Delft University of Technology

Faculty Electrical Engineering, Mathematics and Computer Science (EEMCS)
Department of Applied Mathematical Analysis

Literature Report

Numerical Aspects of Iterative Solving of Linear Systems derived from Helmholtz's Problem

Author
Jok Tang

MSc Thesis Committee
prof. dr. ir. P. Wesseling (*TU Delft*)
dr. ir. C. Vuik (*TU Delft*)
dr. W. Mulder (*Shell Rijswijk*)

Delft
February 2004

Numerical Aspects of Iterative Solving of Linear Systems derived from Helmholtz's Problem

J.M. Tang

February 19, 2004

Version: 0.1.
Compiled at: February 19, 2004.

Abstract

In this report, several numerical aspects and difficulties for solving a large linear system, derived from the Helmholtz equation are overviewed.

The presentation starts with the derivation of the Helmholtz equation. Helmholtz's boundary value problem (HBVP), or briefly Helmholtz's Problem, with absorbing boundary conditions is formulated. After finite difference discretization of the problem we obtain a linear system which can be solved to obtain the solution of HBVP. The resulting matrix is not only large and sparse, but also indefinite and symmetric-complex.

Next, we give the numerical and exact solution of the one-dimensional HBVP in cases when the wavenumber and the source term are both real-valued. Moreover, special problems with variable wavenumber are also discussed. Some examples are worked out in order to get more feeling for the Helmholtz equation and to develop our intuition.

To solve HBVP we use iterative methods. Several CG-related methods are known (CGNR, COCG, GMRES, Bi-CG and Bi-CGSTAB), which are summarized and their algorithms are given. GMRES and BiCGstab are worked out in detail, because these are mostly used to solve HBVP.

We deal with a few preconditioners (like ILU, AILU, separation-of-variables and complex shifted Laplace precondtioners), which can be applied to solve Helmholtz's boundary value problem efficiently. The preconditioners are required to accelerate the iterative methods.

At the end, we give a short outlook for further research.

Contents

List of Figures

Chapter 1

Introduction

Wave propagation through an inhomogeneous acoustic medium with a constant density is described in the frequency domain by the Helmholtz equation

$$-\Delta p(\mathbf{x}) - k(\mathbf{x})^2 p(\mathbf{x}) = f(\mathbf{x}), \tag{1.1}$$

where p is the pressure field and f a point source term. The wavenumber $k = \omega/c$ is a function of the spatial coordinates $\mathbf{x} = (x, y)$ in 2-dimensional case and $\mathbf{x} = (x, y, z)$ in 3-dimensional case, because the velocity c depends on the spatial coordinates in an inhomogeneous medium.

In practical applications, for instance when modeling a seismic survey, the wavenumber is such that the Helmholtz operator has positive and negative eigenvalues. To mimic an infinite space by a finite computational domain, absorbing boundary conditions are added to (1.1), which lead to the Helmholtz boundary value problem (HBVP). More information about the HBVP and the derivation of the Helmholtz equation can be found in Chapter 2.

Chapter 3 deals with one-dimensional examples of the HBVP to get more feeling and intuition of this kind of problems. Both numerical and analytical solutions are discussed. Moreover, some spectral analysis is done as preparation for the next chapters.

To solve equation (1.1) with suitable boundary conditions, a finite-difference discretization is applied leading to the linear system

$$\mathbf{A}\mathbf{u} = \mathbf{f}. \tag{1.2}$$

The matrix $\mathbf{A}$ is a large but sparse symmetric-complex matrix. The solution $\mathbf{u}$ is represented on a grid with between 500 and 2000 points per coordinate direction in typical seismic applications. In 2 dimensions, the system (1.2) can be solved by a direct method based on LU-factorization. In 3-dimension, a direct solver is generally too expensive. To fully take into account the sparseness of $\mathbf{A}$, an iterative method should therefore be applied. We treat iterative methods based on Krylov spaces, for instance CGNR, Bi-CGSTAB

and GMRES methods, see therefore Chapter 4 where these methods and their numerical algorithms are discussed in more detail.

A preconditioner is needed to improve the convergence and robustness of iterative methods. Several preconditioners for solving HBVP are the subject of Chapter 5. Both standard and recently designed preconditioners like AILU, separation-of-variables and complex shifted Laplace are discussed in this chapter.

We end with Chapter 6, where a short summary of this report is given and a temporary plan is made for further research.

Chapter 2

Helmholtz's Boundary Value Problem

The *Helmholtz equation* is given by

$$\Delta p(\mathbf{x}) + k(\mathbf{x})^2 p(\mathbf{x}) = f(\mathbf{x}), \tag{2.1}$$

with the Laplace-operator $\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$ and $\mathbf{x} = (x, y, z)$ in $\Omega \in \mathbb{R}^3$ which is a 3-dimensional region [1]. The pressure $p(\mathbf{x})$ needs to be solved from this partial differential equation. This equation, derived from the wave equation, is used to describe for example scattering phenomena.

In this report we derive the Helmholtz equation (2.1) and formulate the Helmholtz boundary value problem (HBVP). Using a finite difference discretization we obtain a linear system belonging to HBVP.

2.1 Helmholtz equation

The central equation that governs acoustic wave propagation in a medium with constant density is the Helmholtz equation, see (2.1). Strongly related to this is the *wave equation*

$$\frac{\partial^2 p}{\partial t^2}(\mathbf{x}, t) = c(x)^2 \Delta p(\mathbf{x}, t), \tag{2.2}$$

with $\mathbf{x} \in \mathbb{R}^3, t > 0$ and $c(x)$ the speed of sound which is spatial dependent.

In the next subsections, we derive the wave equation (2.2) and the Helmholtz equation.

[1] in 1-dimension: $\mathbf{x} = x$ in the region $\Omega \in \mathbb{R}$ and 2-dimensions: $\mathbf{x} = (x, y)$ in the region $\Omega \in \mathbb{R}^2$.

2.1.1 Derivation of the wave equation

The wave equation appears in various disciplines and as a consequence there are several derivations of this equation. In this report we give the derivation with *acoustic waves*, see for instance Colton & Kress [5] and Erlangga [6]. The derivation with *elastic solids*, which is of our interest, is given in for instance Achenbach [1] and is almost identical to the derivation of acoustic waves.

Sound waves of small amplitude propagate in a region $\Omega \in \mathbb{R}^3$. In an acoustic wave problem, the local motions of particles c.q. elements in the medium are considered, while the medium itself is *motionless*.

We define field variables which are of our interest: pressure $p = p(\mathbf{x}, t)$, particle velocity $\mathbf{v} = \mathbf{v}(\mathbf{x}, t)$, density $\rho = \rho(\mathbf{x}, t)$ and entropy $S = S(\mathbf{x}, t)$. If we assume the medium to be *inviscid* and *weakly compressible*, the wave motion can be represented by the following equations: *equation of Euler* (derived from the *equation of motion*, see for instance p.48-51 of Korving & Corstens [14])

$$\frac{\partial \mathbf{v}}{\partial t} + (\mathbf{v} \cdot \nabla)\mathbf{v} = -\frac{1}{\rho}\nabla p, \tag{2.3}$$

the *continuity equation*

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = 0, \tag{2.4}$$

and the *equation of state*

$$p = f(\rho, S), \tag{2.5}$$

where f is a function depending on the nature of the medium and S follows the *adiabatic hypothesis*

$$\frac{\partial S}{\partial t} + \mathbf{v} \cdot \nabla S = 0. \tag{2.6}$$

Since this adiabatic hypothesis is imposed and no dissipative process occurs from the equations of Euler, the process during the propagation of waves is *isentropic*, which means that the entropy of the whole process is preserved.

If waves propagate through the medium, all field variables are perturbed from their quiescent conditions. If we assume only *small* perturbations caused by wave propagation, the perturbed field variables can be expressed as

$$\begin{cases} p(\mathbf{x}, t) &= p_0 + \delta p(\mathbf{x}, t); \\ \rho(\mathbf{x}, t) &= \rho_0 + \delta \rho(\mathbf{x}, t); \\ \mathbf{v}(\mathbf{x}, t) &= \mathbf{v}_0 + \delta \mathbf{v}(\mathbf{x}, t); \\ S(\mathbf{x}, t) &= S_0 + \delta S(\mathbf{x}, t), \end{cases} \tag{2.7}$$

with respectively $|\delta p| \ll |p_0|$, $|\delta \rho| \ll |\rho_0|$, $|\delta \mathbf{v}| \ll |\mathbf{v}_0|$ and $|\delta S| \ll |S_0|$. In general the parameters $p_0, \rho_0, \mathbf{v}_0$ and S_0 are functions, but in the *static state* they are *constant*. Moreover $\mathbf{v}_0 = \mathbf{0}$. The expressions in (2.7) can be used

to linearize equations (2.3)-(2.6), giving the following equations: *linearized Euler equation*

$$\frac{\partial \mathbf{v}}{\partial t} = -\frac{1}{\rho_0}\nabla p, \tag{2.8}$$

linearized continuity equation

$$\frac{\partial \rho}{\partial t} + \rho_0 \nabla \cdot \mathbf{v} = 0, \tag{2.9}$$

and *linearized equation of state*

$$\frac{\partial p}{\partial t} = \frac{\partial f}{\partial \rho}(\rho_0, S_0)\frac{\partial \rho}{\partial t}. \tag{2.10}$$

Differentiating (2.8) in space and (2.9) in time gives us, respectively,

$$\nabla \cdot \frac{\partial \mathbf{v}}{\partial t} = -\frac{1}{\rho_0}\Delta p, \tag{2.11}$$

and

$$\nabla \cdot \frac{\partial \mathbf{v}}{\partial t} = -\frac{1}{\rho_0}\frac{\partial^2 \rho}{\partial t^2}. \tag{2.12}$$

Combining (2.11) and (2.12) leads to

$$\Delta p = \frac{\partial^2 \rho}{\partial t^2}. \tag{2.13}$$

After differentiating (2.10) in time, we can substitute this in (2.13) to obtain

$$\Delta p = \left(\frac{\partial f}{\partial \rho}(\rho_0, S_0)\right)^{-1} \frac{\partial^2 p}{\partial t^2}. \tag{2.14}$$

The *speed of sound* c is in our assumed situation defined by

$$c^2 = \frac{\partial f}{\partial \rho}(\rho_0, S_0). \tag{2.15}$$

We notice again that in each part of the medium with constant ρ_0 and S_0 has spatial-dependent speed of sound.

From (2.14) and (2.15) we obtain finally the *wave equation*

$$\frac{\partial^2 p}{\partial t^2}(\mathbf{x}, t) = c^2 \Delta p(\mathbf{x}, t). \tag{2.16}$$

However, in applications we are interested in *variable* c. After several computations we can derive

$$\frac{\partial^2 p}{\partial t^2}(\mathbf{x}, t) = c(\mathbf{x})^2 \Delta p(\mathbf{x}, t), \tag{2.17}$$

wherein ρ_0 is no longer constant anymore.

2.1.2 Derivation of the wave equation in elastic solids

The derivation of the Helmholtz equation was given with acoustic waves. However, our interest is in waves in *elastic solids*. Earlier in this section there is mentioned that both derivations are roughly the same. For completeness we give shortly the steps in the elastic solid medium in onedimension. For a full understanding of this steps and for the three-dimensional case we refer to Achenbach [1].

Non-linearized theory

In a purely *one-dimensional* longitudinal motion all material particles move along parallel lines and the motion is uniform in planes normal to the direction of motion.

Suppose the *position* of a material point p at a certain time, say $t = 0$, is defined by the position x. At a later time t the position of the particle can be specified by

$$x = p(x, t). \tag{2.18}$$

The *displacement* is denoted by $u(x, t)$ and can be expressed as

$$u(x, t) = x - p(x, t). \tag{2.19}$$

As a consequence of the nonuniformity in the direction of the motion, the element undergoes a *deformation*. Then we obtain the displacement gradient $\partial u/\partial x$ as a measure of the deformation.

To make use of material derivative of $u(x, t)$ one can derive *Reynolds' transport theorem*

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{x_1}^{x_2} f(x,t)\mathrm{d}x = \int_{x_1}^{x_2}\left[\frac{\partial f(x,t)}{\partial t} + \frac{\partial f(x,t)v(x,t)}{\partial x}\right]\mathrm{d}x, \tag{2.20}$$

where $v(x, t)$ is the velocity of a certain particle and $f(x, t)$ is the total of a global quantity carried by the mass system. Notice that x_1 and x_2 in (2.20) are time-dependent.

Conservation of mass gives

$$\frac{\partial \rho(x,t)}{\partial t} + \frac{\partial \rho(x,t)v(x,t)}{\partial x} = 0, \tag{2.21}$$

where $\rho(x, t)$ is the mass density.

The principle of *balance of linear momentum* implies that

$$\tau(x,t)|_{x_1}^{x_2} = \frac{\mathrm{d}}{\mathrm{d}t}\int_{x_1}^{x_2} \rho(x,t)v(x,t)\mathrm{d}x. \tag{2.22}$$

Here $\tau(x, t)$ defines the *stress* at position x, where body forces are not taken into account. With a few computations we can derive

$$c^2\frac{\partial^2 u}{\partial x^2} = \frac{\partial^2 u}{\partial t^2}, \tag{2.23}$$

with

$$c^2 = \frac{1}{\rho_0} \frac{\mathrm{d}\mathcal{S}}{\mathrm{d}\,(\partial u/\partial x)}. \tag{2.24}$$

In expression (2.24) ρ_0 is the *mass density* as a function of the reference configuration and $\tau(x,t)$ has the following form

$$\tau(x,t) = \mathcal{S}(\partial u/\partial x), \tag{2.25}$$

with $\partial u/\partial x$ the *displacement gradient*.

Linearized theory

Although it is possible to determine solutions for certain one-dimensional problems governed by the nonlinear theory, there are often rather substantial complications. The complications disappear altogether when the theory is appropriately linearized.

Let us consider the constitutive relation (2.25) for the special case that $\tau(x,t)$ is proportional to $\partial u/\partial x$:

$$\tau(x,t) = S_1 \frac{\partial u(x,t)}{\partial x}, \tag{2.26}$$

where S_1 is a constant. For some materials this relation may be an approximation applicable only when $|\partial u/\partial x| \ll 1$. For other materials and within the context of a one-dimensional theory it may be exact in the sense that it may apply for quite large values of $|\partial u/\partial x|$.

If expression (2.26) holds, equation (2.23) reduces finally to the *linear wave equation*

$$\frac{\partial^2 u}{\partial x^2} = \frac{1}{c^2} \frac{\partial^2 u}{\partial t^2}, \tag{2.27}$$

where

$$c^2 = \frac{S_1}{\rho_0}. \tag{2.28}$$

In the same way we can derive the following expression for non-constant c with variable ρ_0:

$$\frac{\partial^2 u}{\partial x^2} = \frac{1}{c(x)^2} \frac{\partial^2 u}{\partial t^2}. \tag{2.29}$$

2.1.3 Derivation of the Helmholtz equation

We consider *time-harmonic* (standing) waves of *frequency* $\omega > 0$ with *time independent* pressure $\tilde{p}$ of the form

$$p(\mathbf{x},t) = \cos(\omega t)\tilde{p}(\mathbf{x}). \tag{2.30}$$

This may also be written as

$$p(\mathbf{x},t) = \Re e^{-i\omega t}\tilde{p}(\mathbf{x}), \tag{2.31}$$

where $i^2 = -1$ and $\Re$ indicates that the real part of the right-hand-side of (2.31) should be taken. We shall omit this symbol $\Re$ in further analysis for brevity and keep in mind that we are interested in the real part of imaginair solutions. However, it appears that sometimes the complex part of the solutions do give information, so these are also treated in some applications.

With the aid of (2.31) one derives that

$$\frac{\partial^2 p(\mathbf{x},t)}{\partial t^2} = -\omega^2 e^{-i\omega t}\tilde{p}(\mathbf{x}). \tag{2.32}$$

Using (2.31) and (2.32), the wave equation (2.17) reduces to the *reduced wave equation* or *Helmholtz equation*

$$\Delta\tilde{p}(\mathbf{x}) + k(\mathbf{x})^2\tilde{p}(\mathbf{x}) = 0, \tag{2.33}$$

where the *wave number* k is given by

$$k(\mathbf{x}) = \frac{\omega}{c(\mathbf{x})}. \tag{2.34}$$

This equation carries the name of *Von Helmholtz* for his contributions to mathematical acoustics and electromagnetics.

Figure 2.1: Hermann Ludwig Ferdinand von Helmholtz (1821-1894), one of the greatest German physicists and mathematicians.

If there is no ambiguity in the context, we denote $p(\mathbf{x})$ by $\tilde{p}(\mathbf{x})$ in this report. Moreover, if we assume a *harmonic source* in the neighbourhood of Ω, i.e.,f a harmonic disturbance $e^{-i\omega t}f(\mathbf{x})$ which is producing the waves, then the source appears on the right-hand-side of (2.33). We obtain the *inhomogeneous Helmholtz equation*

$$\Delta p(\mathbf{x}) + k(\mathbf{x})^2 p(\mathbf{x}) = f(\mathbf{x}). \tag{2.35}$$

where f is a given *source term*. Equation (2.35) is identical to equation (2.1).

2.2 Helmholtz's problem

In this section we define Helmholtz's boundary value problem.

2.2.1 Boundary conditions

Proper boundary conditions are required to find a unique solution of the Helmholtz equation.

For a *sound-soft* obstacle the pressure of the total wave vanishes on the boundary. Consider the scattering of a given incoming wave u^i by sound-soft obstacle $D \in \Omega$ and define u^s by the scattered wave. Then the total wave $u = u^i + u^s$ must satisfy the wave equation in the exterior $\mathbb{R}^3 \setminus D$ of D and a Dirichlet boundary condition $u = 0$ on ∂D.

Similarly, the scattering from *sound-hard* obstacles leads to a Neumann boundary condition $\partial u/\partial n = 0$ on ∂D where n is the unit outward normal to ∂D, since here the normal velocity of the acoustic wave vanishes on the boundary.

Impedance boundary condition

More generally, allowing obstacles for which the normal velocity on the boundary is proportional to the excess pressure on the boundary leads to an *impedance boundary condition* of the form

$$\frac{\partial p(\mathbf{x})}{\partial n} + i\lambda p(\mathbf{x}) = 0 \tag{2.36}$$

on $\mathbf{x} \in \partial D$ with $\lambda > 0$ [2]. Such a condition can be used for the boundaries of $\partial\Omega$.

Sommerfeld radiation condition

Related to condition (2.36) is the first-order *Sommerfeld radiation condition*

$$\frac{\partial p(\mathbf{x})}{\partial n} - ik(\mathbf{x})p(\mathbf{x}) = 0, \quad x \in \partial\Omega. \tag{2.37}$$

This condition is widely used in Helmholtz problems, see for instance Erlangga [7].

[2] See section 2 (p.15) of Colton & Kress [5].

Absorbing condition

If we assume

$$p(\mathbf{x}) = \mu e^{ik\mathbf{x}}, \quad \mu \in \mathbb{C}, \tag{2.38}$$

then it can be derived that

$$\frac{\partial p(\mathbf{x})}{\partial x} - ikp(\mathbf{x}) = 0, \tag{2.39}$$

which is used as condition at the two faces of the boundary, where the unit outward normal is in x-direction. Similar conditions as (2.39) are defined for the other faces.

In section 2 of Mulder & Plessix [17], condition (2.39) is given as

$$\pm\frac{\partial p(\mathbf{x})}{\partial n} - ik(\mathbf{x})p(\mathbf{x}) = 0, \tag{2.40}$$

where n representing x, y or z, depending on the boundary. Expression (2.40) is the so-called *first order absorbing boundary condition*. The aim of this artificial condition is to represent our finite domain Ω as part of an *infinite* domain, as accurately as possible.

One can write (2.40) in the following concise way

$$\frac{\partial p(\mathbf{x})}{\partial n} + ik(\mathbf{x})p(\mathbf{x}) = 0, \tag{2.41}$$

where n is the unit outward normal to the boundaries.

Choice of boundary conditions

In our further analysis we take the absorbing boundary condition (2.41) as our boundary condition on $\partial\Omega$. Later on we shall choose other values of λ in order to represent the real situation as accurately as possible.

We assume our region Ω to be a *box-shaped* computational domain with dimensions $[x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}] \times [z_{\min}, z_{\max}]$, which are real-valued. Then condition (2.37) holds only for the six faces, because the outward normal is *not* defined for the edges and corners of the box.

For each edge of two bordered faces α and β we define the following condition

$$\frac{1}{\sqrt{2}}\left(\frac{\partial p(\mathbf{x})}{\partial n_\alpha} + \frac{\partial p(\mathbf{x})}{\partial n_\beta}\right) + ik(\mathbf{x})p(\mathbf{x}) = 0, \tag{2.42}$$

where n_α and n_β are the unit outward normals of α respectively β.

