DELFT UNIVERSITY OF TECHNOLOGY

REPORT 02-12

Some Numerical Aspects for Solving Sparse Large Linear Systems Derived from the Helmholtz Equation

Y.A. Erlangga

ISSN 1389-6520

Reports of the Department of Applied Mathematical Analysis

Delft 2002

Some Numerical Aspects for Solving Sparse Large Linear Systems Derived from the Helmholtz Equation

Y.A. Erlangga

September 23, 2002

Abstract

In this report, several numerical aspects and difficulties for solving a linear system derived from the time-harmonic wave equations are overviewed. The presentation begins with the derivation of the governing equation for waves propagating in general inhomogeneous media. Due to the need of numerical solutions, various discretizations based on finite difference are discussed. Some numerical methods which are considered applicable for solving the resulting large but sparse, indefinite, non-Hermitian linear system are discussed, including some types of existing preconditioners to possibly accelerate the convergence. Following demands for solving large linear problems in parallel computers, domain decomposition methods are revisited, with particular attentions on methods for solving the discrete Helmholtz equation.

Keywords: Wave equation, radiation condition, finite difference, Krylov subspace methods, multigrid, preconditioners, domain decomposition

Contents

1 Introduction

This report presents some issues related to numerical solutions of the wave equation. More specific, we describe methods that have been used to solve the linear systems obtained from the discrete wave equation which can be used to describe scattering phenomena. In particular, we concentrate on a time-harmonic wave equation represented by the Helmholtz equation

$$\Delta p + k^2 p = f, \quad \text{in } \Omega \in \mathbb{R}^3, \tag{1}$$

with certain boundary conditions at $\partial\Omega$ and varying k in space.

Prior to the discussion of the recent solution techniques to the wave equation, we will present the derivation of the mathematical model for wave propagation. Boundary conditions associated with the problem are also discussed. Since in computations we would have to truncate the infinite domain into a finite region, the so-called *radiation condition* will be our main interest. It will be treated in Chapter 2.

Further, we are interested in numerical solutions to the wave equation. That is, we look for solutions of a linear system

$$\mathbf{A}\mathbf{p} = \mathbf{f}, \quad \mathbf{A} \in \mathbb{C}^{n\times n}, \tag{2}$$

derived from discretization of (1). Therefore, some numerical algorithms will be discussed. In short, the discretization of the wave equation results in a linear system with a large, sparse coefficient matrix which is non-Hermitian and indefinite for a large range of applications (Section 3). The standard direct solution methods will become inefficient to solve such a system if some additional techniques are not incorporated (e.g. domain decomposition). As alternatives, iterative methods can be used to solve the linear system. In either direct or iterative methods, many competing algorithms are available and should be studied. For the direct methods, the *minimum degree algorithm* and *nested dissection* method will be discussed. For iterative methods, some methods developed based on Krylov subspace methods will be discussed (Section 5). Since the iterative methods are only robust and efficient with a preconditioner, discussions on preconditioners appropriate for solving the linear systems will be given. Preconditioners will be the subject of Section 6. In Section 7, the domain decomposition method, which is a powerful technique for parallel computing, will be revisited. Some concluding remarks on the methods for solving the Helmholtz equation are given in Section 8.

2 The wave equation

In this section we derive the time-harmonic Helmholtz equation for a general 3-D inhomogeneous problem. At the end we discuss the boundary conditions. We derive the so-called Sommerfeld boundary condition ensuring outgoing waves without reflections at the outer boundary. The case with at least one Sommerfeld's boundary condition at the outer boundaries, we call *exterior problem.* If none of the outer boundaries is the Sommerfeld condition we call it an *interior problem.* In practical applications, one should approximate the Sommerfeld condition to a certain accuracy. The first order Sommerfeld condition is the simplest one but less accurate for treating outgoing waves with an incidence angle. This will be discussed together with higher order approximations.

2.1 The Helmholtz equation

The central equation that governs the acoustic waves is the Helmholtz equation. The derivation of this equation will be the subject of this section. However, only a brief discussion is given. We refer to Colton and Kress [13, 14] or Ghosh-Roy and Couchman [53] for a more detailed derivation.

We consider a region $\Omega \in R^3$ wherein sound waves of small amplitude propagate. In an acoustic wave problem, we consider the local motions of particles in the medium while the medium itself is motionless. We define field variables which are of our interest: pressure $p = p(x,t)$, particle velocity $\mathbf{v} = \mathbf{v}(x,t)$, density $\rho = \rho(x,t)$, and entropy $S = S(x,t)$. If we assume the medium to be inviscid and compressible, the wave motion can be represented by Euler's equation

$$\partial_t \mathbf{v} + (\mathbf{v} \cdot \nabla)\, \mathbf{v} = -\frac{1}{\rho} \nabla p, \tag{3}$$

the continuity equation

$$\partial_t \rho + \nabla \cdot (\rho \mathbf{v}) = 0, \tag{4}$$

and the equation of state

$$p = p(\rho, S), \tag{5}$$

with S follows the adiabatic hypothesis

$$\partial_t S + \mathbf{v} \cdot \nabla p = 0. \tag{6}$$

When waves propagate through the medium, all field variables are perturbed from their quiescent conditions. If we assume only small perturbations to occur due to wave propagations, the perturbed field variables can be expressed as $p(x,t) = p_0(x,t) + \delta p(x,t)$, $\rho(x,t) = \rho_0(x,t) + \delta\rho(x,t)$, $S(x,t) = S_0(x,t) + \delta S(x,t)$, and $\mathbf{v}(x,t) = \mathbf{v}_0(x,t) + \delta\mathbf{v}(x,t)$, $|\delta\mathbf{v}| \ll |\mathbf{v}_0|$, where subscript "$_0$" denotes the quiescent condition. These expressions can be used to linearize Euler's equation by inserting them to (2.1)–(2.4), giving the following linearized equations:

$$\partial_t \mathbf{v} = -\frac{1}{\rho_0} \nabla p, \tag{7}$$

$$\partial_t \rho + \rho_0 \nabla \cdot \mathbf{v} = 0, \tag{8}$$

$$\partial_t p = \frac{\partial p}{\partial \rho}|_{\rho_0, S_0} \partial_t \rho. \tag{9}$$

After differentiating (2.5) in space and (2.6) in time and substracting the resulting equations one to another, we obtain the following relation:

$$\partial_{tt} \rho = \Delta p, \tag{10}$$

where $\Delta = \partial_{xx} + \partial_{yy} + \partial_{zz}$. By making use of (2.7), we get

$$\frac{\partial p}{\partial \rho}|_{\rho_0, S_0} \partial_{tt} p = \Delta p. \tag{11}$$

If the process during the propagation of waves is isentropic (this can be justified since the adiabatic hypothesis is imposed and, from Euler's equation, no dissipative process occurs), we can use the definition of the speed of sound

$$c^2 = \frac{\partial p}{\partial \rho}|_{\rho_0, S_0} \tag{12}$$

to form the equation of wave motion,

$$\partial_{tt} p = \frac{1}{c^2} \Delta p. \tag{13}$$

We consider time-harmonic waves with time dependent pressure of the form

$$p(x,t) = p(x) \mathrm{e}^{-\mathrm{i}\omega t} \tag{14}$$

where $\omega > 0$ denotes frequency. Eq. (2.11) reduces to

$$\Delta p(x) + k^2(x) p(x) = 0. \tag{15}$$

In (2.13) k is the wavenumber, defined as $k = \omega / c$. Because $\omega = 2\pi / T$, where T is the wave period, we find also $k = 2\pi / \lambda$, where $\lambda = cT$ is the wavelength. Eq. (2.13) is known as *Helmholtz's equation* expressed in pressure.

We can also derive the Helmholtz equation expressed in the velocity by using the velocity potential relation $\mathbf{v} = 1/\rho_0 \nabla \Phi$ and $p = -\partial_t \Phi$, where $\Phi = \Phi(x,t)$ is the velocity potential. Substituting these relations to (2.11), we observe that Φ also satisfies the wave equation. Introducing time-harmonic waves with $\Phi(x,t) = \phi(x) \exp(\mathrm{i}\omega t)$, $\omega > 0$, Helmholtz's equation can be expressed as (see [13, 14, 53])

$$\Delta \phi + k^2 \phi = 0. \tag{16}$$

This equation and (2.13) are identical, and represent wave propagation in a medium. Since we are interested in pressure field solution, we will use (2.13) instead of (2.14). To obtain the solution in ϕ, one can transform p via the potential relations.

2.2 Boundary conditions

Since we are faced with a partial differential equation, proper boundary conditions are required. These boundary conditions should be such that the resulting problem is well-posed. In general, we identify two types of boundary conditions. The first type is a boundary condition for waves propagate to infinite distance. The second type relates to the case of waves scattered by obstacles in a medium. These boundary conditions will be discussed in what follows.

Conditions for a boundary at infinite distance can be derived by considering the physical situation at infinity. This can be viewed from the solution of Helmholtz's equation (2.13). We assume spherical symmetric waves propagating from a source or scatterer in Ω. Eventhough in most cases, at distance close to the source waves may be arbitrary and more complex than just spherical waves, we assume that at infinity the waves will be disentangled and become spherical. Under this assumption, the problem can be easily evaluated if (2.13) is transformed to a spherical coordinate system, giving the relation

$$(rp)'' + k^2(rp) = 0, \tag{17}$$

with the general solution of (2.15) to be

$$p(r) = A\frac{\cos(kr)}{r} + B\frac{\sin(kr)}{r}. \tag{18}$$

Combining with (2.12), the time-harmonic solution reads

$$p(r,t) = A^*\frac{e^{\mathrm{i}(kr-\omega t)}}{r} + B^*\frac{e^{-(ikr+\omega t)}}{r}. \tag{19}$$

From quick inspection, for surfaces of constant phase it is easily deduced that the first term on the right hand side describes waves propagating away from a scatterer, if ωt increases with time. Contrary to the first term, the second term represents waves propagating from infinity. For physical reasons if the region Ω bounded by a spherical surface $\Gamma = \partial\Omega$ contains scatterer, the latter term should be removed. This is because physics only allows outgoing waves. As a consequence, only the first term on the right hand side remains in the solution. Since the remaining term contains the factor r^{-1}, the amplitudes of the wave must also vanish at infinity.

However, the vanishing amplitude condition can not be achieved, not even if the spherical surface goes to infinity. If we choose two spherical surfaces with radius R^1 and R^2 enclosing the scatterers region with $R^2 > R^1$, it is found that $p^{R^1}(x) = p^{R^2}(x)$, exhibiting an invariance property to the radius. [The proof can be found in Ghosh-Roy and Couchman [53]. There, they first calculate $p^{R^1}(x)$ and $p^{R^2}(x)$ using the Helmholtz representation in the interior, and then subtract one from the other. The divergence theorem is applied to the region between bounding surfaces Γ_{R_1} and Γ_{R_2}]. If there exists a nonvanishing contribution in any spheres in the region, then this contribution will not vanish, not even if $R \to \infty$. This fact again should be rejected for physical reasons.

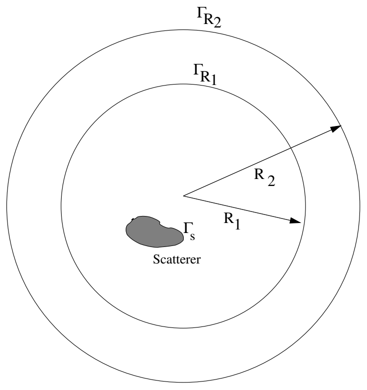

Figure 1: Schematic picture of spherical waves propagating from a scatterer.

The vanishing criterion ensuring $p(r) \to 0$ as $r \to 0$ is provided by the radiation condition. For $k > 0$, if we differentiate (2.17) with respect to r, the following relation results:

$$r(p' - \mathrm{i}kp) = -p. \tag{20}$$

If $r \to \infty$, $p \to 0$. By this we can reformulate the problem as

$$\lim_{r\to\infty} (p' - \mathrm{i}kp) \sim o(r^{-1}); \ \lim_{r\to\infty} rp \sim O(1). \tag{21}$$

In (2.19), symbols 'o' and 'O' are *Landau's symbols* defined as

$$\begin{aligned} f(x) \sim O(g(x)) &\Longrightarrow \frac{f(x)}{g(x)} \to C, \\ f(x) \sim o(g(x)) &\Longrightarrow \frac{f(x)}{g(x)} \to 0, \end{aligned} \tag{22}$$

with C being a constant. The first condition in (2.19) is the *Sommerfeld radiation* condition, whereas the second condition of (2.19) is the *finiteness* condition. (In some references, a different terminology is used for the radiation condition, for instance, *non-reflecting* or *absorbing* condition. In this report, we will use the term "radiation condition" for the boundary condition at infinity.)

The boundary condition in (2.19) is of first-order type and has a non-reflecting property for normal incoming waves. In the case of waves with a finite incidence angle θ, waves are still reflected back into the interior. However the normal speed of the reflecting waves is very small and does not influence too much the solution far from the boundaries if the boundaries are taken very far from the source or scatterer [20].

Engquist and Majda [20] (see also Bamberger, Joly, and Roberts [3]) derive higher order radiation conditions that reduce the amplitude of the reflecting waves on the boundaries. Instead of looking at families of solutions (2.13) in the form of (2.12), a more general solution can be constructed for a 2-dimensional wave as

$$p(x,y,t) = \iint \mathrm{e}^{\mathrm{i}\sqrt{\xi^2-\omega^2}+\xi t+py} \rho(\xi,\omega)\hat{p}(0,\xi,\omega)\,\partial\xi\partial\omega \tag{23}$$

where (ω,ξ) are dual variables to (y,t). Here, we position the boundary at $x=0$ and let waves travel from $x>0$ to the left. $\hat{p}$ is the Fourier transform of p. A condition for perfect wave annihilation on the boundary can be determined by differentiating (2.21) in x giving,

$$\partial_t p - \iint \mathrm{e}^{\mathrm{i}\sqrt{\xi t+\omega y}} \mathrm{i}\sqrt{\xi^2-\omega^2}\rho\hat{p}\,\partial\xi\,\partial\omega = 0. \tag{24}$$

Using the *pseudo-differential operator* after Nirenberg (see [20]), the radiation condition can then be written as

$$\left(\partial_t - \rho\left(\partial_y,\partial_t\right)\sqrt{\partial_{tt}-\partial_{yy}}\right) p|_{x=0} = 0, \tag{25}$$

or

$$\left(\partial_t - \mathrm{i}\xi\sqrt{1-(\omega/\xi)^2}\right) = 0. \tag{26}$$

If a zeroth order approximation is used to the term under the square-root, i.e., $1-(\omega/\xi)^2 = 1+O(\omega^2/\xi^2)$, and knowing that $\mathrm{i}\xi$ is related to ∂_t, a radiation condition based on the first order derivative can be determined, namely

$$(\partial_x - \partial_t)p|_{x=0} = 0. \tag{27}$$

A higher order approximation can be applied to the square root term: $1-(\omega/\xi)^2 = 1-1/2(\omega/\xi)^2+O(\omega^4/\xi^4)$. This gives

$$\left(\partial_x\partial_t - \partial_{tt} + \frac{1}{2}\partial_{xx}\right) p|_{x=0} = 0, \tag{28}$$

which is also a radiation condition but with a second order derivative.

If the *second Padé approximation* is used to approximate the square root term of (2.24), a radiation condition based on third order derivative can be found, namely

$$\left(\partial_{tt}\partial_x - \frac{1}{4}\partial_x\partial_{yy} - \partial_{ttt} + \frac{3}{4}\partial_t\partial_{yy}\right) p|_{x=0} = 0. \tag{29}$$

In [20] it is shown that all higher radiation conditions (2.25), (2.26), and (2.27) are perfectly non-reflecting for normal incoming waves, well-posed, and have a local property. The latter means that solutions on the boundary at the next time level can be computed without having all values on the boundary at the previous time level. This property is beneficial in terms of operation count. Numerical experiments also reveal that the amplitude of the reflecting waves can be reduced up to only 0.5 % for (2.27) in case of $\theta = 45^o$ incident waves.

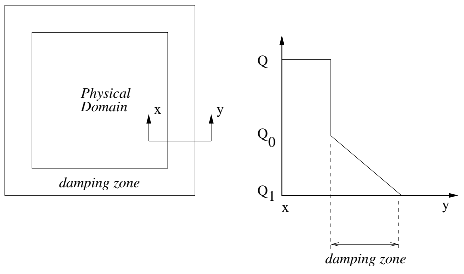

Figure 2: Computational domain with perfectly matched layer (damping zone). Q is a quality variable.

An even more accurate non-reflecting condition is proposed by Clayton and Engquist [11]. By treating the boundary from the solutions of the paraxial equation (by assumming harmonic solutions to the Helmholtz equation) and by approximations to it, a collection of non-reflecting conditions is found; see [11]. This boundary condition turns out to be more effective than the previous ones in removing reflections and has a local property.

A different boundary treatment is by enlarging the computational domain next to the outer boundary of the physical domain (see fig. 2.2). In this additional domain (termed as *damping zone*) a "reducing complex velocity condition" is imposed linearly. Observation on the early numerical results (see [7, 43, 38, 39]) show "perfect" penetration of the outgoing waves. This idea was firstly introduced by Berenger [7] for Maxwell's problem (so-called *Perfectly Matched Layer (PML)*) and was subsequently implemented for the Helmholtz equation by Liao and McMechan [43] and Kim [38, 39]. From a numerical point of view, this property can help in reducing the number of iterations necessary to reach a solution [38, 39].

Even though it is tempting to implement the high order radiation condition at infinity, the choice of radiation condition is not so straightforward. Several numerical issues should be taken into account before choosing the radiation condition. These include the type of discretization on the boundaries, the operation costs due to number of points involved, or the scenario for solving the resulting linear system. For the latter, if we apply the separation of variables method as a preconditioner, the use of the first-order radiation condition is most attractive, since the separation of variables approach can not be implemented if high order radiation conditions are imposed (see Plessix and Mulder [52])

In the scattering situation by obstacles inside the region Ω, additional boundary conditions representing the presence of the obstacles should be added. Consider an impenetrable obstacle D in Ω with the boundary $\Gamma_s = \partial D$. We distinguish two types of boundary condition commonly used for scattering problems. In the case of *sound-soft* obstacles, the velocity of the total wave (i.e. the sum of incoming wave ϕ^i and scattered wave ϕ^s) vanishes at the boundary Γ_s, i.e. $\phi = 0$ on Γ_s. This condition is known as the *Dirichlet* boundary condition. For the case of *sound-hard* obstacles, the condition on Γ_s leads to a *Neumann* boundary condition $\partial\phi/\partial\nu = 0$ on Γ_s, imposing a condition of a vanishing normal velocity on the scatterer surface. A combination of both boundary conditions with at least one radiation condition gives a unique solution. For the proof, see for instance [13, 14, 53].

2.3 Helmholtz's problem

We now define our problem as follows. Let $\Omega \in R^3$ be the region enclosing the medium with scatterers and boundary $\Gamma = \partial\Omega$. Further, let ν denote the unit outward normal to Γ. We introduce the Helmholtz operator $\mathcal{L} \equiv \Delta + k^2$, where $k = k(x, \omega)$. Then, the Helmholtz problem for wave propagation in an inhomogeneous medium can be defined as follows:

Find the total field p such that

$$\begin{aligned} \mathcal{L}\, p(x) &= f(x, \omega) && \text{in } \Omega, \\ \left(\frac{\partial}{\partial \nu} - \mathrm{i}k\right) p(x) &= 0 && \text{on } \Gamma = \partial\Omega. \end{aligned} \tag{30}$$

In (2.28), f is a given source term.

