline{\boldsymbol{v}} = \boldsymbol{M}_\Gamma \boldsymbol{p}$
3. For each nodal point $\boldsymbol{x}_i \in \Gamma_R$ do
     For each harmonic given by the pair $(n, j)$ do
       $[\widehat{\boldsymbol{v}}]_{nj} = [\widehat{\boldsymbol{v}}]_{nj} + [\overline{\boldsymbol{v}}]_i [\boldsymbol{\psi}^*_{nj}]_i$
     Enddo $(n, j)$
   Enddo $i$
4. For each harmonic given by the pair $(n, j)$, $[\widehat{\boldsymbol{v}}]_{nj} = z_{|n|}[\widehat{\boldsymbol{v}}]_{nj}$
5. For each nodal point $\boldsymbol{x}_m \in \Gamma_R$ do
     For each harmonic given by the pair $(n, j)$ do
       $[\widetilde{\boldsymbol{v}}]_m = [\widetilde{\boldsymbol{v}}]_m + [\widehat{\boldsymbol{v}}]_{nj} [\boldsymbol{\psi}_{nj}]_m$
     Enddo $(n, j)$
   Enddo $m$
6. Set $\overline{\boldsymbol{v}} = \boldsymbol{M}_\Gamma \widetilde{\boldsymbol{v}}$
7. Set $\boldsymbol{v} = \boldsymbol{v}_1 - \overline{\boldsymbol{v}}$.
```

While implementing the above algorithm, we calculate and store the matrices $\boldsymbol{K}_\Omega$ and $\boldsymbol{M}_\Gamma$, and the impedance coefficients $z_n(k, R; d)$. For this the storage requirements are of the order $\mathrm{O}(N_{\mathrm{dof}} + N_{\Gamma_R})$, since $N \ll N_{\Gamma_R}$. Recall, that the straightforward matrix–vector product requires storage of the order $\mathrm{O}(N_{\mathrm{dof}} + N_{\Gamma_R}^2)$.Thus the storage requirements associated with the above algorithm are considerably lower. In steps 1, 2, 6 and 7 of the above algorithm we perform $\mathrm{O}(N_{\mathrm{dof}})$ operations. In steps 3 and 5 we perform $\mathrm{O}(N_T N_{\Gamma_R})$ operations and in step 4, $\mathrm{O}(N_T)$ operations. Thus the total number of operations performed in the above algorithm are of the order $\mathrm{O}(N_{\mathrm{dof}} + N_{\Gamma_R} N_T)$. For the standard matrix–vector product, the total number of operations are of order $\mathrm{O}(N_{\mathrm{dof}} + N_{\Gamma_R}^2)$. Typically $N_T \ll N_{\Gamma_R}$, thus we require fewer operations to implement the above algorithm when compared to the naive implementation.

For the modified DtN formulation starting from Eqs. (47) and (43), and following the same steps as for $\boldsymbol{B}_{\mathrm{dtn}}$, we have

$$[\boldsymbol{B}_{\mathrm{mdtn}}]_{ik} = [\boldsymbol{M}_\Gamma]_{il} \left(\sum_{n=\underline{N}^h(d)}^{N} \sum_{j=-J(d)}^{J(d)} [\boldsymbol{\psi}_{nj}]_l \tilde{z}_{|n|}(k, R; d) \left[\boldsymbol{\psi}_{nj}^*\right]_m \right) [\boldsymbol{M}_\Gamma]_{mk}. \tag{52}$$

Based on this definition of $\boldsymbol{B}_{\mathrm{mdtn}}$, we propose the following algorithm to compute matrix–vector products of the kind $\boldsymbol{v} = \boldsymbol{K}_{\mathrm{mdtn}} \boldsymbol{p}$. Formally, this algorithm is identical to Algorithm 3.1.1 for the DtN formulation.

Algorithm 3.1.2. Matrix–vector product for modified DtN.
Input: A vector $\boldsymbol{p} \in \mathbb{C}^{N_{\mathrm{dof}}}$.
Output: The vector $\boldsymbol{v} = \boldsymbol{K}_{\mathrm{mdtn}} \boldsymbol{p}$.

```
1. Set $\boldsymbol{v}_1 = (\boldsymbol{K} - \alpha \boldsymbol{M}_\Gamma)\boldsymbol{p}$
2. Set $\overline{\boldsymbol{v}} = \boldsymbol{M}_\Gamma \boldsymbol{p}$
3. For each nodal point $\boldsymbol{x}_i \in \Gamma_R$ do
     For each harmonic given by the pair $(n, j)$ do
       $[\widehat{\boldsymbol{v}}]_{nj} = [\widehat{\boldsymbol{v}}]_{nj} + [\overline{\boldsymbol{v}}]_i [\boldsymbol{\psi}_{nj}^*]_i$
     Enddo $(n, j)$
   Enddo $i$
4. For each harmonic given by the pair $(n, j)$, $[\widehat{\boldsymbol{v}}]_{nj} = \tilde{z}_{|n|}[\widehat{\boldsymbol{v}}]_{nj}$
5. For each nodal point $\boldsymbol{x}_m \in \Gamma_R$ do
     For each harmonic given by the pair $(n, j)$ do
       $[\widetilde{\boldsymbol{v}}]_m = [\widetilde{\boldsymbol{v}}]_m + [\widehat{\boldsymbol{v}}]_{nj} [\boldsymbol{\psi}_{nj}]_m$
     Enddo $(n, j)$
   Enddo $i$
6. Set $\overline{\boldsymbol{v}} = \boldsymbol{M}_\Gamma \widetilde{\boldsymbol{v}}$
7. Set $\boldsymbol{v} = \boldsymbol{v}_1 - \overline{\boldsymbol{v}}$.
```

While implementing this algorithm, we store the matrices $\boldsymbol{K} - \alpha \boldsymbol{M}_\Gamma$ and $\boldsymbol{M}_\Gamma$, and the impedance coefficients $\tilde{z}_n(k, ; d)$. The storage requirements and the operation count for the above algorithm are of order $\mathrm{O}(N_{\mathrm{dof}} + N_{\Gamma_R})$ and $\mathrm{O}(N_{\mathrm{dof}} + N_{\Gamma_R} N_T)$, respectively. Thus like Algorithm 3.1.1 for the DtN formulation, Algorithm 3.1.2 is an efficient way to compute matrix–vector products for the modified DtN formulation.