In almost similar way as above we can define the boundary condition for the corners of the box

$$\frac{1}{\sqrt{3}}\left(\frac{\partial p(\mathbf{x}, t)}{\partial n_\alpha} + \frac{\partial p(\mathbf{x})}{\partial n_\beta} + \frac{\partial p(\mathbf{x})}{\partial n_\zeta}\right) + ik(\mathbf{x})p(\mathbf{x}) = 0, \tag{2.43}$$

where α, β and ζ are the three faces which border the considered corner. See Heikkola, Rossi, Toivanen [12] for accompanying literature.

Later on it turns out that the conditions at the edges and corners will not be used in the numerical methods, see next chapter.

2.2.2 Helmholtz's boundary value problem

Wave propagation in an *inhomogeneous* medium is considered, which means that the medium can consist of several parts. Let $\Omega \in \mathbb{R}^3$ be the region enclosing the medium with scatterers and boundary $\partial\Omega$. Furthermore, let n denote the *unit outward normal* to ∂D. Then, the *Helmholtz's boundary value problem* for wave propagation in an *inhomogeneous* medium can be defined as follows:

Helmholtz's boundary value problem (HBVP)
Find the total field $p(\mathbf{x})$ in an inhomogenous medium Ω such that

$$\left(\Delta + k(\mathbf{x})^2\right) p(\mathbf{x}) = f(\mathbf{x}), \quad \mathbf{x} \in \Omega, \tag{2.44}$$

with boundary conditions

$$\left(\frac{\partial}{\partial n} + ik(\mathbf{x})\right) p(\mathbf{x}) = 0, \quad \mathbf{x} \in \partial\Omega, \tag{2.45}$$

where f is a given source term, n is the unit outward normal and

$$k = \frac{w}{c(\mathbf{x})}, \tag{2.46}$$

with wave-frequency $\omega > 0$ and sound of speed c.

HBVP can be solved by an integral equation method by transformation into a Fredholm integral equation, see for instance Colton & Kress [5]. 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.

In this report, we aim at numerical solutions of HBVP by applying a *finite difference approach*. In many practical applications, the linear system obtained after discretization is very large. This requires specials methods to solve the linear system *efficiently* with a reasonable *accuracy*.

2.3 Analytical solution of HBVP

In this section we deal with the analytical solutions of the Helmholtz boundary value problem (HBVP), where *Green's functions* are used.

2.3.1 Homogeneous solution

First we consider the homogeneous Helmholtz equation *without* source-term and with *constant* positive wavenumber k, i.e.,

$$\Delta p(\mathbf{x}) + k^2 p(\mathbf{x}) = 0. \tag{2.47}$$

Straightforward differentiation shows that for fixed $\mathbf{y} \in \mathbb{R}^3$ the *fundamental solution*

$$G(\mathbf{x}, \mathbf{y}) := -\frac{e^{ik|\mathbf{x}-\mathbf{y}|}}{4\pi|\mathbf{x}-\mathbf{y}|}, \quad \mathbf{x} \neq \mathbf{y}, \tag{2.48}$$

satisfies this Helmholtz equation in $\mathbb{R}^3 \setminus \{\mathbf{y}\}$.

$G(\mathbf{x}, \mathbf{y})$ is called a Green's function. Later on we will see why this function makes sense.

2.3.2 General solution

By definition the Green's function for the Helmholtz equation satisfies

$$(\Delta + k^2)G(\mathbf{x}, \mathbf{y}) = \delta(\mathbf{x} - \mathbf{y}) \tag{2.49}$$

subject to a suitable boundary (radiation) condition, in such a way that $f(\mathbf{x}) \to 0$ as $|\mathbf{x}| \to \infty$. So in (2.49) the source-term is represented as a *point* source. Then it follows that

$$\Phi(\mathbf{x}) = \int_\Omega G(\mathbf{x}, \mathbf{y}) f(\mathbf{y}) d\mathbf{y}, \tag{2.50}$$

is the solution of the homogeneous Helmholtz equation with again constant positive wavenumber k, i.e.

$$(\Delta + k^2)\Phi(\mathbf{x}) = f(\mathbf{x}). \tag{2.51}$$

Note that no determination of arbitrary constants is required, since $\Phi(\mathbf{x})$ as given by the Green's function integral formula automatically satisfies the boundary conditions.

With simple but laborious computional work we find that $G(\mathbf{x}, \mathbf{y})$ in (2.49) has the following form

$$G(\mathbf{x}, \mathbf{y}) := -\frac{e^{ik|\mathbf{x}-\mathbf{y}|}}{4\pi|\mathbf{x}-\mathbf{y}|}, \quad \mathbf{x} \neq \mathbf{y}, \tag{2.52}$$

This is exactly (2.48).

The usefulness of a Green's function solution rests on the fact that the Green's function is independent of the non-homogeneous term in the differential equation.

By definition of the Green's function, this should satisfy the homogeneous equations except at the source point where an appropiate singularity

must exist. Furthermore, the Green's function should vanish on the boundaries, i.e.,

$$G(\mathbf{x}, \mathbf{y}) = 0 \quad \text{for } \mathbf{x} \in \partial\Omega, \tag{2.53}$$

Moreover, we can notice that the Green's function in the Helmholtz equation is different for the one-, two- and three-dimensional cases. In this section we have only considered the 3-dimensional case.

To summarize: the solution of the inhomogeneous Helmholtz problem

$$(\Delta + k^2)\Phi(\mathbf{x}) = f(\mathbf{x}), \tag{2.54}$$

which satisfies the outward radiation boundary condition is given by

$$\Phi(\mathbf{x}) = -\frac{1}{4\pi} \int_\Omega \frac{e^{ik|\mathbf{x}-\mathbf{y}|}}{|\mathbf{x}-\mathbf{y}|} f(\mathbf{y}) d\mathbf{y}. \tag{2.55}$$

2.4 Finite difference discretization

We apply a discretization to HBVP to obtain a suitable equation for numerical computation. Several discretization schemes are available and can be used to solve the problem. In this research we use the *finite difference discretization* with second order accurary.

2.4.1 Finite difference method

The 3-dimensional domain of interest Ω with boundary $\partial\Omega$ is discretized vertex-centered on an equidistant grid with $L+2$, $M+2$ and $N+2$ points in respectively x-, y- and z-direction, i.e.

$$\begin{cases} x_l = x_0 + l\Delta x, & \text{for } l = 0, 1, \ldots, L+1; \\ y_m = y_0 + m\Delta y, & \text{for } m = 0, 1, \ldots, M+1; \\ z_n = z_0 + n\Delta z, & \text{for } n = 0, 1, \ldots, N+1. \end{cases} \tag{2.56}$$

with constant parameters $\Delta x, \Delta y$ and Δz and where $\mathbf{x}_{min} = (x_0, y_0, z_0)$ and $\mathbf{x}_{max} = (x_{L+1}, y_{M+1}, z_{N+1})$.

We apply a finite difference stencil to approximate the first derivative

$$D_x p_{l,m,n} = \frac{p_{l+1,m,n} - p_{l,m,n}}{\Delta x}, \tag{2.57}$$

which is first order accurate, i.e.

$$\frac{\partial p_{l,m,n}}{\partial x} = D_x p_{l,m,n} + O(\Delta x), \tag{2.58}$$

see for instance Van Kan & Segal [13], p.30, or Vuik [29], p.21-22. The notation $p_{l,m,n} \equiv p(x_l, y_m, z_n)$ is used in (2.58) for simplicity.

Moreover, in the interior we apply a standard finite difference stencil

$$D_{xx}p_{l,m,n} = \frac{p_{l-1,m,n} - 2p_{l,m,n} + p_{l+1,m,n}}{\Delta x^2}. \tag{2.59}$$

Equation (2.59) is derived from a *Taylor's expansion* in order to approximate the second order derivative in the x-direction. For sufficiently smooth p in Ω, the second order derivative can be represented as

$$\frac{\partial^2 p_{l,m,n}}{\partial x^2} = D_{xx}p_{l,m,n} + O(\Delta x^2), \tag{2.60}$$

which means second-order accuracy in space if we use (2.59) as our standard stencil. The proof of (2.60) can be found in for instance [13], p.29.

Equations (2.59) and (2.60) are derivatives in the x-direction. Similar expressions can be found for the y- and z-directions.

2.4.2 Interior points

In the *interior points*, discretization of the Helmholtz equation (2.44) results in the following equation

$$\begin{aligned} &\frac{p_{l-1,m,n} - 2p_{l,m,n} + p_{l+1,m,n}}{\Delta x^2} + \frac{p_{l,m-1,n} - 2p_{l,m,n} + p_{l,m+1,n}}{\Delta y^2} \\ &+\frac{p_{l,m,n-1} - 2p_{l,m,n} + p_{l,m,n+1}}{\Delta z^2} + k_{l,m,n}^2 p_{l,m,n} = f_{l,m,n}, \end{aligned} \tag{2.61}$$

for $l = 1, \ldots, L$, $m = 1, \ldots, M$ and $n = 1, \ldots, N$.

2.4.3 Boundary points

In the case $l = 0$ *forward* discretization of the boundary condition (2.41) gives

$$\frac{p_{0,m,n} - p_{1,m,n}}{\Delta x} + ik_{0,m,n}p_{0,m,n} = 0, \tag{2.62}$$

for $m = 1, \ldots, M$ and $n = 1, \ldots, N$. In (2.62) we used $\partial/\partial n = -\partial/\partial x$ and

$$D_x p_{0,m,n} = \frac{p_{1,m,n} - p_{0,m,n}}{\Delta x}. \tag{2.63}$$

With equation (2.62) we derive the following expression

$$p_{0,m,n} = \frac{1}{1 + ik_{0,m,n}\Delta \mathbf{x}} p_{1,m,n}, \tag{2.64}$$

which we can substitute in equation (2.61) to eliminate $p_{0,m,n}$.

In the case $l = L + 1$ we use *backward* discretization to obtain

$$p_{L+1,m,n} = \frac{1}{1 + ik_{L+1,m,n}\Delta \mathbf{x}} p_{L,m,n}, \tag{2.65}$$

with $m = 1, \ldots, M$ and $n = 1, \ldots, N$.

In similar way as above we derive the expressions for the other boundaries. These boundary-elements will be eliminated. Thus, in each direction there are resp. L, M and N elements left, which have to be determined.

2.4.4 Linear system

After applying equation (2.61) to all *interior* points $p_{l,m,n}$ in Ω and eliminating all *boundary* points, one obtains a *linear system*

$$\mathbf{Ap} = \mathbf{f}, \tag{2.66}$$

where $\mathbf{p}$ and $\mathbf{f}$ both are vectors with LMN elements and

$$\begin{array}{rcl} p(l + (m-1)L + (n-1)L \times M) & = & p_{l,m,n}, \\ f(l + (m-1)L + (n-1)L \times M) & = & f_{l,m,n}, \end{array} \tag{2.67}$$

for all $l = 1, \ldots, L$, $m = 1, \ldots, M$ and $n = 1, \ldots, N$.

In the linear system (2.66) $\mathbf{A}$ is an $(LMN) \times (LMN)$-matrix with seven non-zero diagonals. The non-zero elements of the main diagonal are given by

$$A(d,d) = -\left(\frac{2}{\Delta x^2} + \frac{2}{\Delta y^2} + \frac{2}{\Delta z^2}\right) + k_{l,m,n}^2 + \gamma_{l,m,n}. \tag{2.68}$$

The non-zero elements of the subdiagonals are

$$\begin{array}{lll} A(d, d+1) & = A(d, d-1) & = \dfrac{1}{\Delta x^2}, \\ A(d, d+L) & = A(d, d-L) & = \dfrac{1}{\Delta y^2}, \\ A(d, d+L \times M) & = A(d, d-L \times M) & = \dfrac{1}{\Delta z^2}. \end{array} \tag{2.69}$$

In (2.68) and (2.69) the symbol d represents a counting parameter. For instance, expression (3.11) holds for $d = 1, 2, ..., LMN$.

Furthermore, the parameter $\gamma_{l,m,n}$ is determined by the position of the points $p_{l,m,n}$ in the region. For interior points these are zero, i.e.

$$\gamma_{l,m,n} = 0 \quad \text{for } l = 1, \ldots, L, \ m = 1, \ldots, M \text{ and } \ n = 1, \ldots, N. \tag{2.70}$$

For points at the boundaries, the expression depends on the *type* of boundary discretization.

We have assumed the region Ω as a box with 6 faces, 12 edges and 8 corners. If we choose the discretization defined in paragraph 2.4.3, the

following expression yields for $\gamma_{l,m,n}$ on the *faces*, excluded the edges and corners [3]

$$\gamma_{l,m,n} = \begin{cases} \frac{1}{\Delta x^2(1+ik_{0,m,n}\Delta x)} & \text{if } l = 1; \\ \frac{1}{\Delta x^2(1+ik_{L+1,m,n}\Delta x)} & \text{if } l = L; \\ \\ \frac{1}{\Delta y^2(1+ik_{l,0,n}\Delta y)} & \text{if } m = 1; \\ \frac{1}{\Delta y^2(1+ik_{l,M+1,n}\Delta y)} & \text{if } m = M; \\ \\ \frac{1}{\Delta z^2(1+ik_{l,m,0}\Delta z)} & \text{if } n = 1; \\ \frac{1}{\Delta z^2(1+ik_{l,m,N+1}\Delta z)} & \text{if } n = N. \end{cases} \tag{2.71}$$

The total number of nonzero elements of $\mathbf{A}$ is at most $7LMN$ of $(LMN)^2$ total entries, indicating *sparsity*. Moreover, matrix $\mathbf{A}$ is *symmetric*, but *non-Hermitian* [4]. This follows immediately from (2.71).

[3] The expressions on the edges and corners for $\gamma_{l,m,n}$ are combinations of those of expression (2.71).

[4] $\mathbf{A}$ is *Hermitian* when the matrix has the property $\bar{\mathbf{A}}^T = \mathbf{A}$.

Chapter 3

One-dimensional HBVP and Some Examples

In this report we give two cases of the one-dimensional *Helmholtz's boundary value problem* (HBVP): one with *constant* wavenumber and one with *variable* wavenumber. We give both *analytical* and *numerical* solutions of these problems.

3.1 HBVP with constant wavenumber

We consider the one-dimensional version of the HBVP in the domain $\Omega = [0, \pi]$, where the wavenumber and the sourceterm are *real*-valued constants; in formulae

$$\left(\frac{\mathrm{d}^2}{\mathrm{d}x^2} + k^2\right) p(x) = f, \qquad x \in (0, \pi), \tag{3.1}$$

with boundary conditions

$$\begin{aligned} \left(-\frac{\mathrm{d}}{\mathrm{d}x} + ik\right) p(x) &= 0, \qquad x = 0, \\ \left(\frac{\mathrm{d}}{\mathrm{d}x} + ik\right) p(x) &= 0, \qquad x = \pi, \end{aligned} \tag{3.2}$$

where $k \in \mathbb{R}^+$ and $f \in \mathbb{R}$.

3.1.1 Analytical solution

One can easily derive the following solution of (3.1)

$$p(x) = \frac{f}{k^2} + c_1 \cos kx + c_2 \sin kx, \tag{3.3}$$

with $c_1, c_2 \in \mathbb{C}$, using for instance the method of *variation of parameters*, see section 3.7 of Boyce & DiPrima [4].

The constants c_1 and c_2 can be determined with the aid of (3.2). If we first assume k to be a *positive integer*, i.e., $k = \mathbb{N}$, then the constants can be derived with the following linear system

$$\begin{bmatrix} ik & -k \\ ik\theta & k\theta \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} -i\frac{f}{k} \\ -i\frac{f}{k} \end{bmatrix} \tag{3.4}$$

where

$$\theta = \begin{cases} -1 & \text{if } k \text{ is odd;} \\ 1 & \text{if } k \text{ is even.} \end{cases} \tag{3.5}$$

System (3.4) can be easily solved to obtain the following general solution of (3.1) and (3.2)

$$p(x) = \begin{cases} \dfrac{f}{k^2} + i\dfrac{f}{k^2}\sin kx & \text{if } k \text{ is odd;} \\ \dfrac{f}{k^2} - \dfrac{f}{k^2}\cos kx & \text{if } k \text{ is even.} \end{cases} \tag{3.6}$$

When we are only interested in the *real* part of the solutions, we can write

$$p(x) = \begin{cases} \dfrac{f}{k^2} & \text{if } k \text{ is odd;} \\ \dfrac{f}{k^2} - \dfrac{f}{k^2}\cos kx & \text{if } k \text{ is even.} \end{cases} \tag{3.7}$$

The imaginary part of solution (3.6) can be written as

$$p(x) = \begin{cases} i\dfrac{f}{k^2}\sin kx & \text{if } k \text{ is odd;} \\ 0 & \text{if } k \text{ is even.} \end{cases} \tag{3.8}$$

Observe that for even k the solution is always real-valued.

In similar way we can derive expressions for general k, i.e., for arbitrary $k \in \mathbb{R}^+$. Notice that the wavenumber k has a large influence on the shape of the solution.

3.1.2 Numerical solution

We apply a finite difference discretization with

$$x_l = x_0 + l\Delta x, \quad \text{for } \; l = 0, 1, \ldots, N+1, \tag{3.9}$$

to obtain a linear system of (3.1) and (3.2)

$$\mathbf{Ap} = \mathbf{f}, \tag{3.10}$$

where $\mathbf{p}$ and $\mathbf{f}$ both are vectors with N elements. The elements of the main diagonal of $\mathbf{A}$ are given by

$$\mathbf{A}(d,d) = -\left(\frac{2}{\Delta x^2}\right) + k^2 + \gamma_d, \tag{3.11}$$

where

$$\gamma_d = \begin{cases} \frac{1}{\Delta x^2(1+ik\Delta x)} & \text{if } d = 1 \text{ or } d = N\ ; \\ 0 & \text{otherwise.} \end{cases} \tag{3.12}$$

The non-zero elements of the subdiagonals are

$$\mathbf{A}(d,d+1) \quad = \mathbf{A}(d,d-1) \quad = \frac{1}{\Delta x^2}. \tag{3.13}$$

This linear system (3.10) will be solved with a *direct* method using *Matlab*.

In the next two subsections we shall consider two examples of the HBVP and determine both the analytical and numerical solution.

3.1.3 Example 1: $k = 1$ and $f = 2$

We consider the HBVP with $k = 1$ and $f = 2$, i.e.

$$\begin{cases} \dfrac{\mathrm{d}^2 p(x)}{\mathrm{d}x^2} + p(x) &= 2, & x \in (0,\pi), \\ -\dfrac{\mathrm{d}p(x)}{\mathrm{d}x} + ip(x) &= 0, & x = 0, \\ \dfrac{\mathrm{d}p(x)}{\mathrm{d}x} + ip(x) &= 0, & x = \pi, \end{cases} \tag{3.14}$$

The real part of the analytical solution of system (3.14) is

$$p(x) = 2, \tag{3.15}$$

and the imaginary part is given by

$$p(x) = 2\sin x, \tag{3.16}$$

using expressions (3.7) and (3.8).

We compute the numerical solution of (3.14) with the command

`Ex1(100,1,2)`,

where we have taken 100 elements in the discretization. The code of the program can be found in **Appendix A.1**. The obtained plot of the solution is given in Figure 3.1. We see that this is approximately equal to the analytical solution (3.15). We notice that the numerical solution is not exact,

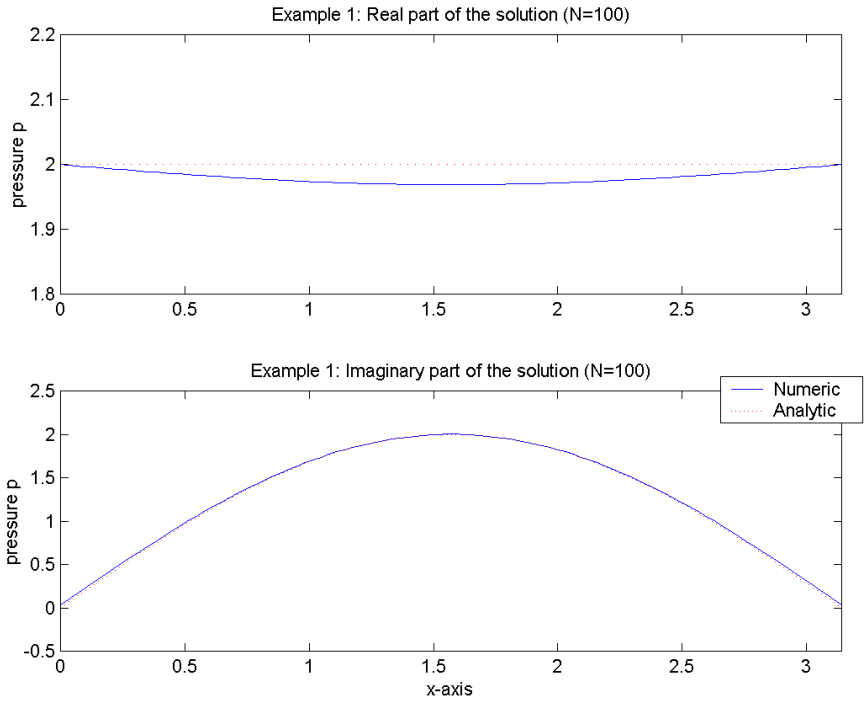

Figure 3.1: PLOT OF THE SOLUTION OF HBVP WITH $k = 1$ AND $f = 2$.

since we have dealt with a *complex* solution where the real and imaginary part have been taken. This causes *truncation* errors.