For the case of impenetrable obstacles D with the boundary $\Gamma_s = \partial D$ in Ω, the problem can be defined as follows:

Find the total field p such that

$$\begin{aligned} &\mathcal{L}\, p(x) = f(x, \omega) && \text{in } \Omega, \\ &\left(\frac{\partial}{\partial \nu} - \mathrm{i}k(x, \omega)\right) p(x) = 0 && \text{on } \Gamma = \partial\Omega, \\ &\textit{Either Neumann or Dirichlet b.c.} && \text{on } \Gamma_s. \end{aligned} \tag{31}$$

Problems (2.28) and (2.29) become indefinite for high wavenumbers, having both negative and positive real eigenvalues. For the Dirichlet problem, Singer and Turkel [56] show that the largest eigenvalue of the continuous problem (2.29) is $\lambda = k^2 - 2$. So, the problem becomes indefinite for $k > \sqrt{2}$. For the Neumann problem, the largest eigenvalue is $\lambda = k^2 - 5/4$. Therefore, if $k > \sqrt{5}/2$ the problem becomes indefinite.

Problems (2.28) and (2.29) can be solved by an integral equation method by transformation into a Fredholm integral equation [4] (of the first or second kind). Colton and Kress [14] give in-depth discussions about the use of the integral equation in Helmholtz's problem. In the discretized form, the Fredholm integral equation results in a large full matrix which requires the inversion for resolving the solution. This is considered too expensive in many practical problems. Furthermore, for large k the solution becomes oscillatory. An asymptotic method can be used to resolve the solution.

In this research, we aim at numerical solutions of the Helmholtz problem (2.28), i.e., high frequency waves propagating in an inhomogeneous medium. The numerical solution is resolved from the linear system, resulting from a so-called "direct" discretization of the continuous Helmholtz equation. In many practical applications, the linear system produced is very large. This requires special method(s) to solve the linear system efficiently in reasonable accuracy.

3 Discretization

To obtain a suitable equation for numerical computation, discretization must be applied to (2.28). Several discretization schemes are available and can be used to solve the problem. In general, we categorize them as finite difference and finite element discretizations. In this section, we discuss the finite difference discretization.

3.1 Second order accurate finite difference

Let us consider again problem (2.28). The 3-D domain of interest Ω with boundary $\Gamma = \partial\Omega$ is discretized on an equidistant grid with L points in x, M points in y, and N points in z, i.e.,

$$\begin{aligned} x_l &= x_0 + l\Delta x, & l &= 0, 1, \cdots, L \\ y_m &= y_0 + m\Delta y, & m &= 0, 1, \cdots, M \\ z_n &= z_0 + n\Delta z, & n &= 0, 1, \cdots, N. \end{aligned} \tag{32}$$

We use a standard finite difference stencil:

$$D_{xx}p(x_l) = \frac{p_{l+1} - 2p_l + p_{l-1}}{\Delta x^2} \tag{33}$$

derived from Taylor's expansion to approximate the second order derivative in the first equation of (2.28) at x_l. For sufficiently smooth p in Ω, we can express the second order derivative as

$$\partial_{xx}p_l = D_{xx}p_l + O(\Delta x^2), \tag{34}$$

which means second order accuracy in space. This formulation is called here the *point-wise representation.*

In the interior points, discretization results in the following equation, for the 3-D case:

$$\begin{aligned} &\frac{p_{l+1,m,n} - 2p_{l,m,n} + p_{l-1,m,n}}{\Delta x^2} + \frac{p_{l,m+1,n} - 2p_{l,m,n} + p_{l,m-1,n}}{\Delta y^2} + \\ &+ \frac{p_{l,m,n+1} - 2p_{l,m,n} + p_{l,m,n-1}}{\Delta z^2} + k^2_{l,m,n}p_{l,m,n} = f_{l,m,n}. \end{aligned} \tag{35}$$

After applying (3.4) to all points in the interior of Ω, one obtains a linear system

$$\mathbf{A}\mathbf{p} = \mathbf{f}, \tag{36}$$

where $\mathbf{A}$ is $(LMN) \times (LMN)$ matrix having seven non-zero diagonals. The non-zero diagonal elements are given by

$$\begin{aligned} A(d,d) &= -\left(\frac{2}{\Delta x^2} + \frac{2}{\Delta y^2} + \frac{2}{\Delta z^2}\right) + k^2_{l,m,n} + \gamma_{bc}, \\ A(d,d+1) &= A(d-1,d) = \frac{1}{\Delta z^2}, \\ A(d,d+L) &= A(d-L,d) = \frac{1}{\Delta y^2}, \\ A(d,d+L\times M) &= A(d-L\times M,d) = \frac{1}{\Delta x^2}, \end{aligned} \tag{37}$$

and

$$
\begin{aligned}
p(d) &= p_{l,m,n}, \\
f(d) &= f_{l,m,n}.
\end{aligned}
\tag{38}
$$

In (3.6), γ_{bc} is determined by the position of the points in the region. For interior points, $\gamma_{bc} = 0$. For points at the boundary, the expression depends on the type of boundary discretization. For the first-order radiation condition, the following expression for γ_{bc} can be used, [52]

$$
\gamma_{bc} = \begin{cases}
\frac{p(x_1,y_m,z_n)}{1+\mathrm{i}k(x_0,y_m,z_n)}, & \text{if}\, l = 0; \\
\frac{p(x_{L-1},y_m,z_n)}{1+\mathrm{i}k(x_L,y_m,z_n)}, & \text{if}\, l = L; \\
\frac{p(x_l,y_1,z_n)}{1+\mathrm{i}k(x_l,y_0,z_n)}, & \text{if}\, m = 0; \\
\frac{p(x_l,y_{M-1},z_n)}{1+\mathrm{i}k(x_l,y_M,z_n)}, & \text{if}\, m = M; \\
\frac{p(x_l,y_m,z_1)}{1+\mathrm{i}k(x_l,y_m,z_0)}, & \text{if}\, n = 0; \\
\frac{p(x_l,y_m,z_{N-1})}{1+\mathrm{i}k(x_l,y_m,z_N)}, & \text{if}\, n = N.
\end{cases}
\tag{39}
$$

The total number of nonzero elements of $\mathbf{A}$ is $7LMN$ of $(LMN)^2$ total entries, indicating sparsity.

3.2 Fourth order accurate finite difference

A more accurate approximation for the second order derivative can be constructed to (2.28). For instance, the following fourth-order accurate stencil can be applied to discretize (2.28):

$$
D_{xx}p = \partial_{xx}p + \frac{1}{12}\Delta x^2 \partial_{xxxx}p + O(\Delta x^4). \tag{40}
$$

This expression can be obtained if we keep the fourth-order term in Taylor's expansion and replace the values p_{l+1}, p_l, p_{l-1} with (3.2). Because of (3.3) we find

$$
\partial_{xxxx}p = D_{xx}\left(\partial_{xx}p\right) + O(\Delta x^2). \tag{41}
$$

Substituting (3.10) into (3.9) we get

$$
\partial_{xx}p = \left(1 + \frac{1}{12}\Delta x^2 D_{xx}\right)^{-1} D_{xx}\partial_{xx}p + O(\Delta x^4). \tag{42}
$$

Inserting (3.11) to the first equation of (2.28) we find, for the 3-D case,

$$
\left(\frac{D_{xx}}{1+\frac{1}{12}\Delta x^2 D_{xx}} + \frac{D_{yy}}{1+\frac{1}{12}\Delta y^2 D_{yy}} + \frac{D_{zz}}{1+\frac{1}{12}\Delta z^2 D_{zz}} + k^2\right)p = f, \tag{43}
$$

or

$$
(D_1 + D_2 + D_3)p + (1 + K_1 + K_2 + K_3)k^2 p = (1 + K_1 + K_2 + K_3)f, \tag{44}
$$

where

$$
\begin{aligned}
D_1 =& D_{xx} + D_{yy} + D_{zz}, \\
D_2 =& \frac{1}{12}((\Delta x^2 + \Delta y^2) D_{xx} D_{yy} + (\Delta x^2 + \Delta z^2) D_{xx} D_{zz} + \\
& + (\Delta y^2 + \Delta z^2) D_{yy} D_{zz}), \\
D_3 =& \frac{1}{144} \left(\Delta x^2 \Delta y^2 + \Delta x^2 \Delta y^2 + \Delta y^2 \Delta z^2 \right) D_{xx} D_{yy} D_{zz}, \\
K_1 =& \frac{1}{12} \left(\Delta x^2 D_{xx} + \Delta y^2 D_{yy} + \Delta z^2 D_{zz} \right), \\
K_2 =& \frac{1}{144} \left(\Delta x^2 \Delta y^2 D_{xx} D_{yy} + \Delta x^2 \Delta z^2 D_{xx} D_{zz} + \Delta y^2 \Delta z^2 D_{yy} D_{zz} \right), \\
K_3 =& \frac{1}{1728} \Delta x^2 \Delta y^2 \Delta z^2 D_{xx} D_{yy} D_{zz}.
\end{aligned}
\tag{45}
$$

This approximation is known as the *Padé approximation* (see Singer and Turkel [56], Harari and Turkel [34]).

Another type of fourth-order accurate stencil for discretizing the Helmholtz equation can be derived by differentiating the Helmholtz equation twice in space. This can be justified since the solution p satisfies the Helmholtz equation. By doing this, we get, for the 1-dimensional Helmholtz equation,

$$
\partial_{xxxx} p + k^2 \partial_{xx} p = \partial_{xx} f. \tag{46}
$$

Inserting (3.15) for the fourth-order derivative term in (3.9), we find

$$
D_{xx} p = \left(1 - \frac{k^2 \Delta x^2}{12} \right) \partial_{xx} p + \frac{\Delta x^2}{12} \partial_{xx} f + O(\Delta x^2). \tag{47}
$$

If we further estimate $\partial_{xx} f$ using second-order approximation (3.3), expression for the second-order derivative estimate can be obtained, i.e.,

$$
\partial_{xx} p = \left(1 - \frac{k^2 \Delta x^2}{12} D_{xx} \right)^{-1} D_{xx} \left(p - \frac{\Delta x^2}{12} f \right) + O(\Delta x^4). \tag{48}
$$

This stencil is also fourth-order accurate. Using this stencil for 3-D cases, the discretized Helmholtz equation reads

$$
(\tilde{D}_1 + \tilde{D}_2 + \tilde{D}_3) p + (1 + \tilde{K}_1 + \tilde{K}_2 + \tilde{K}_3) k^2 p = (1 + \tilde{F}_1 + \tilde{F}_2 + \tilde{F}_3) f, \tag{49}
$$

where

$$\begin{aligned}
\tilde{D}_1 =& D_{xx} + D_{yy} + D_{zz},\\
\tilde{D}_2 =& \frac{k^2}{12}((\Delta x^2 + \Delta y^2)D_{xx}D_{yy} + (\Delta x^2 + \Delta z^2)D_{xx}D_{zz} +\\
&+ (\Delta y^2 + \Delta z^2)D_{yy}D_{zz}),\\
\tilde{D}_3 =& \frac{k^4}{144}\Delta x^2\Delta y^2\Delta z^2 D_{xx}D_{yy}D_{zz},\\
\tilde{K}_1 =& -\frac{k^2}{12}(\Delta x^2 D_{xx} + \Delta y^2 D_{yy} + \Delta z^2 D_{zz}),\\
\tilde{K}_2 =& \frac{k^4}{144}(\Delta x^2\Delta y^2 D_{xx}D_{yy} + \Delta x^2\Delta z^2 D_{xx}D_{zz} + \Delta y^2\Delta z^2 D_{yy}D_{zz}),\\
\tilde{K}_3 =& -\frac{k^6}{1728}\Delta x^2\Delta y^2\Delta z^2 D_{xx}D_{yy}D_{zz},\\
\tilde{F}_1 =& \frac{1-k^2}{12}(\Delta x^2 D_{xx} + \Delta y^2 D_{yy} + \Delta z^2 D_{zz}),\\
\tilde{F}_2 =& \frac{k^2(2-k^2)}{144}(\Delta x^2\Delta y^2 D_{xx}D_{yy} + \Delta x^2\Delta z^2 D_{xx}D_{zz} +\\
&+ \Delta y^2\Delta z^2 D_{yy}D_{zz}),\\
\tilde{F}_3 =& \frac{k^4(3-k^2)}{1728}\Delta x^2\Delta y^2\Delta z^2 D_{xx}D_{yy}D_{zz}.
\end{aligned} \tag{50}$$

In both 3D fourth-order discretizations, at most 27 points are required in the stencil. The linear system obtained from these discretizations has 27 nonzero diagonals, giving the total nonzero entries of $27LMN$.

3.3 Second-order versus fourth-order accurate stencil

It seems natural to expect more a accurate solution from a higher order discretization for a given mesh size. However for Helmholtz problems, this is not always the case.

Singer and Turkel [56] make some comparison tests on second-order and fourth-order accurate stencils for solving the 2-D Helmholtz equation. The tests were performed for different values of k. Numerical experiments were performed with given source term f that allows the exact solution to be obtained analytically. The numerical solutions were then compared to the exact solution. The results show that for a low value of k, fourth-order accurate stencils give better accuracy than the second-order one. However, whenever k is sufficiently large, the second-order accurate scheme becomes more accurate compared to the fourth-order on coarse grids. The fourth-order accurate schemes gain more accuracy if a smaller mesh size is taken. Theoretically, for a second-order stencil we can estimate that the necessary mesh size in each direction is proportional to $1/k$ with ten mesh points needed to resolve one wavelength. In 3-D cases, approximately $O((10k)^3)$ mesh points are needed to resolve the solution [19]. For large k, if fourth-order stencils are used the order may be larger than this estimate, and the size of the linear system can be very large for a sufficiently accurate solution. From this view, the second-order stencil may become more attractive. (Furthermore, the number of nonzero diagonals is smaller for the second-order stencil than the fourth-order ones, implying that storage issues for the first stencil may not be as critical as for the latter.

In problem (2.28), no restriction is made with respect to k in deriving the Helmholtz equation. In general, k may be complex. If $k \in \mathbb{C}$, the waves are damped as they travel away from the source. This is the natural case. In constrast, the radiation condition in Chapter 2 was derived under restriction that $k > 0$ is real. If we stick to this assumption (which is correct for approximation), discretization in the interior of Ω leads to a real valued linear system. However, imposing the radiation condition on the boundaries results in a complex valued $\mathbf{A}$ in the resulting linear system. Matrix $\mathbf{A}$ of (3.5) is therefore complex and symmetric discretized using either second- or fourth-order stencils. The only contribution of complex values is from the boundary condition. For high values of k, the linear system becomes indefinite (see Plessix and Mulder [52], and Elman and O'Leary [18]). We call a linear system to be (positive) definite if $\mathbf{x}^T\mathbf{A}\mathbf{x} > 0$ for $\mathbf{x} \neq 0$. For definite matrices, many nice algorithms to solve the linear system are available. However, these nice algorithms typically fail to work for the indefinite linear system or are very slow to converge. This is why in general no algorithms are available to solve the Helmholtz equation with unrestricted value of k with a similar efficiency as for the symmetric possitive definite (SPD) cases.

In [56] an eigenvalues analysis for the second- and fourth-order discretized Helmholtz equation is derived. Since $\mathbf{A}$ is large and sparse, solution methods which exploit sparsity of $\mathbf{A}$ are desirable. Standard direct methods may become inattractive.

4 Direct methods for solving discrete Helmholtz equations

In general practice, one can solve linear system (3.5) with coefficient matrix **A** by implementing either direct or iterative methods. In direct methods, one tries to solve a linear system in the spirit of the Gauss elimination. Since the standard Gauss elimination is extremely expensive for solving very large, sparse matrices, it becomes prohibitive to directly use such a procedure. However, in the case of sparse matrices modifications to the standard Gaussian elimination can lead to an efficient algorithm and also a reduction of the required storage (at least for the 2-D cases).

In this section, we will first discuss the basics of Gauss elimination for solving a linear system and the equivalent formulation of the coefficient matrix **A** in a simpler-to-solve form, the so-called *Cholesky* factor. Since we are concerned with the sparsity of the matrix **A** some additional algorithms will also be revisited, which are in practice often used. These include numbering scenarios (reordering) and storage issues. We refer to [28] and the references therein for a more detailed discussion.

4.1 Gauss elimination

Consider a general linear system

$$\mathbf{A}\mathbf{x} = \mathbf{b}. \tag{51}$$

Solutions can be obtained by implementing the Gauss elimination algorithm, i.e. by reducing **A** into an upper triangular matrix followed by back substitution. The basic Gaussian elimination algorithm is explained below.

Algorithm 4.1 Gauss elimination

```
1. For $k = 1, 2, \cdots, n-1$ do
2.   if $a_{kk} = 0$ pivot
3.   for $i = k+1, \cdots, n$ do
4.     $\alpha = a_{ik}/a_{kk}$
5.     For $j = k, \cdots, n$ do
6.       $a_{ij} = a_{ij} - \alpha a_{kj}$
7.     enddo
8.   enddo
9. enddo
```

Algorithm 4.2 explains backward substitution to find the solution x from the upper triangular matrix resulted from the algorithm 4.1.

Algorithm 4.2 Backward substitution

```
1. For $i = n, n-1, \cdots, 1$ do
2.    $x_i = b_i$
3.    For $j = i+1, \cdots, n$ do
4.       $x_i = x_i - a_{ij} b_j$
5.    enddo
6. enddo
```

The algorithm is exactly terminated after a finite number of steps and if the matrix $\mathbf{A}$ is invertible the solution exists. Assuming the matrix $\mathbf{A}$ is full, the number of storage needed to store the matrix $\mathbf{A}_{n\times n}$ and $\mathbf{b}_n$ are $n(n+1)$. The operation counts required to obtain the solution is of order of $2n^3/3$.

A more efficient way to solve (4.1) can be designed for a symmetric positive definite (SPD) matrix $\mathbf{A}$, i.e., if $\mathbf{A}$ satisfies

$$\begin{aligned} a_{ij} &= a_{ji}, \ i, j = 1, 2, \cdots, N \\ \mathbf{x}^T\mathbf{A}\mathbf{x} &> 0, \text{ for any } \mathbf{x} \neq \mathbf{0}. \end{aligned} \tag{52}$$

For such a matrix, the following theorem is always satisfied.

Theorem 4.1
If $\mathbf{A}$ is SPD then it has a unique triangular factorization $\mathbf{L}\mathbf{L}^T$, where $\mathbf{L}$ is the lower triangular matrix with positive diagonal entries.

The factorization $\mathbf{A} = \mathbf{L}\mathbf{L}^T$ is called the Cholesky factorization. Substituting the Cholesky factorization into (4.1), the following linear systems can be solved sequentially to obtain the solution $\mathbf{x}$:

$$\begin{aligned} \mathbf{L}\mathbf{y} &= \mathbf{b}, \\ \mathbf{L}^T\mathbf{x} &= \mathbf{y}. \end{aligned} \tag{53}$$

Both linear systems are easier to solve than the original one, e.g., by applying forward and backward substitution to the first and second equation respectively. Using either forward or backward substitution requires operation counts of order n^2. The following describes the Cholesky factorization of an SPD matrix $\mathbf{A}$ [28], the so-called *bordering method.* Suppose $\mathbf{A}$ is partitioned as

$$\mathbf{A} = \begin{pmatrix} \mathbf{M} & \mathbf{u} \\ \mathbf{u}^T & s \end{pmatrix}. \tag{54}$$

The Cholesky factorization of $\mathbf{A}$ is given as

$$\mathbf{A} = \begin{pmatrix} \mathbf{L}_M & \mathbf{0} \\ \mathbf{w} & t \end{pmatrix} \begin{pmatrix} \mathbf{L}_M^T & \mathbf{w} \\ \mathbf{0} & t \end{pmatrix}, \tag{55}$$

where

$$\mathbf{w} = \mathbf{L}_M^{-1}\mathbf{u}, \quad t = (s - \mathbf{w}^T\mathbf{w})^{1/2}. \tag{56}$$

Alternate algorithms for finding the Cholesky factor are the *outer product* form and the *inner product* form; see [28].

4.2 Matrix reordering

Since discretization of the Helmholtz equation results in a coefficient matrix **A** which is sparse, it becomes important to maintain this sparsity. Unfortunately, a direct use of the Cholesky factorization typically leads to fill-in, i.e. adding a number of nonzeros in the factor **L**.