Remarks.

(1) The special structure of the matrix $\boldsymbol{B}_{\mathrm{dtn}}$ was first recognized by Malhotra and Pinsky [17], and used to develop a procedure similar to Algorithm 3.1.1 to compute matrix–vector products for the

matrix-free implementation of the DtN boundary condition. Note that Eq. (50) involves the representation $\psi_n^h(\boldsymbol{x}; d) = \boldsymbol{\psi}_{nj} \cdot \boldsymbol{N}(\boldsymbol{x})$, used in (49) which enables us to express $\boldsymbol{B}_{\text{dtn}}$ in terms of $\boldsymbol{M}_\Gamma$. This is an extension of the matrix-free interpretation proposed in [17], where such an approximation was not made.

(2) In Algorithm 3.1.1, the major computational cost is involved in performing steps 3 and 5. In these steps, calculating $[\boldsymbol{\psi}_{nj}]_m$, $n = \underline{N}^h(d), \dots, N$, $j = -J(d), \dots, J(d)$, for each $\boldsymbol{x}_m \in \Gamma_R$ can be expensive. One way to reduce these costs is to store the vectors $\boldsymbol{M}\boldsymbol{\psi}_{nj}$. This approach would require storage of the order of $\mathrm{O}(N_{\text{dof}} + N_T N_{\Gamma_R})$ words. To avoid these storage costs, we calculate $[\boldsymbol{\psi}_{nj}]_m$ for a given m efficiently by utilizing recurrence relations between successive harmonics evaluated at a given point. For example, in two dimensions we make use of Algorithm 3.1.3 and in three dimension we use Algorithm 3.1.4.

Algorithm 3.1.3. To calculate $\exp(\mathrm{i}ny)$, $n = 1, \dots, N$.
Input: A scalar $y \in (0, 2\pi)$.
Output: The vector $[\exp(\mathrm{i}ny)]$, $n = 1, \dots, N$.

```
1. For $n = 1, \dots, N$ do
     $\exp(\mathrm{i}ny) = \exp(\mathrm{i}(n-1)y) \times \exp(\mathrm{i}y)$
   Enddo $n$
```

Algorithm 3.1.4. To calculate $P_n^j(y)$, $n = 0, \dots, N$, $j = 0, \dots, n$.
Input: A scalar $y \in (-1, 1)$.
Output: The vector $[P_n^j(y)]$, $n = 0, \dots, N$, $j = 0, \dots, n$.

```
1. Set $P_0^0(y) = 1$.
2. For $m = 1, \dots, N$ do
     $P_m^m(y) = (1-2m)\sqrt{1-x^2}\,P_{m-1}^{m-1}(y)$.
     If $m \neq n$ then
       $P_{m-1}^m(y) = (2m+1)yP_m^m(y)$
       If $m \neq N-1$ then
         For $l = m+2, \dots, N$ do
           $P_l^m(y) = (1/(l-m))((2l-1)yP_{l-1}^m(y) - (l+m-1)P_{l-2}^m(y))$
         Enddo $l$
       Endif
     Endif
   Enddo $m$
```

3.2. DtN and discrete Fourier and spherical transforms

In this subsection we establish a relationship between the DtN map and the discrete Fourier transform in two dimensions, and the discrete spherical transform in three dimensions. These relations hold for the the modified DtN map also.

From the definition of $\boldsymbol{\psi}_{nj}$ in two dimensions and by interchanging the do loops in step 3 of Algorithm 3.1.1 (or 3.1.2) we get

$$[\widehat{\boldsymbol{v}}]_{n0} = \frac{1}{\sqrt{2\pi R}} \sum_{\theta_j \in \Gamma_R} [\overline{\boldsymbol{v}}]_j \exp(-\mathrm{i}n\theta_j), \quad n = -N, \ldots, N. \tag{53}$$

Similarly from step 5 in Algorithm 3.1.1 (or 3.1.2) we get

$$[\widetilde{\boldsymbol{v}}]_m = \frac{1}{\sqrt{2\pi R}} \sum_{n=-N}^{N} [\widehat{\boldsymbol{v}}]_{n0} \exp(\mathrm{i}n\theta_m), \quad \theta_m \in \Gamma_R. \tag{54}$$

Up to scaling by a constant, in Eq. (53) $\widehat{\boldsymbol{v}}$ is the forward discrete Fourier transform of $\overline{\boldsymbol{v}}$ and in Eq. (54) $\widetilde{\boldsymbol{v}}$ is the inverse discrete Fourier transform of $\widehat{\boldsymbol{v}}$. Thus Algorithms 3.1.1 and 3.1.2 can be restated in two dimensions as follows.

Algorithm 3.2.1. Matrix–vector product for DtN and modified DtN formulations in 2 dimensions.
Input: A vector $\boldsymbol{p} \in \mathbb{C}^{N_{\mathrm{dof}}}$.
Output: The vector $\boldsymbol{v} = \boldsymbol{K}_{\mathrm{dtn}}\boldsymbol{p}$ for DtN; the vector $\boldsymbol{v} = \boldsymbol{K}_{\mathrm{mdtn}}\boldsymbol{p}$ for modified DtN.

```
1. Set $\boldsymbol{v}_1 = \boldsymbol{K}_\Omega \boldsymbol{p}$ for DtN;
       $\boldsymbol{v}_1 = (\boldsymbol{K}_\Omega - \alpha \boldsymbol{M}_\Gamma)\boldsymbol{p}$ for modified DtN
2. Set $\overline{\boldsymbol{v}} = \boldsymbol{M}_\Gamma \boldsymbol{p}$
3. Set $\widehat{\boldsymbol{v}}$ = forward discrete Fourier transform of $\overline{\boldsymbol{v}}$
4. For each harmonic $(n, 0)$,
       for DtN $[\widehat{\boldsymbol{v}}]_{n0} = z_{|n|}[\widehat{\boldsymbol{v}}]_{n0}$;
       for modified DtN $[\widehat{\boldsymbol{v}}]_{n0} = \tilde{z}_{|n|}[\widehat{\boldsymbol{v}}]_{n0}$
5. Set $\widetilde{\boldsymbol{v}}$ = inverse discrete Fourier transform of $\widehat{\boldsymbol{v}}$
6. Set $\overline{\boldsymbol{v}} = \boldsymbol{M}_\Gamma \widetilde{\boldsymbol{v}}$
7. Set $\boldsymbol{v} = \boldsymbol{v}_1 - \overline{\boldsymbol{v}}$.
```

This interpretation of the DtN map in terms of discrete Fourier transforms is useful when the number of terms in the DtN kernel is approximately the same as the number of points on Γ_R ($N_T \approx N_{\Gamma_R}$). For this case Algorithms 3.1.1 and 3.1.2 require the same order of flops as the standard implementation, i.e., $\mathrm{O}(N_{\mathrm{dof}} + N_{\Gamma_R}^2)$. However, in Algorithm 3.2.1 we can use a fast Fourier transform in steps 3 and 5 to reduce the order of operations in these steps and in the overall algorithm to $\mathrm{O}(N_{\mathrm{dof}} + N_{\Gamma_R} \log N_{\Gamma_R})$.