The real part of the eigenvalues of this problem are both positive and negative, since the extreme eigenvalues (i.e., $\lambda_{\max}$ and $\lambda_{\min}$) are given by

```
1.0e+003 *

0.0007 - 0.0005i

-4.0508 - 0.0000i
```

So the system is *indefinite*, which has direct consequences for the choice of iterative methods to solve the linear system, see next chapter.

3.1.4 Example 2: $k = 2$ and $f = 1$

We consider the HBVP with $k = 2$ and $f = 1$, i.e.

$$\begin{cases} \dfrac{\mathrm{d}^2 p(x)}{\mathrm{d}x^2} + 4p(x) &= 1, & x \in (0, \pi), \\ -\dfrac{\mathrm{d}p(x)}{\mathrm{d}x} + 4ip(x) &= 0, & x = 0, \\ \dfrac{\mathrm{d}p(x)}{\mathrm{d}x} + 4ip(x) &= 0, & x = \pi, \end{cases} \tag{3.17}$$

The analytical solution of system (3.17) is

$$p(x) = \frac{1}{4} - \frac{1}{4}\cos 2x. \tag{3.18}$$

We compute the numerical solution of (3.14) with the command

```
Ex2(100,2,1).
```

See **Appendix A.2** for the code.

The obtained plot of the solution is given in Figure 3.2, which is practically equal to the analytical solution (3.18).

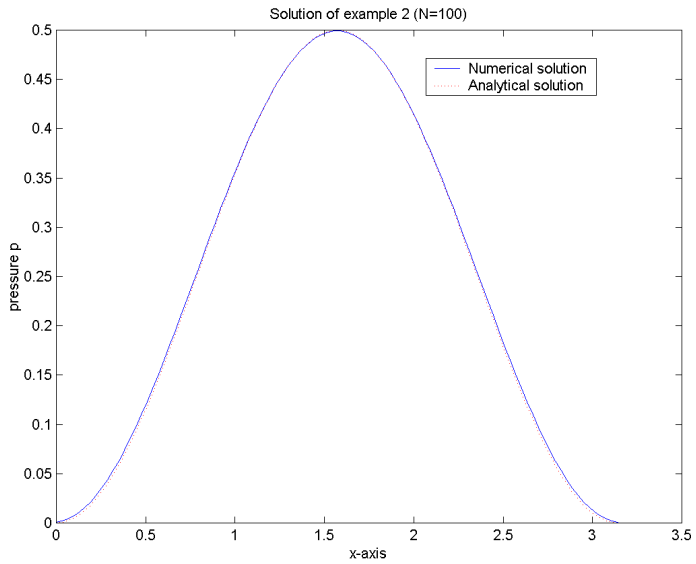

Figure 3.2: Plot of the solution of HBVP with $k = 2$ and $f = 1$.

The extreme eigenvalues are

```
1.0e+003 *
0.0033 - 0.0005i
-4.0478 - 0.0000i
```

So this system is also indefinite.

3.2 HBVP with variable wavenumber

One deals with *non*-constant values of k in the most of practical problems. To illustrate this we consider HBVP wherein k behaves as a step function, so we consider the following problem

$$\begin{cases} \dfrac{\mathrm{d}^2 p(x)}{\mathrm{d}x^2} + k^2 p(x) &= f, \qquad x \in (0, \frac{1}{2}\pi) \cup (\frac{1}{2}\pi, \pi), \\ -\dfrac{\mathrm{d}p(x)}{\mathrm{d}x} + ikp(x) &= 0, \qquad x = 0, \\ \dfrac{\mathrm{d}p(x)}{\mathrm{d}x} + ikp(x) &= 0, \qquad x = \pi, \end{cases} \tag{3.19}$$

where $f \in \mathbb{R}$ and

$$k = \begin{cases} k_1, & x \in [0, \frac{1}{2}\pi], \\ k_2, & x \in (\frac{1}{2}\pi, \pi], \end{cases} \tag{3.20}$$

where $k_1, k_2 > 0$.

The same boundary conditions as in the previous problem are used here. We need two extra conditions in order to derive a unique solution of problem (3.19)

$$\begin{cases} p_1(x) &= p_2(x), \qquad x = \frac{1}{2}\pi, \\ \dfrac{\mathrm{d}p_1(x)}{\mathrm{d}x} &= \dfrac{\mathrm{d}p_2(x)}{\mathrm{d}x}, \qquad x = \frac{1}{2}\pi, \end{cases} \tag{3.21}$$

where $p_1(x)$ is the solution at $x \in [0, \frac{1}{2}\pi]$ and $p_2(x)$ is the solution at $x \in (\frac{1}{2}\pi, \pi]$. Thus, *connecting* conditions (3.21) imply that $p(x)$ has to be *continuous* and *differentiable* in $x = \frac{1}{2}\pi$.

3.2.1 Analytical solution

One can derive the following solution of (3.19)

$$p(x) = \begin{cases} p_1(x) &= \dfrac{f}{k_1^2} + c_1 \cos k_1 x + c_2 \sin k_1 x, \qquad x \in [0, \frac{1}{2}\pi], \\ p_2(x) &= \dfrac{f}{k_2^2} + c_3 \cos k_2 x + c_4 \sin k_2 x, \qquad x \in (\frac{1}{2}\pi, \pi]. \end{cases} \tag{3.22}$$

Using the boundary conditions and connecting conditions (3.21) the constants $c_1, \ldots, c_4 \in \mathbb{C}$ can be determined, see example 3 for an illustration.

3.2.2 Numerical solution

Using the same finite difference discretization, like in the previous problem, we can derive

$$\mathbf{A}\mathbf{p} = \mathbf{f}, \tag{3.23}$$

where $\mathbf{p}$ and $\mathbf{f}$ both are vectors with N elements, where we shall take an *odd*-valued N for simplicity.

The connecting conditions (3.21) can be implemented in the following way.

- Continuity in $\frac{1}{2}\pi$ is automatically satisfied, because we have exact one gridpoint in that point, namely $p_{\frac{1}{2}(N+1)}$.
- The second condition can be discretized in several ways. In the next subsections we treat two methods and give the resulting matrix $\mathbf{A}$.

Method 1

We discretize the second condition of (3.21) as follows

$$\frac{p_{\frac{1}{2}(N+1)} - p_{\frac{1}{2}(N-1)}}{\Delta x} = \frac{p_{\frac{1}{2}(N+3)} - p_{\frac{1}{2}(N+1)}}{\Delta x}. \tag{3.24}$$

Remember that we have taken odd-valued N. Furthermore, notice that one-side difference is applied in (3.24) to discretize the first derivative. Now, expression (3.24) leads to the equation

$$-p_{\frac{1}{2}(N-1)} + 2p_{\frac{1}{2}(N+1)} - p_{\frac{1}{2}(N+3)} = 0. \tag{3.25}$$

Notice that this indirect implies that the second derivative in $p_{\frac{1}{2}(N+1)}$ is equal to zero.

The original equation of the point $p_{\frac{1}{2}(N+1)}$ can be replaced by equation (3.25).

Now, the elements of the main diagonal of $\mathbf{A}$ are given by

$$\mathbf{A}(d,d) = \begin{cases} -\left(\dfrac{2}{\Delta x^2}\right) + k_1^2 + \gamma_d, & d < \frac{1}{2}(N+1); \\ 2, & d = \frac{1}{2}(N+1); \\ -\left(\dfrac{2}{\Delta x^2}\right) + k_2^2 + \gamma_d, & d > \frac{1}{2}(N+1), \end{cases} \tag{3.26}$$

where

$$\gamma_d = \begin{cases} \frac{1}{\Delta x^2(1+ik_1\Delta x)} & \text{if } d = 1 \ ; \\ \frac{1}{\Delta x^2(1+ik_2\Delta x)} & \text{if } d = N \ ; \\ 0 & \text{otherwise.} \end{cases} \tag{3.27}$$

The non-zero elements of the subdiagonals are

$$\begin{cases} \mathbf{A}(d,d-1) = \dfrac{1}{\Delta x^2}, \quad \mathbf{A}(d,d+1) = \dfrac{1}{\Delta x^2}, & \text{if } d \neq \frac{1}{2}(N+1) \ ; \\ \\ \mathbf{A}(d,d-1) = -1, \quad \mathbf{A}(d,d+1) = -1, & \text{if } d = \frac{1}{2}(N+1) \ . \end{cases} \tag{3.28}$$

The linear system (3.23) can now be solved in the same way as in the previous problem.

Method 2

The second method to treat the second connecting condition is to assume that

$$k = \begin{cases} k_1 & \text{if } d < \frac{1}{2}(N+1); \\ \frac{k_1+k_2}{2} & \text{if } d = \frac{1}{2}(N+1); \\ k_2 & \text{if } d > \frac{1}{2}(N+1). \end{cases} \tag{3.29}$$

which can be justified using Finite Elements method (FEM). Now, the elements of the main diagonal of $\mathbf{A}$ are given by

$$\mathbf{A}(d,d) = -\left(\frac{2}{\Delta x^2}\right) + k^2 + \gamma_d, \tag{3.30}$$

where

$$\gamma_d = \begin{cases} \frac{1}{\Delta x^2(1+ik_1\Delta x)} & \text{if } d = 1; \\ \frac{1}{\Delta x^2(1+ik_2\Delta x)} & \text{if } d = N; \\ 0 & \text{otherwise.} \end{cases} \tag{3.31}$$

The non-zero elements of the subdiagonals are

$$\mathbf{A}(d,d+1) \ = \mathbf{A}(d,d-1) \ = \frac{1}{\Delta x^2}. \tag{3.32}$$

Notice that expressions (3.30) and (3.32) are the same as (3.11) respectively (3.13).

3.2.3 Example 3: $k_1 = 1, k_2 = 3$ and $f = 9$

We consider

$$\begin{cases} \dfrac{\mathrm{d}^2p(x)}{\mathrm{d}x^2} + k^2p(x) &= 9, & x \in (0,\pi), \\ \\ -\dfrac{\mathrm{d}p(x)}{\mathrm{d}x} + ikp(x) &= 0, & x = 0, \\ \\ \dfrac{\mathrm{d}p(x)}{\mathrm{d}x} + ikp(x) &= 0, & x = \pi, \end{cases} \tag{3.33}$$

where

$$k = \begin{cases} 1, & x \in [0, \frac{1}{2}\pi], \\ 3, & x \in (\frac{1}{2}\pi, \pi]. \end{cases} \tag{3.34}$$

Analytical solution

The general solution of (3.33) and (3.34) is

$$\begin{cases} p_1(x) = 9 - (ic_1 + 9)\cos x + c_1 \sin x, \\ p_2(x) = 1 + c_2 \cos 3x + i(1 - c_2)\sin 3x, \end{cases} \tag{3.35}$$

with $c_1, c_2 \in \mathbb{R}$. The constants c_1 and c_2 can be determined with the aid of (3.21), resulting in the following linear system

$$\begin{bmatrix} 1 & -i \\ 1 & 3i \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} -(8+i) \\ 9i \end{bmatrix}. \tag{3.36}$$

The solution of (3.36) is

$$\begin{cases} c_1 = \frac{3}{2}i - 6, \\ c_2 = 2\frac{1}{2} - 2i, \end{cases} \tag{3.37}$$

which can be substituted in (3.35) to obtain

$$\begin{cases} p_1(x) = 9 + (6i - 7\frac{1}{2})\cos x + (\frac{3}{2}i - 6)\sin x, \\ p_2(x) = 1 + (2\frac{1}{2} - 2i)\cos 3x - (\frac{3}{2}i + 2)\sin 3x. \end{cases} \tag{3.38}$$

Thus, the *real* parts of solution (3.38) are

$$\begin{cases} p_1(x) = 9 - \frac{15}{2}\cos x - 6\sin x, & x \in [0, \frac{1}{2}\pi], \\ \\ p_2(x) = 1 + \frac{5}{2}\cos 3x - 2\sin 3x, & x \in (\frac{1}{2}\pi, \pi], \end{cases} \tag{3.39}$$

and the *imaginairy* parts are given by

$$\begin{cases} p_1(x) = 6\cos x + \frac{3}{2}\sin x, & x \in [0, \frac{1}{2}\pi], \\ \\ p_2(x) = -2\cos 3x - \frac{3}{2}\sin 3x, & x \in (\frac{1}{2}\pi, \pi]. \end{cases} \tag{3.40}$$

Numerical solution

Making use of the code in **Appendix A.3** we obtain the plot of the solution for method 1 and 2. We typ the following two commando's in *Matlab*:

```
Ex3met1(101,1,3,9),
Ex3met2(101,1,3,9).
```

See Figure 3.3 for the results.

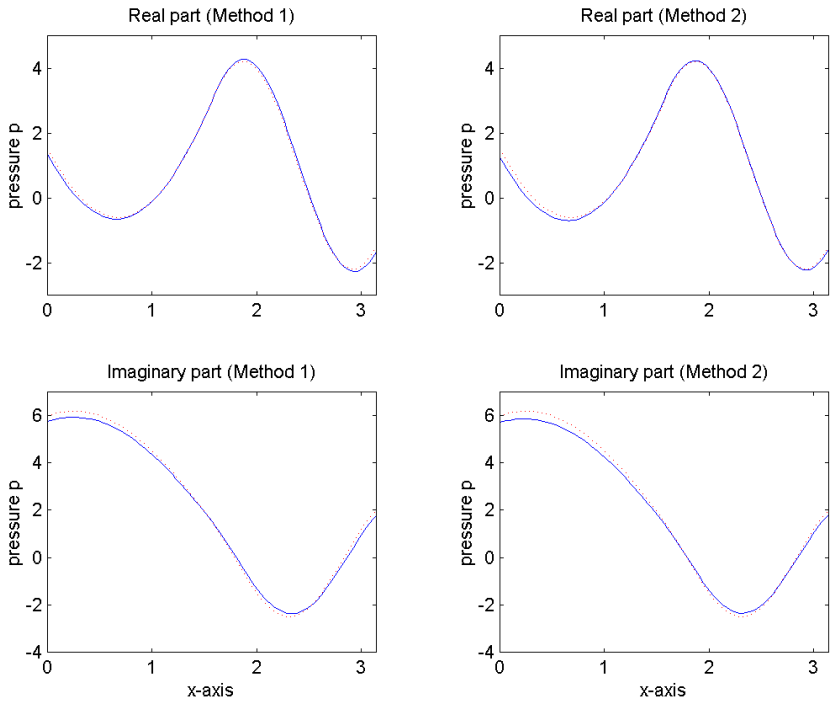

Figure 3.3: PLOTS OF THE SOLUTION OF EXAMPLE 3 WITH METHOD 1 AND 2 WITH $N = 101$.

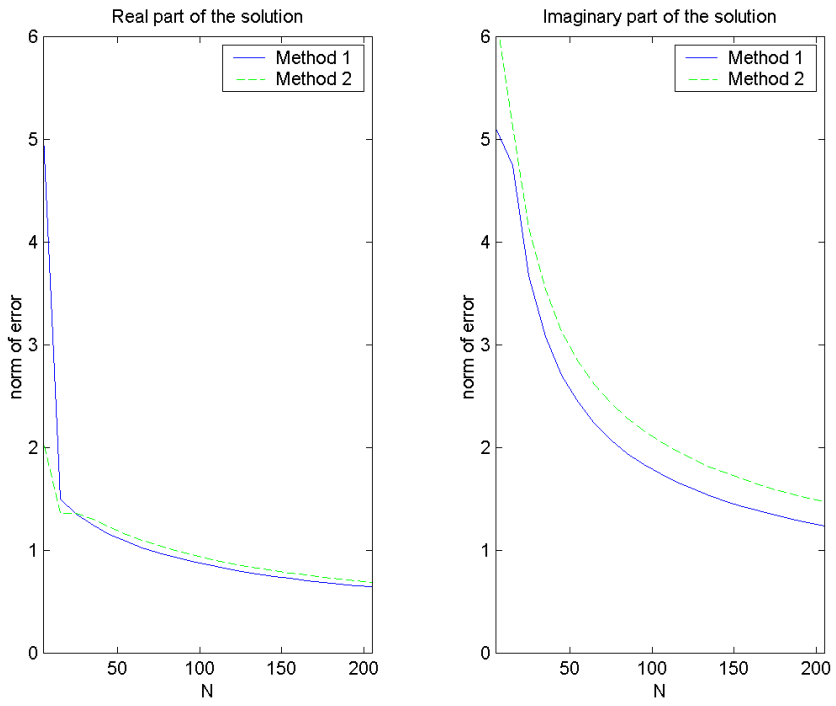

Figure 3.4: PLOTS OF THE NORM OF THE RESIDUALS OF EXAMPLE 3 WITH VARIED N.

The exact solution has been included in this figure for a comparison between the methods. We see that the three solutions are consistent with each other.

We compare the two numerical methods by observing the norm of the residuals

$$\mathbf{r}(\mathbf{x}) = ||\tilde{\mathbf{p}}(\mathbf{x}) - \mathbf{p}(\mathbf{x})||_2 = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\tilde{\mathbf{p}}(x_i) - \mathbf{p}(x_i)\right)^2} \tag{3.41}$$

where $\tilde{\mathbf{p}}(\mathbf{x})$ is the numerical solution of method 1 or 2 and $\mathbf{p}(x)$ is the exact solution. We vary the number of elements N. Therefore we take consecutive $N = 5, 15, \ldots, 205$. For each case and for both real and part of the solutions we compute the norm of the residuals as given in expression (3.41). This results in the *Matlab*-code as seen in **Appendix A.5**. Plots of the results are represented in Figure 3.4. In further research it may be advantageous to plot the figure logarithmically.

Since we are interested in relative high values of N, method 1 is the most accurate one in this example, see Figure 3.4. More research is needed

to seek out this item.

The extreme eigenvalues (i.e., $\lambda_{\max}$ and $\lambda_{\min}$) of method 1 (with $N = 101$ elements) are, respectively,

```
1.0e+003 *
0.0057 - 0.0017i
-4.1291 - 0.0000i
```

The extreme eigenvalues of method 2 (with $N = 101$ elements) are

```
1.0e+003 *
0.0066 - 0.0009i
 -4.1306 - 0.0000i
```

So both systems are *indefinite*.

3.2.4 Example 4: $k_1 = 1e-6, k_2 = 30$ and $f = 9$

We end the examples with the same case as example 3, but now we define

$$k = \begin{cases} 1 \times 10^{-6}, & x \in [0, \frac{1}{2}\pi], \\ 30, & x \in (\frac{1}{2}\pi, \pi]. \end{cases} \tag{3.42}$$

instead of expression (3.34).

The analytical solution is omitted here. We compute with the following commando's the numerical solution of this problem

```
Ex4met1(101,1e-6,30,9),
Ex4met2(101,1e-6,30,9),
```

using **Appendix A.4**. The resulting plots can be found in Figure 3.5. In the figure we can see some differences, especially in the plot of the imaginary part of the solution. More research is needed to decide which method is the best in this case, comparing the corresponding solutions with the analytical one.

The extreme eigenvalues of method 1 are

```
1.0e+003 *
0.8959 - 0.0002i
-4.1301 - 0.0000i
```

The extreme eigenvalues of method 2 are

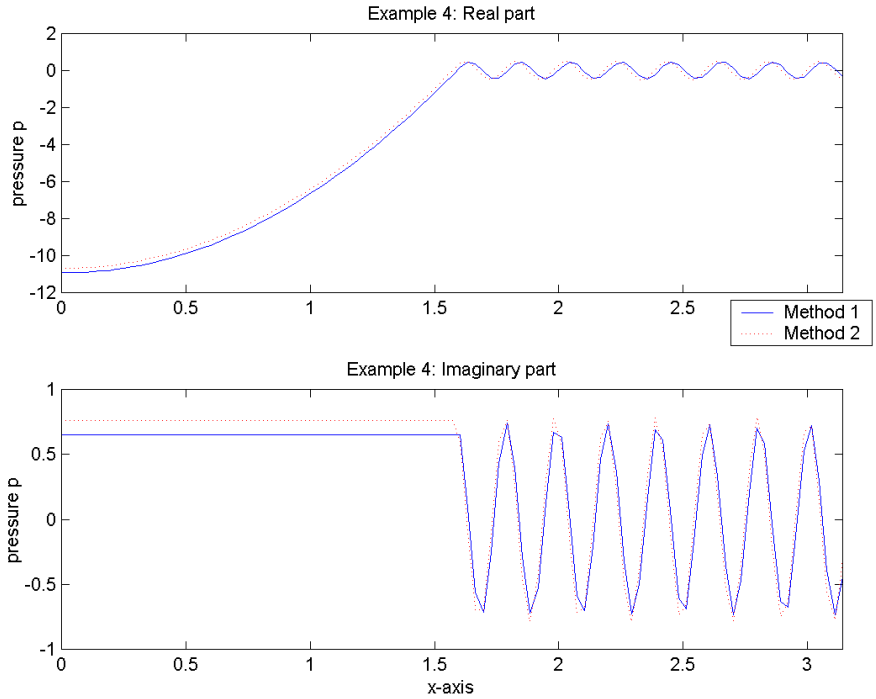

Figure 3.5: PLOTS OF THE SOLUTION OF EXAMPLE 4 WITH $N = 101$.

```
1.0e+003 *
0.8961 - 0.0002i
-4.1303 - 0.0000i
```

Both systems are again *indefinite*.