This is something one should avoid since such fill-in increases the storage requirements and the number of operations. By introducing permutations, the fill-in can be kept minimum after factorization. If we introduce a permutation matrix **P** to the linear system (4.1) so that

$$\mathbf{P}\mathbf{A}\mathbf{P}^T\mathbf{P}\mathbf{x} = \mathbf{P}\mathbf{b}, \tag{57}$$

we factorize the permuted system to obtain the following system

$$\tilde{\mathbf{L}}\tilde{\mathbf{L}}^T\tilde{\mathbf{x}} = \tilde{\mathbf{b}}. \tag{58}$$

It is practically impossible to find permutations which are favorable for all sparse matrice cases. Introducing permutations, however, can be regarded as introducing a reordering of the matrix. From this point of view, one can set heuristic reordering methods without having to find specific permutation matrices.

Standard reordering techniques are the *band* and the *envelope* reordering. In the band reordering, the matrix elements are reordered in a way that the off-diagonal elements are positioned as close as possible to the diagonal. This reordering is easy to implement and sometimes efficient. However the band reordering exploits only the sparsity of the matrix outside the bandwidth. If the matrix has a very wide bandwidth, the band method becomes inefficient.

An alternative is the envelope method which also exploits sparsity inside the band. Falling into this category is the Reverse Cuthill-McKee (RCM) method. Following the idea of Cuthill-McKee reordering, if a *labelled* y and an *unlabelled* z are a pair of connected nodes, the node z should be reordered directly after y reordering to minimize the bandwidth of the matrix. In the RCM method, renumbering is done by reversing the Cuthill-McKee reordering. This step is easily invoked by the statement $y_i = x_{n-i+1}$, $i = 1, 2, \cdots, n$. The RCM method is found to be always superior over the original Cuthill-McKee even though it never reduces the bandwidth of the matrix.

The envelope method usually becomes even more superior if a starting node (or *root*) is appropriately chosen. One defines *distance* $d(V, y)$ as the length of a shortest path joining two nodes x and y in the graph $G^A(V^A, E^A)$ of the matrix **A**. The appropriate starting node is often the one with maximum distance. Therefore, problems of finding the starting node can be viewed as finding a pair of nodes having maximum distance. The *rooted level structure* can be used to find a so-called pseudo-peripheral node [28].

4.3 Minimum degree (MD)

It has been noted earlier that it is not trivial to obtain an appropriate permutation matrix **A**. In order to minimize fill-in, a heuristic strategy which is based on the elimination of the given graph can be designed. The first type of ordering within this category is the so-called *minimum degree algorithm* (MD). The first description of MD and its use is due to Tinney and Walker [61]. In the MD, ordering is done by first selecting the vertex in the graph G which has the smallest *degree*. (Degree of the vertex x_i, $deg(i)$ is defined as the number of neighbors). This vertex is numbered next in the ordering and then eliminated from the graph G.

The MD algorithm is described below for a given unlabelled graph $G_0 = (V, E)$; see [28]. In line 2, the search for a minimum-degree vertex is performed. This vertex is then eliminated to form the elimination graph.

Algorithm 4.3 Minimum degree algorithm

```
Let $G_0 = (V, E)$ be an unlabelled graph.
1. Initialization: $i \leftarrow 1$
2. Minimum degree selection: In $G_{i-1} = (V_{i-1}, E_{i-1})$, choose a node $x_i$ of minimum degree in $G_{i-1}$
3. Graph transformation: Form new elimination $G_{i-1} = (V_{i-1}, E_{i-1})$, deleting node $x_i$ from $G_{i-1}$
4. Loop or stop: $i \leftarrow i + 1$. If $i > |V|$, stop. Else, go to 2.
```

The $|V|$ in Algorithm 4.3 indicates the size of "V". In this basic algorithm, an elimination graph is constructed on each loop and the nodes are renumbered. This process is very costly and makes the algorithm inefficient. It is apparent that in minimum-degree-type reorderings, the most expensive operation takes part in the update step [44]. Improvements are proposed by interpreting the construction of the elimination graph from a *reachable set* point of view. Representing in this set a modified algorithm can be stated which is only requiring information of the original graph. The degree then requires update only in the adjacent nodes of the eliminated one. It is also possible to improve the algorithm by eliminating two or more nodes having the same reachable set with respect to the elimination (or *indistinguishable* nodes).

One of the most popular algorithms falling into the MD-type ordering is the *multiple* MD reordering (or MMD) due to Liu [44]. In order to reduce operations in the degree update step, updating is only performed after several node eliminations. This process results in a different minimum reordering. However, it is very effective in reducing operations and in many cases it gives better reordering quality (less nonzero elements) than the standard MD algorithm. Another popular variant is the *Approximate* MD ordering (or AMD) after Davis, Amestoy, Duff [15]. As an improvement to the standard MD ordering, the AMD offers faster ordering and better ordering quality. However, no significant improvements are indicated from the results compared to the MMD ordering in terms of fill-in.

4.4 Nested dissection (ND)

Since we want to exploit the sparsity of matrix $\mathbf{A}$, it is expected that the number of nonzero elements is preserved after reordering. If $\mathbf{A}$ is permuted using a permutation matrix $\mathbf{P}$, the nonzero structure of $\mathbf{A}$ and $\mathbf{PAP^T}$ may differ after permutation. The ideal problem of minimum reordering is given as follows.

$$|\mathrm{nz}(\mathbf{F}(\mathbf{P}^*\mathbf{A}\mathbf{P}^{*\mathbf{T}}))| = \min_{\mathbf{P}} \quad |\mathrm{nz}(\mathbf{F}(\mathbf{PAP^T}))| \tag{59}$$

where $\mathbf{F}(\mathbf{X})$ indicates the fill-in of $\mathbf{A}$. In Eq. (4.9), $\mathrm{nz}(\mathbf{A})$ means the number of *nonzero elements* of matrix $\mathbf{A}$. However, nonzero preservation is hard to fulfill during reordering. Rather than trying to preserve the nonzero size, one applies a weaker constraint, i.e. to find a permutation $\mathbf{P}^*$ which gives an acceptable small number of nonzero elements.

For matrix $\mathbf{A}$ symmetric positive definite (SPD), the *nested dissection* method, see e.g. George and Liu [28], can be used. This method offers advantages in comparison with the minimum degree algorithm, in terms of speed and storage requirement. This method also attemp to minimize fill-ins which is usually suffered by many direct solution methods.

The idea underlying the nested dissection method is explained as follows. Let G^A be the undirected graph of symmetric coefficient matrix $\mathbf{A}$. We consider a separator S in G^A which disconnects the graph into two parts whose node sets are C^1 and C^2. The nodes in separator S are numbered following those in C^1 and C^2, introducing a partioning of the ordered matrix. Since we try to preserve as much as possible nonzero elements, the separator plays an important role during ordering; a proper choice of separator may preserve zero elements in matrix $\mathbf{A}$. A similar process of ordering can be done recursively to the submatrices by choosing another sets of vertices

$$S^j \subset R^j, j = 1, 2. \tag{60}$$

The process is repeated until the components could not be dissected further.

The key feature in the ND method is finding a separator which disconnects a given grids into two pieces with approximately equal size. Several methods have been proposed for finding the separator; see [28].

4.5 Direct methods for the discrete Helmholtz equation

Plessix, Mulder, and Pratt [51] and Plessix and Mulder [52] note that the nested dissection method can be applied to solve the linear system at efficient cost. However, such application is restricted to 2-D cases. A case of 1000×1000 grid points has been tested on a workstation. In 3-D cases, problems arise in reordering because the amount of fill-in is too large.

It should be noted that the minimum degree and nested dissection method are derived mainly for symmetric positive definite matrices, for which the Cholesky factorization exists and is unique. For the Helmholtz equation, however, this necessary condition is hardly satisfied except for very low frequency (or small wavenumber cases). Unfortunately, applications are typically defined beyond this low frequency.

For high aspect ratio grids, often the MD and ND reordering do not consistently produce good quality orderings. Ashcraft and Liu [1] show that the good performance of ND ordering degrades in the case of high aspect ratio grids. In such a case, the multisection reordering can be implemented, showing performance consistency (and therefore robustness) in high aspect ratio grids.

A combination method is proposed by Hendrickson and Rothberg [35] to improve the operation and quality of nested dissection orderings. The method is a hybrid of the ND and MD algorithm. Starting with constructing a compressed graph, the incomplete nested dissection is applied, i.e. the ND algorithm is used until fairly small sections have been obtained. To find separators, a direct multilevel vertex separator approach is used. Local improvement to the separator is done using a vertex Fiduccia-Mattheyses graph partitioning. Following this, the MD is invoked on the small sections, improving the ordering obtained from the ND. Implementation results show, in general, a gain of 10% and 39% reduction of operations during ordering can be obtained compared to the ND and MD alone respectively. Further, the nonzeros reduces about 10% compared to the ND and 17% compared to the MD alone. However, for 3-D cases, it is expected that the cost to solve the linear system remains unacceptably large.

5 Iterative methods for solving discrete Helmholtz equation

It has been cited in the previous section that application of the nested dissection method on Helmholtz problem is restricted to the 2-D case. The problem lies in the fact that reordering leads to fill-in, reducing the method's efficiency when implemented in 3-D.

Solution methods with an acceptable efficiency can still be pursued by implementing iterative methods for $\mathbf{A}\mathbf{x} = \mathbf{b}$. Recently, several iterative methods have been developed. The underlying concept in deriving iteration methods is the so-called *Krylov subspace* method. In this section, we will first describe some iterative methods relevant to problems at hand. Since we aim at the numerical solution of a complex symmetric, non Hermitian, and (highly) indefinite linear system, we will consider iterative methods feasible for this linear system. We can list, e.g., Conjugate Gradient (CG), CGNR, Generalized Minimal Residual (GMRES), BiCGSTAB, COCG, Quasi Minimum Residual (QMR) as the candidates. We include also the conjugate gradient algorithm since, eventhough it is seemly not applicable to our problem, this algorithm is of importance as a basis for deriving several iterative methods. In the early research on the numerical solutions of Helmholtz's equation, the conjugate gradient method gained popularity due to its better efficieny in comparison with direct methods. However, modifications should be done to the standard conjugate gradient method as, for instance, proposed Bayliss, Goldstein, and Turkel [4].

In practice, standard iterative methods are not sufficiently efficient for solving a sparse, large linear problem without a modification of the linear system. It is known that in order to obtain a very efficient algorithm, the linear system should be transformed into a formulation which is similar and, therefore, gives the same solution but is much simpler to solve. This process is called *preconditioning*. Without preconditioning, iterative methods are not so attractive. We will deal with preconditioners in the next section.

For simplicity and to avoid ambiguity, we redefine the linear system (3.5) with new variables. All related variables with (3.5) are no longer used throughout this section. Let us consider the linear system

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \quad \mathbf{A} \in \mathbb{C}^{n\times n}, \tag{61}$$

where $\mathbf{A}$ is an $n \times n$ matrix. The matrix $\mathbf{A}$ is sparse and assumed in general to be complex. We intend to solve (5.1) iteratively.

For general discussions, we refer to [2, 9, 32, 33, 54].

5.1 Krylov subspace method

The iterative methods explained in the subsequent subsections are basically developed from constructing consecutive iterants on Krylov subspaces, i.e., subspaces of the form

$$\mathcal{K}^j(\mathbf{A}, \mathbf{r}_0) = \text{span}\{\mathbf{r}_0, \mathbf{A}\mathbf{r}_0, \mathbf{A}^2\mathbf{r}_0, \cdots, \mathbf{A}^{j-1}\mathbf{r}_0\}. \tag{62}$$

The dimension of $\mathcal{K}^j$ is equal to j and increases by one at each step of the approximation process.

The idea of Krylov subspace methods can be outlined as follows. For an initial solution $\mathbf{x}_0$, approximations to the solution $\mathbf{x}$ are computed every step by iterants $\mathbf{x}_j$ of the form

$$\mathbf{x}_j \in \mathbf{x}_0 + \mathcal{K}^j(\mathbf{A}, \mathbf{r}_0), \; j > 1. \tag{63}$$

Krylov subspaces $\mathcal{K}^j$ are constructed by basis $\mathbf{v}_1, \cdots, \mathbf{v}_j$ where

$$\mathbf{V}_j = [\mathbf{v}_1, \cdots, \mathbf{v}_j] \in \mathcal{K}^j. \tag{64}$$

Defining residual $\mathbf{r}_j = \mathbf{b} - \mathbf{A}\mathbf{x}_j$, application of condition (5.2) gives an expression for the residual at j-step,

$$\mathbf{r}_j = \mathbf{r}_0 - \mathbf{A}\mathbf{V}_j\mathbf{y}_j, \tag{65}$$

where $\mathbf{y}_j \in \mathbb{C}^j$ and $\mathbf{x}_j = \mathbf{x}_0 + \mathbf{V}_j\mathbf{y}_j$. From (5.5) we can observe that Krylov subspace methods rely on constructing basis a $\mathbf{V}_j$ and a vector $\mathbf{y}_j$. In general, we identify two general methods can be used for constructing basis the $\mathbf{V}_j$: Arnoldi's method and Lanczos' method. $\mathbf{y}_j$ can be constructed by the residual projection or the residual norm minimization method.

5.2 Conjugate Gradient (CG) method

We consider the matrix $\mathbf{A}$ in $\mathbf{A}\mathbf{x} = \mathbf{b}$ to be a symmetric and positive definite (SPD) matrix. In the CG, we want to construct a vector $\mathbf{x}_j \in \mathcal{K}^j(\mathbf{A}, \mathbf{r}_0)$ such that $\|\mathbf{x} - \mathbf{x}_j\|_A$ is minimal. For this purpose, the vector $\mathbf{x}_{j+1}$ is expressed as

$$\mathbf{x}_{j+1} = \mathbf{x}_j + \alpha_j\mathbf{p}_j, \tag{66}$$

where $\mathbf{p}_j$ is the search direction. The residual vectors satisfy the recurrence

$$\mathbf{r}_{j+1} = \mathbf{r}_j + \alpha_j\mathbf{A}\mathbf{p}_j. \tag{67}$$

For $\mathbf{r}_j$'s to be orthogonal, it is necessary that $(\mathbf{r}_j - \alpha_j\mathbf{A}\mathbf{p}_j, \mathbf{r}_j) = 0$, resulting in

$$\alpha_j = \frac{(\mathbf{r}_j, \mathbf{r}_j)}{(\mathbf{A}\mathbf{p}_j, \mathbf{r}_j)}. \tag{68}$$

Since the next search direction $\mathbf{p}_{j+1}$ is a linear combination of $\mathbf{r}_{j+1}$ and $\mathbf{p}_j$, i.e.,

$$\mathbf{p}_{j+1} = \mathbf{r}_{j+1} + \beta_j\mathbf{p}_j, \tag{69}$$

the denominator in Eq. (5.8) can be written as $(\mathbf{A}\mathbf{p}_j, \mathbf{p}_j - \beta_{j-1}\mathbf{p}_{j-1}) = (\mathbf{A}\mathbf{p}_j, \mathbf{p}_j)$, knowing that $\mathbf{A}\mathbf{p}_j$ is orthogonal to $\mathbf{p}_{j-1}$. Because $\mathbf{p}_{j+1}$ is also orthogonal to $\mathbf{A}\mathbf{p}_j$, by making use of Eq. (5.9), β_j can be written as

$$\beta_j = \frac{(\mathbf{r}_{j+1}, \mathbf{r}_{j+1})}{(\mathbf{r}_j, \mathbf{r}_j)}. \tag{70}$$

The algorithm for CG method is given as follows.

Algorithm 5.1 Conjugate Gradient Algorithm

```
1. Compute $\mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0$, $\mathbf{p}_0 := \mathbf{r}_0$
2. For $j = 0, 1, \cdots$, *until convergence* Do:
3.     $\alpha_j := (\mathbf{r}_j, \mathbf{r}_j)/(\mathbf{A}\mathbf{p}_j, \mathbf{p}_j)$
4.     $\mathbf{x}_{j+1} := \mathbf{x}_j + \alpha_j \mathbf{p}_j$
5.     $\mathbf{r}_{j+1} := \mathbf{r}_j - \alpha_j \mathbf{A}\mathbf{p}_j$
6.     $\beta_j := (\mathbf{r}_{j+1}, \mathbf{r}_{j+1})/(\mathbf{r}_j, \mathbf{r}_j)$
7.     $\mathbf{p}_{j+1} := \mathbf{r}_{j+1} + \beta_j \mathbf{p}_j$
8. Enddo
```

CG algorithm works well for an SPD matrix with real coefficients. With a suitable choice of preconditioner, the algorithm runs very efficiently. The CG method encounters problems when $\mathbf{A}$ is indefinite, i.e., when matrix $\mathbf{A}$ has real positive and negative eigenvalues. Paige and Saunders [49] show that if definiteness is not guaranteed, the matrix $\mathbf{T}_j = \mathbf{V}_j^T \mathbf{A} \mathbf{V}_j$ is possibly singular or near-singular. Since $\mathbf{T}_j$ is required to determine x_j (i.e. $\mathbf{x}_j = \mathbf{V}_j \mathbf{T}_j \beta e_1$), at each sequence x_j is poorly determined.

To overcome the indefiniteness of $\mathbf{A}$, and if $\mathbf{A}$ is nonsymmetric, modifications are introduced to the standard CG algorithm. These lead to several variants of CG, for instance: CGS and BiCGSTAB. These variants preserve not only the nice property of the CG algorithm but also extend the methods to be able to solve non-SPD matrices. Differently, by transformation into the normal equation, the CG algorithm still works nicely to solve (5.1) as reported in [4]. We will discuss this approach in the next subsection.

5.3 CGNR

For nonsymmetric, indefinite linear systems the CGNR can be used. Consider the linear system

$$\mathbf{A}^T \mathbf{A} \mathbf{x} = \mathbf{A}^T \mathbf{b}. \tag{71}$$

Applying CG to this system we need to compute the residual $\mathbf{z}_{j+1} = \mathbf{z}_j - \alpha_j \mathbf{A}^T \mathbf{A} \mathbf{p}_j$. Actually we can compute this residual in two steps: $\mathbf{r}_{j+1} := \mathbf{r}_j - \alpha_j \mathbf{A}\mathbf{p}_j$ and $\mathbf{z}_{j+1} = \mathbf{A}^T \mathbf{r}_{j+1}$. The CGNR algorithm is given as follows, according to [54].

Algorithm 5.2 CGNR algorithm

```
1. Compute $\mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0$, $\mathbf{z}_0 = \mathbf{A}^T \mathbf{r}_0$, $\mathbf{p}_0 := \mathbf{z}_0$
2. For $j = 0, 1, \cdots$, *until convergence* Do:
3.     $\mathbf{w}_j = \mathbf{A}\mathbf{p}_j$
4.     $\alpha_j := (\mathbf{z}_j, \mathbf{z}_j)/(\mathbf{A}\mathbf{p}_j, \mathbf{A}\mathbf{p}_j)$
5.     $\mathbf{x}_{j+1} := \mathbf{x}_j + \alpha_j \mathbf{p}_j$
6.     $\mathbf{r}_{j+1} := \mathbf{r}_j - \alpha_j \mathbf{A}\mathbf{p}_j$
7.     $\mathbf{z}_{j+1} = \mathbf{A}^T \mathbf{r}_{j+1}$
8.     $\beta_j := (\mathbf{z}_{j+1}, \mathbf{z}_{j+1})/(\mathbf{z}_j, \mathbf{z}_j)$
```

```
9.     $\mathbf{p}_{j+1} := \mathbf{z}_{j+1} + \beta_j \mathbf{p}_j$
10. Enddo
```

CGNR is considered the simplest algorithm to solve nonsymmetric or indefinite linear systems. Expression (5.11) ensures that the related linear system is symmetric and definite. However, the convergence becomes slow in comparison with the original linear system due to the square of the condition number. In line 4, α can be written as $\alpha_j := (\mathbf{z}_j, \mathbf{z}_j)/(\mathbf{p}_j, \mathbf{A}^T\mathbf{A}\mathbf{p}_j)$ which is the direct result of applying CG algorithm to (5.11). However, algorithm using this formulation is less stable than the algorithm 5.2.