Fast algorithms to perform both the forward and reverse Fourier transform have traditionally been available for uniformly spaced data [3]. In the present context, this corresponds to the case of uniform mesh on Γ_R. More recently, efficient and robust Fast Fourier transform algorithms have been developed for non-uniformly spaced data [4,2], and these may be used for more general meshes in the algorithm above. Thus for the two-dimensional case, it is possible to implement Algorithm 3.2.1 in $\mathrm{O}(N_{\mathrm{dof}} + N_{\Gamma_R} \log N_{\Gamma_R})$ operations.

In three dimensions, using Eq. (51) in step 3 of Algorithm 3.1.1 (or 3.1.2) we get

$$[\widehat{\boldsymbol{v}}]_{nj} = \sum_{(\theta_k,\phi_k)\in\Gamma_R} [\overline{\boldsymbol{v}}]_k \sqrt{\frac{(2n+1)(n-|j|)!}{4\pi R^2 (n+|j|)!}} P_n^{|j|}(\cos\theta_k) \exp(-\mathrm{i}j\phi_k),$$
$$n = 0, \ldots, N; \; j = -n, \ldots, n, \tag{55}$$

and using Eq. (51) in step 5 of Algorithm 3.1.1 (or 3.1.2) we get

$$[\widetilde{\boldsymbol{v}}]_m = \sum_{n=0}^{N} \sum_{j=-n}^{n} [\widehat{\boldsymbol{v}}]_{nj} \sqrt{\frac{(2n+1)(n-|j|)!}{4\pi R^2 (n+|j|)!}} P_n^{|j|}(\cos\theta_m) \exp(\mathrm{i}j\phi_m), \quad (\theta_m, \phi_m) \in \Gamma_R. \tag{56}$$

From the definition of a discrete spherical transform (see [13], for example), we conclude that up to scaling by a constant, $\widehat{\boldsymbol{v}}$ is the forward discrete spherical transform of $\overline{\boldsymbol{v}}$ and $\widetilde{\boldsymbol{v}}$ defines the inverse discrete spherical transform of $\widehat{\boldsymbol{v}}$. Thus our observation regarding the relation between the DtN and discrete Fourier transforms in two dimensions, extends in three dimensions to discrete spherical transforms. However, unlike the two-dimensional case there are difficulties associated with the application of a "fast" spherical transform. These are:

(1) There is no "fast" spherical convolution algorithm for non-uniformly spaced data. (Uniformly spaced data on a sphere implies an equi-angular mesh which suffers from the drawback that it is relatively coarse at the equator and fine at the poles and thus is not the optimal discretization of the surface of a sphere.)

(2) Even for uniformly spaced data, currently there are no fast spherical convolution subroutines or programs that can be easily incorporated into an existing code. However, recently fast spherical convolution algorithms have been developed and implemented by Healy and co-workers [13] and look promising.

3.3. *Preconditioning*

The notion of preconditioning is crucial to the application of a Krylov-subspace iterative method for solving linear systems. For a symmetric coefficient matrix the basic idea is as follows. Let the original system we wish to solve be given by $\boldsymbol{K}\boldsymbol{d} = \boldsymbol{f}$, where $\boldsymbol{K} \in \mathbb{C}^{N_{\mathrm{dof}} \times N_{\mathrm{dof}}}$ and $\boldsymbol{K}^{\mathrm{T}} = \boldsymbol{K}$. Let $\boldsymbol{M} \in \mathbb{C}^{N_{\mathrm{dof}} \times N_{\mathrm{dof}}}$ be a matrix that approximates the matrix $\boldsymbol{K}$ in some sense. Further, we assume that $\boldsymbol{M}$ admits the product decomposition

$$\boldsymbol{M} = \boldsymbol{M}_1\boldsymbol{M}_2, \tag{57}$$

where $\boldsymbol{M}_1$ and $\boldsymbol{M}_2$ are the so-called left and right preconditioners. Instead of the original system, the Krylov subspace method is used to solve the preconditioned linear system

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \tag{58}$$

where $\boldsymbol{A} := \boldsymbol{M}_1^{-1}\boldsymbol{K}\boldsymbol{M}_2^{-1}$, $\boldsymbol{x} := \boldsymbol{M}_2\boldsymbol{d}$, $\boldsymbol{b} := \boldsymbol{M}_1^{-1}\boldsymbol{f}$ and $\boldsymbol{M}$ is called the preconditioning matrix. The requirements on $\boldsymbol{M}$ and hence on $\boldsymbol{M}_1$ and $\boldsymbol{M}_2$ are twofold. First, for the system of Eq. (58) to have better convergence properties than the original system, $\boldsymbol{M}^{-1}$ must approximate $\boldsymbol{K}^{-1}$ well. Second, since solving (58) using a Krylov-subspace method will involve vector products with the matrix $\boldsymbol{M}_1^{-1}\boldsymbol{K}\boldsymbol{M}_2^{-1}$ at each iteration, the systems

$$\boldsymbol{M}_1\boldsymbol{z}_1 = \boldsymbol{q}, \tag{59}$$
$$\boldsymbol{M}_2\boldsymbol{z}_2 = \boldsymbol{q} \tag{60}$$

must be cheap to solve. These two requirements are difficult to satisfy simultaneously. Often, a compromise is sought between these two requirements in designing a useful preconditioner.