3.3 Eigenvalues

In all examples we have observed the systems are indefinite, i.e., the eigenvalues have both positive and negative real parts. Notice that we have $|\lambda_{\max}| \ll |\lambda_{\min}|$ in examples 1-3. In this section we are looking for the eigenvalues of the same systems in an analytical way.

3.3.1 Analytical system

First we examine the analytical eigenvalues of the Helmholtz problem with Dirichlet conditions, i.e.,

$$\begin{cases} \left(\dfrac{\mathrm{d}^2}{\mathrm{d}x^2} + k^2\right) p(x) = f, \qquad x \in (0,1), \\ \\ \quad p(0) = p(1) = 0. \end{cases} \tag{3.43}$$

Notice that in this section we have taken $x \in (0,1)$ instead of $(0,\pi)$. Then, the eigenvalue differential equation has to satisfy

$$\begin{cases} \left(\dfrac{\mathrm{d}^2}{\mathrm{d}x^2} + k^2\right) \phi(x) = \lambda\phi(x), \qquad x \in (0,1), \\ \\ \quad \phi(0) = \phi(1) = 0, \end{cases} \tag{3.44}$$

where λ and $\phi(x)$ are, respectively, the eigenvalue and eigenfunction of system (3.43). The solution of (3.44) is

$$\phi(x) = c_1 \cos(\sqrt{k^2 - \lambda}x) + c_2 \sin(\sqrt{k^2 - \lambda}x), \tag{3.45}$$

where c_1 and c_2 can be derived using the Dirichlet conditions $p(0) = p(1) = 0$. Since we are not interested in the trivial solution, we obtain the equation

$$\sin(\sqrt{k^2 - \lambda}) = 0. \tag{3.46}$$

This gives us immediately the eigenvalues

$$\lambda_n = k^2 - (n\pi)^2, \tag{3.47}$$

for $n = 1, 2, \ldots$. Observe that there are an *infinite* number of eigenvalues and also eigenfunctions.

Moreover, if we multiply both sides of the differential equation in (3.43) with -1, then the eigenvalues as given in (3.47) get an extra minus-sign, i.e., the eigenvalues for the negative Helmholtz-operator are given by

$$\tilde{\lambda}_n = (n\pi)^2 - k^2. \tag{3.48}$$

Hence, there exists a positive integer $\tilde{n}$ such that

$$\tilde{\lambda}_n > 0 \quad \forall n > \tilde{n}, \tag{3.49}$$

which is favourable in some practical situations, see next chapters.

3.3.2 Discretized system

In Peters, Eiermann, Daniels [18] one found for the Toeplitz matrix

$$\mathbf{A} = \begin{bmatrix} \alpha & \beta & & & \emptyset \\ \gamma & \alpha & \beta & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \beta \\ \emptyset & & & \gamma & \alpha \end{bmatrix}, \text{where } \mathbf{A} \in \mathbb{R}^{N\times N}, \tag{3.50}$$

the following eigenvalues

$$\lambda_j = \alpha + 2\sqrt{\beta\gamma}\cos(\pi j \Delta x), \quad j = 1, 2, \ldots, N, \tag{3.51}$$

where $\Delta x = \frac{1}{N}$ and $\alpha, \beta, \gamma \in \mathbb{R}$. These eigenvalues can be found by using the solution $\phi_j = \sin(\delta j h)$ with $\delta \in \mathbb{R}$. Then the boundary conditions leads to the expressions given in (3.51).

Consider the matrix obtained from discretizing system (3.43), i.e., consider again example 1 with modified boundary conditions $p(0) = p(1) = 0$ instead of absorbing conditions (see (3.14)) and therefore $\gamma_d = 0$ instead of expression (3.12). Then the eigenvalues are given by

$$\lambda_j = \left(-\frac{2}{\Delta x^2} + k^2\right) + \frac{2}{\Delta x^2}\cos(\pi j \Delta x), \quad j = 1, 2, \ldots, N, \tag{3.52}$$

with the aid of (3.51) [1].

Therefore, the smallest eigenvalue is

$$\lambda_{\min} = -\frac{4}{\Delta x^2} + k^2, \tag{3.53}$$

since $\cos(\pi) = -1$ by taking $j = N$. As a consequence, $\lambda_{\min}$ is *negative* when $k < 2N$.

[1] Using the fact that $1 - 2\cos 2x = 2\sin^2 x$, we can rewrite (3.52) as $\lambda_j = k^2 - \frac{4}{\Delta x^2}\sin^2(\frac{1}{2}\pi j \Delta x)$.

The largest eigenvalue is

$$\lambda_{\max} = -\frac{2}{\Delta x^2} + k^2 + \frac{2}{\Delta x^2}\cos(\pi \Delta x), \tag{3.54}$$

by taking $j = 1$ in expression (3.52). We can write (3.54) as

$$\lambda_{\max} = k^2 - \pi^2 + \mathcal{O}(\Delta x^2), \tag{3.55}$$

using the Taylor serie of the cosinus, i.e., $\cos(\pi\Delta x) = 1 - \frac{(\pi\Delta x)^2}{2} + \mathcal{O}(\Delta x^4)$. Now, if we let $N \to \infty$, i.e., $\Delta x \to 0$ then we derive

$$\lambda_{\max} = k^2 - \pi^2. \tag{3.56}$$

In other words, if we increase the dimensions of matrix $\mathbf{A}$, then the largest eigenvalue tends to $k^2 - \pi^2$. Therefore,

$$\begin{cases} \lambda_{\max} < 0 & \text{if } k < \pi, \\ \lambda_{\max} > 0 & \text{if } k > \pi\ , \\ \lambda_{\max} = 0 & \text{if } k = \pi\ . \end{cases} \tag{3.57}$$

We can conclude that if we choose k such that $\pi < k < 2N$ with sufficient large N, then we have $\lambda_{\min} < 0$ and $\lambda_{\max} > 0$, thus then we deal with an *indefinite* system.

Moreover, notice that if we have a sufficient large N and a fixed (relative small) k, then it yields $|\lambda_{\max}| \ll |\lambda_{\min}|$ with the aid of expressions (3.53) and (3.56). This is exactly the case in examples 1-3.

However, in example 1 we have considered *absorbing* instead of *Dirichlet* conditions. In that case we can not use $p_j = \sin(\delta j \Delta x)$ to derive the eigenvalues, but we expect that the (real part of the) eigenvalues as given in (3.52) will not change much.

3.3.3 Comparing eigenvalues of the analytical and discretized system

In theory, the eigenvalues of the discretized system, found in (3.52), have to converge asymptotically to the eigenvalues of the analytical system given in (3.47). Below we shall give the proof.

We take the limit $\Delta x \to 0$ in (3.52) which results in

$$\begin{aligned} \lambda_j &= \lim_{\Delta x \to 0} -\tfrac{2}{\Delta x^2} + k^2 + \tfrac{2}{\Delta x^2}\cos(\pi j \Delta x), \\ &= \lim_{\Delta x \to 0} -\tfrac{2}{\Delta x^2} + k^2 + \tfrac{2}{\Delta x^2}\left(1 - \tfrac{(\pi j \Delta x)^2}{2} + \mathcal{O}(\Delta x^4)\right), \\ &= \lim_{\Delta x \to 0} k^2 - (j\pi)^2 + \mathcal{O}(\Delta x^2), \\ &= k^2 - (j\pi)^2, \end{aligned} \tag{3.58}$$

for $j = 1, 2, \ldots$. In (3.58) we have used again the Taylor expansion of the cosinus. Indeed, the eigenvalues of the discretized system are asymptotically *equal* to those of the analytical system.

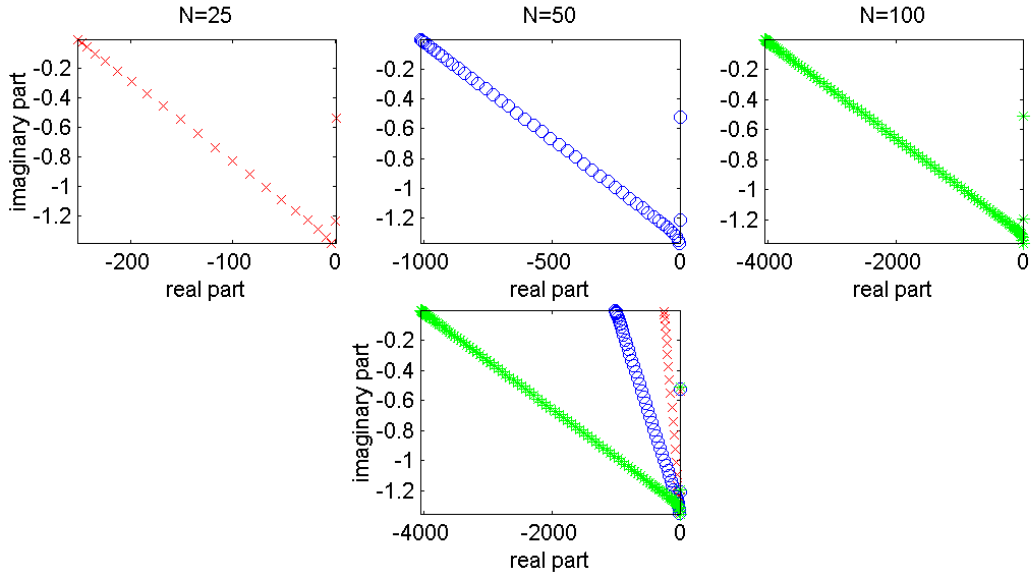

Figure 3.6: PLOTS OF THE EIGENVALUES OF EXAMPLE 1 WITH N = 25, 50, 100.

Now, we return to example 1 and in the next subsections one deals with some aspects with respect to the eigenvalues. Firstly, we compute the eigenvalues for different values of N. Secondly we compare the eigenvalues of the original system with a system where $\mathbf{A}$ is a *Toeplitz* matrix.

3.3.4 Eigenvalues of example 1 with varied N

We examine the eigenvalues of example 1 with three different numbers of elements: $N = 25, 50, 100$. The code can be found in **Appendix A.6**. Now, the results are given in figure 3.6 [2].

We conclude that the imaginary part of the eigenvalues do not change if we vary N. However, the real part of the eigenvalues are much dependent of N. In this case we see obviously that the system is more indefinite when we increase N. This is the expected result, considering expression (3.53).

3.3.5 Comparing the original with a modified example 1

We compare matrix $\mathbf{A}$ found in example 1 with a modified example where we have taken the following Dirichlet boundary conditions instead of absorbing

[2] Denote Re(λ)= values of the real part of the eigenvalues. Then figure 3.6 will be more clear if we plot Re(λ) $* \frac{1}{N}$ in the horizontal axis instead of Re(λ). This is left here.

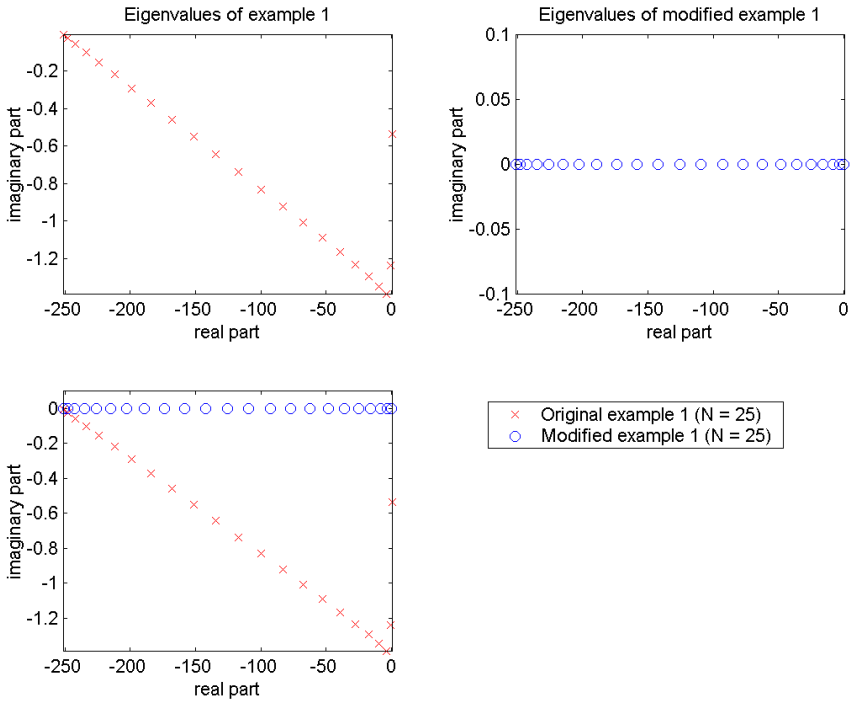

Figure 3.7: Plots of the eigenvalues of the original (with absorbing conditions) and modified (with Dirichlet conditions) example 1 with $N = 25$.

conditions,

$$p(0) = p(\pi) = 0. \tag{3.59}$$

Then we get a matrix $\tilde{\mathbf{A}}$ which has a Toeplitz structure.

We compute the eigenvalues of this modified matrix $\tilde{\mathbf{A}}$ and compare this with the eigenvalues of $\mathbf{A}$, see figure 3.7. The code can be found in **Appendix A.7**.

One can conclude that the real part of the eigenvalues of both systems has the same range, while the exact values are not the same, see also figure 3.8. Actually, there is only a shift in the range of the imaginary part of the eigenvalues. This imaginary shift is not disadvantageous in iterative methods, see next chapter.

Considering this example we conclude that we can work with the Toeplitz version of the original matrix $\mathbf{A}$ to derive the real-valued spectrum of the eigenvalues of $\mathbf{A}$. More research is needed to generalize this conclusion.

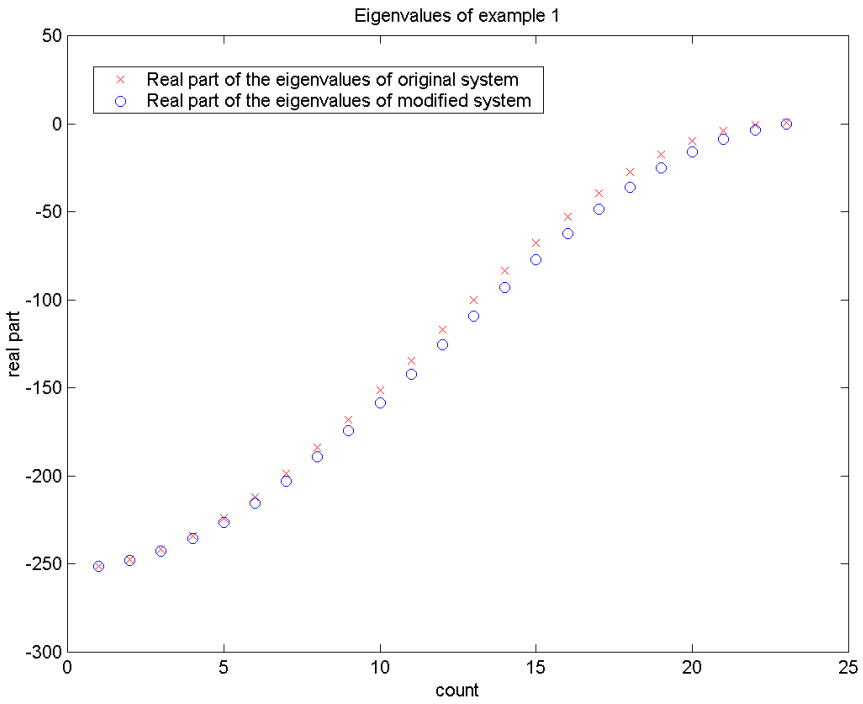

Figure 3.8: Plots of the *sorted* set of real parts of the eigenvalues of both the original and modified example 1 with $N = 25$.

Chapter 4

Krylov Iterative Methods

4.1 Introduction

Consider the linear system

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \quad \mathbf{A} \in \mathbb{C}^{n \times n}, \tag{4.1}$$

where $\mathbf{A}$ is an $n \times n$ matrix and $\mathbf{x}$ and $\mathbf{b}$ are both complex vectors with n elements. Moreover, matrix $\mathbf{A}$ is sparse and assumed in general to be complex. We intend to solve (4.1).

The one or two dimensional version of Helmholtz's boundary value problem (HBVP) gives a linear system, which in general can be solved with *direct* numerical methods, like *Gauss-elimination* or *nested dissection method*, see Mulder & Plessix [17]. A case with 1000×1000 gridpoints has been tested on a workstation and the nested dissection method can indeed be applied to solve the linear system at efficient cost. In 3-dimensional cases, the direct methods are in general *inefficient* to use. For instance the nested dissection method gives problems because the amount of fill-in is too large, see chapter 4 of Erlangga [6].

However, solution methods with an acceptable efficiency can still be pursued by implementing *iterative* numerical methods for the linear system (4.1). Recently, several iterative methods have been developed. The underlying concept in deriving this methods is the so-called *Krylov subspace* method.

In this report, we first describe some iterative methods, relevant to HBVP. Since we aim at the numerical solution of a *complex symmetric* and (highly) *indefinite* linear system, we will consider iterative methods feasible for this kind of linear systems. We treat respectively Conjugate Gradient (CG), CGNR, COCG, GMRES, Bi-CG and Bi-CGSTAB, see next subsections. We shall give special attention to GMRES and Bi-CGSTAB, because these methods are frequently used to deal with HBVP.

Conjugate Gradient (CG) method is not applicable to HBVP, because the linear system has to be real and positive-definite to secure an accurate

solution, see e.g. section 6.7 of Saad [19]. Nevertheless, we include the CG method, since this algorithm is of importance as a basis for deriving several related iterative methods.

In practice, standard iterative methods are not sufficiently efficient for solving a sparse and large linear system without modifications. 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 approximately the same solution, but which is much easier to solve. This process is called *preconditioning*. Krylov iterative methods are inattractive without preconditioning. We deal with preconditioners in the next section.

4.2 Krylov subspace method

In standard iterative methods, the iterant in the $(j+1)$-th step can be determined from the j-th step in the following way

$$\mathbf{x}_{j+1} = \mathbf{x}_j + \mathbf{P}^{-1}\mathbf{r}_j, \tag{4.2}$$

where $\mathbf{P}$ is a $n \times n$ matrix and the j-th *residu* $\mathbf{r}_j$ is defined as

$$\mathbf{r}_j = \mathbf{b} - \mathbf{A}\mathbf{x}_j, \tag{4.3}$$

which is a measure of the difference of the iterative and the exact solution of the problem. If we work out the first iterations of (4.2), one obtains

$$\begin{cases} \mathbf{x}_0 & & \\ \mathbf{x}_1 & = & \mathbf{x}_0 + \mathbf{P}^{-1}\mathbf{r}_0 \\ \mathbf{x}_2 & = & \mathbf{x}_1 + \mathbf{P}^{-1}(\mathbf{b} - \mathbf{A}\mathbf{x}_0 - \mathbf{A}\mathbf{P}^{-1}\mathbf{r}_0) \\ & = & \mathbf{x}_0 + 2\mathbf{P}^{-1}\mathbf{r}_0 - \mathbf{P}^{-1}\mathbf{A}\mathbf{P}^{-1}\mathbf{r}_0 \\ \mathbf{x}_3 & = & \dots \\ & \vdots & \end{cases} \tag{4.4}$$

Using (4.4) we get the following expression for (4.2)

$$\mathbf{x}_{j+1} \in \mathbf{x}_0 + \text{ span}\left\{\mathbf{P}^{-1}\mathbf{r}_j, \mathbf{P}^{-1}\mathbf{A}(\mathbf{P}^{-1}\mathbf{r}_j), \dots, (\mathbf{P}^{-1}\mathbf{A})^{i-1}(\mathbf{P}^{-1}\mathbf{r}_j)\right\}. \tag{4.5}$$

Subspaces of the form

$$\mathcal{K}_j(\mathbf{A}, \mathbf{r}_0) = \text{ span}\left\{\mathbf{r}_0, \mathbf{A}\mathbf{r}_0, \mathbf{A}^2\mathbf{r}_0, \dots, \mathbf{A}^{j-1}\mathbf{r}_0\right\}. \tag{4.6}$$

are called *Krylov subspaces* with dimension j, belonging to $\mathbf{A}$ and $\mathbf{r}_0$. Using (4.6) we get the following expression for standard iterative methods

$$\mathbf{x}_{j+1} \in \mathbf{x}_0 + \mathcal{K}_j(\mathbf{P}^{-1}\mathbf{A}, \mathbf{P}^{-1}\mathbf{r}_0). \tag{4.7}$$

Expression (4.7) is equivalent to (4.2) and (4.5).