5.4 Generalized Minimum Residual (GMRES) method

The GMRES method is of Galerkin type and minimizes the residual norm over the Krylov subspace.

The GMRES algorithm is given as follows (see Saad [54] and Saad and Schultz [55]).

Algorithm 5.3 Generalized Minimum Residual Algorithm

```
1.  Choose $x_0$. Compute $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$, $\beta := \|\mathbf{r}_0\|_2$, and $\mathbf{v}_1 := \mathbf{r}_0/\beta$
2.  For $j = 1, 2, \cdots, m$ do:
3.      Compute $\mathbf{w}_j := \mathbf{A}\mathbf{v}_j$
4.      For $i = 1, 2, \cdots, m$ do:
5.          $h_{i,j} := (\mathbf{w}_j, \mathbf{v}_i)$, $\mathbf{w}_j := \mathbf{w}_j - h_{i,j}\mathbf{v}_i$
6.      Enddo
7.      $h_{j+1,j} = \|\mathbf{w}_j\|_2$.
8.      $\mathbf{v}_{j+1} = \mathbf{w}_j/h_{j+1,j}$
9.  Enddo
10. Compute $\mathbf{y}_m$ the minimized of $\|\beta e_1 - \bar{\mathbf{H}}_m y\|_2$ and $\mathbf{x}_m = \mathbf{x}_0 + \mathbf{V}_m\mathbf{y}_m$
```

Line 2 to 9 represent the Arnoldi algorithm for orthogonalization. In line 10, we define a minimalization process by solving a least squares problem

$$J(y) = \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2, \tag{72}$$

where $\mathbf{x} = \mathbf{x}_0 + \mathbf{V}_k\mathbf{y}$ is any vector in $\mathcal{K}^k$. Except for line 10, the GMRES algorithm is almost identical with the Full Orthogonalization Method (FOM). (In FOM, we compute $\mathbf{y}_m = \mathbf{H}_m^{-1}(\beta e_1)$, where e_1 is the first unit vector.) Inserting the expression of $\mathbf{x}$ to (5.12) and making use of the following property:

$$\mathbf{A}\mathbf{V}_k = \mathbf{V}_{k+1}\bar{\mathbf{H}}_k \tag{73}$$

(see Preposition 6.5 in [54]), we arrive at the following result:

$$J(y) = \|\beta e_1 - \bar{\mathbf{H}}_m y\|_2, \tag{74}$$

which is subjected to minimization. In (5.14), $\beta = \|\mathbf{r}_0\|$, and $e_1 \equiv \{1, 0, \cdots, 0\}^T$.

It is worth noting some properties of the GMRES algorithm. The GMRES algorithm may break down if at iteration step j, $h_{j+1,j} = 0$ (see line 8). However, this situation implies that residual vector is zero and therefore, the algorithm gives the exact solution at this step. Hence, examination of value $h_{j+1,j}$ in algorithm step 7 becomes important.

If the iteration number m is large, the GMRES algorithm becomes impractical because of memory and computational requirements. This is understandable from the fact that during the Arnoldi steps the number of vectors requiring storage increases. To remedy this problem, a restarted algorithm can be designed. The restarted GMRES follows the idea of the original GMRES except after the l-th step, the algorithm is repeated by setting $l = 0$. Below is the restarted GMRES algorithm.

Algorithm 5.4 Restarted GMRES(l) Algorithm

```
1. Choose $j_{max} = l$. Start the unrestarted GMRES algorithm
2. Restart: Compute $\mathbf{r}_l = \mathbf{b} - \mathbf{A}\mathbf{x}_l$. If satisfied then stop.
Else, compute $\mathbf{x}_0 := \mathbf{x}_l$ and $\mathbf{v}_1 := \mathbf{r}_l/\|\mathbf{r}_l\|$, go to 1.
```

However, the restarted GMRES algorithm may lead to difficulties if $\mathbf{A}$ is not positive definite [54]. For such a type of matrix, the algorithm can stagnate, which is not the case for the unrestarted GMRES algorithm since the latter is guaranteed to converge in at most N steps [55].

5.5 Biconjugate Gradient (BiCG) method

For solving non-Hermitian systems, an approach can be designed by combining the Lanczos biorthogonalization method with the Petrov-Galerkin condition. The method based on this approach is known as the Biconjugate Gradient (BiCG) method. In the algorithm, a dual linear system is solved $\mathbf{A}^T\mathbf{x}^* = \mathbf{b}^*$ together with the original $\mathbf{A}\mathbf{x} = \mathbf{b}$. If there is a dual system to solve with $\mathbf{A}^T$, then the Petrov-Galerkin condition is satisfied by scaling the initial residual $\mathbf{r}_0^* = \mathbf{b}^* - \mathbf{A}^T\mathbf{x}_0^*$. The BiCG algorithm is given as below.

Algorithm 5.5 Biconjugate Gradient algorithm

```
1.  Compute $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$, choose $\mathbf{r}_0^*$ such that $(\mathbf{r}_0, \mathbf{r}_0^*) \neq 0$
2.  Set $\mathbf{p}_0 = \mathbf{r}_0$, $\mathbf{p}_0^* = \mathbf{r}_0^*$
3.  For $j = 0, 1, \cdots$, *until convergence* Do:
4.      $\alpha_j = (\mathbf{r}_j, \mathbf{r}_j^*)/(\mathbf{A}\mathbf{p}_j, \mathbf{p}_j^*)$
5.      $\mathbf{x}_j = \mathbf{x}_j + \alpha_j \mathbf{p}_j$
6.      $\mathbf{r}_{j+1} = \mathbf{r}_j - \alpha_j \mathbf{A}\mathbf{p}_j$
7.      $\mathbf{r}_{j+1}^* = \mathbf{r}_j^* - \alpha_j \mathbf{A}^T \mathbf{p}_j^*$
8.      $\beta_j = (\mathbf{r}_{j+1}, \mathbf{r}_{j+1}^*)/(\mathbf{r}_j, \mathbf{r}_j^*)$
9.      $\mathbf{p}_{j+1} = \mathbf{r}_{j+1} + \beta_j \mathbf{p}_j$
10.     $\mathbf{p}_{j+1}^* = \mathbf{r}_{j+1}^* + \beta_j(\mathbf{u}_j + \beta_j \mathbf{p}_j^*)$
11. Enddo
```

In many cases, the algorithm may break down if the linear system determining the vector $\mathbf{y}_j$ becomes rank deficient. To overcome the breakdown, a look-a-head strategy needs to be incorporated. Also, if the system for determining $\mathbf{y}_j$ is poorly conditioned, the convergence becomes irregular. Computation involving $\mathbf{A}^T$ can also be costly if the solution for the dual system is not of interest.

5.6 Conjugate Gradient Squared (CGS) method

In [58] Sonneveld proposed a modification of the CG algorithm for handling nonsymmetric linear systems. Prior to the modification, the residual $\mathbf{r}_j$ and search direction $\mathbf{p}_j$ recursions in CG are reinterpreted in a polynomial sense as

$$\begin{aligned} \mathbf{r}_j &= \phi(\mathbf{A})\mathbf{r}_0, \\ \mathbf{p}_j &= \psi(\mathbf{A})\mathbf{p}_0. \end{aligned} \tag{75}$$

with $\phi(\mathbf{A})$ and $\psi(\mathbf{A})$ polynomials of degree less than or equal to n.

In such relations, these polynomials behave like contracting operators to $\mathbf{r}_0$ since in a sequential process, $\mathbf{r}_j \rightarrow 0$ as j increases. Using polynomial formulations in the CG algorithm $(\mathbf{r}_j, \mathbf{r}_j) = [\phi_j(\mathbf{A}), \phi_j(\mathbf{A})]$ and $(\mathbf{A}\mathbf{p}_j, \mathbf{p}_j) = [\psi_j(\mathbf{A}), \nu\psi_j(\mathbf{A})]$, where $\nu(\tau) = \tau$. During implementation, such polynomials are not involved explicitly in the algorithm.

The Conjugate Gradient-Squared (CGS) algorithm is described as follows.

Algorithm 5.5 Conjugate Gradient-Squared

```
1.  Compute $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$, $\mathbf{r}_0^*$ arbitrary
2.  Set $\mathbf{p}_0 = \mathbf{u}_0 = \mathbf{r}_0$
3.  For $j = 0, 1, \cdots$, *until convergence* Do:
4.      $\alpha_j = (\mathbf{r}_j, \mathbf{r}_0^*)/(\mathbf{A}\mathbf{p}_j, \mathbf{r}_0^*)$
5.      $\mathbf{q}_j = \mathbf{u}_j - \alpha_j \mathbf{A}\mathbf{p}_j$
6.      $\mathbf{x}_{j+1} = \mathbf{x}_j + \alpha_j(\mathbf{u}_j + \mathbf{q}_j)$
7.      $\mathbf{r}_{j+1} = \mathbf{r}_j - \alpha_j \mathbf{A}(\mathbf{u}_j + \mathbf{q}_j)$
8.      $\beta_j = (\mathbf{r}_{j+1}, \mathbf{r}_0^*)/(\mathbf{r}_j, \mathbf{r}_0^*)$
9.      $\mathbf{u}_{j+1} = \mathbf{r}_{j+1} + \beta_j \mathbf{q}_j$
10.     $\mathbf{p}_{j+1} = \mathbf{r}_{j+1} + \beta_j(\mathbf{u}_j + \beta_j \mathbf{q}_j)$
11. Enddo
```

From the algorithm, CGS avoids matrix-by-vector product with $\mathbf{A}^T$ as in the BiCG algorithm. In many cases, this causes the algorithm to converge twice as fast as BiCG [58]. However, because of squaring of polynomials, jumps in the residual errors tend to be more damaging than in the standard BiCG algorithm (see [58, 66]).

5.7 Biconjugate Gradient Stabilized (BiCGSTAB) method

In CGS, we take the square of polynomial $\phi(\mathbf{A})$ in defining the new recursion factor $\mathbf{r}$. However, this squaring may lead to build-up of rounding errors in the case of irregular convergence. To avoid irregular convergence, van der Vorst [66] uses a different expression for the residual vector. Rather than taking the square of the polynomial, another polynomial $\psi(\mathbf{A})$ is defined and the residual vector to be determined during reccursion is given as

$$\mathbf{r}_j = \psi_j(\mathbf{A})\phi_j(\mathbf{A})\mathbf{r}_0. \tag{76}$$

In (5.16), ψ is defined recursively at each step to stabilize the convergence behaviour according to the relation

$$(1 - \omega_j \mathbf{A}), \tag{77}$$

which ω_j is selected at the j-th iteration to minimize $\mathbf{r}_j$. Furthermore,

$$\phi_j(\mathbf{A})\mathbf{r}_0 = (\phi_{j-1}(\mathbf{A}) - \alpha_j \mathbf{A}\mathbf{T}_{j-1}(\mathbf{A}))\mathbf{r}_0, \tag{78}$$

and

$$\mathbf{T}_j(\mathbf{A})\mathbf{r}_0 = (\phi_j(\mathbf{A}) + \beta_{j+1}\mathbf{A}\mathbf{T}_{j-1}(\mathbf{A}))\mathbf{r}_0. \tag{79}$$

The parameters β_j and α_j are determined from the relations below:

$$\begin{aligned} \alpha_j &= \frac{\tilde{\rho}_j}{(\mathbf{A}\mathbf{p}_j, \mathbf{r}_0^*)}, \\ \beta_j &= \frac{\rho_{j+1}}{\rho_j} \times \frac{\alpha_j}{\omega_j}, \\ \rho_j &= (\hat{\mathbf{r}}_0, \mathbf{r}_{j-1}). \end{aligned} \tag{80}$$

The optimal value for ω_j is given as

$$\omega_j = \frac{(\mathbf{A}\mathbf{s}_j, \mathbf{s}_j)}{(\mathbf{A}\mathbf{s}_j, \mathbf{A}\mathbf{s}_j)}, \tag{81}$$

which optimizes the 2-norm of the vector $(\mathbf{I} - \omega_j\mathbf{A})\psi_j(\mathbf{A})\phi_{j+1}(\mathbf{A})\mathbf{r}_0$. The BiCGSTAB algorithm is described as follows.

Algorithm 5.6 BiCGSTAB algorithm

```
1. Compute $\mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0$; $\mathbf{r}_0^*$ arbitrary.
2. $\mathbf{p}_0 := \mathbf{r}_0$
3. For $j = 1, 2, \cdots$, *until convergence* Do:
4.     $\alpha_j := (\mathbf{r}_j, \mathbf{r}_0^*)/(\mathbf{A}\mathbf{p}_j, \mathbf{r}_0^*)$
5.     $\mathbf{s}_j := \mathbf{r}_j - \alpha_j \mathbf{A}\mathbf{p}_j$
6.     $\omega_j := (\mathbf{A}\mathbf{s}_j, \mathbf{s}_j)/(\mathbf{A}\mathbf{s}_j, \mathbf{A}\mathbf{s}_j)$
7.     $\mathbf{x}_{j+1} := \mathbf{x}_j + \alpha_j \mathbf{p}_j + \omega_j \mathbf{s}_j$
```

```
8.      $\mathbf{r}_{j+1} := \mathbf{s}_j - \omega_j \mathbf{A}_j \mathbf{s}_j$
9.      $\beta_j := (\mathbf{r}_{j+1}, \mathbf{r}_0^*)/(\mathbf{r}_j, \mathbf{r}_0^*) \times \alpha_j/\omega_j$
10.     $\mathbf{p}_{j+1} := \mathbf{r}_{j+1} + \beta_j(\mathbf{p}_j - \omega_j \mathbf{A}\mathbf{p}_j)$
11. Enddo
```

Investigation of the BiCGSTAB algorithm has been reported as in [66] for various applications and compared to BiCG and CGS. In general, BiCGSTAB converges more smoothly than the CGS or the BiCG. However, the convergence rate is typically the same. This becomes more significant in cases where the linear system is nonsymmetric. In some nonsymmetric cases, it is also revealed that when CGS fails to converge and shows spurious irregularity, BiCGSTAB still converges. The convergence rate is also faster than CGS or BiCG. However, for symmetric matrix cases, these significances are not clearly seen.

Even though BiCGSTAB is an attractive alternative to CGS, further investigations reveal a weakness of this algorithm. If the parameter ω becomes very close to zero during recursion, the algorithm may stagnate or break down. Numerical experiments confirm that this is likely to happen if $\mathbf{A}$ is real and has complex eigenvalues with imaginary part larger than the real part. To overcome this, improvements to BiCGSTAB have been proposed, e.g., by Gutknecht [31] and Sleijpen and Fokkema [57]. In such a case, one may expect that the situation where ω is very close to zero can be better handled by the minimum-residual polynomial of higher order. Gutknecht [31] proposed the use of second order of this polynomial in his BiCGSTAB2 algorithm. However it becomes clear from experiments that for the same situation in which BiCGSTAB breaks down, BiCGSTAB2 stagnates or also breaks down. Sleijpen and Fokkema [57] proposed a modification by forming a general l-order minimum-residual polynomial, resulting in the BiCGSTAB(l) algorithm. For $l = 1$, the algorithm resembles van der Vorst's BiCGSTAB.

Numerical experiments reveal that BiCGSTAB(l) improves BiCGSTAB in the case of stagnation or breakdown. It is known also that a higher order of l can be used to improve the convergence. In general, BiCGSTAB(l) is more superior than BiCGSTAB or BiCGSTAB2.

The original BiCGSTAB has been used, e.g., in [52] to solve the discrete Helmholtz equation.

5.8 Conjugate Orthogonal Conjugate Gradient (COCG) method

In CG we find that the solution of a linear system can be found from an iterative process using a three term recursion of the residual vectors $\mathbf{r}$. However, CG is restricted only to the Hermitian matrix, i.e. $\mathbf{A} = \mathbf{A}^H$. From Section 3, we know that discretization of the Helmholtz problem leads to a symmetric non-Hermitian matrix.

Van der Vorst and Melissen [67] observe that the nice property of three term recursion in CG still can be used to solve a symmetric non Hermitian matrix but with an additional modification to the orthogonality condition. Instead of using Hermitian orthogonality, they propose the use of conjugate orthogonality, i.e.,

$$(\overline{\mathbf{r}}_j, \mathbf{r}_k) = 0 \text{ if } j \neq k. \tag{82}$$

The sequence $\mathbf{v}_j$ is generated using the Lanczos algorithm by forcing the conjugate orthogonality condition to define α and β in recurrence $\mathbf{v}_{j+1} = \mathbf{A}\mathbf{v}_j + \alpha_j \mathbf{v}_j + \beta_j \mathbf{v}_{j-1}$. The vectors $\mathbf{v}_0, \mathbf{v}_1, \cdots, \mathbf{v}_j$ will then form the basis for the Krylov subspace $\mathcal{K}^{j+1}(\mathbf{A}, \mathbf{r})$. We can construct an algorithm known as Conjugate Orthogonal Conjugate Gradient (COCG).

Algorithm 5.7 Conjugate Orthogonal Conjugate Gradient

```
1. Choose $\mathbf{x}_0$. Compute $\mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0$, $\mathbf{p}_0 := \mathbf{r}_0$
2. For $j = 0, 1, \cdots$, until convergence Do:
3.     $\mathbf{v}_j := \mathbf{A}\mathbf{p}_j$
3.     $\alpha_j := (\overline{\mathbf{r}}_j, \mathbf{r}_j)/(\overline{\mathbf{v}}_j, \mathbf{p}_j)$
4.     $\mathbf{x}_{j+1} := \mathbf{x}_j + \alpha_j \mathbf{p}_j$
5.     $\mathbf{r}_{j+1} := \mathbf{r}_j - \alpha_j \mathbf{A}\mathbf{p}_j$
6.     $\beta_j := (\overline{\mathbf{r}}_{j+1}, \mathbf{r}_{j+1})/(\overline{\mathbf{r}}_j, \mathbf{r}_j)$
7.     $\mathbf{p}_{j+1} := \mathbf{r}_{j+1} + \beta_j \mathbf{p}_j$
8. Enddo
```

5.9 Quasi Minimum Residual (QMR) method

One advantage of the Conjugate Gradient-type algorithms is the small storage requirement needed during iteration. Because there is no minimization process incorporated in the algorithm the convergence may suffer from irregularity. As mentioned in subsection 5.6 – 5.7, CGS and BiGSTAB were proposed to attain smoother convergence behaviour of CG-like algorithms. In comparison with the GMRES, however, no minimization process is involved. Another CG-type algorithm which maintains the low storage property and the small number of recursion but also imposes an minimization process is Quasi Minimum Residual (QMR) method.

The idea of QMR is based on the look-ahead Lanczos algorithm [24] and solving a least-squares problem

$$\|d^n - \Omega^n \mathbf{H}_e^n z^n\| = \min_{z \in \mathcal{C}^n} \|d^n - \Omega^n \mathbf{H}_e^n z^n\| \tag{83}$$

The basic algorithm of the QMR is described as follows.

Algorithm 5.8 Quasi Minimum Residual

```
1. Choose $\mathbf{x}_0$. Compute $\mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0$, $\mathbf{p}_0 := \mathbf{r}_0$
2. For $j = 0, 1, \cdots$, until convergence Do:
3.     Perform the $n$-th iteration of the look-ahead Lanczos algorithm
3.     Update the QR factorization of $\Omega^n \mathbf{H}_e^n$ and the vector $\mathbf{t}^n$
4.     Compute $\mathbf{x}_n = \mathbf{x}_0 + \mathbf{V}^n \left(\mathbf{R}^n\right)^{-1} \mathbf{t}^n$
5. Enddo
```

In comparison with the CG-like methods, e.g., BiCG, CGS, and BiCGSTAB, QMR shows smoother convergence. This property becomes significant when problems to be solved are near breakdown in the case of the BiCG algorithm. However, QMR is not guaranteed to converge faster than BiCG. In fact, in cases in which BiCG shows irregular convergence, QMR and BiCG converge with typically the same number of iterations. Even though the GMRES converges smoother than the QMR, both methods have basically the same convergence behaviour. In many cases, QMR is outperformed by the restarted GMRES algorithm.