The SSOR preconditioner is one of the more popular preconditioning approaches. It is based on an additive split of the matrix $\boldsymbol{K}$. We observe that we can split the symmetric coefficient matrix $\boldsymbol{K}$ as follows:

$$\boldsymbol{K} = \boldsymbol{L} + \boldsymbol{\Delta} + \boldsymbol{L}^{\mathrm{T}}, \tag{61}$$

where $\boldsymbol{\Delta}$ is a non-singular diagonal matrix and $\boldsymbol{L}$ is strictly lower triangular. Since the original matrix is indefinite we may require some permutations to make $\boldsymbol{\Delta}$ non-singular. The left and the right preconditioners for the SSOR preconditioner are given by

$$\boldsymbol{M}_1 = (\boldsymbol{\Delta} + \omega \boldsymbol{L})\boldsymbol{\Delta}^{-1/2}, \tag{62}$$

$$\boldsymbol{M}_2 = \boldsymbol{\Delta}^{-1/2}(\boldsymbol{\Delta} + \omega \boldsymbol{L}^{\mathrm{T}}), \tag{63}$$

where $\omega \in (0,2)$. For the discussion of the "closeness" of $\boldsymbol{M} = \boldsymbol{M}_1\boldsymbol{M}_2$ to $\boldsymbol{K}$ the reader is referred to [8].

For the SSOR preconditioner, matrix–vector products of the kind $\boldsymbol{v} = \boldsymbol{A}\boldsymbol{p}$ imply solving Eqs. (59) and (60), with $\boldsymbol{M}_1$ and $\boldsymbol{M}_2$ defined by Eqs. (62) and (63), and a matrix–vector product with $\boldsymbol{K}$. Since $\boldsymbol{M}_1$ and $\boldsymbol{M}_2$ are triangular, this means two triangular back-solves together with one multiplication with $\boldsymbol{\Delta}$ and a matrix–vector product with $\boldsymbol{K}$. However, by writing the preconditioned matrix A in the following way:

$$\begin{aligned}\boldsymbol{A} &= \boldsymbol{M}_1^{-1}\boldsymbol{K}\boldsymbol{M}_2^{-1} = \boldsymbol{\Delta}^{1/2}(\boldsymbol{\Delta} + \omega\boldsymbol{L})^{-1}(\boldsymbol{L} + \boldsymbol{\Delta} + \boldsymbol{L}^{\mathrm{T}})(\boldsymbol{\Delta} + \omega\boldsymbol{L}^{\mathrm{T}})^{-1}\boldsymbol{\Delta}^{1/2}\\ &= \frac{1}{\omega}\boldsymbol{\Delta}^{1/2}\big((\boldsymbol{\Delta} + \omega\boldsymbol{L}^{\mathrm{T}})^{-1} + (\boldsymbol{\Delta} + \omega\boldsymbol{L})^{-1}(\boldsymbol{I} + (\omega - 2)\boldsymbol{\Delta}(\boldsymbol{\Delta} + \omega\boldsymbol{L}^{\mathrm{T}})^{-1})\big)\boldsymbol{\Delta}^{1/2},\end{aligned} \tag{64}$$

the matrix–vector product $\boldsymbol{v} = \boldsymbol{A}\boldsymbol{p}$ may be calculated as follows.

Algorithm 3.3.1. Eisenstat's trick for SSOR preconditioner.
Input: A vector $\boldsymbol{p} \in \mathbb{C}^{N_{\mathrm{dof}}}$.
Output: The vector $\boldsymbol{v} = \boldsymbol{A}\boldsymbol{p}$.

```
1. Set $\boldsymbol{t} = \boldsymbol{\Delta}^{1/2}\boldsymbol{p}$
2. Solve $(\boldsymbol{\Delta} + \omega\boldsymbol{L}^T)\widehat{\boldsymbol{t}} = \boldsymbol{t}$ for $\widehat{\boldsymbol{t}}$
3. Set $\boldsymbol{t} := \boldsymbol{t} + (\omega - 2)\boldsymbol{\Delta}\widehat{\boldsymbol{t}}$
4. Solve $(\boldsymbol{\Delta} + \omega\boldsymbol{L})\widetilde{\boldsymbol{t}} = \boldsymbol{t}$ for $\widetilde{\boldsymbol{t}}$
5. Set $\boldsymbol{v} = (1/\omega)(\widehat{\boldsymbol{t}} + \widetilde{\boldsymbol{t}})$.
```

Note that Eisenstat's trick [5], involves two solves with triangular matrices, together with a multiplication with $\boldsymbol{\Delta}$ and a multiplication with $\boldsymbol{\Delta}^{1/2}$. Therefore by using Eisenstat's trick the multiplication with $\boldsymbol{K}$ is replaced by multiplication with $\boldsymbol{\Delta}^{1/2}$, a considerably cheaper operation.

We now address the question of formulating SSOR preconditioners for Eqs. (37) and (44), given the matrix–vector products are performed using Algorithms 3.1.1. and 3.1.2, respectively (or Algorithm 3.2.1 in two dimensions). The standard SSOR preconditioner (with or without Eisenstat's trick) cannot be used for these problems because we do not assemble the matrices $\boldsymbol{K}_{\mathrm{dtn}}$ and $\boldsymbol{K}_{\mathrm{mdtn}}$. In particular, for the DtN formulation we do not assemble or store $\boldsymbol{B}_{\mathrm{dtn}}$ and similarly for the modified DtN formulation we do not assemble $\boldsymbol{B}_{\mathrm{mdtn}}$. With this restriction, the triangular back-solves, which are an integral part of the SSOR preconditioner, cannot be performed efficiently.

However, for the DtN formulation we do assemble and store the matrix $\boldsymbol{K}_{\Omega}$. Similarly, for the modified DtN formulations we assemble the matrix $\boldsymbol{K}_{\Omega} - \alpha\boldsymbol{M}_{\Gamma}$. Thus, if these matrices are "close" to $\boldsymbol{K}_{\mathrm{dtn}}$ and $\boldsymbol{K}_{\mathrm{mdtn}}$, respectively, we may use a SSOR preconditioner based on the additive split of these matrices for solving Eqs. (37) and (44).