From (4.7) we can observe that Krylov subspace methods rely on finding a matrix $\mathbf{P}$ and a basis for $\mathcal{K}_j$ such that the iterative method converge fast with reasonable accuracy and efficiency with respect to memory and computational time. Matrix $\mathbf{P}$ is called the *preconditioner*.

Standard iterative methods like *Gauss-Jacobi* and *Gauss-Seidel* are described in e.g. section 11.2 of Segal & Van Kan [13], chapter 3 of Tang [24] and section 5.1 of Vuik [28]. If we denote $\mathbf{A} = \mathbf{D} - \mathbf{L} - \mathbf{L}^T$ with $\mathbf{D}$ the matrix with the main diagonal and $\mathbf{L}$ the (strict) undertriagonal of $\mathbf{A}$, then we get the following expressions of matrix $\mathbf{P}$:

- Gauss-Jacobi: $\mathbf{P}_{JAC} = \mathbf{D}$;
- Gauss-Seidel: $\mathbf{P}_{GS} = \mathbf{D} - \mathbf{L}$.

It is well-known that this methods converge very slow in (3-dimensional) practical problems. Successive Over Relaxation (SOR) can be used to accelerate this methods.

4.3 Conjugate Gradient (CG) method

The CG method is the most prominent iterative method for solving sparse systems $\mathbf{Ax} = \mathbf{b}$, see (4.1), where $\mathbf{A}$ is *Hermitian* and *positive definite*. The whole idea of this method can be found in e.g. Shewchuk [20].

4.3.1 CG idea

For convenience we take $\mathbf{P} = \mathbf{I}$ and $\mathbf{x}_0 = 0$, thus $\mathbf{r}_0 = \mathbf{b}$. This is no important restriction, because if $\mathbf{x}_0 \neq 0$, then $\mathbf{Az} = \mathbf{b} - \mathbf{Ax}_0 = \tilde{b}$ with $\mathbf{z} \equiv \mathbf{x} - \mathbf{x}_0$. And then we can take $z_0 = 0$ as initial vector for the system $\mathbf{Az} = \tilde{\mathbf{b}}$.

A standard method is to find the iterants in the Krylov-subspace, so

$$\mathbf{x}_{j+1} \in \mathcal{K}_j(\mathbf{A}, \mathbf{r}_0). \tag{4.8}$$

The CG method also has its iterants in this subspace, but it attemps to minimize the distance between the jth iterant and the exact solution. Minimizing the true solution in the standard norm is attractive, i.e.,

$$\min_{\mathbf{y} \in \mathcal{K}_j(\mathbf{A}, \mathbf{r}_0)} ||\mathbf{y} - \mathbf{x}||_2, \tag{4.9}$$

where $\mathbf{x}$ is the exact solution of $\mathbf{Ax} = \mathbf{b}$. This is impossible to compute, because $\mathbf{x}$ is clearly unknown. However, we do know $\mathbf{Ax}$ $(= \mathbf{b})$, so it is obvious to define the following norm

$$||\mathbf{v}||_{\mathbf{A}} = \sqrt{(\mathbf{v}, \mathbf{Av})}, \tag{4.10}$$

and we attemp to minimize the distance in this norm (4.10), i.e.

$$\min_{\mathbf{y}\in\mathcal{K}_j(\mathbf{A},\mathbf{r}_0)} ||\mathbf{y}-\mathbf{x}||_{\mathbf{A}}, \tag{4.11}$$

It will appear that (4.11) is useful. This leads to the CG-method.

4.3.2 CG derivation

We express the vector $\mathbf{x}_{j+1}$ as

$$\mathbf{x}_{j+1} = \mathbf{x}_j + \alpha_j \mathbf{p}_j, \tag{4.12}$$

where $\alpha_j \in \mathbb{C}$ and $\mathbf{p}_j$ is the search direction. We shall choose the initial search direction $\mathbf{p}_0 = \mathbf{r}_0$. Therefore the residual vectors must satisfy the recurrence

$$\mathbf{r}_{j+1} = \mathbf{r}_j - \alpha_j \mathbf{A}\mathbf{p}_j. \tag{4.13}$$

If we wish $\mathbf{r}_j$ for all j to be orthogonal to each other, it is necessary that

$$(\mathbf{r}_j, \mathbf{r}_{j+1}) = (\mathbf{r}_j - \alpha_j \mathbf{A}\mathbf{p}_j, \mathbf{r}_j) = 0, \tag{4.14}$$

resulting in

$$\alpha_j = \frac{(\mathbf{r}_j, \mathbf{r}_j)}{(\mathbf{A}\mathbf{p}_j, \mathbf{r}_j)}. \tag{4.15}$$

Recall that $(\mathbf{a}, \mathbf{b}) \equiv \sum_{i=1}^{n} \overline{a}_i \cdot b_i$ for arbitrary $\mathbf{a}, \mathbf{b} \in \mathbb{C}^n$.

It is known that the next search direction $\mathbf{p}_{j+1}$ is a linear combination of $\mathbf{r}_{j+1}$ and $\mathbf{p}_j$. After rescaling the $\mathbf{p}$ vectors appropriately, it follows that

$$\mathbf{p}_{j+1} = \mathbf{r}_{j+1} + \beta_j \mathbf{p}_j, \tag{4.16}$$

where $\beta \in \mathbb{C}$. A first consequence of (4.16) is that the denominator in (4.15) can be written as

$$(\mathbf{A}\mathbf{p}_j, \mathbf{r}_j) = (\mathbf{A}\mathbf{p}_j, \mathbf{p}_j - \beta_{j-1}\mathbf{p}_{j-1}) = (\mathbf{A}\mathbf{p}_j, \mathbf{p}_j), \tag{4.17}$$

because $\mathbf{A}\mathbf{p}_j$ is orthogonal to $\mathbf{p}_{j-1}$, using (4.16) and $(\mathbf{A}\mathbf{p}_j, \mathbf{r}_{j-1}) = 0$. Then, expression (4.15) becomes

$$\alpha_j = \frac{(\mathbf{r}_j, \mathbf{r}_j)}{(\mathbf{A}\mathbf{p}_j, \mathbf{p}_j)}. \tag{4.18}$$

In addition, knowing that $\mathbf{p}_{j+1}$ is orthogonal to $\mathbf{A}\mathbf{p}_j$ yields

$$(\mathbf{p}_{j+1}, \mathbf{A}\mathbf{p}_j) = (\mathbf{r}_{j+1} + \beta_j \mathbf{p}_j, \mathbf{A}\mathbf{p}_j) = 0. \tag{4.19}$$

From (4.19) we derive

$$\beta_j = -\frac{(\mathbf{r}_{j+1}, \mathbf{A}\mathbf{p}_j)}{(\mathbf{p}_j, \mathbf{A}\mathbf{p}_j)}. \tag{4.20}$$

Note that from (4.13)

$$\mathbf{A}\mathbf{p}_j = -\frac{1}{\alpha_j}(\mathbf{r}_{j+1} - \mathbf{r}_j). \tag{4.21}$$

Using (4.16) and (4.21), we can write expression (4.20) in the following way

$$\beta_j = \frac{(\mathbf{r}_{j+1}, \mathbf{r}_{j+1})}{(\mathbf{r}_j, \mathbf{r}_j)}. \tag{4.22}$$

Above we have seen that the vectors $\mathbf{p}_j$ are $\mathbf{A}$-orthogonal, or *conjugate*. That is why the method is called the *conjugate gradient* method.

4.3.3 CG algorithm

In the previous subsection we derived expressions that are applied in the algorithm for the CG method.

Conjugate Gradient Algorithm

1. Compute $\mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0$ and $\mathbf{p}_0 := \mathbf{r}_0$.

2. **For** $j := 0, 1, \ldots,$ *until convergence* **Do** :

3. $\quad \mathbf{w}_j := \mathbf{A}\mathbf{p}_j$
4. $\quad \alpha_j := \frac{(\mathbf{r}_j, \mathbf{r}_j)}{(\mathbf{w}_j, \mathbf{p}_j)}$
5. $\quad \mathbf{x}_{j+1} := \mathbf{x}_j + \alpha_j \mathbf{p}_j$
6. $\quad \mathbf{r}_{j+1} := \mathbf{r}_j - \alpha_j \mathbf{w}_j$
7. $\quad \beta_j := \frac{(\mathbf{r}_{j+1}, \mathbf{r}_{j+1})}{(\mathbf{r}_j, \mathbf{r}_j)}$
8. $\quad \mathbf{p}_{j+1} := \mathbf{r}_{j+1} + \beta_j \mathbf{p}_j$

9. **EndDo**

Notice that we defined $\mathbf{w}_j \equiv \mathbf{A}\mathbf{p}_j$ in the above algorithm for simplicity.

In terms of storage, in addition to the matrix $\mathbf{A}$, four vectors $\mathbf{x}, \mathbf{p}, \mathbf{A}\mathbf{p}$ and $\mathbf{r}$ must be saved in the CG-algorithm.

The CG-algorithm works well for *symmetric*, *Hermitian* and *positive definite* (PD) matrices 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 $\mathbf{A}$ has both positive and negative eigenvalues, see subsection 11.3.2 of Segal & Van Kan [13].

To overcome the indefiniteness, non-Hermite and nonsymmetry of $\mathbf{A}$, modifications are introduced to the standard CG-method. These lead to several variants of CG, for example SYMMLQ, CGS and Bi-CGSTAB. These variants preserve some of the nice properties of the CG algorithm (e.g. short

recurrences or optimal method with respect to the $||\cdot||$–norm) and also extend the methods to be able to solve non-PD matrices.

Alternatively, by transformation into normal equations, the CG algorithm sitll works nicely. We discuss this approach in the next section.

4.4 CGNR method

If we multiply $\mathbf{Ax} = \mathbf{b}$ with the Hermitian of $\mathbf{A}$, i.e., $\mathbf{A}^H$, then this leads to the normal equations

$$\mathbf{A}^H\mathbf{Ax} = \mathbf{A}^H\mathbf{b}. \tag{4.23}$$

If matrix A is not singular, then it can easily be derived that $\mathbf{A}^H\mathbf{A}$ is always *positive definite*, *Hermitian* and *symmetric*. The consequence is that we can apply CG to (4.23) and all properties of the method are still valid. Notice that now we want

$$\min_{\mathbf{y}\in\mathcal{K}_j(\mathbf{A}^H\mathbf{A},\mathbf{A}^H\mathbf{r}_0)} ||\mathbf{y}-\mathbf{x}||_{\mathbf{A}^H\mathbf{A}}. \tag{4.24}$$

We derive easily

$$||\mathbf{y}-\mathbf{x}||_{\mathbf{A}^H\mathbf{A}} = ||\mathbf{A}(\mathbf{y}-\mathbf{x})||_2 = ||\mathbf{r}||_2, \tag{4.25}$$

where the residu $\mathbf{r} = \mathbf{b} - \mathbf{Ay}$.

This is why the method is called CGNR, because it is the Conjugate Gradient method applied to the Normal equations where the Residu is minimized.

4.4.1 CGNR algorithm

The residual vectors $\mathbf{z}$ satisfy the recurrence

$$\mathbf{z}_{j+1} = \mathbf{z}_j - \alpha_j\mathbf{A}^H\mathbf{Ap}_j. \tag{4.26}$$

We may compute this residual in two parts:

$$\begin{array}{rcl} \mathbf{r}_{j+1} & = & \mathbf{r}_j - \alpha_j\mathbf{Ap}_j; \\ \mathbf{z}_{j+1} & = & \mathbf{A}^H\mathbf{r}_{j+1}. \end{array} \tag{4.27}$$

The CGNR-algorithm can now be cast in the following form.

CGNR Algorithm

1. Compute $\mathbf{r}_0 := \mathbf{b} - \mathbf{Ax}_0$, $\mathbf{z}_0 = \mathbf{A}^H\mathbf{r}_0$ and $\mathbf{p}_0 := \mathbf{r}_0$.

2. **For** $j := 0, 1, \ldots,$ *until convergence* **Do** :

3. $\quad \mathbf{w}_j := \mathbf{Ap}_j$

4. $\alpha_j := \frac{(\mathbf{z}_j, \mathbf{z}_j)}{(\mathbf{w}_j, \mathbf{w}_j)}$
5. $\mathbf{x}_{j+1} := \mathbf{x}_j + \alpha_j \mathbf{p}_j$
6. $\mathbf{r}_{j+1} := \mathbf{r}_j - \alpha_j \mathbf{w}_j$
7. $\mathbf{z}_{j+1} = \mathbf{A}^H \mathbf{r}_{j+1}$
8. $\beta_j := \frac{(\mathbf{z}_{j+1}, \mathbf{z}_{j+1})}{(\mathbf{z}_j, \mathbf{z}_j)}$
9. $\mathbf{p}_{j+1} := \mathbf{z}_{j+1} + \beta_j \mathbf{p}_j$
10. **EndDo**

The disadvantage of this method is that both *round-off-errors* and *convergence speed* depend on the *spectral condition number* of $\mathbf{A}^H\mathbf{A}$ in the Euclidian norm, i.e. they depend on $K_2(\mathbf{A}^H\mathbf{A}) = K_2(\mathbf{A})^2$ with $K_2(\mathbf{A}) = ||A||_2 \cdot ||A^{-1}||_2$. This means that if $\mathbf{A}$ has a relative bad condition number, $K_2(\mathbf{A}^H\mathbf{A})$ can only be worse.

Furthermore, another disadvantage is that CGNR requires the matrix-multiplication $\mathbf{A}^H\mathbf{x}$ which may be difficult to compute. Notice that $\mathbf{A}^H\mathbf{A}$ is usually not determined. When we need to compute $\mathbf{A}^H\mathbf{A}\mathbf{x}$, we apply $\mathbf{A}^H(\mathbf{A}\mathbf{x})$ which is often cheaper.

In spite of these disadvantages there are several categories of matrices where CGNR gives good results.

4.5 COCG method

The Conjugate Orthogonal Conjugate Gradient (COCG) method can be used for symmetric-complex matrix, which is of our interest. It is the same method as CG, but with an additional modification of the orthogonality condition. Instead Herminitian orthogonality, we use the *conjugate orthogonality*, i.e.

$$(\overline{\mathbf{r}}_j, \mathbf{r}_k) = 0 \quad \text{if } j \neq k. \tag{4.28}$$

4.5.1 COCG algorithm

The COCG algorithm is given below.

COCG Algorithm

1. Compute $\mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0$ and $\mathbf{p}_0 := \mathbf{r}_0$.
2. **For** $j := 0, 1, \ldots,$ *until convergence* **Do** :
3. $\mathbf{w}_j := \mathbf{A}\mathbf{p}_j$
4. $\alpha_j := \frac{(\overline{\mathbf{r}}_j, \mathbf{r}_j)}{(\overline{\mathbf{w}}_j, \mathbf{p}_j)}$
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. $\beta_j := \frac{(\bar{\mathbf{r}}_{j+1}, \mathbf{r}_{j+1})}{(\bar{\mathbf{r}}_j, \mathbf{r}_j)}$
8. $\mathbf{p}_{j+1} := \mathbf{r}_{j+1} + \beta_j \mathbf{p}_j$

9. **EndDo**

We see that COCG has short recurrences. Moreover, notice that for arbitrary $a, b \in \mathbb{C}^n$ yields

$$(\bar{\mathbf{a}}, \mathbf{b}) = \sum_{i=1}^{n} \overline{\bar{a}}_i \cdot b_i = \sum_{i=1}^{n} a_i \cdot b_i, \tag{4.29}$$

which does *not* satisfy all properties of the inner product, i.e., the method is optimal to the defined *semi*-inner product. Furthermore, COCG-method is not robust. See Van der Vorst & Melissen [26] for more details about COCG.

4.6 GMRES

The generalized minimal residual (GMRES) is an extension of the minimal residual method (MINRES) for *unsymmetric* systems. GMRES minimizes the residual norm over the Krylov subspace. In order to reach this, it generates a sequence of orthogonal vectors with long recurrences due to the unsymmetry. The method uses a variant of the Arnoldi's procedure to find a set of orthonormalized vectors.

4.6.1 Arnoldi's method

Arnoldi's procedure is an algorithm for building an orthonormal basis of the Krylov subspace $\mathcal{K}_m$. One variant of the algorithm is given below.

Arnoldi's algorithm

1. Choose a vector $\mathbf{v}_1$ of norm 1

2. **For** $j := 1, 2, \ldots, m$ **Do** :

3. $h_{i,j} := (\mathbf{A}\mathbf{v}_j, \mathbf{v}_i)$ for $i = 1, 2, \ldots, j$
4. $\mathbf{w}_j := \mathbf{A}\mathbf{v}_j - \sum_{i=1}^{j} h_{ij}\mathbf{v}_i$
5. $h_{j+1,j} := ||\mathbf{w}_j||_2$

6. $\mathbf{v}_{j+1} := \dfrac{\mathbf{w}_j}{h_{j+1,j}}$

7. **EndDo**

At each step, the algorithm multiplies the previous Arnoldi vector $\mathbf{v}_j$ by $\mathbf{A}$ and then orthonormalizes the resulting vector $\mathbf{w}_j$ against all previous $\mathbf{v}_i$'s by a standard Gram-Schmidt procedure.

One can prove the following propositions (see p.146-148 of Saad [19]).

Proposition 1 *Assume that Arnoldi's algorithm does not stop before the m-th step. Then the vectors* $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_m$ *form an orthonormal basis of the Krylov subspace*

$$\mathcal{K}_m = \text{span}\left\{\mathbf{v}_1, \mathbf{A}\mathbf{v}_1, \ldots, \mathbf{A}^{m-1}\mathbf{v}_1\right\}. \tag{4.30}$$

Proposition 2 *Denote by* $\boldsymbol{V}_m$ *the* $n \times m$ *matrix with column vectors* $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m$. *Denote by* $\bar{\boldsymbol{H}}_m$ *the* $(m+1) \times m$ *Hessenberg matrix whose nonzero entries* $h_{i,j}$ *are defined by Arnoldi's algorithm. Furthermore, denote by* $e_m = \{0, 0, \ldots, 1\}^T$ *and by* $\boldsymbol{H}_m$ *the matrix obtained from* $\bar{\boldsymbol{H}}_m$ *by deleting its last row. Then the following relations hold:*

$$\begin{aligned} \mathbf{A}\mathbf{V}_m &= \mathbf{V}_m\mathbf{H}_m + \mathbf{w}_m e_m^T & (4.31)\\ &= \mathbf{V}_{m+1}\bar{\mathbf{H}}_m, & (4.32)\\ \mathbf{V}_m^T\mathbf{A}\mathbf{V}_m &= \mathbf{H}_m. & (4.33) \end{aligned}$$

4.6.2 GMRES idea

GMRES method is a projection method based on the m-th Krylov subspace $\mathcal{K}_m$ with initial vector $\mathbf{v}_1 = \mathbf{r}_0/||\mathbf{r}_0||_2$. The basic idea of this method will be described below.

Any vector $\mathbf{x}$ in $\mathbf{x}_0 + \mathcal{K}_m$ can be written as

$$\mathbf{x} = \mathbf{x}_0 + \mathbf{V}_m\mathbf{y}, \tag{4.34}$$

where $\mathbf{y}$ is an m-vector. Defining

$$\begin{aligned} J(\mathbf{y}) &= ||\mathbf{b} - \mathbf{A}\mathbf{x}||_2\\ &= ||\mathbf{b} - \mathbf{A}\left(\mathbf{x}_0 + \mathbf{V}_m\mathbf{y}\right)||_2 & (4.35)\\ &= ||\mathbf{r}_0 - \mathbf{A}\mathbf{V}_m\mathbf{y}||_2, \end{aligned}$$

relation (4.32) results in

$$\begin{aligned} \mathbf{b} - \mathbf{A}\mathbf{x} &= \mathbf{r}_0 - \mathbf{A}\mathbf{V}_m\mathbf{y}\\ &= \beta\mathbf{v}_1 - \mathbf{V}_{m+1}\bar{\mathbf{H}}_m\mathbf{y} & (4.36)\\ &= \mathbf{V}_{m+1}\left(\beta e_1 - \bar{\mathbf{H}}_m\mathbf{y}\right), \end{aligned}$$

where we have used $\beta = ||\mathbf{r}_0||_2$ and $e_1 = \{1, 0, \ldots, 0\}^T$.