5.10 Iterative methods for solving Helmholtz's equation

Even though various iteration methods have been developed, not all of them are applicable for Helmholtz's problem. CGS, for example, is not known to be used for solving Helmholtz's problem. Also CG so far is only used for comparison studies since theoretically it is not suitable for indefinite problems. For indefinite problems, the CG for normal equations – CGNR – is used due to its simplicity. However, without good preconditioners CGNR is not practical because of slow convergence. We refer to Bayliss, Goldstein, and Turkel [4] and Laird [41] for CGNR applications. Some authors use GMRES, BiCGSTAB, COCG, and QMR to solve Helmholtz problems. Again, without good preconditioners these methods are not efficient.

Since all authors use different test cases, it is somewhat difficult to compare one iterative method with another. Furthermore, the fact that the methods show slow convergences without being preconditioned makes it impossible to compare the methods without incorporating preconditioners. Just for a performance estimation, Table 5.1 provides the cost of the iteration methods measured in the number of matrix/vector multiplications without preconditioners.

Table 5.1: *Matrix/vector multiplications for one iteration step without preconditioners*

Iter. methods	No. of mat/vec mults.
CG	1
CGNR	2
BiCGSTAB	2
GMRES	1
QMR	2
COCG	2

For large wavenumbers, GMRES usually encounters memory problems as shown, e.g., in [41] using an SPD perturbed Helmholtz operator. This is because one tends to use a high resolution grid leading to a very large matrix. The restarted GMRES can be used to overcome this issue. However, the iteration is not guaranteed to converge after a prescribed number of iterations. At the end, the number of inner iterations should be made close to full GMRES, even though this does not always ensure convergence [41].

BiCGSTAB and QMR, although still show comparable performance with GMRES for $k < 20$, become inefficient in terms of mat/vec multiplications for high wavenumbers. However, a different preconditioner used in [27] allows QMR to obtain converged solutions even for high wavenumbers at acceptable costs.

6 Preconditioners

The weak point in solving $\mathbf{Ax} = \mathbf{b}$ with iterative methods comes from the lack of robustness and efficiency [54]. In general, this linear system can be transformed into another form having the same solution but being simpler to solve with iterative methods. The original linear system is preconditioned. A proper choice of preconditioner is expected to improve the robustness and efficiency of iterative methods. With a preconditioner, the problem of solving linear systems can be reformulated as follows:

Find $\mathbf{C}_L$ *and* $\mathbf{C}_R$ *such that* $\mathbf{C}_L\mathbf{A}\mathbf{C}_R\mathbf{y} = \mathbf{C}_L\mathbf{b}$, $\mathbf{y} = \mathbf{C}_R^{-1}\mathbf{x}$ *is easier to solve*

A linear system obtained from discretizations of a PDE can have a highly distributed spectrum and can result in an indefinite linear system (the spectrum consists of both positive and negative real eigenvalues). For such problems the iterative methods show slow convergence or even breakdown. A good preconditioner can transform the original linear system into a system with a clustered spectrum. It is also important that a preconditioned system does not have an eigenvalue close to zero.

In the subsequent subsections, the Jacobi and Gauss-Seidel iteration will be revisited because they do not only pave the way of constructing the widely-used *incomplete* ILU preconditioner but also the recently popular method to improve convergence of iterative methods: multigrid. We will see how the Jacobi and Gauss-Seidel iterations can be interpreted as a preconditioner to the linear systems and be generalized to the so-called ILU preconditioner.

6.1 Jacobi preconditioners

In this type of preconditioner, one takes a splitting to the matrix $\mathbf{A}$ according to the following:

$$\mathbf{A} = \mathbf{M} - \mathbf{N}, \tag{84}$$

where $\mathbf{M}$ and $\mathbf{N}$ are any suitable matrices for improving the iterative algorithms. Substituting (6.1) in the linear system $\mathbf{Ax} = \mathbf{b}$, an iterative formulation can be constructed to obtain the solution $\mathbf{x}$, namely

$$\mathbf{x}_{j+1} = \mathbf{G}\mathbf{x}_j + \mathbf{f}, \tag{85}$$

where $\mathbf{f} = \mathbf{M}^{-1}\mathbf{b}$, and

$$\mathbf{G} = \mathbf{M}^{-1}\mathbf{N} = \mathbf{I} - \mathbf{M}^{-1}\mathbf{A}. \tag{86}$$

This formulation is identical for Jacobi and Gauss-Seidel iteration to obtain solution $\mathbf{x}$ with $\mathbf{G}_{JAC} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{A}$ and $\mathbf{G}_{GS} = \mathbf{I} - (\mathbf{D} - \mathbf{E})^{-1}\mathbf{A}$, wherein $\mathbf{A} = \mathbf{D} - \mathbf{E} - \mathbf{F}$, and $\mathbf{E}$ and $\mathbf{F}$ strictly lower and upper triangular part of $\mathbf{A}$, respectively. In this sense, Jacobi or Gauss-Seidel iteration can be seen as preconditioners for solving a preconditioned linear systems with $\mathbf{M}_{JAC} = \mathbf{D}$ and $\mathbf{M}_{GS} = \mathbf{D} - \mathbf{E}$.

Generalization can be made for the Gauss-Seidel-type preconditioner by relaxing the preconditioner with a factor ω. In such a case, a *Successive Overrelaxation* (SOR) preconditioning is obtained, i.e.,

$$\mathbf{M}_{SOR} = \frac{1}{\omega}(\mathbf{D} - \mathbf{E}). \tag{87}$$

If extra sweeping is performed over unknowns in reverse order of the SOR algorithm, a symmetric version of SOR results. The Symmetric SOR (or SSOR) preconditioner is given as

$$\mathbf{M}_{SSOR} = (\mathbf{D} - \omega\mathbf{E})^{-1}\mathbf{D}^{-1}(\mathbf{D} - \omega\mathbf{F}). \tag{88}$$

If $\omega = 1$ then the symmetric Gauss-Seidel iteration is found:

$$\mathbf{M}_{SGS} = (\mathbf{D} - \mathbf{E})^{-1}\mathbf{D}^{-1}(\mathbf{D} - \mathbf{F}) \tag{89}$$

which can be expressed in $\mathbf{L}$ and $\mathbf{U}$ as

$$\mathbf{M}_{SGS} = \mathbf{L}\mathbf{U}, \tag{90}$$

where $\mathbf{L} = (\mathbf{D} - \mathbf{E})\mathbf{D}^{-1} = \mathbf{I} - \mathbf{E}\mathbf{D}^{-1}$ is unit lower triangular and $\mathbf{U} = \mathbf{D} - \mathbf{F}$ is upper triangular. Since all methods described above rely on a splitting of $\mathbf{A}$, (see (6.1)), these preconditioners are categorized as the matrix-splitting class. Generalization in the form (6.7) leads to an important result which becomes the foundation for developing another type of preconditioners discussed in the next subsection. In many cases, the performance of the Jacobi or SOR/SSOR preconditioner is rather poor unless modifications are incorporated into the original algorithm.

6.2 Preconditioners for normal equation

In Section 5 we discussed CGNR and considered it to be the simplest algorithm to solve indefinite or nonsymmetric linear systems. However, the algorithm produces very slow convergence rates. In this subsection, we discuss two preconditioners especially for the standard CGNR algorithm for solving Helmholtz's problem.

6.3 Laplace operator as a preconditioner

We first consider a classical method of Bayliss, Goldstein, and Turkel [4]. The method was developed based on solving a preconditioned normal equation with CG. To construct the algorithm, the standard SOR preconditioner is modified by approximating $\mathbf{A}$ with its discrete Laplacian term $\mathbf{A}_L$. This can be simply done by setting $k = 0$ in the Helmholtz operator $\mathcal{L}$. The preconditioner $\mathbf{M}$ then follows (6.4).

Since the system is highly indefinite for large k, the first step to be performed is to transform the linear system into a normal equation by multiplying it with its adjoint $\mathbf{A}^T$. This results in a positive semidefinite matrix for any type of boundary conditions and a positive definite matrix for Sommerfeld's radiation condition. However, the resulting normal equation is ill-conditioned, and therefore the CG method is very slow to converge. To improve the convergence, Bayliss, Goldstein, and Turkel used the discrete Laplace operator as a preconditioner to approximate the inverse of the discrete Helmholtz operator.

Let $\mathbf{A}'$ be the preconditioned $\mathbf{A}$. The equivalent linear system in normal form takes the form

$$\mathbf{A}'^{*}\mathbf{A}'\mathbf{x}' = \mathbf{A}'^{*}\mathbf{b}', \tag{91}$$

where $\mathbf{A}'^{*}$ the adjoint of $\mathbf{A}'$. Further, $\mathbf{A}' = \mathbf{M}^{-1}\mathbf{A}\mathbf{M}$, $\mathbf{x}' = \mathbf{M}^{-1}\mathbf{x}$, $\mathbf{b}' = \mathbf{M}^{-1}\mathbf{b}$, and $\mathbf{M} = \mathbf{D}_L - \mathbf{E}_L$. With this formulation, the preconditioned linear system can be solved iteratively using CG method (see Section 5.3).

In the implementation, the factor $\mathbf{A}'^{*}\mathbf{A}'$ is never computed. However, this approach requires the inversion of $\mathbf{M}$. This matrix is obtained from point SSOR applied to $\mathbf{A}_L$. This matrix is then multiplied by $\mathbf{A}$ and $\mathbf{A}^{*}$. The latter is an additional multiplication which doubles the number of operations in one CG-iteration.

6.3.1 Generalized positive definite operator as a preconditioner

The approach used by Bayliss, Goldstein, and Turkel can be regarded as preconditioning the normal equation with a perturbed Helmholtz equation. The constraint which allows such a perturbation is that the perturbed Helmholtz equation should have positive definite properties. In their case, setting $k = 0$ reduces the Helmholtz equation to the Laplace equation which is positive definite. This approach can be generalized by replacing the positive k^2 with $-k^2$ ensuring a positive definite system.

We can set e.g.

$$M = (\Delta - k^2)p, \tag{92}$$

with prescribed boundary conditions as the preconditioner. Reference [41] gives detailed investigation of this type of preconditioner for 2D Helmholtz problems. It is shown there that the preconditioner efficiently clusters eigenvalues and reduces the condition number to a value independent of grid resolution. This property is proven for both the standard system and the normal equation. From the numerical results, the number of matrix/vector multiplications needed to reach convergence is by factor of 4 larger compared to the non-restarted GMRES with the same preconditioner. The matrix/vector multiplications also increase rapidly for higher wavenumbers. However, GMRES can not be used further for sufficiently high wavenumbers, since the grid resolution requirement causes storage problems. The restarted GMRES can solve the storage problems, however, the number of restart should be large to avoid stagnation. Even though BiCGSTAB and QMR outperform the CGNR for low wavenumber cases, their performances become severely worse for high wavenumber cases.

6.4 ILU preconditioner

In Section 6.1, the general pattern of preconditioner $\mathbf{M}$ can be written as an $\mathbf{L}$- and $\mathbf{U}$-part of $\mathbf{A}$. In practice, the exact factorization of $\mathbf{A}$ into $\mathbf{L}$ and $\mathbf{U}$ is not necessarily required. Rather, an approximate factorization is still useful for preconditioning. Since the degree of approximation is rather arbitrary, constraints should be added to the factorization.

An *Incomplete* LU (ILU) factorization can be obtained by computing a sparse lower triangular matrix $\mathbf{L}$ and upper triangular matrix $\mathbf{U}$ by which the residual matrix $\mathbf{R} = \mathbf{LU} - \mathbf{A}$ satisfies certain constraints. In a practical implementation, the ILU factorization depends on the implementation of the Gaussian elimination. The general ILU factorization is given in the following algorithm [54]. In this algorithm, the IJK version of Gaussian elimination has been incorporated. In the algorithm below, P is the non zero pattern set such that $P \subset \{(i,j)|i \neq j; 1 \leq i,j \leq n\}$.

Algorithm 6.1. General ILU factorization

```
1. For $i = 2, \cdots, n$ Do:
2.    For $k = 1, \cdots, i-1$ and if $(i,k) \ni P$, Do:
3.      $a_{ik} := a_{ik}/a_{kk}$
4.      For $j = k+1, \cdots, n$ and for $(i,j) \ni P$, Do:
5.        $a_{ij} := a_{ij} - a_{ik}a_{kj}$
6.      Enddo
7.    Enddo
8. Enddo
```

6.4.1 Variants of ILU factorization

The first ILU variant is the so-called zero fill-in ILU factorization or ILU(0). In general, if one takes any lower triangular matrix $\mathbf{L}$ and upper triangular matrix $\mathbf{U}$ the product $\mathbf{LU}$ does not have the same structure as $\mathbf{A}$. In ILU(0), one takes any pair of $\mathbf{L}$ and $\mathbf{U}$ having the nonzero pattern set to be precisely the same as the nonzero pattern of the lower and upper triangular of $\mathbf{A}$. This defines the ILU(0): if $\mathbf{A} - \mathbf{LU}$ is computed, then the nonzero elements of $\mathbf{A}$ are set to zero.

In order to improve the convergence rate, more accurate ILU factorizations can be performed by allowing some fill-ins. Falling into this category is ILU(p), especially ILU(1). The ILU(1) factorization results from taking the zero pattern to be that of the product $\mathbf{LU}$ obtained from ILU(0). In other words, we consider a matrix with additional off-diagonal components which are actually zero in the original matrix. The factors $\mathbf{L}_1$ and $\mathbf{U}_1$ of ILU(1) are obtained by performing ILU(0) to this matrix.

Also within this class is the *Modified* ILU (or MILU) factorization in which the dropping process for the extra diagonals is compensated at the k-loop of Algorithm 6.1. In an MILU factorization, after the k-loop the diagonal element a_{ii} is modified by adding it with the sum of the row i. Also, preconditioners such as ILUT can be used to have a more accurate factorization that improves the convergence rate. Details about MILU and ILUT can be found in [54].

Made [45] used the positive definite or slightly indefinite preconditioner to construct the factorization. Since one of the requirements to have a good convergence performance is that the real part of the eigenvalues of the preconditioned system is positive, Made proposed perturbations to the real components of the matrix $\mathbf{A}$. In determining the perturbation, a parameter is introduced acting only on the diagonal components of **A**. This parameter is determined so that the perturbed **A** is positive definite or slightly indefinite.

To reduce the number of fill-in, the ordering may become very important in any type of ILU factorizations. Several ordering methods are possible to be incorporated into ILU preconditioners, e.g., Cuthill-McKee, minimum degree, and nested dissection ordering, discussed in Section 4.

The use of ILU(0) is reported in [27] and compared with AILU (Section 6.3.3). Numerical experiments show that the ILU(0) breaks down for large k (highly indefinite problems).

6.4.2 Nested Grids ILU (NGILU) and Matrix Renumbering ILU (MRILU)

Van der Ploeg, Botta, and Wubs [65] use another strategy to factorize **A** using partitioning based on multigrid. For the m-th grid level, Ω_m is the set of unknowns involved in the grid and is defined as $W_m = \Omega_m \setminus \Omega_{m+1}$, where Ω_{m+1} is one level coarser than Ω_m. Using a lexicographical-type numbering, a system of linear equations is obtained in the form

$$\begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{bmatrix} \tag{93}$$

for the finest grid level Ω_1 and its one-level coarser grid Ω_2. This partitioning can be done repeatedly until a coarsest grid is reached.

After renumbering, the preconditioner is constructed using incomplete LU-decomposition with a drop tolerance. In order to obtain this preconditioner, the standard splitting $\mathbf{A} = \mathbf{LU} + \mathbf{R}$ is used with a condition that the elements of residual matrix **R** satisfy $r_{ij} \leq \epsilon_{ij}$, where ϵ_{ij} is the threshold parameter. In [65], to obtain the factors **L** and **U**, first the diagonal of **A** is scaled to unity. This is then followed by a row-by-row incomplete decomposition. To construct row i of **L** and **U**, the relation

$$r_{ik} + l_{ik}u_{kk} = a_{ik} - \sum_{j=1}^{k-1} l_{ij}u_{jk}, \ k < i \tag{94}$$

is used. If the absolute value of the right-hand side of (6.10) is less then ϵ_{ij}, fill-in on position (i,k) is added to the main diagonal. The drop tolerance ϵ_{ij} is determined as follows,

$$\epsilon_{ij} = c^{m-1}\epsilon^{(1)}, \ \max(i,j) \in W_m. \tag{95}$$

For such a choice of drop tolerance, one allows the increase of fill-in per grid level. According to [65], in 2-D the choice $c = 0.25$ is sufficient.

Comparisons with ILU and MILU(0.02) preconditioners for solving Poisson's equation shows the superiority of NGILU. All three preconditioners are combined with the BiCGSTAB iteration for solving the discrete Poisson equation. The number of flops after using NGILU shows an almost 10-times reduction of that of MILU(0.02). However, the MRILU and NGILU have not been tested to accelerate convergence on the Helmholtz problem.

The limitation NGILU has is that the numbering is based solely on the grid and does not take into account the size of nonzero elements of the matrix. For obtaining an efficient factorization process it is crucial that $\mathbf{A}_{11}$ is always well-conditioned. Eventhough for standard rectangular grids the ill-conditioned $\mathbf{A}_{11}$ may not appear, this problem can arise if (highly) stretched grids are used or in the case of finite element grids. Botta and Wubs suggest that the numbering should be determined during the incomplete factorization process, not on the basis of grids or sparsity patterns [10]. This results in a new preconditioner called Matrix Renumbering ILU (MRILU). During the factorization, one guarantees that the diagonal blocks to be inverted are strongly diagonally dominant. This condition can be expressed mathematically as

$$\sum_{i \neq k} |a_{ik}| \leq \epsilon |a_{ii}| \text{ with } \epsilon < 1. \tag{96}$$

The threshold ϵ can be chosen small to approximate $\mathbf{A}_{11}$ with a diagonal matrix, which simplifies the factorization process on the next coarser grid level. During factorization, small elements are dropped to limit the number of nonzeros. However such a dropping is only limited to the diagonal block $\mathbf{A}_{11}$ and two blocks $\mathbf{A}_{12}$ and $\mathbf{A}_{21}$. Modification in $\mathbf{A}_{22}$ is to be avoided because it can cause the diagonal of $\mathbf{A}_{22}$ on the next coarser grid level to become very small. In such situations, the factorization may break down.

6.4.3 Analytic ILU preconditioner

Recently, Gander and Nataf [25, 26, 27] proposed an ILU preconditioner based on the analytic parabolic factorization for Helmholtz problems. Instead of finding a proper preconditioner for the discrete Helmholtz problems, the preconditioner is determined from an analytical factorization of the continuous differential operator. It is expected that the resulting preconditioner, which is called *Analytical* ILU (AILU) preconditioner, will be a good approximation to the differential operator and will also be easy to invert.

The main idea of AILU preconditioner is based on the parabolic factorization of the Helmholtz operator $\mathcal{L} = \Delta + k^2$ into the form

$$\mathcal{L} = (\partial_x + \Lambda_1)(\partial_x - \Lambda_2), \tag{97}$$

where Λ_1 and Λ_2 are positive operators. The term $(\partial_x + \Lambda_1)$ represents a parabolic operator acting in the positive x-direction, while the term $(\partial_x + \Lambda_2)$ represnts a parabolic operator acting in the negative x-direction.

If Fourier transformation is applied in y-direction to the operator $\mathcal{L}$, we get the relation

$$\mathcal{F}_y(k^2 + \Delta) = (\partial_x + \sqrt{k^2 - \omega^2})(\partial_x - \sqrt{k^2 - \omega^2}). \tag{98}$$

The continuous parabolic operator (6.13) can be obtained with $\Lambda_1 = \Lambda_2 = \mathcal{F}_y^{-1}(\sqrt{k^2 - \omega^2})$. Both Λ_1 and Λ_2 are nonlocal operators in y because of the square root.