We examine the "closeness" of $\boldsymbol{K}_{\Omega}$ and $\boldsymbol{K}_{\mathrm{dtn}}$ first. Consider the strong form of the continuous operators that these two matrices approximate. For a given right hand side $\boldsymbol{f}$ the system $\boldsymbol{K}_{\mathrm{dtn}}\boldsymbol{z} = \boldsymbol{f}$ is an approximate solution to Eqs. (1)–(4), the original exterior Helmholtz problem. On the other hand,

$\boldsymbol{K}_\Omega \boldsymbol{z} = \boldsymbol{f}$ is an approximate solution to Eqs. (5)–(8) with $M(u) = 0$ on Γ_R in Eq. (8). This is an interior acoustics problem with homogeneous Neumann data on Γ_R. The qualitative character of the solution to this problem is very different from the solution to the exterior problem. Further, while the exterior problem has a unique solution for any finite value of the wavenumber k, the interior problem may have non-unique solutions or no solutions at all for certain wavenumbers. Thus it is not natural to propose a SSOR preconditioner for the DtN formulation based on $\boldsymbol{K}_\Omega$.

Next we examine the "closeness" of $\boldsymbol{K}_\Omega - \alpha \boldsymbol{M}_\Gamma$ to $\boldsymbol{K}_{\mathrm{mdtn}}$. The strong form of the continuous problem associated with $\boldsymbol{K}_{\mathrm{mdtn}} \boldsymbol{z} = \boldsymbol{f}$ is the original exterior problem given by Eqs. (1)–(4). The strong form of the the continuous problem associated with $(\boldsymbol{K}_\Omega - \alpha \boldsymbol{M}_\Gamma)\boldsymbol{z} = \boldsymbol{f}$ is given by Eqs. (5)–(8), with $M(u) = \alpha u$. With this definition of $M(u)$, this problem has the following properties that make it similar to the exterior acoustics problem.

(1) With α defined by Eq. (28) this problem has a unique solution for all values of the wavenumber k.

(2) Recall, from Section 2.2 that α is chosen such that $M(u)$ imposes an approximate non-reflecting boundary condition at Γ_R. This makes the solution of this problem an approximation to the solution of the exterior problem.

Thus it is feasible to formulate a SSOR preconditioner based on the additive split of the matrix $\boldsymbol{K}_\Omega - \alpha \boldsymbol{M}_\Gamma$ for Eqs. (44). However, it is obvious that for the efficient calculation of matrix–vector products with this preconditioned system we cannot use Eisenstat's trick described in Algorithm 3.3.1. Instead we propose a modified Eisenstat's trick that is applicable whenever the matrix $\boldsymbol{K}$ can be written as the sum of two matrices

$$\boldsymbol{K} = \boldsymbol{K}_1 + \boldsymbol{K}_2 \tag{65}$$

and the SSOR preconditioner is based on the additive split of one of these matrices say, $\boldsymbol{K}_1$. That is $\boldsymbol{L}$ and $\boldsymbol{\Delta}$ are defined by

$$\boldsymbol{K}_1 = \boldsymbol{L} + \boldsymbol{\Delta} + \boldsymbol{L}^{\mathrm{T}}. \tag{66}$$

Since $\boldsymbol{M}_1$ and $\boldsymbol{M}_2$ are given by Eqs. (62) and (63), we can write the preconditioned matrix $\boldsymbol{A}$ as follows:

$$\begin{aligned} \boldsymbol{A} &= \boldsymbol{M}_1^{-1}(\boldsymbol{K}_1 + \boldsymbol{K}_2)\boldsymbol{M}_2^{-1} = \boldsymbol{\Delta}^{1/2}(\boldsymbol{\Delta} + \omega \boldsymbol{L})^{-1}(\boldsymbol{L} + \boldsymbol{\Delta} + \boldsymbol{L}^{\mathrm{T}} + \boldsymbol{K}_2)(\boldsymbol{\Delta} + \omega \boldsymbol{L}^{\mathrm{T}})^{-1}\boldsymbol{\Delta}^{1/2} \\ &= \frac{1}{\omega}\boldsymbol{\Delta}^{1/2}\big((\boldsymbol{\Delta} + \omega \boldsymbol{L}^{\mathrm{T}})^{-1} + (\boldsymbol{\Delta} + \omega \boldsymbol{L})^{-1}(\boldsymbol{I} + ((\omega - 2)\boldsymbol{\Delta} + \omega \boldsymbol{K}_2)(\boldsymbol{\Delta} + \omega \boldsymbol{L}^{\mathrm{T}})^{-1})\big)\boldsymbol{\Delta}^{1/2}. \end{aligned} \tag{67}$$

Using the above equation, the matrix–vector product $\boldsymbol{v} = \boldsymbol{A}\boldsymbol{p}$ can be calculated as follows.

Algorithm 3.3.2. Modified Eisenstat's trick.
Input: A vector $\boldsymbol{p} \in \mathbb{C}^{N_{\mathrm{dof}}}$.
Output: The vector $\boldsymbol{v} = \boldsymbol{A}\boldsymbol{p}$.

```
1. Set $t = \Delta^{1/2} v$
2. Solve $(\Delta + \omega L^{T})\hat{t} = t$ for $\hat{t}$
3. Set $t := t + (\omega - 2)\Delta\hat{t} + \omega K_2 \hat{t}$
4. Solve $(\Delta + \omega L)\tilde{t} = t$ for $\tilde{t}$
5. Set $v = (1/\omega)(\hat{t} + \tilde{t})$.
```

The situation described above is representative of a problem where only part of the coefficient matrix is assembled. The effect of the other part is known only through its operation on a vector. In such a case if the SSOR preconditioner is derived from the assembled part of the matrix, then the straightforward calculation of the vector $\boldsymbol{v} = \boldsymbol{A}\boldsymbol{p}$ involves two back-solves, one multiplication with $\boldsymbol{\Delta}$, a matrix–vector product with $\boldsymbol{K}_2$ and a matrix–vector product with $\boldsymbol{K}_1$. However using Algorithm 3.3.2 the operation of multiplication with $\boldsymbol{K}_1$ is replaced by a multiplication with $\boldsymbol{\Delta}^{1/2}$.