Since the column-vectors of $\mathbf{V}_{m+1}$ are orthonormal, expression (4.35) can be written as

$$\begin{aligned} J(\mathbf{y}) &= ||\mathbf{r}_0 - \mathbf{A}\mathbf{V}_m\mathbf{y}||_2 \\ &= ||\mathbf{V}_{m+1}\left(\beta e_1 - \bar{\mathbf{H}}_m\mathbf{y}\right)||_2 \\ &= ||\beta e_1 - \bar{\mathbf{H}}_m\mathbf{y}||_2. \end{aligned} \tag{4.37}$$

The GMRES approximation is the unique vector of $\mathbf{x}_0+\mathcal{K}_m$ which minimizes (4.35). By (4.34) and (4.37), this approximation can be obtained quite simply as $\mathbf{x}_m = \mathbf{x}_0 + \mathbf{V}_m\mathbf{y}_m$ where $\mathbf{y}_m$ minimizes the function $J(\mathbf{y}) = ||\beta e_1 - \bar{\mathbf{H}}_m\mathbf{y}||_2$, i.e.,

$$\begin{aligned} \mathbf{x}_m &= \mathbf{x}_0 + \mathbf{V}_m\mathbf{y}_m, \quad \text{with} \\ \mathbf{y}_m &= \arg\min_{\mathbf{y}} ||\beta e_1 - \bar{\mathbf{H}}_m\mathbf{y}||_2. \end{aligned} \tag{4.38}$$

The minimizer $\mathbf{y}_m$ is *inexpensive* to compute since it requires the solution of an $(m+1) \times m$ least-squares probles where m is typically small.

4.6.3 GMRES algorithm

The GMRES algorithm is now given as follows:

Generalized Minimum Residual (GMRES) Algorithm

1. Choose $\mathbf{x}_0$ and 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** $j := 1, 2, \ldots, m$ **Do** :
3. $\quad \mathbf{w}_j := \mathbf{A}\mathbf{v}_j$
4. $\quad$ **For** $i := 1, 2, \ldots, j$ **Do** :
5. $\quad\quad h_{i,j} := (\mathbf{w}_j, \mathbf{v}_i)$
6. $\quad\quad \mathbf{w}_j := \mathbf{w}_j - h_{ij}\mathbf{v}_i$
7. $\quad$ **EndDo**
8. $\quad h_{j+1,j} := ||\mathbf{w}_j||_2$
9. $\quad \mathbf{v}_{j+1} := \dfrac{\mathbf{w}_j}{h_{j+1,j}}$
10. **EndDo**
11. Compute $\mathbf{y}_m := \arg\min_{\mathbf{y}} ||\beta e_1 - \bar{\mathbf{H}}_m\mathbf{y}||_2$
12. Compute $\mathbf{x}_m := \mathbf{x}_0 + \mathbf{V}_m\mathbf{y}_m$

As we can see, line 2 to 10 represent Arnoldi's algorithm for orthogonalization.

The GMRES algorithm may break down if $h_{j+1,j} = 0$ at iteration step j (see line 9). However, this situation implies that the residual vector is zero and therefore, the algorithm gives the exact solution at this step. Hence, examination of value $h_{j+1,j}$ becomes important.

If the iteration number m is *large*, the GMRES algorithm becomes *impractical* as consequence of a lack of memory and increasing computational requirements. This is understandable from the fact that during the Arnoldi steps (lines 2-10) the number of vectors requiring storage increases. There are several ways to remedy this problem, like *restarting* and *truncating*, see p.79 of Vuik [28].

4.7 Bi-CG method

The CG method is *not* suitable for *non-Hermitian* systems, because the residual vectors cannot be made orthogonal with *short* recurrences, as proved in Voevodin [25] and Faber and Manteuffel [9]. The GMRES method retains orthogonality of the residuals by using *long* recurrences, at the cost of a larger storage demand. The Biconjugate Gradient (Bi-CG) method takes another approach, replacing the orthogonal sequence of residuals by two mutually orthogonal sequences, at the price of *no* longer providing a minimization.

The update relations for residuals in the CG method are augmented in the biconjugate gradient method by relations that are similar but based on $\mathbf{A}^H$ instead of $\mathbf{A}$. Thus we update two sequences of residuals

$$\begin{cases} \mathbf{r}_{j+1} &= \mathbf{r}_j - \alpha_j \mathbf{A}\mathbf{p}_j, \\ \tilde{\mathbf{r}}_{j+1} &= \tilde{\mathbf{r}}_j - \alpha_j \mathbf{A}^H \tilde{\mathbf{p}}_j, \end{cases} \tag{4.39}$$

and two sequences of search directions

$$\begin{cases} \mathbf{p}_{j+1} &= \mathbf{r}_{j+1} + \beta_j \mathbf{p}_j, \\ \tilde{\mathbf{p}}_{j+1} &= \tilde{\mathbf{r}}_{j+1} + \beta_j \tilde{\mathbf{p}}_j. \end{cases} \tag{4.40}$$

The choices

$$\begin{cases} \alpha_j &= \frac{(\mathbf{r}_j, \tilde{\mathbf{r}}_j)}{(\mathbf{A}\mathbf{p}_j, \tilde{\mathbf{p}}_j)}, \\ \beta_j &= \frac{(\mathbf{r}_{j+1}, \tilde{\mathbf{r}}_{j+1})}{(\mathbf{r}_j, \tilde{\mathbf{r}}_j)}, \end{cases} \tag{4.41}$$

ensure the orthogonality relations

$$(\mathbf{r}_i, \tilde{\mathbf{r}}_j) = (\mathbf{A}\mathbf{p}_i, \tilde{\mathbf{p}}_j) = 0 \quad \text{for } i \neq j. \tag{4.42}$$

4.7.1 Bi-CG algorithm

Below we describe the Bi-CG algorithm.

Biconjugate Gradient (Bi-CG) Algorithm

1. Compute $\mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0$ and set $\mathbf{p}_0 := \mathbf{r}_0$
2. Choose $\tilde{\mathbf{r}}_0$ such that $(\mathbf{r}_0, \tilde{\mathbf{r}}_0) \neq 0$ and set $\tilde{\mathbf{p}}_0 := \tilde{\mathbf{r}}_0$

3. **For** $j := 0, 1, \ldots,$ *until convergence* **Do** :

4. $\quad \mathbf{w}_j := \mathbf{A}\mathbf{p}_j$
5. $\quad \tilde{\mathbf{w}}_j := \mathbf{A}^H \tilde{\mathbf{p}}_j$
6. $\quad \alpha_j := \frac{(\mathbf{r}_j, \tilde{\mathbf{r}}_j)}{(\mathbf{w}_j, \tilde{\mathbf{p}}_j)}$
7. $\quad \mathbf{x}_{j+1} := \mathbf{x}_j + \alpha_j \mathbf{p}_j$
8. $\quad \mathbf{r}_{j+1} := \mathbf{r}_j - \alpha_j \mathbf{w}_j$
9. $\quad \tilde{\mathbf{r}}_{j+1} := \tilde{\mathbf{r}}_j - \alpha_j \tilde{\mathbf{w}}_j$
10. $\quad \beta_j := \frac{(\mathbf{r}_{j+1}, \tilde{\mathbf{r}}_{j+1})}{(\mathbf{r}_j, \tilde{\mathbf{r}}_j)}$
11. $\quad \mathbf{p}_{j+1} := \mathbf{r}_{j+1} + \beta_j \mathbf{p}_j$
12. $\quad \tilde{\mathbf{p}}_{j+1} = \tilde{\mathbf{r}}_{j+1} + \beta_j \tilde{\mathbf{p}}_j$

13. **EndDo**

For symmetric positive definite systems, Bi-CG is equivalent with CG, but it takes twice the cost per iteration.

A small disadvantage of Bi-CG is the extra computation involving $\mathbf{A}^H$ which can be costly if the solution for the dual system is not of interest.

It is difficult to make a fair comparison between GMRES and Bi-CG. GMRES really minimizes a residual, but at the cost of increasing work for keeping all residuals orthogonal and increasing demands for memory space. Bi-CG does not minimize a residual, but often its accuracy is comparable to GMRES, at the cost of twice the amount of matrix vector products per iteration step. However, the generation of the basis vectors is relatively cheap and the memory requirements are modest. Several variants of Bi-CG have been proposed (e.g., conjugate gradient squared (CGS) method and biconjugate gradient stabilized (Bi-CGstab) method, see next subsections) that increase the effectiveness of this class of methods in certain circumstances.

4.8 CGS method

The conjugate gradient squared (CGS) method was developed in 1984 by Sonneveld [22].

In the BiCG method, the residual vector $\mathbf{r}_j$ can be regarded as the product of $\mathbf{r}_0$ and the j-th degree polynomial $\mathcal{R}$ in $\mathbf{A}$, i.e.,

$$\mathbf{r}_j = \mathcal{R}_j(\mathbf{A})\mathbf{r}_0, \tag{4.43}$$

satisfying the constraint $\mathcal{R}_j(0) = 1$. This same polynomial satisfies also

$$\tilde{\mathbf{r}}_j = \mathcal{R}_j(\mathbf{A}^T)\tilde{\mathbf{r}}_0, \tag{4.44}$$

by construction of the BiCG method.

Similarly, the conjugate-direction polynomial $\mathcal{P}_j(t)$ satisfies the following two expressions

$$\mathbf{p}_j = \mathcal{P}_j(\mathbf{A})\mathbf{r}_0, \quad \tilde{\mathbf{p}}_j = \mathcal{P}_j(\mathbf{A}^T)\tilde{\mathbf{r}}_0. \tag{4.45}$$

Also, the scalar α_j in BiCG is given by

$$\alpha_j = \frac{(\mathbf{r}_j, \tilde{\mathbf{r}}_j)}{(\mathbf{A}\mathbf{p}_j, \tilde{\mathbf{p}}_j)} = \frac{(\mathcal{R}_j(\mathbf{A})\mathbf{r}_0, \mathcal{R}_j(\mathbf{A}^T)\tilde{\mathbf{r}}_0)}{(\mathbf{A}\mathcal{P}_j(\mathbf{A})\mathbf{r}_0, \mathcal{P}_j(\mathbf{A}^T)\tilde{\mathbf{r}}_0)} = \frac{(\mathcal{R}_j^2(\mathbf{A})\mathbf{r}_0, \tilde{\mathbf{r}}_0)}{(\mathbf{A}\mathcal{P}_j^2(\mathbf{A})\mathbf{r}_0, \tilde{\mathbf{r}}_0)}. \tag{4.46}$$

which indicates that if it is possible to obtain a recursion for the vectors $\mathcal{R}_j^2(\mathbf{A})\mathbf{r}_0$ and $\mathcal{P}_j^2(\mathbf{A})\mathbf{r}_0$, then computing α_j and, similarly, β_j cause no problem.

Expression (4.46) suggests that if $\mathcal{R}_j(\mathbf{A})$ reduces $\mathbf{r}_0$ to a smaller vector $\mathbf{r}_j$, then it might be advantageous to apply this 'contraction'-operator twice and compute $\mathcal{R}_j^2(\mathbf{A})\mathbf{r}_0$. The iteration coefficients can still be recovered from these vectors (as shown above), and it turns out to be easy to find the corresponding approximations for $\mathbf{x}$. This approach is the conjugate gradient squared (CGS) method.

4.8.1 CGS algorithm

The derivation of CGS relies on simple but laborous algebra only. To establish the desired recurrences for the squared polynomials, recurrences are used which define $\mathcal{R}_j$ and $\mathcal{P}_j$,

$$\begin{cases} \mathcal{R}_{j+1}(t) &= \mathcal{R}_j(t) - \alpha_j t\mathcal{P}_j, \\ \mathcal{P}_{j+1}(t) &= \mathcal{R}_{j+1}(t) + \beta_j\mathcal{P}_j, \end{cases} \tag{4.47}$$

which can be compared with lines (6) and (8) of the CG algorithm.

The whole derivation of CGS can be found at p.214-216 of Saad [19]. Below we describe the CGS algorithm.

Conjugate Gradient Squared (CGS) Algorithm

1. Compute $\mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0$ and set $\mathbf{p}_0 := \mathbf{u}_0 := \mathbf{r}_0$
2. Choose $\tilde{\mathbf{r}}_0$ arbitrary.
3. **For** $j := 0, 1, \ldots,$ *until convergence* **Do** :

4. $\quad \mathbf{w}_j := \mathbf{A}\mathbf{p}_j$
5. $\quad \alpha_j := \frac{(\mathbf{r}_j, \tilde{\mathbf{r}}_0)}{(\mathbf{w}_j, \tilde{\mathbf{r}}_0)}$
6. $\quad \mathbf{q}_j := \mathbf{u}_j - \alpha_j\mathbf{w}_j$

7. $\mathbf{x}_{j+1} := \mathbf{x}_j + \alpha_j \left(\mathbf{u}_j + \mathbf{q}_j\right)$
8. $\mathbf{r}_{j+1} := \mathbf{r}_j - \alpha_j \mathbf{A} \left(\mathbf{u}_j + \mathbf{q}_j\right)$
9. $\beta_j := \frac{(\mathbf{r}_{j+1}, \tilde{\mathbf{r}}_0)}{(\mathbf{r}_j, \tilde{\mathbf{r}}_0)}$
10. $\mathbf{u}_{j+1} = \mathbf{r}_{j+1} + \beta_j \mathbf{q}_j$
11. $\mathbf{p}_{j+1} := \mathbf{u}_{j+1} + \beta_j \left(\mathbf{q}_j + \beta_j \mathbf{p}_j\right)$
12. **EndDo**

CGS avoids using $\mathbf{A}^H$ as in BiCG, which is an advantage when $\mathbf{A}^H$ is not of our interest.

In the BiCG method, the residual vector is given by $\mathbf{r}_j = \mathcal{R}_j(\mathbf{A})\mathbf{r}_0$. In CGS we have $\tilde{\mathbf{r}}_j = \mathcal{R}_j^2(\mathbf{A})\mathbf{r}_0$ (see p.75-76 of Vuik [28]), therefore the name 'CG Squared'. In many cases, this causes the algorithm to converge twice as fast as BiCG, while the same number of operations per iteration is required. However, because of squaring of polynomials, jumps in the residual errors tend to be more damaging than in the standard BiCG algorithms, see Sonneveld [22].

4.9 Bi-CGSTAB method

The CGS algorithm is based on squaring the residual polynomial and therefore it shows often *irregular* convergence patterns which may lead to substantial build-up of rouding errors, see Van der Vorst [27]. The Biconjugate Gradient Stabilized (Bi-CGSTAB) algorithm is a variation of CGS, which was developed by Van der Vorst to remedy this difficulty. Instead of computing the residual vector $\tilde{\mathbf{r}}_j = \mathcal{R}_j^2(\mathbf{A})\mathbf{r}_0$, Bi-CGSTAB computes

$$\tilde{\mathbf{r}}_j = \mathcal{Q}_j(\mathbf{A})\mathcal{R}_j(\mathbf{A})\mathbf{r}_0, \tag{4.48}$$

with $\mathcal{Q}_j$ a new polynomial which is defined recursively at each step with the goal of 'stabilizing' or 'smoothing' the convergence behaviour of the original algorithm. Specifically, $\mathcal{Q}_j$ is defined by the simple recurrence

$$\mathcal{Q}_{j+1}(t) = (1 - \omega_j t)\mathcal{Q}_j. \tag{4.49}$$

This is equivalent with

$$\mathcal{Q}_{j+1}(t) = (1 - \omega_0 t)(1 - \omega_1 t) \cdots (1 - \omega_j t), \tag{4.50}$$

in which the scalars ω_k for all $k = 0, \ldots, j$ are to be determined.

4.9.1 Bi-CGSTAB derivation

We give the derivation of Bi-CGSTAB, which is almost the same as in subsection 7.4.2 of Saad [19].

Ignoring the scalar coeffients at first, we start with a relation for the residual polynomial $\mathcal{Q}_{j+1}\mathcal{R}_{j+1}$. We immediately obtain

$$\begin{aligned} \mathcal{Q}_{j+1}\mathcal{R}_{j+1} &= (1-\omega_j t)\mathcal{Q}_j\mathcal{R}_{j+1} \\ &= (1-\omega_j t)\mathcal{Q}_j(\mathcal{R}_j - \alpha_j t\mathcal{P}_j) \\ &= (1-\omega_j t)(\mathcal{Q}_j\mathcal{R}_j - \alpha_j t\mathcal{Q}_j\mathcal{P}_j). \end{aligned} \tag{4.51}$$

Moreover we can write

$$\begin{aligned} \mathcal{Q}_j\mathcal{P}_j &= \mathcal{Q}_j(\mathcal{R}_j(t) + \beta_{j-1}\mathcal{P}_{j-1}) \\ &= \mathcal{Q}_j\mathcal{R}_j + \beta_{j-1}\mathcal{Q}_j\mathcal{P}_{j-1} \\ &= \mathcal{Q}_j\mathcal{R}_j + \beta_{j-1}(1-\omega_{j-1}t)\mathcal{Q}_{j-1}\mathcal{P}_{j-1}. \end{aligned} \tag{4.52}$$

In expressions (4.51) and (4.52) we used recurrences (4.47) formulated in the CGS method. Define

$$\left\{ \begin{array}{rcl} \mathbf{r}_j &=& \mathcal{Q}_j(\mathbf{A})\mathcal{R}_j(\mathbf{A})\mathbf{r}_0, \\ \mathbf{p}_j &=& \mathcal{Q}_j(\mathbf{A})\mathcal{P}_j(\mathbf{A})\mathbf{r}_0. \end{array} \right. \tag{4.53}$$

According to formulae (4.53), these vectors can be updated from a double recurrence provided the scalars α_j and β_j were computable. Using (4.51) and (4.52) this recurrence is

$$\left\{ \begin{array}{rcl} \mathbf{r}_{j+1} &=& (I-\omega_j A)(\mathbf{r}_j - \alpha_j A\mathbf{p}_j), \\ \mathbf{p}_{j+1} &=& \mathbf{r}_j + \beta_j(I-\omega_j A)\mathbf{p}_j. \end{array} \right. \tag{4.54}$$

Now, we consider the computation of the scalars α_j and β_j needed in the recurrence. According to line 10 of the original Bi-CG algorithm, one can write $\beta_j = \rho_{j+1}/\rho_j$ with

$$\begin{aligned} \rho_j &= (\mathcal{R}_j(\mathbf{A})\mathbf{r}_0, \mathcal{R}_j(\mathbf{A}^T)\tilde{\mathbf{r}}_0) \\ &= (\mathcal{R}_j^2(\mathbf{A})\mathbf{r}_0, \tilde{\mathbf{r}}_0). \end{aligned} \tag{4.55}$$

Unfortunately, ρ_j is not computable from (4.55) because none of the vectors $\mathcal{R}_j(\mathbf{A})\mathbf{r}_0, \mathcal{R}_j(\mathbf{A}^T)\tilde{\mathbf{r}}_0$ or $\mathcal{R}_j^2(\mathbf{A})\mathbf{r}_0$ is available. However, ρ_j can be related to the scalar

$$\tilde{\rho}_j = (\mathcal{R}_j(\mathbf{A})\mathbf{r}_0, \mathcal{Q}_j(\mathbf{A}^T)\tilde{\mathbf{r}}_0), \tag{4.56}$$

which is computable via

$$\begin{aligned} \tilde{\rho}_j &= (\mathcal{R}_j(\mathbf{A})\mathbf{r}_0, \mathcal{Q}_j(\mathbf{A}^T)\tilde{\mathbf{r}}_0) \\ &= (\mathcal{Q}_j(\mathbf{A})\mathcal{R}_j(\mathbf{A})\mathbf{r}_0, \tilde{\mathbf{r}}_0) \\ &= (\mathbf{r}_j, \tilde{\mathbf{r}}_0), \end{aligned} \tag{4.57}$$

using (4.53). To relate the scalars $\tilde{\rho}_j$ and $\tilde{\rho}_j$, expand $\mathcal{Q}_j(\mathbf{A}^T)\tilde{\mathbf{r}}_0$ explicitly in the power basis, to obtain

$$\tilde{\rho}_j = (\mathcal{R}_j(\mathbf{A})\mathbf{r}_0, \eta_1^{(j)}(\mathbf{A}^T)^j\tilde{\mathbf{r}}_0 + \eta_2^{(j)}(\mathbf{A}^T)^{j-1}\tilde{\mathbf{r}}_0 + \ldots), \tag{4.58}$$

where $\eta_1^{(j)}, \eta_2^{(j)}, \ldots$ are scalars. Since $\mathcal{R}_j(\mathbf{A})\mathbf{r}_0$ is orthogonal to all vectors $(\mathbf{A}^T)^j\tilde{\mathbf{r}}_0$ with $k \neq j$ (see (4.42) of Bi-CG method), only the leading power is relevant in the expansion on the right side of the inner product in (4.58), i.e., expression (4.58) can be written as

$$\tilde{\rho}_j = (\mathcal{R}_j(\mathbf{A})\mathbf{r}_0, \eta_1^{(j)}(\mathbf{A}^T)^j\tilde{\mathbf{r}}_0). \tag{4.59}$$