The parabolic factorization can be related to an exact LU decomposition of the linear system after discretizing the Helmholtz operator $\mathcal{L}$ in x-direction and computing another analytic factorization for the semi-discrete operator $\mathcal{L}_h = \Delta_h + k^2$, where $\Delta_h = D^+D^- + \partial_{yy}$ where D^+ and D^- are the first order forward and backward discretizations in x-direction. Following the same idea with the exact analytical factorization of the Helmholtz operator, the following relation is obtained,

$$\mathcal{F}_y(k^2 + \Delta_h) = \left(D_x^- + \left(\tau h - \frac{1}{h}\right)\right) \frac{1}{h^2 \tau} \left(D_x^+ - \left(\tau h - \frac{1}{h}\right)\right), \tag{99}$$

with h the mesh size and τ the nonlocal operator in y introduced due to discretization. It is easy to find that (6.13) is recovered for $h \to 0$.

Eq. (6.15) can be used to solve the Helmholtz problem using the defect correction-type technique as explained in [30] (or [47]). However, since this equation contains nonlocal operators in y, we should first approximate these nonlocal operators. Approximation can be made to the nonlocal operator τ in the following form:

$$\tau_{app} = \frac{1}{h^2} + \frac{k^2 - \omega^2}{2} + \frac{2}{2h}(q + pk^2), \quad p, q \in \mathbb{C}\,,\ \mathcal{R}e(p) > 0. \tag{100}$$

In (6.16), p and q are optimization parameters and the subscript $_{app}$ indicates "approximation". This approximate operator is then used as the ILU-type preconditioner. After substituting τ_{app} into (6.15) and denotinf the right-hand side as $\mathcal{L}_{app}$, the approximate factorization can be cast, namely

$$\mathcal{L}_{app} = D_x^+ D_x^- - \tau_{app} - \frac{1}{\tau_{app} h^4} + \frac{2}{h^2}. \tag{101}$$

A more detail analysis including the convergence properties of the AILU can be found in [25]. One important result from the discussion is that the convergence rate for a stationary iterative method with given optimized parameters p and q is bounded by $1 - \mathcal{O}(h^{2/3})$. For CG-like iterative method the convergence rate is bounded by $1 - \mathcal{O}(h^{1/3})$.

In [27], comparisons have been made with ILU(0), ILU(1e-2) preconditioners for 2D Helmholtz problem. The QMR algorithm solves the linear system with the initial guess $p^{(0)} = 0$. In addition, the unpreconditioned QMR is used. A wide frequency range is used during the test campaign, from $k = 5$ to 50 with the stepsize of 5. It can be deduced that with the AILU preconditioner, the QMR shows better computation performance in terms of iteration count and flops count for the solution process. A significant improvement is shown for high frequency cases whenever QMR with two other preconditioners fails to converge within the allocated iteration hour. In such cases, QMR with the AILU preconditioner converges to the solution. In terms of operation count required for computing the preconditioner, AILU is comparable to ILU(0).

Because of the interesting computational performance provided by the AILU preconditioner, numerical experiments on the Helmholtz problem as described in Section 2 will be performed. The results are given in a separate report.

6.4.4 Preconditioner for a highly indefinite matrix

One problem arising in solving linear equations derived from the discretization of the Helmholtz equation is the indefiniteness of matrix $\mathbf{A}$. Freund [23] proposes a method for constructing indefinite preconditioner for the symmetric QMR algorithm. In this case, two types of preconditioners can be developed.

First, a preconditioner can be derived based on block SSOR preconditioner. A permutation $\mathbf{P}$ is determined such that

$$\mathbf{PAP}^T = \mathbf{L} + \mathbf{\Lambda} + \mathbf{L}^T \tag{102}$$

where $\mathbf{L}$ is strictly lower triangular and $\mathbf{\Lambda}$ is a non-singular and "close" to diagonal matrix. The preconditioner developed is

$$\mathbf{M} = \mathbf{P}^T(\mathbf{\Lambda} + \mathbf{L})\mathbf{\Lambda}^{-1}(\mathbf{\Lambda} + \mathbf{L}^T)\mathbf{P}. \tag{103}$$

Using (6.19) and (6.20) the "error" of the SSOR preconditioner can be determined, i.e.,

$$\mathbf{M} - \mathbf{A} = \mathbf{P}^T\mathbf{L}\mathbf{\Lambda}^{-1}\mathbf{L}^T\mathbf{P}. \tag{104}$$

Relation (6.21) suggests that $\mathbf{P}$ should be chosen such that $\mathbf{\Lambda}$ is as "large" as possible.

Another type of preconditioner suitable for an indefinite matrix can be developed from an adaptation of the ILUT preconditioner. To construct the preconditioner, we compute an incomplete factorization

$$\mathbf{PAP}^T = \mathbf{L}\mathbf{\Lambda}\mathbf{L}^T + \mathbf{R}. \tag{105}$$

where $\mathbf{\Lambda}$ is block diagonal with 1×1 and 2×2 blocks which are non-singular and "close" to diagonal, and $\mathbf{L}$ is block lower triangular. The sparsity of $\mathbf{L}$ is controlled by removing undesired elements and putting them into $\mathbf{R}$.

A test using QMR algorithm is performed in [23] for an elliptic PDE system (Stokes equation) and compared with the MINRES algorithm. It is seen at least for the experiment presented that QMR is superior to MINRES in terms of number of iteration. Further, it is shown that the preconditioner based on an ILUT adaptation clusters the spectrum better than SSOR. Numerical experiments show a faster convergence with the preconditioner based on ILUT adaptation than on SSOR.

6.5 Preconditioner based on separation of variables

The Helmholtz operator can be viewed a the Laplace operator Δ with an additional term k. For the Laplace equation, an analytic solution can be obtained using the separation of variables method. One may consider the same solution procedure could work nicely for the Helmholtz equation. However, the presence of k actually prevents application of the same method on the latter problem. We can decompose k leading to a formulation suitable for a separation of variables method and use it as a preconditioner for solving the Helmholtz equation. Refering back to the discretization given in Section 3, the matrix $\mathbf{A}$

consists of two parts: Laplace's equation with related boundary conditions and a block diagonal matrix $\mathbf{K}$ consisting of $k^2\mathbf{I}$. Since k prevents us from directly implementing the separation of variables method, we look for a decomposed formulation of k which allows us to include k in the separation of variables method. We decompose k as follows (Plessix and Mulder [52]):

$$k^2(x,y,z) = k_x^2(x) + k_{yz}^2(y,z) + \tilde{k}^2(x,y,z), \tag{106}$$

which $\tilde{k}$ satisfies conditions

$$\begin{aligned} \int \tilde{k}^2(x,y,z)dx = 0 \ \forall y,z \\ \int \tilde{k}^2(x,y,z)dydz = 0 \ \forall x. \end{aligned} \tag{107}$$

Introducing this decomposition, we can rewrite matrix $\mathbf{A}$ as the following:

$$\mathbf{A} = (\mathbf{A}_x - \mathbf{K}_x) \otimes \mathbf{I}_{yz} + \mathbf{I}_x \otimes (\mathbf{A}_{yz} - \mathbf{K}_{yz}) - \tilde{\mathbf{K}}. \tag{108}$$

Let the eigenvalues and eigenvectors of the linear system $\mathbf{A}_x - \mathbf{K}_x$ satisfy the relation

$$\mathbf{W}_L^H(\mathbf{A}_x - \mathbf{K}_x)\mathbf{W}_R = \mathbf{\Lambda}. \tag{109}$$

If we multiply $\mathbf{A}$ from the left with $\mathbf{W}_L^H \otimes \mathbf{I}_{yz}$ and with $\mathbf{W}_R \otimes \mathbf{I}_{yz}$ from the right, we get

$$\begin{aligned} \mathbf{B} &= (\mathbf{W}_L^H \otimes \mathbf{I}_{yz})\mathbf{A}(\mathbf{W}_R \otimes \mathbf{I}_{yz}) \\ &= \mathbf{\Lambda} \otimes \mathbf{I}_{yz} + \mathbf{I}_x \otimes \mathbf{A}_{yz} - (\mathbf{W}_L^H \otimes \mathbf{I}_{yz})\tilde{\mathbf{K}}(\mathbf{W}_R \otimes \mathbf{I}_{yz}). \end{aligned} \tag{110}$$

If we introduce a permutation matrix $\mathbf{P}$ such that the nonzero elements of $\mathbf{P}$ are

$$P(i+(j-1)M, j+(i-1)NL) = 1;\ 1 \geq i \geq M \text{ and } 1 \geq j \geq NL \tag{111}$$

the following simple relation is obtained, namely

$$\mathbf{P}^T\mathbf{B}\mathbf{P} = \mathbf{D} + \tilde{\tilde{\mathbf{K}}}, \tag{112}$$

where $\mathbf{D}$ is a block diagonal matrix consisting of M blocks with

$$\mathbf{D}_m = \lambda_m\mathbf{I}_{yz} + \mathbf{A}_{yz} - \mathbf{K}_{yz}, \tag{113}$$

and

$$\mathbf{P}^T(\mathbf{W}_L^H \otimes \mathbf{I}_{yz})\tilde{\mathbf{K}}(\mathbf{W}_R \otimes \mathbf{I}_{yz})\mathbf{P}. \tag{114}$$

Finally, the separation of variables method gives the solution, if we assume constant k implying $\tilde{\tilde{\mathbf{K}}} = 0$,

$$\mathbf{D}\tilde{\mathbf{p}} = \tilde{\mathbf{f}} \tag{115}$$

with new variables

$$\begin{aligned} \tilde{\mathbf{p}} &= \mathbf{P}^T(\mathbf{W}_L^H \otimes \mathbf{I}_{yz})\mathbf{p}, \\ \tilde{\mathbf{f}} &= \mathbf{P}^T(\mathbf{W}_L^H \otimes \mathbf{I}_{yz})\mathbf{f}. \end{aligned} \tag{116}$$

The linear system (6.31) is easy to solve, because $\mathbf{D}$ is block diagonal. Moreover, since $\tilde{\mathbf{p}}$ and $\tilde{\mathbf{f}}$ are actually comprising of M separate blocks of size NL the overall problem also reduces to solving M 2D problems of size NL.

Even though the final cost for computing the solution is low, one should anticipate the overhead cost involving the computation of M eigenvectors and eigenvalues of the system $\mathbf{A}_x - \mathbf{K}_x$ and also matrix/matrix (of order $\mathbf{N}$) multiplication to construct the block diagonal $\mathbf{D}$.

Until this point, we have only discussed solution of the discrete Helmholtz equation using the separation of variables method which is workable for constant k due to the condition $\tilde{\mathbf{K}} = 0$. Extension to inhomogeneous media can not be done in the same way because of the restriction imposed by $\tilde{\mathbf{K}} = 0$. However we can use (6.32) as the preconditioner of the original linear system and solve the preconditioned linear system iteratively. To cope with the varying k in the medium, we interprete the subsurface interfaces as boundaries with outgoing conditions imposed. Since on the adjacent of these boundaries the values of k are varied, we simply take the averaged value of two adjacent k. With this we can construct the preconditioner. The preconditioned linear system for iterative solution is now given as

$$\mathbf{M}^{-1}\mathbf{A}\mathbf{p} = \mathbf{M}^{-1}\mathbf{f}. \tag{117}$$

where

$$\mathbf{M}^{-1} = (\mathbf{W}^R \otimes \mathbf{I}_{yz})\mathbf{P}\mathbf{D}^{-1}\mathbf{P}^T(\mathbf{W}_L^H \otimes \mathbf{I}_{yz}). \tag{118}$$

Since $\mathbf{M}$ can be considered as an approximate system of $\mathbf{A}$, $\mathbf{M}^{-1}\mathbf{A}$ multiplication should provide a system which is relatively easy to solve. This is proved for very low frequency cases and relatively smooth media. However, the degree of accuracy of this approximate depends largely on how k varies in the medium. Numerical examples show that for largely varying k, the methods fail. Moreover, the methods also fail for large frequency cases, which are of practical interest.

6.6 Approximate inverse

We have cited when discussing the standard ILU preconditioner that this class of preconditioner can often fail if the matrix $\mathbf{A}$ is not an M-matrix. Moreover, breakdown due to zero pivoting may be encountered if $\mathbf{A}$ is indefinite. The approximate inverse of $\mathbf{A}$ can be determined by seeking a sparse matrix $\mathbf{M}$ which minimizes the residual matrix $\mathbf{R} = \mathbf{I} - \mathbf{A}\mathbf{M}$. A standard method is by minimizing the Frobenius norm of $\mathbf{R}$.

For example, for the left-preconditioning matrix the minimization process can be defined as

Find $\boldsymbol{M}$ such that $||\boldsymbol{I} - \boldsymbol{M}\boldsymbol{A}||_F$ is minimal.

The Frobenius norm of an arbitrary matrix $\mathbf{X}$ is defined as

$$||\mathbf{X}||_F = \left[\mathrm{tr}(\mathbf{X}^H\mathbf{X})\right]^{1/2} = \left[\mathrm{tr}(\mathbf{X}\mathbf{X}^H)\right]^{1/2} \tag{119}$$

A similar expression can be determined for the right-preconditioning and for splitting preconditioning.

Two approaches can be used in order to proceed the minimization. First, a global minimization is applied which is similar to a gradient-type method. In global minimization the steepest descent method can be implemented. Secondly, we can minimize the residual row-wise by considering the equivalence

$$||\mathbf{I} - \mathbf{M}\mathbf{A}||_F^2 = \sum_{j=1}^{n} ||\mathbf{e}_j - \mathbf{m}_j\mathbf{A}||_2^2, \tag{120}$$

in which $\mathbf{e}$ and $\mathbf{m}$ are the j-th rows of the identity matrix $\mathbf{I}$ and of the matrix $\mathbf{M}$. Therefore, we minimize the function

$$f_j(m) = ||\mathbf{e}_j - \mathbf{m}\mathbf{A}||_2^2. \tag{121}$$

For detailed discussions, we refer to [54]. In the case of a nonsymmetric or an ill-conditioned linear system, the left-preconditioning usually results in a good approximate inverse [8].

Another method is to construct an approximate inverse based on the factorization $\mathbf{A} = \mathbf{LDU}$. Inverting $\mathbf{A}$ gives $\mathbf{A} = \mathbf{Z}\mathbf{D}^{-1}\mathbf{W}^T$, where $\mathbf{Z} = \mathbf{U}^{-1}$ and $\mathbf{W} = \mathbf{L}^{-T}$. The approximate inverse of the factorized $\mathbf{A}$ is determined by computing sparse approximations $\overline{\mathbf{Z}} \approx \mathbf{Z}$, $\overline{\mathbf{W}} \approx \mathbf{W}$, and $\overline{\mathbf{D}} \approx \mathbf{D}$, giving $\mathbf{M} \approx \mathbf{A}^{-1}$. In this method, there are several techniques that can be used to compute $\mathbf{M}$. These include the FSAI method and the incomplete (bi)conjugation-based method. We refer to [8] and references therein for further discussions of the two methods.

6.7 Multi-Grid Method

In this subsection another class of convergence accelerators is described, namely multigrid. General idea of multigrid can be observed from a stationary iteration case.

6.7.1 Stationary iterations as smoothers

A detailed explanation of multigrid can be found, e.g., in Wesseling [68] and Trottenberg, Oosterlee, and Schüller [62]. Using the standard Gauss-Seidel iteration a given linear equation system can be solved. The iteration usually converges fast in the initial stage of the iteration but then slows down and tends to stall. A Fourier analysis, e.g. as described in [68], shows that the Gauss-Seidel iteration effectively removes high frequency errors but fails to remove the low ones. Rather the errors are smoothened (see e.g. Lahaye [40] for a more illustrative example). From this point of view the Gauss-Seidel iteration can be considered as a smoother. The same property is also possessed by the Jacobi iteration.

On coarser grids, the smooth, low frequency error can become oscillatory. In such an oscillatory case, the Gauss-Seidel iteration becomes a very effective method to remove oscillatory errors. The idea of multigrid is derived from this fact. Multigrid uses coarse grid discretizations to remove smooth errors which can not be wiped out efficiently by stationary iteration techniques in the fine grids. In terms of errors the smoothers can be expressed by the following:

$$e^{m+1} = \mathbf{S}e^{m},\ \mathbf{S} = (\mathbf{I} - \mathbf{M}^{-1}\mathbf{A}) \tag{122}$$

where $\mathbf{M}$ the splitting matrix depending on the type of the stationary iteration.

6.7.2 Geometric Multi-Grid

Let the domain Ω be discretized with h and H are mesh sizes of a fine and coarse grid discretization. A classical way of coarsening is by taking information of the fine grid structure and then removing some gridpoints from the finer grid in a systematic way. There are several ways to coarsen the grids. For the most standard technique is by halving the grid in all directions (identified as standard $h \to 2h$ coarsening). This method falls into the so-called *Geometric Multi-Grid* (GMG).

Discretization on the fine and coarse grid results in the linear systems

$$\mathbf{A}^h\mathbf{x}^h = \mathbf{b}^h \tag{123}$$

and

$$\mathbf{A}^H\mathbf{x}^H = \mathbf{b}^H \tag{124}$$

respectively. To relate the two grid levels, prolongation and restriction operator are used, denoted by $\mathbf{P}_H^h$ and $\mathbf{R}_h^H$ respectively. These operators map the solution of the coarse grids into the fine grids and vice versa.

In a two-grid case, a multigrid cycle is described as follows. Let $\mathbf{x}_m^h$ be an approximate solution of (6.24) after $(m-1)$ multigrid cycles. The high frequency errors are eliminated by applying a few steps of the smoother on the fine grid. This smoothing – so-called pre-smoothing – results in a new approximate solution $\bar{\mathbf{x}}_m^h$. The low frequency errors will be eliminated by restricting the residual after the smoothing onto the coarse grid, by solving the defect equation

$$\mathbf{A}^H\mathbf{e}_m^H = \mathbf{R}_h^H\mathbf{r}_m^h \tag{125}$$

where $\mathbf{r}_m^h = \mathbf{b}^h - \mathbf{A}^h\bar{\mathbf{x}}_m^h$ the residual of the fine grid after smoothing. This process is called the coarse grid correction. The error after the coarse grid correction is interpolated onto the fine grid using the prolongation operator and then added to the approximate solution $\bar{\mathbf{x}}_m^h$, i.e.,

$$\bar{\bar{\mathbf{x}}}_m^h = \bar{\mathbf{x}}_m^h + \mathbf{P}_H^h\mathbf{e}_m^H \tag{126}$$

where, from (6.25), $\mathbf{e}_m^H = (\mathbf{A}^H)^{-1}\mathbf{R}_h^H\mathbf{r}_m^h$. Because of adding the correction, the approximation (6.26) may contain high frequency errors which can be removed by applying post-smoothing (see [40, 68]).

Extension to more-than-two (or *multi*)grid levels can be done by applying the two-grid algorithm recursively on the subsequent coarser grids. In the multigrid algorithm, several standard cycles can be distinguished by implementation of the pre-smoothing, coarse grid correction, and post-smoothing on each grid level. Figure 6.1 shows an illustration of the often-used standard cycles, namely the V-, W-, and F-cycle.

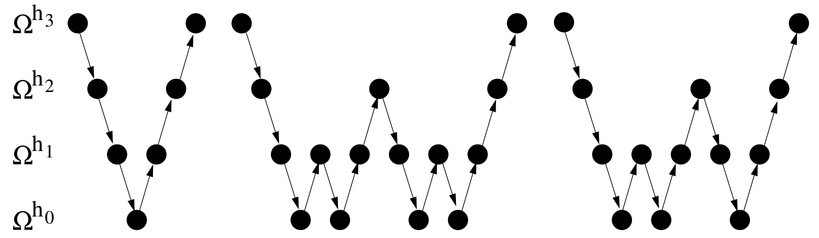

Figure 3: Types of standard multigrid cycles: V-,W-, and F-cycle.