Finally, for the modified DtN formulation with SSOR preconditioner we present an algorithm that uses the modified Eisenstat's trick and the special structure of $\boldsymbol{B}_{\text{mdtn}}$ to compute the matrix–vector product $\boldsymbol{v} = \boldsymbol{A}\boldsymbol{p}$ efficiently. Here $\boldsymbol{A} = \boldsymbol{M}_1^{-1}\boldsymbol{K}_{\text{mdtn}}\boldsymbol{M}_2^{-1}$, where $\boldsymbol{M}_1$ is given by Eq. (62), $\boldsymbol{M}_2$ is given by Eq. (63), and $\boldsymbol{L}$ and $\boldsymbol{\Delta}$ are defined by the following additive split

$$\boldsymbol{K}_\Omega - \alpha\boldsymbol{M}_\Gamma = \boldsymbol{L} + \boldsymbol{\Delta} + \boldsymbol{L}^{\mathrm{T}}. \tag{68}$$

Algorithm 3.3.3. Modified DtN formulation with SSOR preconditioner.
Input: A vector $\boldsymbol{p} \in \mathbb{C}^{N_{\text{dof}}}$.
Output: The vector $\boldsymbol{v} = \boldsymbol{A}\boldsymbol{p}$.

```
1. Set $t = \Delta^{1/2} p$
2. Solve $(\Delta + \omega L^{T})\hat{t} = t$ for $\hat{t}$
3. Set $\bar{v} = M_\Gamma \hat{t}$
4. For each nodal point $x_i \in \Gamma_R$ do
     For each harmonic given by the pair $(n, j)$ do
       $[\hat{v}]_{nj} = [\hat{v}]_{nj} + [\bar{v}]_i[\psi^*_{nj}]_i$
     Enddo $(n, j)$
   Enddo $i$
5. For each harmonic given by the pair $(n, j)$, $[\hat{v}]_{nj} = \tilde{z}_{|n|}[\hat{v}]_{nj}$
6. For each nodal point $x_m \in \Gamma_R$ do
     For each harmonic given by the pair $(n, j)$ do
       $[\tilde{v}]_m = [\tilde{v}]_m + [\hat{v}]_{nj}[\psi_{nj}]_m$
     Enddo $(n, j)$
   Enddo $m$
7. Set $\bar{v} = M_\Gamma \tilde{v}$
8. Set $t := t + (\omega - 2)\Delta\hat{t} + \omega\bar{v}$
9. Solve $(\Delta + \omega L)\tilde{t} = t$ for $\tilde{t}$
10. Set $v = (1/\omega)(\hat{t} + \tilde{t})$.
```

Note that, for a two-dimensional problem, FFTs can be employed to perform steps 4 and 6 in the above algorithm.

4. Numerical examples

In this section we present two examples to compare the algorithms developed in the previous section. All numerical calculations presented in this section were performed on a R1000 processor. For solving the system of linear equations, we have used the QMR algorithm based on a three-term recurrence relation [7]. We denote by N_{it} the number of QMR iterations required to achieve

$$\frac{\|\boldsymbol{r}_n\|_2}{\|\boldsymbol{r}_0\|_2} < 1\text{e}{-06}, \tag{69}$$

where $\boldsymbol{r}_n$ is the residual after n iterations. The storage required for various formulations and algorithms is measured in terms of N_w, the number of real double-precision words stored.

4.1. Radiation from an infinitely long rigid circular cylinder

The problem of time-harmonic radiation from an infinitely long, rigid circular cylinder of unit radius in an unbounded fluid domain reduces to a problem in two dimensions if we assume that the prescribed pressure on the wet surface does not vary along the axis of the cylinder. Then the pressure field on any two planes perpendicular to the axis of the cylinder is identical and is given by the solution to the Helmholtz equation in two dimensions. The problem is posed on a domain exterior to a circle, and the radiator is represented by the circle. The strong form of the problem is given by (1)–(4) with $d = 2$, $\Gamma_h = \emptyset$, Γ_g defined to be the circumference of the circle and Ω the unbounded domain outside it. The wave number for this problem is $k = 20$ and the forcing function $f \equiv 0$. The function g is chosen such that it is a random number between -1 and 1 on the nodes, and varies linearly between them.

To solve this problem we truncate the unbounded domain with a circle of radius 1.25 and impose the DtN or the modified DtN boundary condition on this circle. We use bilinear finite elements and compute the pressure field by solving (37) or (44) using the QMR algorithm. We use a uniform mesh in (r, θ), with 1025 elements in the circumferential direction and 25 elements in the radial direction. The DtN map is truncated at $N = 171$, which implies $N_T = 343$. For every harmonic included in the DtN kernel we have at least 6 nodal points per wavelength. We compare

(1) The DtN map, via the naive approach of assembling B_{dtn} and then calculating $\boldsymbol{K}_{\text{dtn}}\boldsymbol{p}$.
(2) The DtN map via Algorithm 3.1.1, which utilizes the special structure of $\boldsymbol{B}_{\text{dtn}}$.
(3) The DtN map via Algorithm 3.2.1, which uses the FFT.
(4) The modified DtN map with the SSOR ($\omega = 1$) preconditioner via Algorithm 3.3.3. In this algorithm we have used the FFT.

Table 1 contains the results obtained from solving this problem. From this table we observe:

(1) all the algorithms implemented above require about 9 times less storage than the naive version,
(2) using the FFT in Algorithm 3.2.1 makes it about 2.5 times faster than the naive algorithm,
(3) the effect of the SSOR preconditioner in lowering N_{it} is significant (about 2.5 times).

In the previous section it was indicated that in two dimensions, Algorithm 3.3.3 with the FFT is the most efficient algorithm developed in this study. This observation is verified by comparing the first and

Table 1
Storage requirements and performance results for numerical example 1

Method	Storage N_w ($\times 10^3$)	Iterations N_{it}	Wall-clock time (sec)
DtN (Naive)	1172	726	114
DtN (Algorithm 3.1.1)	121	725	104
DtN (Algorithm 3.2.1)	121	725	49
Modified DtN (Algorithm 3.3.3)	121	264	29

the last rows of Table 1. When compared to the naive algorithm, Algorithm 3.3.3 requires 9 times less storage and is about 4 times faster.