In particular, if $\gamma_1^{(j)}$ is the leading coefficient for the polynomial $\mathcal{R}_j$, then

$$\tilde{\rho}_j = (\mathcal{R}_j(\mathbf{A})\mathbf{r}_0, \frac{\eta_1^{(j)}}{\gamma_1^{(j)}}\mathcal{R}_j(\mathbf{A}^T)\tilde{\mathbf{r}}_0) = \frac{\eta_1^{(j)}}{\gamma_1^{(j)}}\rho_j. \tag{4.60}$$

When examining the recurrence relations for $\mathcal{R}_{j+1}$ and $\mathcal{Q}_{j+1}$, leading coefficients for these polynomials are found to satisfy the relations

$$\left\{ \begin{array}{lcl} \eta_1^{(j+1)} & = & -\omega_j\eta_1^{(j)}, \\ \gamma_1^{(j+1)} & = & -\alpha_j\gamma_1^{(j)}. \end{array} \right. \tag{4.61}$$

As a result,

$$\begin{array}{rcl} \frac{\tilde{\rho}_{j+1}}{\tilde{\rho}_j} & = & \frac{\eta_1^{(j+1)}}{\gamma_1^{(j+1)}}\frac{\gamma_1^{(j)}}{\eta_1^{(j)}}\frac{\rho_{j+1}}{\rho_j} \\ & = & \frac{\omega_j}{\alpha_j}\frac{\rho_{j+1}}{\rho_j}, \end{array} \tag{4.62}$$

which yields the following relation for β_j

$$\beta_j = \frac{\alpha_j}{\omega_j}\frac{\rho_{j+1}}{\rho_j}. \tag{4.63}$$

Similarly, a simple recurrence formula for α_j can be derived. By definition,

$$\alpha_j = \frac{(\mathcal{R}_j(\mathbf{A})\mathbf{r}_0, \mathcal{R}_j(\mathbf{A}^T)\tilde{\mathbf{r}}_0)}{(\mathbf{A}\mathcal{P}_j(\mathbf{A})\mathbf{r}_0, \mathcal{P}_j(\mathbf{A}^T)\tilde{\mathbf{r}}_0)}, \tag{4.64}$$

see lines (3) and (4) of the CG algorithm. As in the previous case, the polynomials in the right sides of the inner products in both the numerator and denominator can be replaced by their leading terms. However, in this case, the leading coefficient for $\mathcal{R}_j(\mathbf{A}^T)\tilde{\mathbf{r}}_0$ and $\mathcal{P}_j(\mathbf{A}^T)\tilde{\mathbf{r}}_0$ are identical, since we have chosen $\mathbf{p}_0 = \mathbf{r}_0$, and therefore,

$$\begin{array}{rcl} \alpha_j & = & \frac{(\mathcal{R}_j(\mathbf{A})\mathbf{r}_0, \mathcal{R}_j(\mathbf{A}^T)\tilde{\mathbf{r}}_0)}{(\mathbf{A}\mathcal{P}_j(\mathbf{A})\mathbf{r}_0, \mathcal{R}_j(\mathbf{A}^T)\tilde{\mathbf{r}}_0)} \\ & = & \frac{(\mathcal{R}_j(\mathbf{A})\mathbf{r}_0, \mathcal{Q}_j(\mathbf{A}^T)\tilde{\mathbf{r}}_0)}{(\mathbf{A}\mathcal{P}_j(\mathbf{A})\mathbf{r}_0, \mathcal{Q}_j(\mathbf{A}^T)\tilde{\mathbf{r}}_0)} \\ & = & \frac{(\mathcal{Q}_j(\mathbf{A})\mathcal{R}_j(\mathbf{A})\mathbf{r}_0, \tilde{\mathbf{r}}_0)}{(\mathbf{A}\mathcal{Q}_j(\mathbf{A})\mathcal{P}_j(\mathbf{A})\mathbf{r}_0, \tilde{\mathbf{r}}_0)} \\ & = & \frac{\tilde{\mathbf{p}}_j}{(\mathbf{A}\mathbf{p}_j, \tilde{\mathbf{r}}_0)}, \end{array} \tag{4.65}$$

where we used expressions (4.53) and (4.56) in the last equality.

Next, the parameter ω_j must be defined. This can be thought of as an additional free parameter. One of the simplest choices, and perhaps the most natural, is to select ω_j to achieve a steepest descent step in the residual direction obtained before multiplying the residual vector by $(I - \omega_j \mathbf{A})$ in (4.54). In other words, ω_j is chosen to minimize the 2-norm of the vector $\tilde{\mathbf{r}}_{j+1} = (I - \omega_j \mathbf{A})\mathcal{Q}_j(\mathbf{A})\mathcal{R}_{j+1}(\mathbf{A})\mathbf{r}_0$, i.e.,

$$\min_{\omega \in \mathbb{R}} ||(I - \omega_j \mathbf{A})\mathcal{Q}_j(\mathbf{A})\mathcal{R}_{j+1}(\mathbf{A})\mathbf{r}_0||_2. \tag{4.66}$$

Notice that the process minimalizes in each iteration. Equation (4.54) can be rewritten as

$$\mathbf{r}_{j+1} = (I - \omega_j A)\mathbf{s}_j, \tag{4.67}$$

in which

$$\mathbf{s}_j \equiv \mathbf{r}_j - \alpha_j A\mathbf{p}_j. \tag{4.68}$$

Then the optimal value for ω_j is given by

$$\omega_j = \frac{(\mathbf{A}\mathbf{s}_j, \mathbf{s}_j)}{(\mathbf{A}\mathbf{s}_j, \mathbf{A}\mathbf{s}_j)}. \tag{4.69}$$

Finally, a formula is needed to update the approximate solution $\mathbf{x}_{j+1}$ from $\mathbf{x}_j$. Equation (4.67) can be written as

$$\mathbf{r}_{j+1} = \mathbf{s}_j - \omega_j A\mathbf{s}_j = \mathbf{r}_j - \alpha_j A\mathbf{p}_j - \omega_j A\mathbf{s}_j, \tag{4.70}$$

which is equivalent to

$$\mathbf{A}^{-1}(\mathbf{r}_j - \mathbf{r}_{j+1}) = \alpha_j \mathbf{p}_j + \omega_j \mathbf{s}_j. \tag{4.71}$$

This yields the following update formula

$$\mathbf{x}_{j+1} = \mathbf{x}_j + \alpha_j \mathbf{p}_j + \omega_j \mathbf{s}_j. \tag{4.72}$$

4.9.2 Bi-CGSTAB algorithm

The resulting algorithm is given below.

Biconjugate Gradient Stabilized (Bi-CGSTAB) Algorithm

1. Compute $\mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0$ and set $\mathbf{p}_0 := \mathbf{r}_0$
2. Choose $\tilde{\mathbf{r}}_0$ arbitrary
3. **For** $j := 0, 1, \ldots,$ *until convergence* **Do** :

4. $\quad \mathbf{w}_j := \mathbf{A}\mathbf{p}_j$
5. $\quad \mathbf{v}_j := \mathbf{A}\mathbf{s}_j$

6. $\alpha_j := \frac{\tilde{\mathbf{p}}_j}{(\mathbf{w}_j, \tilde{\mathbf{r}}_0)}$
7. $\mathbf{s}_j := \mathbf{r}_j - \alpha_j \mathbf{w}_j$
8. $\omega_j := \frac{(\mathbf{v}_j, \mathbf{s}_j)}{(\mathbf{v}_j, \mathbf{v}_j)}$
9. $\mathbf{x}_{j+1} := \mathbf{x}_j + \alpha_j \mathbf{p}_j + \omega_j \mathbf{s}_j$
10. $\mathbf{r}_{j+1} := \mathbf{s}_j - \omega_j \mathbf{v}_j$
11. $\beta_j := \frac{\alpha_j}{\omega_j} \frac{\rho_{j+1}}{\rho_j}$
12. $\mathbf{p}_{j+1} := \mathbf{r}_j + \beta_j(\mathbf{p}_j - \omega_j \mathbf{w}_j)$
13. **EndDo**

Investigation of the Bi-CGSTAB algorithm has been reported in Van der Vorst [27] for various applications and compared to BiCG and CGS. In general, Bi-CGSTAB converges *more smoothly* than CGS of 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, Bi-CGSTAB still converges. The convergence rate is also faster than CGS or BiCG. However, for symmetric matrix cases, these significances are *not* clearly seen.

Though Bi-CGSTAB is an attractive alternative to CGS, further investigation reveal a weakness of this algorithm, as mentioned in Erlangga [6]. 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 Bi-CGSTAB have been proposed, resulting in Bi-CGSTAB(l) where $l \in \mathbb{N}$, see Sleijpen and Fokkema [21]. This are modifications by forming a general l-order minimum-residual polynomial, instead of using $l = 1$ in the original Bi-CGSTAB.

The original Bi-CGSTAB has been tried to solve the discrete Helmholtz equation, see e.g. Mulder & Plessix [17], but unfortunately without any success.

4.10 Stopcriterium

In all iterative algorithms given in this section contain the following statement

For $j := 0, 1, \ldots,$ *until convergence* **Do** :

To specify '*until convergence*' one can choose various stopcriteria. It depends on each practical situation which of them should be chosen for an accurate and efficient solution. Standard stopcriteria are for instance

$$||\mathbf{r}_j||_2 < \varepsilon, \tag{4.73}$$

and

$$K(\mathbf{A})\frac{||\mathbf{r}_j||_2}{||\mathbf{b}||_2} < \varepsilon, \tag{4.74}$$

with $K(\mathbf{A})$ the condition number of $\mathbf{A}$ and a given $\varepsilon > 0$. In expressions (4.73) and (4.74) one aims to minimize, respectively, the norm of the residual and the norm of the relative error.

4.11 Discussion

We have dealt with several iterative methods in the previous sections. In this section we discuss the applicability of these methods to HBVP. For example, CGS and Bi-CG are not known to be used for solving HBVP (see p.32 of [6]). Also, CG so far is only used for comparison studies, since theoretically it is not suitable for indefinite problems. Recall that in HBVP we have a matrix $\mathbf{A}$ which is *indefinite* and *symmetric complex*. For indefinite problems, CGNR is used due to its simplicity. However, without good preconditioners CGNR is often not attractive because of slow convergence.

Some authors use GMRES, Bi-CGSTAB and COCG to solve HBVP, but again, without good preconditioners these methods are *not* efficient.

It appears that the choice of the *wavenumber* has a large impact on the convergence behaviour of the iterative methods. Therefore, there doesn't exist a best iterative method which can be used to deal all variants of HBVP.

Since all authors use different test cases, it is somewhat difficult to compare the iterative methods with each other. Furthermore, the fact that the methods show slow convergence, without being preconditioned, makes it impossible to compare the methods without incorporating preconditioners.

Chapter 5

Preconditioning techniques

5.1 Introduction

Lack of *robustness* and *efficiency* are a widely recognized weakness of iterative solvers, relative to direct solvers. This is mainly the consequence of the fact that the convergence behaviour of Krylov subspace methods depends strongly on the eigenvalue distribution of the coefficient matrix.

Both robustness and efficiency can be improved by using *preconditioning*. Preconditioning is simply a means of transforming the original linear system into one which has the *same* solution, but which is likely to be *easier* to solve with an iterative solver.

If we want to solve $\mathbf{Ax} = \mathbf{b}$ with a preconditioner $\mathbf{P}$ which is a matrix, then we could solve the following preconditioned system:

$$\mathbf{P}^{-1}\mathbf{Ax} = \mathbf{P}^{-1}\mathbf{b} \tag{5.1}$$

or

$$\mathbf{AP}^{-1}\mathbf{y} = \mathbf{b}, \quad \mathbf{x} = \mathbf{P}^{-1}\mathbf{y}. \tag{5.2}$$

We are looking for a preconditioner $\mathbf{P}$ such that (5.1) or (5.2) is easier to solve, relative to the original system. In general, we aim to find $\mathbf{P}_L$ and $\mathbf{P}_R$ such that

$$\mathbf{P}_L^{-1}\mathbf{AP}_R^{-1}\mathbf{y} = \mathbf{P}_L^{-1}\mathbf{b}, \quad \mathbf{x} = \mathbf{P}_R^{-1}\mathbf{y}. \tag{5.3}$$

is easier to solve. The ideal choice is $\mathbf{P}_L = \mathbf{A}$ and $\mathbf{P}_R = \mathbf{I}$, which is obviously impractical since in general the inverse of $\mathbf{A}$ is expensive to compute. One can look for a preconditioner $\mathbf{P}_L^{-1}$ which approximate the inverse of $\mathbf{A}$ and is relatively inexpensive to compute.

A linear system obtained from discretizations of a PDE can have a highly distributed spectrum and can result in an indefinite system, i.e., 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, i.e., the spectrum consists of eigenvalues which are concentrated in a coherent region. It is also important that a preconditioned system does *not* have an eigenvalue close to zero.

To summarize, we can distinguish two approaches for constructing preconditioners:

- *Matrix*-based approach. Within this class are, e.g., several variants of ILU-factorization [19].

- *Operator*-based approach. Examples of these kind of preconditioners are shifted Laplace preconditioners [7, 8], AILU [10] and separation–of–variables [17].

In general, the *reliability* of iterative techniques, when dealing with various applications, depends much more on the *quality* of the preconditioner than on the particular Krylov subspace accelerators used. In this chapter we shall cover some of these preconditioners which can be applied to the HBVP, where we consider the Helmholtz operator with a *minus*-sign, i.e., $-\Delta - k^2$.

5.2 Diagonal preconditioner

We transforms the original system in the following preconditioned system

$$\tilde{\mathbf{A}}\tilde{\mathbf{x}} = \tilde{\mathbf{b}}, \tag{5.4}$$

where $\tilde{\mathbf{A}} = \mathbf{P}^{-1}\mathbf{A}\mathbf{P}^{-T}, \mathbf{x} = \mathbf{P}^{-T}\tilde{\mathbf{x}}, \tilde{\mathbf{b}} = \mathbf{P}^{-1}\mathbf{b}, \mathbf{P}$ is a non-singular matrix and $\mathbf{A}$ is *symmetric positive definite* matrix with dimensions $N \times N$.

A simple choice for $\mathbf{P}$ is a diagonal matrix with diagonal elements

$$\mathbf{P}_{ii} = \sqrt{\mathbf{A}_{ii}}, \quad \text{for } i = 1, 2, \ldots, N. \tag{5.5}$$

Then we can easily derive

$$\tilde{\mathbf{A}}_{ii} = \mathbf{P}_{ii}^{-1}\mathbf{A}_{ii}\mathbf{P}_{ii}^{-T} = 1, \quad \text{for } i = 1, 2, \ldots, N. \tag{5.6}$$

In Van der Sluis [23] it has been shown that this choice for $\mathbf{P}$ minimizes the condition number of $\tilde{\mathbf{A}}$, if $\mathbf{P}$ is restricted to be a diagional matrix. For this preconditioner it is advantageous to apply CG to $\tilde{\mathbf{A}}\tilde{\mathbf{x}} = \tilde{\mathbf{b}}$, since $\tilde{\mathbf{A}}$ is easy to calculate.

However, in HBVP we have a Hermitian and symmetric-complex matrix $\mathbf{A}$, instead of a symmetric positive definite one. More research is required to decide of the diagonal preconditioner can be used in this case.

5.3 Matrix-splitting preconditioners

In this section we treat, successively, Jacobi, Gauss-Seidel, SOR and SSOR preconditioners. In many cases, the performance of this preconditioners is rather poor, unless modifications are incorporated into the original algorithm. It appears that the results found in the next subsections have become the basis for developing another types of preconditioners.

5.3.1 Jacobi and Gauss-Seidel preconditioners

In section 4.2 (Krylov subspace method) we have seen that Jacobi and Gauss-Seidel methods give the following expressions for $\mathbf{P}$ in (5.1):

$$\begin{cases} \mathbf{P}_{JAC} &= \mathbf{D}; \\ \mathbf{P}_{GS} &= \mathbf{D} - \mathbf{L}, \end{cases} \tag{5.7}$$

where $\mathbf{A} = \mathbf{D} - \mathbf{L} - \mathbf{L}^T$ with $\mathbf{D}$ the matrix with the main diagonal and $\mathbf{L}$ the (strict) lower triangular part of $\mathbf{A}$.

5.3.2 SOR and SSOR preconditioners

Generalization of *Jacobi* and *Gauss-Seidel* preconditioners can be made by relaxing them with a factor $\omega > 0$. In such a case, a *Successive Overrelaxation* (SOR) preconditioning is obtained, i.e.,

$$\mathbf{P}_{SOR} = \frac{1}{\omega}(\mathbf{D} - \mathbf{L}), \tag{5.8}$$

Notice that $\mathbf{P}_{SOR} = \mathbf{P}_{GS}$ when $\omega = 1$.

SOR can not be used, because in the preconditioned CG $\mathbf{M}_\omega^{-1}\mathbf{N}_\omega$ is *not symmetric*, where $\mathbf{M}_\omega = \mathbf{D} + \omega\mathbf{L}$ and $\mathbf{N}_\omega = (1-\omega)\mathbf{D} - \omega\mathbf{U}$ with $\mathbf{U}$ the (strict)upper triangular matrix of $\mathbf{A}$.

Generalizing SOR in order to obtain a symmetric matrix one obtains *Symmetric SOR* (SSOR). In SSOR one SOR step is followed by a backward SOR step. In this backward step the unknowns are updated in reversed order. This leads to

$$\mathbf{P}_{SSOR} = (\mathbf{D} - \omega\mathbf{L})\mathbf{D}^{-1}(\mathbf{D} - \omega\mathbf{U}), \tag{5.9}$$

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

$$\mathbf{P}_{SGS} = (\mathbf{D} - \mathbf{L})\mathbf{D}^{-1}(\mathbf{D} - \mathbf{U}), \tag{5.10}$$

which can be expressed as

$$\mathbf{P}_{SGS} = \mathbf{L}_{SGS}\mathbf{U}_{SGS}, \tag{5.11}$$

where $\mathbf{U}_{SGS} = \mathbf{D} - \mathbf{U}$ and $\mathbf{L}_{SGS} = (\mathbf{D} - \mathbf{L})\mathbf{D}^{-1} = \mathbf{I} - \mathbf{L}\mathbf{D}^{-1}$ are respectively upper and unit lower triangular. In this case, we could solve the preconditioned system

$$\mathbf{U}_{SGS}^{-1}\mathbf{L}_{SGS}^{-1}\mathbf{A}\mathbf{x} = \mathbf{U}_{SGS}^{-1}\mathbf{L}_{SGS}^{-1}\mathbf{b}. \tag{5.12}$$

However, most practical wave problems result in an *indefinite* linear system, for which SSOR or SGS are *not* guaranteed to converge.

5.4 ILU preconditioners

In the previous section, the general pattern of preconditioner $\mathbf{P}$ 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 approximation factorization is still useful for preconditioning. Since the degree of approximation is rather arbitrary, constraints should be added to the factorization. This leads to the general *Incomplete LU* (ILU) factorization.

An ILU factorization process computes a sparse lower triangular matrix $\mathbf{L}$ and a sparse triangular matrix $\mathbf{U}$ so the residual matrix $\mathbf{R} = \mathbf{L}\mathbf{U} - \mathbf{A}$ satisfies certain constraints, such as having zero entries in some locations.

In a practical implementation, the ILU factorization depends on the implementation of the Gaussian elimination. In section 10.1 of Saad [19] the IKJ variant of Gaussian elimination is used and therefore the general ILU factorization is given as follows:

Algorithm: General ILU factorization (IKJ variant)

1. **For** $i = 2, 3, \ldots, N$ **Do** :
2. **For** $k = 1, 2, \ldots, i-1$ **and** $(i,k) \notin P$ **Do** :
3. $a_{ik} := \frac{a_{ik}}{a_{kk}}$
4. **For** $j = k+1, k+2, \ldots, N$ **and** $(i,j) \notin P$ **Do** :
5. $a_{ij} := a_{ij} - a_{ik}a_{kj}$
6. **EndDo**
7. **EndDo**
8. **EndDo**

Here P is the *zero pattern set* such that

$$P \subset \{(i,j) : i \neq j; 1 \leq i,j \leq N\}, \tag{5.13}$$

and a_{ij} with $i, j = 1, 2, \ldots, N$ are the elements of matrix $\mathbf{A}$.