6.7.3 Algebraic Multi-Grid (AMG)

In applications with unstructured grids the GMG becomes unattractive because of the difficulty of constructing the coarse grid. Applications of the GMG to problems with discontinuities also often give very low convergence rate [40].

Another class of multigrid methods is the so-called Algebraic Multi-Grid (AMG). In contrast with GMG, the construction of the coarse grid in AMG does not require explicit information about the fine grid structure. Rather the coarse grid is constructed using the matrix $\mathbf{A}^h$ which implicitly contains the information of the fine grid structure. The general AMG algorithm uses the standard GMG algorithm for fine grid-coarse grid cycling except for an additional *setup* phase. The setup phase is needed to automatically construct a hierarcy of coarser grids. After the setup phase, the cycle algorithm follows.

The first coarse level is constructed by selecting a proper subset C^h of the fine grid space Ω^h, inducing a partitioning of Ω^h. This results in a linear system for the coarse grid equivalent to the linear system for the fine grid, namely

$$\mathbf{A}^H \mathbf{x}^H = \mathbf{b}^H. \tag{127}$$

Having obtained the coarse level system, the prolongation and restriction operator should be determined. Since AMG does not rely on geometrical information of the fine and coarse grid, both operators should be constructed directly from the matrix $\mathbf{A}^h$. The relation between the fine level system and the coarse level system is built, satisfying the Galerkin formula, namely

$$\mathbf{A}^H = \mathbf{R}_h^H \mathbf{A}^h \mathbf{P}_H^h. \tag{128}$$

The next coarser level system can then be constructed with the same algorithm as the above-mentioned. In practice one may prescribe the number of grid levels to be constructed. In this case the automatic coarser level construction will be terminated after reaching the prescribed number of grid levels. Another option is to let the setup algorithm continues until the fill-in elements in the coarsest level becomes inefficiently large.

Lahaye [40] implements AMG on the 2D time-harmonic magnetic field equation (which has different properties from our Helmholtz equation due to a complex sign). For Helmholtz's equation Vanek, Mandel, and Brezina [64, 63] have developed a solver based on a two-level AMG method. A common disadvantage of multigrid is that the coarser grid must be fine enough to capture the wave character of the problem. One way to overcome this is by employing basis functions for coarse grids which are derived from plane waves. Another way is by contructing coarse spaces based on a plane wave within an aggregate of nodes, being zero outside the aggregate, and then smoothened by a Chebyshev type iteration using the original fine level matrix. Compared to the first coarsening the latter shows better computational complexity and scalability.

7 Domain decomposition

So far, we have surveyed methods working on one domain Ω. In the case Ω is very large or a discretization is very fine, discretization may result in an extremely large matrix. In this situation, it will be helpful to devide the domain under consideration into subdomains.

The early work on the domain decomposition method (DDM) to solve the Helmholtz equation is due to Despres [16]. Improvements to this original work are given, e.g., by Ghanemi [29], Colloni, Ghanemi, and Joly [12]. Similar work can also be found in Benamou and Despres [5, 6], and Susan-Resiga and Atassi [59]. Other contributions can be also found in the work of Kim [36, 37, 38, 39] on finite difference and finite element discretizations. Extension to the case with variation of the wavenumber is done by introduding an Alternating Direction Optimal Procedure (ADOP). Larsson [42] approaches the problem by constructing DDM subproblems identical with a single domain problem with Dirichlet boundary conditions. In the later problem results in Otto and Larsson [48] shows the effectiveness of a fast Poisson-type preconditioner incorporated in the iterative methods. Having identical problem on the subdomain level, DDM can be solved combined with iterative methods.

In DDM, several issues are still open for further research. The DDM is easy to implement in parallel. Parallelization is considered a powerful method to speed-up computations. In this section, we will outline some methods related to the DDM. We restrict to the methods developed for Helmholtz's equation.

7.1 DDM for constant k problems

Consider a domain Ω with the boundary $\partial\Omega$. This domain is partitioned into several (finite) nonoverlapped subdomains Ω_j, $j = 1, \cdots, M$, satisfying:

$$\begin{aligned} \overline{\Omega} &= \bigcup_{j=1}^{M} \overline{\Omega}_j, \\ \Omega_j \cap \Omega_k &\neq \varnothing. \end{aligned} \tag{129}$$

Let $\Gamma_j = \Gamma \cap \partial\Omega_j$, $\Gamma_{jk} = \Gamma_{kj} = \partial\Omega_j \cap \partial\Omega_k$. On each subdomain, the Helmholtz problem can be restated as

$$\begin{aligned} \Delta p_j + k^2 p_j = f_j \quad & \text{in } \Omega_j \\ \frac{\partial p_j}{\partial \mu} - \mathrm{i}k p_j = 0 \quad & \text{on } \Gamma_j \end{aligned} \tag{130}$$

with the consistency condition

$$p_j = p_k \text{ and } \frac{\partial p_j}{\partial \mu} = \frac{\partial p_k}{\partial \mu} \text{ on } \Gamma_{jk}. \tag{131}$$

Condition (7.3) is refered to as transmission condition [16]. In a more convenient way, this condition can be expressed by the *Robin* condition [16, 36], namely

$$\frac{\partial p_j}{\partial \mu_j} + \mathrm{i}k p_j = -\frac{\partial p_k}{\partial \mu_k} + \mathrm{i}k p_k, \; i \neq j. \tag{132}$$

Together with (7.2), condition (7.4) gives a well-posed problem. This problem is subjected to the local solution in the subdomains, and with the transmission condition leads to the complete solution. In an iterative way, the algorithm is written as follows.

$$\begin{aligned}
&\Delta p_j^n + k^2 p_j^n = f_j && \text{in } \Omega_j \\
&\frac{\partial p_j^n}{\partial \mu} - \mathrm{i}k p_j^n = 0 && \text{on } \Gamma_j \\
&\frac{\partial p_j^n}{\partial \mu_j} + \mathrm{i}k p_j^n = -\frac{\partial p_k^{n-1}}{\partial \mu_k} + \mathrm{i}k p_k^{n-1} && \text{in } \Gamma_{jk}.
\end{aligned} \tag{133}$$

On a differential level, the convergence analysis has been given in [16]. On a discrete level, the convergence analysis depends on the discretization scheme used. For the centered second order accurate discretization, [36] gives details about the convergence analysis for 2-D problems. Similar results can be found in [38] for finite element discretizations.

Enhancing the convergence of the DDM can be done via a generalization of the Robin condition by replacing k with an arbitrary constant θ. The constant θ is subject to optimization with the condition that the spectral radius of the iteration matrix should become close to zero; i.e $\rho(\mathbf{G}) \approx 0$ where $\mathbf{G}$ is the iteration matrix derived from (7.5) (see [36]). This can be done automatically and in a cheap way by performing an eigenvalue analysis on the reduced iteration matrix $\mathbf{G}''$, where $\mathbf{G}''$ is obtained by reducing zero columns in $\mathbf{G}$. This process is effectively done using a permutation matrix.

7.2 DDM for non-constant k problems

For a possibly non-constant k inside the domain Ω, the optimal θ_0 obtained from the above procedure would not be the optimal value. In such case, the DDM will converge very slowly. Enhancement by the so-called Alternating Direction Optimal Procedure (ADOP) is introduced in [36] and further used in [37, 38, 39], keeping the idea of an optimal Robin condition. The approach is based on strip-type problems allowing to reduce the 2-D Helmholtz problem into a set of 1-D problems. Instead of finding an optimal θ for the complete 2-D domain, with this approach the problem now reduces to finding an optimal θ on a strip.

Consider a domain Ω decomposed into three subdomains (fig. 7.1) with the bold-lines the interfaces of the subdomains. The method tries to find the optimum value θ_{ADOP} in one grid layer (in fig. 7.1 is the horizontal grid line y_0). The optimality condition is that for the selected θ_{ADOP}, the spectral radius of the iteration matrix of the reduced problem is zero. After obtaining θ_{ADOP} in layer y_0, the procedure is repeated on the next layer.

The overall method can be made automatic and is cheap. However, the latter property can not be maintained if the number of subdomain grows. If direct methods are used, the convergence rate of the large number of subdomains case deteriorates and becomes very slow. (One tends to reduce the size of the subdomains to save memory when implementing direct methods.)

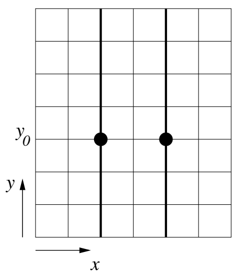

Figure 4: A strip-type decomposition (from [36]).

Instead of using direct methods to attack the problem locally, iterative methods can also be used for the local problems. Since iterative methods are not so competitive without preconditioners, a preconditioner should be incorporated. However, rather than constructing a preconditioner for the complete linear system, the preconditioner is constructed locally in the subdomains. What follows is a DDM method based on generalization of the single domain problem provided in [48].

In the general case (see Larsson [42]), a linear system of the decomposed domain after proper reordering can be written as

$$\begin{pmatrix} \mathbf{A}_{00} & \mathbf{A}_{01} \\ \mathbf{A}_{10} & \mathbf{A}_{11} \end{pmatrix} \begin{pmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \end{pmatrix} = \begin{pmatrix} \mathbf{b}_0 \\ \mathbf{b}_1 \end{pmatrix}, \tag{134}$$

where $\mathbf{p}_0$ consisting of interface points of $\mathbf{p}$ and $\mathbf{p}_1$ containing the interior points. In (7.6), $\mathbf{A}_{11}$ contains coupling between points in the interior of the subdomains and therefore is very large. By block Gauss elimination, system (7.6) reduces to a system for the unknowns on the interface, with the coefficient matrix

$$\mathbf{C} = \mathbf{A}_{00} - \mathbf{A}_{01}\mathbf{A}_{11}^{-1}\mathbf{A}_{10}, \tag{135}$$

is the Schur complement. However, to solve (7.7) exactly is impractical. A more efficient way can be done by suggesting an approximate to $\mathbf{A}_{11}$.

It is found that $\mathbf{A}_{11}$ takes a similar form as the complete matrix $\mathbf{A}$ of the single domain problem when Dirichlet boundary conditions are imposed. However, from the DDM point of view this single domain problem now belongs to each subdomain. Therefore, efficient methods to solve the single domain problem can be adopted on the subdomain level (we think of a possible generalization to the method explained by Larsson [42]). In [42] preconditioner for each subdomain is defined based on a fast Poisson-type preconditioner. In a single domain with Dirichlet boundary conditions, the eigenfunctions at the boundary would take the form of sine functions. These functions will be used as the fast transforms in the fast Poisson preconditioner, which is found to be very efficient [48].

The subdomain preconditioner $\mathbf{M}_{11}$ is constructed with the same block structure as $\mathbf{A}_{11}$ given as

$$\mathbf{M}_{11} = \text{nonad}_{k,m_2}\left(\mathbf{M}_{k,-4}, \mathbf{M}_{k,-3}, \mathbf{M}_{k,-2}, \mathbf{M}_{k,-1}, \mathbf{M}_{k,0}, \mathbf{M}_{k,1}, \mathbf{M}_{k,2}, \mathbf{M}_{k,3}, \mathbf{M}_{k,4}\right), \tag{136}$$

where each block is diagonalized by a block diagonal matrix $\mathbf{Q}$ consisting of the eigenfunctions

$$S_m(j,k) = \sqrt{\frac{2}{m+1}} \sin\left(jk\frac{\pi}{m+1}\right). \tag{137}$$

The $\mathbf{M}$'s are given as follows:

$$\mathbf{M}_{k,r} = \mathbf{Q}\Lambda_{k,r}\mathbf{Q}^T,\ k = 1, \cdots, m_2,\ r = -4, \cdots, 4, \tag{138}$$

with

$$\begin{aligned} &\Lambda_{k,r} = 0,\ k = 2, \cdots, m_2 - 1,\ |r| > 1; \\ &\Lambda_{k,r} = \operatorname{diag}\left(\mathbf{Q}^T\mathbf{A}_{k,r}\mathbf{Q}\right),\ k = 1, m_2,\ |r| > 1; \\ &\Lambda_{k,r} = \operatorname{diag}\left(\mathbf{Q}^T\mathbf{A}_{k,r}\mathbf{Q}\right),\ k = 1, \cdots, m_2,\ r = -1, 0, 1. \end{aligned} \tag{139}$$

The resulting structure now has decoupled subdomain problems allowing (in general) any type of iterative methods to be applied. A comparative study between the restarted GMRES(l) and the band Gauss elimination is given in [42] indicating a further reduction in terms of total complexity and memory requirement. However, to form the preconditioner is an expensive process.

7.3 Domain decomposition based on Lagrange multipliers

Another approach developed for finite element discretizations of the Helmholtz equation is the so-called FETI-H method [21, 60]. This method is favourable not only in terms of computational performance but also from a numerical scalability point of view. The method has been proved to be scalable both with respect to mesh size and subdomain size. Literature [21] provides a concise discussion about the FETI method and its extension to the FETI-H method.

The main concept of the FETI method is based on expressing the discrete Helmholtz equation in a discrete Lagrange multipliers form. This formulation allows the splitting of the complete formulae into two parts: the classical Lagrange function and the interface quantity.

Using the non-overlapping domains (7.1), we partition the solution vectors $\mathbf{p}_j$ into two parts: the internal solution $\mathbf{p}_j^i$ and boundary solution $\mathbf{p}_j^b$. The system $\mathbf{Ap} = \mathbf{f}$ can be rewritten in the form of the system of constrained equations

$$\begin{gathered} \left(\mathbf{K}_j + \mathrm{i}k\mathbf{M}_{I,j}\right)\mathbf{p}_j = \mathbf{f}_j - \mathbf{B}_j^T\lambda \\ \sum_{j=1}^{M} \mathbf{B}_j\mathbf{p}_j = 0 \end{gathered} \tag{140}$$

where

$$\mathbf{M}_{I,j} = \begin{pmatrix} 0 & 0 \\ 0 & \sum_{\Omega_j \cap \Omega_k \neq \emptyset} \epsilon^{j,k}\mathbf{M}_{j,k}^{bb} \end{pmatrix}. \tag{141}$$

Here $\mathbf{B}_j$ are signed Boolean matrices due to interface continuity conditions and $\epsilon^{j,k} = -\epsilon^{k,j} = \pm 1$.

Solving the first equation of (7.12) for $\mathbf{p}_j$ and substituting the results to the second equation of (7.12), we have a dual interface problem:

$$\mathbf{F}_I \lambda = \mathbf{d}, \tag{142}$$

where

$$\begin{aligned} \mathbf{F}_I &= \sum_{j=1}^{M} \mathbf{B}_j (\mathbf{K}_j + \mathrm{i}k\mathbf{M}_{I,j})^{-1} \mathbf{B}_j^T, \\ \mathbf{d} &= \sum_{j=1}^{M} \mathbf{B}_j (\mathbf{K}_j + \mathrm{i}k\mathbf{M}_{I,j})^{-1} \mathbf{f}_j^T. \end{aligned} \tag{143}$$

The dual interface problem (7.15) is numerically scalable with respect to the mesh size. However, it is not ensured that the problem is also scalable with respect to the subdomain size. In order to attain the latter scalability, the domain decomposition is globally preconditioned. This is achieved by introducing a second level FETI-H problem derived by forcing the residual of (7.14) to be orthogonal to $\mathbf{Q}$, an interface matrix subspace. However $\mathbf{Q}$ should be chosen coarse enough to make this process efficient. This is realized by introducing the splitting

$$\lambda = \lambda^0 + \mathbf{P}\bar{\lambda} \tag{144}$$

with $\mathbf{P}$ a projector given by

$$\mathbf{P} = \mathbf{I} - \mathbf{Q}(\mathbf{Q}^T \mathbf{F}_I \mathbf{Q})^{-1} \mathbf{Q}^T \mathbf{F}_I \tag{145}$$

and λ is the initial solution according to the following:

$$\lambda^0 = \mathbf{Q}(\mathbf{Q}^T \mathbf{F}_I \mathbf{Q})^{-1} \mathbf{Q}^T \mathbf{d}. \tag{146}$$

Introducing this splitting transforms (7.14) into

$$(\mathbf{P}^T \mathbf{F}_I \mathbf{P})\bar{\lambda} = (\mathbf{F}_I \mathbf{P})\bar{\lambda} = \mathbf{P}^T \mathbf{d}, \tag{147}$$

where $\mathbf{P}$ can be considered as the right preconditioner for the dual interface problem. In [60] the GCR algorithm is used to solve (7.19). For this algorithm, at every iteration, the product $\mathbf{P}\bar{\lambda}$ entails the solution of a problem

$$(\mathbf{Q}^T \mathbf{F}_I \mathbf{Q})\mathbf{g}^p = \mathbf{h}^p, \tag{148}$$

where $\mathbf{g}^p$ and $\mathbf{h}^p$ are two interface vectors. The problem (7.20) is a projection of problem (7.14) onto the subspace $\mathbf{Q}$ and is a second-level problem. If this subspace is chosen small, the second-level problem also becomes a coarse level. If one looks at the domain decomposition where $\mathbf{Q}$ is determined inside the subdomain, the factor $\mathbf{Q}^T \mathbf{F}_I \mathbf{Q}$ becomes sparse with a structure determined from the connectivity with the other subdomains.

Extension of the FETI-H method to the case of multiple right hand sides

$$\mathbf{A}\mathbf{p}_i = \mathbf{f}_i,\ i = 1, \cdots, n_{rhs} \tag{149}$$

can be done by considering properties of GCR iterations. Since at each GCR step a search direction is generated which is $\mathbf{A}^T\mathbf{A}$-orthogonal, we have

$$(\mathbf{A}^T\mathbf{A}\mathbf{z}^i, \mathbf{z}^j) = (\mathbf{A}\mathbf{z}^i, \mathbf{A}\mathbf{z}^j) = 0. \tag{150}$$

where $\mathbf{z}$ is the search direction. If the first problem ($n_{rhs} = 1$) has been solved in n_1 GCR iterations, we have n_1 search directions available and (7.22) is valid for $i, j = 1, \cdots, n_1$, $i \neq j$.

To solve the second problem ($n_{rhs} = 2$), an initial solution which approximates the Krylov space of the first problem is constructed. This initial solution can be written as

$$\mathbf{p}_2^0 = \mathbf{S}_1\mathbf{y}_2, \tag{151}$$

where $\mathbf{S}_1 = [\mathbf{z}_1^1, \mathbf{z}_1^2, \cdots, \mathbf{z}_1^{n_1}]$ and $\mathbf{y}_2$ is determined from

$$(\mathbf{A}^T\mathbf{A}(\mathbf{x}_2 - \mathbf{x}_2^0), \mathbf{S}_1\mathbf{y}_2) = 0. \tag{152}$$

Because $\mathbf{A}\mathbf{p}_2 = \mathbf{f}_2$, we find that

$$(\mathbf{S}_1^T\mathbf{A}_1^T\mathbf{A}\mathbf{S}_1)\mathbf{y}_2 = \mathbf{S}_1^T\mathbf{A}^T\mathbf{f}_2. \tag{153}$$

Solving (7.25) is trivial because $\mathbf{S}_1^T\mathbf{A}_1^T\mathbf{A}\mathbf{S}_1$ is a diagonal matrix. To find the research direction in the second problem $\mathbf{z}_2$, the GCR algorithm is applied on $\mathbf{A}\mathbf{p}_2 = \mathbf{f}_2$.

From the fact that the solution of the $(j+1)$th problem is initiated by solving (7.25), we can extend the method to $n_{rhs} > 2$.

8 Summary and outlook

In Sections 4-7, we have given some details about methods implemented for the Helmholtz problem. In this section, we will conclude and summarize the information. We restrict on the methods which are used to solve the discrete Helmholtz problem. We also discuss our plans towards developing a general Helmholtz solver for 3-D inhomogeneous media. This is not meant to restrict ourselves within the methods listed in this section.