4.2. Three-dimensional rigid scattering from a cigar-shaped object

As our second example, we consider the time-harmonic scattering of plane waves from a sound-hard cigar-shaped object. The scatterer is a cylinder, whose axis is aligned with the Cartesian x axis, with conical-to-spherical end caps. The length of the scatterer is 8 units and the diameter of the cylindrical part is 1 unit. The incident pressure field is a time-harmonic plane wave and we wish to solve for the scattered pressure. The strong form of the problem for the scattered pressure field u is given by (1)–(4), with $d = 3$, $k = 2$, $\Gamma_g = \emptyset$, Γ_h defined to be the surface of the scatterer, and Ω the unbounded domain outside Γ_h. Since we have assumed the scattering object to be sound-hard, the prescribed Neumann data h is given by

$$h(\boldsymbol{x}) = -\boldsymbol{n} \cdot \boldsymbol{q} \exp(\mathrm{i}k\boldsymbol{q} \cdot \boldsymbol{x}), \quad \boldsymbol{x} \in \Gamma_h. \tag{70}$$

In the above equation $\boldsymbol{q}$, is the wave vector associated with the incident plane wave. For results presented in this subsection $\boldsymbol{q} = [0.5, 0.5, 0.707]$. We have found that the convergence of the iterative scheme is insensitive to the direction of the plane wave, and the results described below are typical for any wave vector $\boldsymbol{q}$.

To solve this problem, we have truncated the fluid domain with a sphere of diameter 12 units and modeled the fluid domain between the scatterer and the surface of the sphere using 82,944 trilinear hexahedral finite elements. The surface mesh for one-half of the finite element model for this problem is shown in Fig. 2. The total number of degrees of freedom is $N_{\mathrm{dof}} = 78{,}370$, and the number of degrees of freedom on Γ_R is $N_{\Gamma_R} = 4{,}610$. We solved the problem using both the DtN and the modified DtN boundary condition, with $N = 24$ ($N_T = 625$). Observe that $N = 2N_{\min}$ for the DtN boundary condition and hence uniqueness is assured.

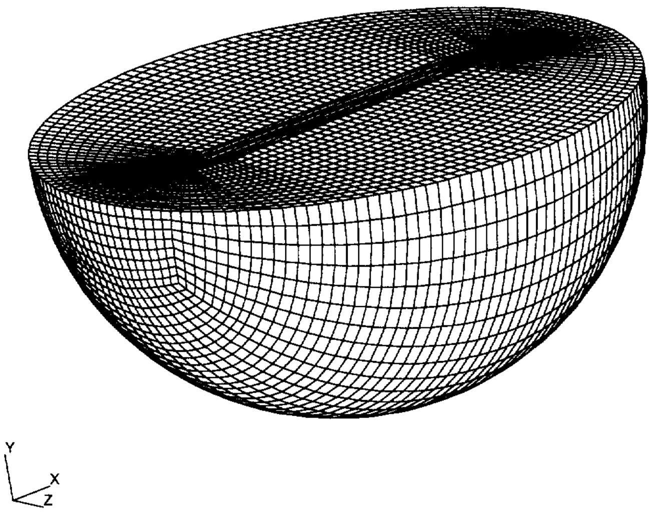

Fig. 2. Surface mesh for one-half of the finite element model for numerical example 2.

For this problem we have implemented the following algorithms.

(1) The DtN map using Algorithm 3.1.1 for matrix–vector products.
(2) The modified DtN map using Algorithm 3.1.2 for matrix–vector products.
(3) The modified DtN map with SSOR ($\omega = 1$) preconditioner using Algorithm 3.3.2 for matrix–vector products.

We have not implemented the DtN or the modified DtN formulation with the fully assembled stiffness matrices, because for these cases the memory required for storing the dense boundary matrix is extremely large; an estimate of the requirements given in Table 2 shows that it is about 20 times more than the other approaches.

Table 2
Storage requirements and performance results for numerical example 2

Method	Storage N_w ($\times 10^6$)	Iterations N_{it}	Time per iteration	Wall-clock time (sec)
DtN (Naive)	10.6	—	—	—
DtN (Algorithm 3.2.1)	0.573	1378	1.69	2333
Modified DtN (Algorithm 3.2.2)	0.573	1362	1.76	2400
Modified DtN (Algorithm 3.3.1)	0.573	275	2.29	632

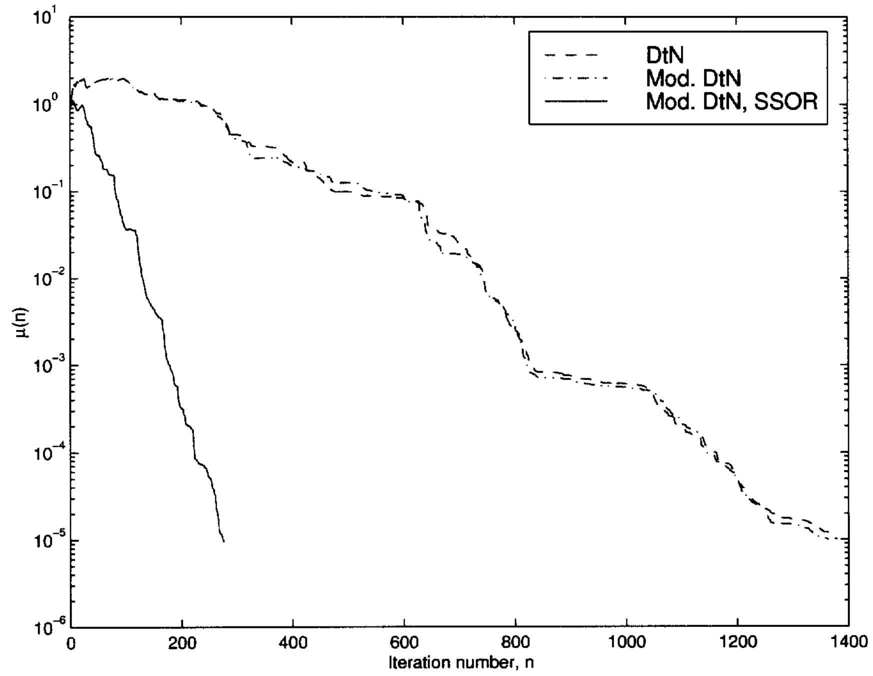

Fig. 3. Iterative convergence for numerical example 2.