8.1 Review on the present methods

Various methods and approaches have been used to solve the discrete Helmholtz equation. Most efforts are aimed at methods to attack indefinite non-Hermitian matrices arising from various discretizations. The degree of indefiniteness increases with the increase of wavenumber k. High wavenumber problems are usually of practical interest.

Reference [52] notes that the use of direct methods based on Nested Dissection is restricted to 2-D cases. In general 3-D cases with inhomogenity, the direct methods suffer from too much fill-in. Some authors direct the research towards iterative methods. The early work of Bayliss, Goldstein, and Turkel [4] marked the use of iterative methods (CG) on a Helmholtz problem. It is a well-known fact that the iterative methods are not competitive without a preconditioner incorporated. For example, Bayliss, Goldstein, and Turkel [4] used the discrete Laplace operator to precondition the normal equations. For a 2-D problem with one radiation condition and three Dirichlet conditions at the boundaries, the number of iterations increases significantly for high wavenumbers. For $k = 4.16$ and $k = 21.33$ the numbers of iterations required are 308 and 915, respectively, which is considerably large in comparison with recent preconditioners.

Recently, investigations on good preconditioners for Helmholtz problems are heavily pursued. An example is AILU by Gander and Nataf [25, 26]. Numerical experiments for the 2-D homogeneous case show that AILU outperforms the ILU(0) and ILU(0.02). (For $k = 5.0$ the number of iterations to converge is 25). However, no results are reported for general 3-D inhomogeneous problems so far. Another example is the preconditioner based on separation of variables [52]. For 2-D test cases with simple inhomogenity the linear system can be solved nearly independent of the domain and mesh size. However, the preconditioner breaks down for complex inhomogeneous media.

Table 8.1 lists and summarizes some methods currently used for solving the Helmholtz equation with some notes regarding their observable properties.

The domain decomposition method has also been investigated. From a practical implementation point of view, the domain decomposition method allows us to exploit parallelism. Some methods are found to be nonscalable, e.g. the ADOP [36, 37, 38, 39]. For scalability, the FETI-H [21, 60] method seems to be a good candidate for domain decomposition. This method is not only scalable with respect to mesh size h but also with respect to the subdomain size.

For the application of Algebraic Multigrid (AMG), we refer to references [64, 63].

Table 8.1: *Iterative methods for solving the discrete Helmholtz equation for a single domain and homogeneous medium (except indicated differently).*

Iter. method	Preconditioner	Ref	note
CGNR	Laplace operator	[4]	easy to implement
			slowly converging
	Generalized SPD	[41]	easy to implement
			fast converging
			works for high k
			easy for inhomog. medium
			enhancement by MG
GMRES	Spectral prec.	[50]	
	Fast Poisson-type	[42]	multi layer DDM
	Multigrid	[17]	difficult to implement
	Generalized SPD	[41]	fast for low k
			storage problem for $k > 30$
BiCGSTAB	Sep. of Variables	[52]	inhomogeneous medium
			mesh independent
			breakdowns for high inhomogenity
QMR	SSOR	[46, 23]	Never tested for Helmholtz eq.
	ILU(0)	[27]	comparison test
			breakdowns
	ILU(0.02)	[27]	comparison test
			memory problems
	AILU	[27]	fast converging
			difficult to optimize p and q
			further testing required
GCR	Local lump	[60, 21]	DDM: FETI-H
			promising for parallelism

8.2 Problem generalization in 2D

We aim at the solutions of the Helmholtz problem for general situations, i.e. the wave propagates in a 3D inhomogeneous medium. As an intermediate step, investigations on a 2D inhomogeneous problem will be pursued. We will start with a simple problem: a 2D problem with two different media constant as depicted in Figure 8.1a. Our first idea is to generalize the AILU preconditioner to such a problem. So far, the AILU preconditioner has only been used on a constant medium. Since AILU is constructed using the parabolic factorization, the issue stemming directly from this problem is how to tackle the varying k in the parabolic factorization.

For a more realistic geophysical problem, we first consider radiation conditions on all boundaries except for the earth surface as a test case (corresponding to $z = 0$). There, we impose either the Dirichlet or Neumann conditions.

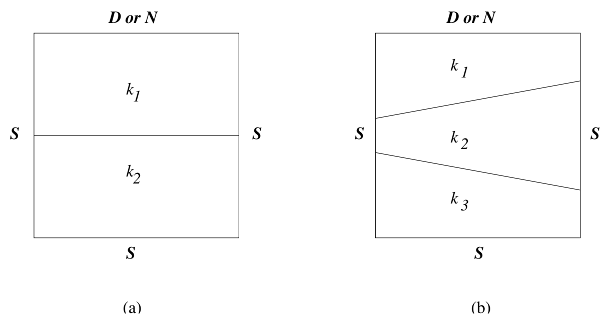

Figure 5: 2D test cases. S = Sommerfeld, D = Dirichlet, N = Neumann

A more difficult problem is the 2-D wedge problem [52] which is roughly sketched in Figure 8.1b. The same boundary conditions will be imposed.

The AILU preconditioner has a similarity with the separation of variables from the fact that the waves are decomposed allowing us to reduce the dimension of the problem. Therefore, some ideas from the separation of variables may also be implemented on the AILU preconditioner.

In this part, the study will include the spectral analysis of AILU. The analysis will be done for a moderate wavenumber which already causes the discrete system to be indefinite. From this analysis, we aim for two objectives: first, to gain a deeper insight in AILU, and secondly to find a possible improvement of the present AILU preconditioner. We summarize our plans in Table 8.2.

In order to allow less gridpoints per wavelength (about 10 to 20 per wavelenght) higher order finite difference discretizations will be used. We will investigate the nine-point stencil discussed in Section 3 providing fourth-order accuracy.

8.3 3-D Problem generalization

Generalization to 3-D problem is done by first considering a 3-D homogeneous medium case. After discretizing with the fourth-order accurate finite difference scheme, the Helmholtz problem is solved using several iterative methods. We consider QMR, BiCGSTAB, and GMRES following the experience we have from 2-D. To enhance convergence, the AILU preconditioner for 3-D problems will be constructed. We compare the results with several other preconditioners, e.g. the ILU(0) and ILU(*tol*). As an addition, spectral analysis will also be performed to look for possible enhancements.

Table 8.2: *Plans for developing 3D Helmholtz solver.*

2-D problem	Deeper insight on AILU with possible generalizations
	Relation with separation of variables and FF
	Spectral analysis
	Test 1: Simple imhomogeneous medium (Figure 8.1a)
	Test 2: Wedge problem (Figure 8.1b)
	p and q optimization issue
	Resonance problem
	Comparison with CGNR
3-D problem	Generalization of AILU in 3-D inhomogeneous problem
(AILU-related)	Spectral analysis
	Convergence acceleration: Deflation
	Convergence acceleration: FETI-H method
	Parallelisation
	Resonance problem
	Comparison with CGNR
3D problem	ILUT [54]
(non-AILU)	MRILU [10] and NGILU [65]
	AMG-type [40]
	Normal equation (with CGNR)

Following this is the solution of a 3D inhomogeneous medium problem with a typical wedge. This problem is solved using the same iterative algorithms: QMR, BiCGSTAB, and GMRES with the AILU as the preconditioner. Improvement of the convergence is investigated. We will study and implement two domain-decomposition-type methods to accelerate convergence. The first one is the Deflation [22] and the second one is the FETI-H method [21]. At this stage parallelization will also be part of the implementation. Recently, results on 3-D inhomogeneous problems are not yet available and the research wiil be directed towards a more general problem.

In the case that AILU can not improve the computational performance, investigations is turned to other matrix-based preconditioners. Falling into this category is a preconditioner with more fill-in. We will investigate the use of ILUT [54]. The MRILU and NGILU which have less complexity compared to AMG but have very similar convergence will also be studied. Investigations include the algebraic multigrid (AMG) and the normal equation with CGNR (see Table 8.2).

From our first experience with the preconditioned CGNR algorithm, it is found that this algorithm is very simple to implement and shows comparable convergence performance with QMR and the AILU precondtioner. We will also investigate an efficient implementation of CGNR in 3-D case, especially in handling the matrix inversion of the preconditioned system. The resulting performance will be compared with other methods.

References

[1] C. Ashcraft, J.W. Liu. "Robust ordering of sparse matrices using multisection". *SIAM J. Matrix Anal. Appl.* **19**(3):816-832, 1998

[2] O. Axelsson. "A survey of preconditioned iterative methods for linear systems of algebraic equations". BIT **25**:166–187, 1985

[3] A. Bamberger, P. Joly, J.E. Roberts. "Second-order absorbing boundary conditions for the wave equation: a solution for corner problem". *SIAM J. Numer. Anal.* **27**(2):323–352, 1990

[4] A. Bayliss, C.I. Goldstein, E. Turkel. "An iterative method for Helmholtz equation". *J. Comput. Phy.* **49**:443–457, 1983

[5] J.-D. Benamou, B. Despres. "Domain decomposition method for the Helmholtz equation and related optimal control problems". *INRIA RR-2791*, 1996

[6] J.-D. Benamou, B. Despres. "Domain decomposition method for the Helmholtz equation and related optimal control problems". *J. Comput. Phy.* **136**:62–88, 1997

[7] J.-P. Berenger. "A perfectly matched layer for the absorption of electromagnetic waves". *J. Comput. Phy.* **114**:185-200, 1994

[8] M. Benzi, M. Tuma. "A comparative study of sparse approximate inverse preconditioner". *Appl. Numer. Math.* **30**:305-340, 1999

[9] A.M. Bruaset. "A survey of preconditioned iterative methods". Longman Scientific & Technical, England, 1995

[10] E.F.F. Botta, F.W. Wubs. "Matrix renumbering ILU: an effective algebraic multilevel ILU preconditioner for sparse matrix". *SIAM J. Matrix Anal. Appl.* **20**(4): 1007–1026, 1999

[11] R.W. Clayton, B. Engquist. "Absorbing boundary conditions for wave-equation migration". *Geophysics* **45**(5):895–904, 1980

[12] F. Colloni, S. Ghanemi, P. Joly. "Domain decomposition methods for harmonic wave propagation: a general presentation". *INRIA RR-3473* , 1998

[13] D. Colton, R. Kress. "Invers matrix and electromagnetic scattering theory". Springer-Verlag Berlin-Heidelberg, 1998.

[14] D. Colton, R. Kress. "Integral equation methods in scattering theory". John Willey & Sons, Canada, 1983

[15] T.A. Davis, P. Amestoy, I.S. Duff. "An approximate minimum degree ordering algorithm". *Comp. Inform. Sci. Dept. TR-94-039, University of Florida*, 1995

[16] B. Despres. "Domain decomposition method and Helmhotz problems." on *in G. Cohen, L. Halpern, P. Joly (eds) Mathematical and Numerical Aspects of Wave Propagation Phenomena*:42–51, *SIAM*, 1991

[17] H.C. Elman, D.P. O'Leary. "Efficient iterative solution of the three dimensional Helmholtz equation". *J. Comput. Phy.* **142**:163-181, 1998

[18] H.C. Elman, D.P. O'Leary. "Eigenanalysis of some preconditioned Helmholtz problems". *Numer. Math.*, **83**:231–257, 1999

[19] H.C. Elman, O.G. Ernst, D.P. O'Leary. "A multigrid method enhanced by Krylov subspace iteration for discrete Helmholtz equations". *UMIACS TR No. 99-36*, University of Maryland, 1999

[20] B. Engquist, A. Majda. "Absorbing boundary conditions for the numerical simulation of waves". *Math. Comp.* **31**:629–651, 1977

[21] C. Farhat, A. Macedo, M. Lesoinne. "A two-level domain decomposition method for the iterative soluton of high frequency exterior Helmholtz problems". *Numer. Math.*: **85**: 283–308

[22] J. Frank, C. Vuik. "On the construction of deflation-based preconditioners". *SIAM J. Sci. Comput.* **23**(2):442-462, 2001

[23] R.W. Freund. "Preconditioning of symmetric, but highly indefinite linear systems". *Num. Anal. Manus. 97-3-03*, Bell Labs., Murray Hill, NJ, 1997

[24] R.W. Freund, N.M. Nachtigal. "QMR: a quasi minimum residual method for non-Hermitian Linear Systems". *Numer. Math.* **60**:315–339, 1991

[25] M.J. Gander, F. Nataf. "AILU: a preconditioner based on the analytic factorization of the elliptic operator". *Num. Linear Algebra Appl.* **7**(7):543–567,2000

[26] M.J. Gander, F. Nataf. "AILU for Helmholtz problems: a new preconditioner based on the analytic parabolic factorization". *C.R. Acad. Sci. Paris* **331**(I):261–266, 2000.

[27] M.J. Gander, F. Nataf. "AILU for Helmholtz problems: a new preconditioner based on the analytic parabolic factorization". *J. Comp. Acoustics*, **9**(4): 1499–1509, 2001

[28] A. George, J.W. Liu. "Computer solution of large sparse positive definite systems". Prentice-Hall Inc., NJ, 1981

[29] S. Ghanemi. "A domain decomposition method for Helmholtz scattering problems". On Bjϕrstad, Espedal, Keyes (editors) *the Ninth Intl. Conf. on Domain Decomposition Methods*, pp. 105–112, 1998

[30] E. Giladi, J.B. Keller. "Iterative solution of elliptic problems by approximate factorization". *J. Comp. Appl. Math.* **85**:287–313, 1997

[31] M.H. Gutknecht. "Variants of BICGSTAB for matrices with complex spectrum", *SIAM J. Sci. Comput.* **14**(5):1022–1033, 1993

[32] M.H. Gutknecht. "A completed theory of the unsymmetric Lanczos process and related algorithms, Part I", *SIAM J. Matrix Anal. Appl.***13**(2):594–639,1992

[33] M.H. Gutknecht. "A completed theory of the unsymmetric Lanczos process and related algorithms, Part II". *SIAM J. Matrix Anal. Appl.***15**(1):15–58,1994

[34] I. Harari, E. Turkel. "Accurate finite difference methods for time-harmonic wave propagation", *J. Comput. Phy.* **119**:252–270, 1995

[35] B. Hendrickson, E. Rothberg. "Improving the run time and quality of nested dissection ordering". *SIAM J. Sci. Comput.* **20**(2):468–489, 1998

[36] S. Kim. "A parallezable iterative procedure for the Helmholtz equation". *Appl. Numer. Math.* **14**:435–449, 1994

[37] S. Kim. "Parallel multidomain iterative algorithms for the Helmholtz wave equation". *Appl. Numer. Math* **17**: 411–429, 1995

[38] S. Kim. "Domain decomposition iterative procedures for solving scalar waves in the frequency domain ". *Numer. Math.* **79**:231–259, 1998

[39] S. Kim. "On the use of rational iterations and domain decomposition methods for the Helmholtz problem". *Numer. Math.* **79**:529–552, 1998

[40] D. Lahaye. "Algebraic multigrid for two-dimensional time-harmonic magnetic field computations". *PhD Thesis.* KU Leuven, Belgium, 2001

[41] A.L. Laird. "Preconditioned iterative solution of the 2D Helmholtz equation". *Report St. Hugh's College*, 2001

[42] E. Larsson. "Domain decomposition method for the Helmhotlz equation in a multilayer domain." *SIAM J. Sci. Comput.* **20**(5):1713-1731, 1999

[43] Q. Liao, G.A. McMachen. "Multifrequency viscoacoustic modeling and inversion". *Geophysics* **61**(5): 1371–1378, 1996

[44] J.W.H. Liu. "Modification of minimum degree algorithm by using multiple elimination". *ACM Trans. Math. Software* **11**:141–153, 1985

[45] M.M.M. Made. "Incomplete factorization-based preconditionings for solving the Helmholtz equation". *Int. J. Numer. Meth. Engng* **50**:1077-1101, 2001

[46] M. Malhotra, R.W. Freund, P.M. Pinsky. "Iterative solution of multiple radiation and scattering problems in structural acoustiics using a block quasi-minimal residual algorithm". *Comput. Methods Appl. Mech. Engrg.* **146**: 173–196, 1997

[47] F. Nataf, J.P. Loheac, M. Schatzman. "Parabolic approximation of the convection-diffusion equation". *Math. Comp.* **60**:515–530, 1993

[48] K. Otto, E. Larsson. "Iterative solution of the Helmholtz equation by a second order method". *SIAM J. Matrix Anal. Appl.* **21**(1):209–229, 1999

[49] C.C. Paige, M.A. Saunders. "Solution of sparse indefinite systems of linear equations". *SIAM J. Numer. Anal.* **12**(4):617–629, 1975

[50] L.F. Pavarino, E. Zampieri. "Preconditioners for spectral discretizations of Helmholtz's equation with Sommerfeld boundary conditions". *Comput. Methods Appl. Mech. Engrg.* **190**: 5341–5356, 2001

[51] R.-E. Plessix, W.A. Mulder, R.G. Pratt. "Frequency-domain finite difference migration with only few frequencies". Expanded Abstract paper MIG P1.2, Annual SEG meeting, San Antonia, 2001

[52] R.-E. Plessix, W.A. Mulder. "Separation of variables as a preconditioner for an iterative Helmholtz solver". To appear in *Applied Numerical Mathematics*, 2002

[53] D.N. Ghosh Roy, L.S. Couchman. "Inverse problems and inverse scattering of plane waves". Academic Press London, 2002

[54] Y. Saad. "Iterative methods for sparse linear systems". PWS Publishing Company Boston, 1996

[55] Y. Saad, M.H. Schultz. "GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems". *SIAM J. Sci. Stat. Comput.* **7**(12):856–869, 1986

[56] I. Singer, E. Turkel. "High-order finite difference methods for the Helmholtz equation". *Comput. Methods Appl. Mech. Engrg.* **163**:343-358, 1998

[57] G.L.G. Sleijpen, D.R. Fokkema. "BICGSTAB(L) for linear equations involving unsymmetric matrices with complex spectrum". *Elec. Trans. Numer. Anal. (ETNA) **1**:11–32, Kent State Univ., 1993*

[58] P. Sonneveld. "CGS, a fast Lanczos-type solver for nonsymmetric linear systems". *SIAM J. Sci. Stat. Comput.* **10**(1):36–52, 1989

[59] R.F. Susan-Resiga, H.M. Atassi. "A domain decomposition method for the exterior Helmholtz problem". *J. Comput. Phy.* **147**: 388–401, 1998

[60] R. Tezaur, A. Macedo, C. Farhat. "Iterative solution of large-scale acoustic scattering problems with multiple right hand-sides by a domain desomposition method with Lagrange multipliers". *Int. J. Numer. Meth. Engng* **51**:1175–1193,2001

[61] W.F. Tinney, J.W. Walker. "Direct solutions of sparse network equations using optimally ordered triagular factorization". *J. Proc. IEEE* **55**: 1801–1809,1967

[62] U. Trottenberg, C.W. Oosterlee, A. Schüller. "Multigrid". Academic Press, 2001

[63] P. Vanek, J. Mandel, M. Brezina. "Two-level algebraic multigrid for the Helmholtz problem". *Contem. Math.* **218**:349–356, 1998

[64] P. Vanek, J. Mandel, M. Brezina. "Solving a two-dimensional Helmholtz problem using algebraic multigrid". *CCM Report 111.* University of Colorado at Denver, 1997

[65] A. van der Ploeg, E.F.F. Botta, F.W. Wubs. "Nested grids ILU-decomposition (NGILU)". *J. Comput. Appl. Math.* **66**:515–526, 1996

[66] H.A. van der Vorst, "BI-CGSTAB: A fast and smoothly converging variant of BI-CG for the solution of nonsymmetric linear systems". *SIAM J. Sci. Stat. Comput.* **13**(2): 631–644, 1992

[67] H.A. van der Vorst, J.B.M. Melissen. "A Petrov-Galerkin type method for solving $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{A}$ is symmetric complex". *IEEE Trans. on Magnetics* **26**(2):706–708, 1990

[68] P. Wesseling. "An introduction to multigrid methods". John Willey and Sons, England, 1992