In Fig. 3 we compare the convergence for the three formulations considered. In the figure, the abscissa represents the iteration number n and the ordinate axis represents $\mu(n)$, a measure of the relative residual $\|\boldsymbol{r}_n\|_2/\|\boldsymbol{r}_0\|_2$. The quantity μ is calculated at each iteration without performing additional matrix–vector products and satisfies the following relation:

$$\mu(n) \leqslant \frac{\|\boldsymbol{r}_n\|}{\|\boldsymbol{r}_0\|}, \quad \forall n. \tag{71}$$

From Fig. 3 and Table 2, we observe that the DtN as well as the modified DtN formulations converge in approximately 1400 QMR iterations, while the solution for the modified DtN formulation with the SSOR preconditioner based on $\boldsymbol{K}_\Omega - \alpha \boldsymbol{M}_\Gamma$ converges in only 275 iterations. Further, by using Eisenstat's trick for this preconditioner, these improvements in the iterative convergence directly translate into improvement in timings. This is illustrated by the timing results shown in the last two columns of the table.

5. Conclusions

The implementation of the DtN and the modified DtN boundary conditions in the finite element solution of the exterior acoustics problem has been examined for the case when the resulting system of linear equations is solved using a Krylov-subspace iterative method.

Algorithms to compute a matrix–vector product in $\mathrm{O}(N_{\mathrm{dof}} + N_{\Gamma_R} N_T)$ flops with significantly low storage requirements of $\mathrm{O}(N_{\mathrm{dof}} + N_{\Gamma_R})$ words are proposed. Here, N_{dof} is the total number of degrees of freedom in the problem, N_{Γ_R} is the number of degrees of freedom on the truncating surface Γ_R, and N_T is the number of terms in the kernel of the DtN (or the modified DtN) map. Since $N_T \ll N_{\Gamma_R}$, these algorithms are typically much faster than the standard matrix–vector products that require $\mathrm{O}(N_{\mathrm{dof}} + N_{\Gamma_R}^2)$ flops and $\mathrm{O}(N_{\mathrm{dof}} + N_{\Gamma_R}^2)$ words of storage. For problems in two dimensions, the fast Fourier transform was employed to further reduce the computational cost per matrix–vector product to $\mathrm{O}(N_{\mathrm{dof}} + N_{\Gamma_R} \log N_{\Gamma_R})$ flops.

A preconditioning approach that utilizes an SSOR splitting of local, approximate NRBC based finite element matrix has been studied to precondition the modified DtN-based finite element linear systems. An efficient implementation of this preconditioner has been proposed, by exploiting Eisenstat's trick for the case when the SSOR operator is constructed from only a part of the coefficient matrix, that is assembled explicitly. The remaining part, consisting of the non-local DtN contribution, is needed only for computing the matrix–vector product. The resulting algorithms perform a matrix–vector product with the preconditioned matrix at a cost that is only nominally more than that required for performing the product with the coefficient matrix itself.

Numerical examples illustrating the performance of proposed preconditioner and matrix–vector product algorithms have been presented. Significant savings in storage costs, of a factor of 20, and computational costs, of around a factor of 4, have been observed.

In this study we have focused on the scalar Helmholtz equation with a circular or spherical truncating boundary. However, with only slight modifications, algorithms developed in this paper can be extended to implement these boundary conditions on more general truncating surfaces like ellipses and prolate spheroids, and also to problems involving the vector Helmholtz and Maxwell's equations.

Acknowledgements

The authors would like to thank Julia Schmidt for carefully proofreading this article and the U.S. Office of Naval Research (ONR) for their support through grant N00014-96-10109.

References

[1] A. Bayliss, M. Gunzberger and E. Turkel, Boundary conditions for the numerical solution of elliptic equations in exterior domains, *SIAM J. Appl. Math.* 42 (1982) 430–451.

[2] G. Beylkin, On the fast Fourier transform of functions with singularities, *Appl. Comput. Harmon. Anal.* 2 (1995) 363–381.

[3] J.W. Cooley and J.W. Tukey, An algorithm for the machine computation of complex Fourier series, *Math. Comp.* 19 (1965) 297–301.

[4] A. Dutt and V. Rokhlin, Fast Fourier transform for nonequispaced data, *SIAM J. Sci. Statist. Comput.* (1993).

[5] S.C. Eisenstat, Efficient implementation of a class of preconditioned conjugate gradient methods, *SIAM J. Sci. Statist. Comput.* 2(2) (1984) 130–138.

[6] K. Feng, Asymptotic radiation conditions for reduced wave equation, *J. Comput. Math.* 2 (1984) 1–4.

[7] R.W. Freund, Conjugate gradient-type methods for linear systems with complex symmetric coefficient matrices, *SIAM J. Sci. Statist. Comput.* 13 (1992) 425–448.

[8] R.W. Freund and F. Jarre, A QMR-based interior point algorithm for solving linear programs, Numerical Analysis Manuscript 94-19, AT&T Bell Laboratories, Murray Hill (1994).

[9] R.W. Freund and N.M. Nachtigal, QMR: a quasi-minimal residual method for non-Hermitian linear systems, *Numer. Math.* 60 (1991) 315–339.

[10] D. Givoli, *Numerical Methods for Problems in Infinite Domains* (Elsevier, Amsterdam, 1992).

[11] M.J. Grote and J.B. Keller, On non-reflecting boundary conditions, *J. Comput. Phys.* (1993).

[12] I. Harari and T.J.R. Hughes, Analysis of continuous formulations underlying the computation of time harmonic acoustics in exterior domains, *Comput. Methods Appl. Mech. Engrg.* 97 (1992) 103–124.

[13] D.M. Healy Jr, D. Rockmore and S.S.B. Moore, FFTs for the 2-sphere-improvements and variations, Technical Report PCS-TR96-292, Dartmouth College, Department of Computer Science (1996).

[14] J.B. Keller and D. Givoli, Numerical methods for problems in infinite domains, *J. Comput. Phys.* 82(1) (1989) 172–192.

[15] M. Malhotra, R.W. Freund and P.M. Pinsky, Solution of multiple radiation and scattering problems in structural acoustics using a block quasi-minimal residual iterative method, *Comput. Methods Appl. Mech. Engrg.* 146 (1997) 173–196.

[16] M. Malhotra and P.M. Pinsky, Parallel preconditioning based on h-hierarchical finite elements with application to acoustics, *Comput. Methods Appl. Mech. Engrg.* 155 (1998) 97–117.

[17] M. Malhotra and P.M. Pinsky, A matrix-free interpretation of the nonlocal Dirichlet-to-Neumann radiation boundary condition, *Internat. J. Numer. Methods Engrg.* 39 (1996) 3705–3713.