If we choose $P = \emptyset$, then the algorithm leads to the IKJ variant of the Gaussian elimination where elements of $\mathbf{A}$ are overwritten with elements of the $\mathbf{L}$ and $\mathbf{U}$ matrices of the factorization. Observe that diagonal entries need not be stored, since $\mathbf{L}$ is *unit* lower triangular.

5.4.1 Zero fill-in ILU (ILU(0))

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 any upper triangular matrix $\mathbf{U}$, then the product $\mathbf{LU}$ does *not* have the same structure as $\mathbf{A}$. In ILU(0), one takes any pair of $\mathbf{L}_0$ and $\mathbf{U}_0$ having the zero pattern set to be precisely the same as the zero pattern of the lower and upper triangular of $\mathbf{A}$. This defines the ILU(0) factorization in general terms: choose any pair of $\mathbf{L}_0$ and $\mathbf{U}_0$ so that $(\mathbf{A} - \mathbf{L}_0\mathbf{U}_0)_{ij} = 0$ if $\mathbf{A}_{ij} \neq 0$ where $\mathbf{L}$ and $\mathbf{U}$ must have the same structure as the lower respectively the upper triangular of $\mathbf{A}$. These constraints do not define ILU(0) factors uniquely since there are, in general, infinitely many pairs of $\mathbf{L}_0$ and $\mathbf{U}_0$ which satisfy these requirements. However, the standard ILU(0) is defined constructively using the previous algorithm with the pattern $P_{ILU(0)}$ equal to the zero pattern of $\mathbf{A}$, i.e.,

$$P_{ILU(0)} = Z(\mathbf{A}), \tag{5.14}$$

where $Z(\mathbf{A})$ is the set of pairs $(i,j), 1 \leq i,j \leq N$ such that $a_{i,j} = 0$. In other words, the previous algorithm leads to an unique decomposition of $\mathbf{L}_0$ and $\mathbf{U}_0$, adding the requirement $(\mathbf{L}_0)_{ii} = 1$ for $i = 1, 2, \ldots, N$, i.e., $\mathbf{L}_0$ has been chosen as an *unit* lower triangular matrix.

The algorithm of ILU(0) can now be represented in the following way.

Algorithm: ILU(0)

1. **For** $i = 2, 3, \ldots, N$ **Do** :
2. **For** $k = 1, 2, \ldots, i-1$ **and** $(i,k) \notin Z(\mathbf{A})$ **Do** :
3. $a_{ik} := \frac{a_{ik}}{a_{kk}}$
4. **For** $j = k+1, k+2, \ldots, N$ **and** $(i,j) \notin Z(\mathbf{A})$ **Do** :
5. $a_{ij} := a_{ij} - a_{ik}a_{kj}$
6. **EndDo**
7. **EndDo**
8. **EndDo**

The accuracy of the ILU(0) incomplete factorization may be insufficient to yield an adequate rate of convergence, see e.g. section 10.3.2 of Saad [19]. Numerical experiments show that ILU(0) breaks down in HBVP for large k (highly indefinite problems), see section 6.4.1 of Erlangga [6].

5.4.2 ILU(p)

In order to improve this convergence rate as well the efficiency, 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 P to be the zero pattern of the product $\mathbf{L}_0\mathbf{U}_0$ of the factors $\mathbf{L}_0$ and $\mathbf{U}_0$ obtained from ILU(0). In other words, we consider a matrix with additional off-diagonal components which are actually zero in the original matrix $\mathbf{A}$. The factors $\mathbf{L}_1$ and $\mathbf{U}_1$ of ILU(1) are obtained by performing ILU(0) to this matrix.

In similar way we can define ILU(p) for $p = 2, 3, \ldots$. Observe that increasing p leads to more fill-in in $\mathbf{L}_p$ and $\mathbf{U}_p$ which could be inefficient. Therefore, choosing an 'ideal' value for p, one has to look for a good mix of restricted amount of fill-ins and fast convergence rate of the method.

5.4.3 Other variants of ILU

There are several variants of the ILU(p) factorization which differ slightly from the original ILU(p). We have for instance the *Modified ILU* (MILU) factorization in which the dropping process for the extra diagonals is compensated at the k-loop of the algorithm of general ILU factorization. In an MILU factorization, after the k-loop, the diagonal element a_{ii} is modified by subtracting it with the sum of the row i.

Also, *ILUT* or ILU(tol) can be used to have a more accurate factorization that improves the convergence rate. In ILUT one drops elements which are smaller than a specified value. Details about MILU and ILUT can be found in sections 10.3 and 10.4 of Saad [19].

However, preconditioners from this class are not effective for general indefinite problems, see for instance Gander & Nataf [10]. For *high* wavenumbers k, ILU(0) converges slowly, while ILU(tol) encounters storage problems and also slow convergence. Recently, some shifted ILU preconditioners have been investigated, see for instance Made [16].

5.5 Incomplete Choleski factorization

Made [16] has introduced a new incomplete factorization based preconditioning technique, which consists in adding small perturbations to the *diagonal* entries of the real part of the matrix. In doing so, the real part is made positive definite, or less indefinite.

5.5.1 Idea of incomplete Cholesky factorization

As model problem, we consider again the large-scale linear system $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{A}$ is *complex-symmetric* and the linear system is deduced from the Helmholtz problem. To solve this system with a *direct* method, one may factorize $\mathbf{A}$ as $\mathbf{A} = \tilde{\mathbf{L}}\tilde{\mathbf{L}}^T$, $\tilde{\mathbf{L}}^T$ being lower triangular, and solves successively two triangular systems. This is known as the *Cholesky factorization*. Even if $\mathbf{A}$ is sparse, $\tilde{\mathbf{L}}^T$ is in general *less sparse* due to fill-in. This makes direct

methods both memory and time consuming for N fairly large, as in real-life scientific or industrial problems.

A common remedy consists in ignoring certain fill-in entries, which yields an *incomplete* factorization preconditioning matrix $B = \mathbf{L}\mathbf{L}^T$. There are two basic strategies for accepting or discarding fill-in:

- By *level of fill-in* (or by position). The level 'lev($l_{i,j}$)' of the coefficient $l_{k,i}$ of matrix $\mathbf{L}$ is defined by Saad, see section 10.3.3 of [19],

  1. *initialization*:

$$\text{lev}(l_{i,j}) := \begin{cases} 0 & \text{if } l_{i,j} \neq 0 \text{ or } k = i, \\ \infty & \text{otherwise,} \end{cases} \tag{5.15}$$

  2. *factorization*:

$$\text{lev}(l_{i,j}) = \min\{\text{lev}(l_{i,j}), \text{lev}(l_{i,k}) + \text{lev}(l_{k,j}) + 1\}, \tag{5.16}$$

     which is updated each time in line 5 of the algorithm of Gauss elimination (IKJ-variant).

  The set $\mathcal{D}$ of fill-in entries to be discarded is taken as

$$\mathcal{D} = \{(i,j) \mid \text{lev}(l_{i,j} > \xi)\}, \tag{5.17}$$

  where the integer ξ denotes a user specified maximal fill-in level.

- By (numerical) *value*. Fill-in is ignored if it is 'too small' with respect to some prescribed tolerance.

Dual approaches that combine ingredients from both structural and numerical strategies are also used. The choice of both ξ and the drop tolerance depends, among other things, on the problem at hand and the available workspace. Several variants of the basic incomplete factorization have been designed, ranging from *modified* methods in which the discarded fill-in entries are added to the diagonal, to more sophisticated multilevel versions that use multigrid like (re)numbering strategies, see for instance Axelsson [2].

5.5.2 Variants of preconditioners

Made has used six variants of preconditioners based on the incomplete Cholesky factorization, i.e.,

1. **IC**: the standard incomplete Cholesky applied to **A**;
2. **MIC**: the standard modified incomplete Cholesky applied to **A** (see for instance section 5.1 of Vuik [28] for more details about this method);

3. $\mathbf{IC}_0$: IC applied to $\mathbf{A}_0 \equiv \mathrm{Re}(\mathbf{A}) + \mathbf{Q}$ $(\gamma = 1)$;
4. $\widetilde{\mathbf{IC}}_0$: IC applied to $\widetilde{\mathbf{A}} \equiv \mathbf{A}_0 + i\ \mathrm{Im}(\mathbf{A}) = \mathbf{A} + \mathbf{Q}$;
5. $\mathbf{MIC}_0$: MIC applied to $\mathbf{A}_0$ $(\gamma = 1)$;
6. $\widetilde{\mathbf{MIC}}$: MIC applied to $\widetilde{\mathbf{A}}$;

where $\mathbf{Q}$ stands for the diagonal matrix whose diagonal entries q_{ii} are defined by

$$q_{ii} = -\gamma \min\{0, \mathrm{Re}((\mathbf{Ae})_i)\}, \tag{5.18}$$

with $\mathbf{e}$ the all-one vector and γ a given real parameter. In fact, $\mathbf{Ae}$ is the vector with the sum of the rows of $\mathbf{A}$. It can be proved, using spectral analysis (see section 4.2 of [16]), that $\mathbf{A}_0$ is a diagonal perturbation of the *Hermitian* part of the system matrix, while with $\tilde{\mathbf{A}}$, the same diagonal perturbation is added to the *whole* matrix.

In numerical experiments using restarted GMRES (section 5.2 of [16]), we can see that varying values for, respectively, wavenumber k, stepsize h, number of fill-levels, parameter γ etcetera lead to different results. It appears that there is no best preconditioner in all situations, although often $\widetilde{\mathbf{MIC}}$ is the better one.

5.6 Shifted Laplace preconditioners

Another approach is found in looking for an approximate inverse of the discrete indefinite operator $\mathbf{A}$, but merely looking for a form of $\mathbf{P}$, for which $\mathbf{P}^{-1}\mathbf{A}$ has satisfactory properties for Krylov subspace methods. A first effort to construct a preconditioner in such a way was done in Bayliss, Goldstein & Turkel [3]. An easy-to-construct $\mathbf{P} = \Delta$ preconditioner is incorporated for CGNR, where one SSOR iteration is used whenever operations involving $\mathbf{P}^{-1}$ are required.

Instead of the Laplace operator as preconditioner, Laird [15] investigates possible improvements if an *extra* term k^2 is added to the Laplace operator Δ, i.e., the Helmholtz equation with reversed sign is proposed as preconditioner $\mathbf{P}$. This preconditioner is applied in CGNR, where one multigrid iteration is employed whenever $\mathbf{P}^{-1}$ must be computed.

One can generalize both preconditioners, mentioned above, to a *complex* shifted Laplace preconditioner where an extra term $(\alpha + \beta i)k^2$ is added to the Laplace operator Δ with $\alpha, \beta \in \mathbb{R}$ and $\alpha \geq 0$. This is proposed by Erlangga, Oosterlee & Vuik [7, 8].

In the next sections, we motivate the real and complex shifted Laplace preconditioner. This motivations are identical to section 5.1 respectively section 5.2 of Erlangga, Oosterlee & Vuik [7].

5.6.1 Real shifted Laplace preconditioner

Consider the continuous 1-dimensional Helmholtz equation, subject to discretization. For simplicity, suppose that both boundary conditions are either Dirichlet or Neumann conditions.

We first consider the eigenvalues for the 1-dimensional (negative) operator without any preconditioning, i.e.,

$$\mathcal{L} = -\Delta - k, \quad k > 0. \tag{5.19}$$

Eigenvalues of this standard problem were found in section 3.3.2, i.e.,

$$\lambda_n^s = k_n^2 - k^2, \quad k_n = n\pi, \quad n = 1, 2, \ldots. \tag{5.20}$$

In expression (5.20), k_n is the *natural frequency* of the system. If one considers the modulus of the eigenvalues, which in this case is simply their absolute value, it is easily seen that $|\lambda|$ becomes *unbounded* if either n or k is large. If the condition number $K = |\lambda_{\max}|/|\lambda_{\min}|$ is used to evaluate the quality of the eigenvalue clustering (which is possible since the considered matrix is assumed to be symmetric), one concludes that for any sufficient small $\lambda_{\min}$ this condition number is *extremely large*.

Now, assume an operator of the form

$$\mathcal{L}_p = \frac{\mathrm{d}^2}{\mathrm{d}x^2} - \alpha k^2 \quad , \alpha \geq 0, k > 0, \tag{5.21}$$

is used as a preconditioner, constructed with the same discretization stencil and boundary conditions. The following generalized eigenvalue problem is obtained, i.e.,

$$\left(\frac{\mathrm{d}^2}{\mathrm{d}x^2} + k^2\right)\phi(x) = \lambda\left(\frac{\mathrm{d}^2}{\mathrm{d}x^2} - \alpha k^2\right)\phi(x), \quad x \in (0, l), \quad l > 0. \tag{5.22}$$

To find the eigenvalues we divide the problem in two parts: $\alpha > 0$ and $\alpha = 0$.

Case $\alpha > 0$

For expression (5.22) we find easily the eigenvalues

$$\lambda_n = \frac{k_n^2 - k^2}{k_n^2 + \alpha k^2} = \frac{1 - (k/k_n)^2}{1 + \alpha(k/k_n)^2}, \quad k_n = n\pi, \quad n = 1, 2, \ldots. \tag{5.23}$$

For $n \to \infty$ we get $\lambda_n \to 1$, i.e., the eigenvalues are *bounded* above by one. Moreover, for $n \to 0$ we obtain $\lambda_n \to -1/\alpha$, which is also *bounded*. So, we can write

$$|\lambda_{\max}| = \max\left(\frac{1}{\alpha}, 1\right), \quad \alpha > 0. \tag{5.24}$$

To estimate the minimum eigenvalue, one can use a simple but rough analysis as follows. Below it is assumed that $|\lambda_{\min}|$ is very close but not

equal to zero, since this is of our interest. The assumption indicates the condition $k_j \approx k$ for a specific $j = 1, 2, \ldots$ as obtained from (5.23). To be more precise, let $k_j = k + \epsilon$ where $\epsilon \in \mathbb{R}$ is any small number. If this relation is substituted into (5.23) and if higher order terms are neglected, then we find

$$|\lambda_{\min}| = \frac{2|\epsilon|}{k(1+\alpha)}. \tag{5.25}$$

where $\epsilon k \ll k^2$ has been assumed. From (5.25), one obtains $\lambda_{\min} \to 0$ if $\alpha \to \infty$.

Now, the condition number K of the preconditioned Helmholtz operator reads

$$K_{\alpha>0} = \frac{|\lambda_{\max}|}{|\lambda_{\min}|} = \begin{cases} \dfrac{(1+\alpha)k}{2|\epsilon|} & \text{if } \alpha \geq 1, \\ \\ \dfrac{(1+\alpha)k}{2\alpha|\epsilon|} & \text{if } 0 < \alpha \leq 1. \end{cases} \tag{5.26}$$

If $\alpha \geq 1$, then it is simple to see that K is a monotonically *increasing* function with respect to α. In this case, the best choice for choosing α is $\alpha = 1$ which gives *minimal* $K_{\alpha>0}$. If $0 < \alpha \leq 1$, then K is a monotonically *decreasing* function, which is also *minimal* if $\alpha = 1$. So, we find

$$\lim_{\alpha \downarrow 1} K_{\alpha>0} = \lim_{\alpha \uparrow 1} K_{\alpha>0} = \frac{k}{|\epsilon|}, \tag{5.27}$$

to be the minimum value of $K_{\alpha>0}$ for $\alpha > 0$. In other words, when we have to choose a suitable positive real-valued α, the best choice is $\alpha = 1$, which minimizes the condition number K. Decreasing the condition number means a narrowing of the eigenvalue clustering, which is favorable in iterative methods.

The choice $\alpha = 1$ leads exactly to the preconditioner proposed by Laird [15].

Case $\alpha = 0$

Substituting $\alpha = 0$ in expression (5.22) leads to the eigenvalues

$$\lambda_n = \frac{k_n^2 - k^2}{k_n^2} = 1 - \left(\frac{k}{k_n}\right)^2, \quad k_n = n\pi, \quad n = 1, 2, \ldots. \tag{5.28}$$

Then we find

$$|\lambda_{\max}| = \max_n |\lambda_n| = \max_n \left| 1 - \left(\frac{k}{k_n}\right)^2 \right| \quad k_n = n\pi, \quad n = 1, 2, \ldots. \tag{5.29}$$

We can distinguish three cases.

- $|\lambda_{\max}| = 1$ when $1 - (\frac{k}{k_n})^2 > 0 \ \forall n$ implying $k < k_n \ \forall n$. This is the case when $k < \pi$, since $\min_n k_n = \pi$.

- $|\lambda_{\max}| = (\frac{k}{\pi})^2 - 1$ if $1 - (\frac{k}{k_n})^2 < 0 \ \forall n$ implying $k > k_n \ \forall n$. In this case we have $k = \infty$ which never happens, since we assumed $0 < k < \infty$.
- $|\lambda_{\max}| = \max_n \left(1, (\frac{k}{\pi})^2 - 1\right)$ when we have the situation

$$\begin{aligned} 1 - \left(\frac{k}{k_{j_1}}\right)^2 &> 0 \ \forall j_1, \quad j_1 \in \Omega_1 \subset [1, 2, \ldots], \\ 1 - \left(\frac{k}{k_{j_2}}\right)^2 &< 0 \ \forall j_2, \quad j_2 \in \Omega_2 \subset [1, 2, \ldots], \end{aligned} \tag{5.30}$$

in the remaining domain $k \in (\pi, \infty)$ such that $\Omega_1 \cup \Omega_2 = [1, 2, \ldots]$ and $\Omega_1 \cap \Omega_2 = \emptyset$. Then, we deduce

$$\left.\begin{array}{l} k < k_{j_1} \ \forall j_1 \\ k > k_{j_2} \ \forall j_2 \end{array}\right\} \quad \rightarrow \quad k_{j_2} < k_{j_1} \ \forall j_1 \ \forall j_2. \tag{5.31}$$

Hence, there is a $q \in [1, 2, \ldots]$ such that

$$[\underbrace{1, 2, \ldots, q-1}_{=\Omega_2 \ni j_2}, q, \underbrace{q+1, q+2, \ldots}_{=\Omega_1 \ni j_1}]. \tag{5.32}$$

Now, we have

$$|\lambda_{\max}| = \max_{j_1, j_2} \left\{ 1 - \left(\frac{k}{k_{j_1}}\right)^2, \left(\frac{k}{k_{j_2}}\right)^2 - 1 \right\}. \tag{5.33}$$

We deal with both cases:

- If $|\lambda_{\max}| = 1 - \left(\frac{k}{k_{j_1}}\right)^2$, then k_{j_1} has to be *maximized* which gives $k_{j_1} = \infty$ using (5.32). Therefore: $|\lambda_{\max}| = 1$.
- If $|\lambda_{\max}| = \left(\frac{k}{k_{j_2}}\right)^2 - 1$, then k_{j_2} has to be *minimized*, so that $k_{j_2} = \pi$ using again (5.32). Therefore: $|\lambda_{\max}| = \left(\frac{k}{\pi}\right)^2 - 1$.

Indeed, we obtain $|\lambda_{\max}| = \max\left(1, (\frac{k}{\pi})^2 - 1\right)$ in this case. Now, to find the conditions for k, we has to distinguish here also two cases:

- we get $|\lambda_{\max}| = (\frac{k}{\pi})^2 - 1$ if $1 < (\frac{k}{\pi})^2 - 1$, so if $k > \sqrt{2}\pi$,
- and otherwise $|\lambda_{\max}| = 1$ if $1 > (\frac{k}{\pi})^2 - 1$, so if $\pi < k < \sqrt{2}\pi$. Using case (1) above we get finally $k < \sqrt{2}\pi$.

We can summarize the above results as follows

$$|\lambda_{\max}| = \left\{ \begin{array}{ll} 1 & , \text{if } k < \sqrt{2}\pi, \\ (\frac{k}{\pi})^2 - 1 & , \text{if } k > \sqrt{2}\pi. \end{array} \right. \tag{5.34}$$

Observe that $|\lambda_{\max}|$ depends on the choice of k.

Again, we assume that $|\lambda_{\min}|$ is very close to zero, so let $k_j = k + \epsilon$ with some $j = 1, 2, \ldots$ and $\epsilon \in \mathbb{R}$ a small number such that $|\epsilon| \ll k$. As a consequence we obtain $k + \epsilon > 0$. Then, expression (5.28) turns out to be

$$|\lambda_{\min}| = \left|1 - \left(\frac{k}{k+\epsilon}\right)^2\right| = \begin{cases} 1 - \left(\frac{k}{k+\epsilon}\right)^2 & \text{if } \epsilon > 0; \\ \left(\frac{k}{k+\epsilon}\right)^2 - 1 & \text{if } \epsilon < 0. \end{cases} \tag{5.35}$$

We deal with both cases of the right hand side of (5.35).

- First, we assume $\epsilon > 0$. Then the condition number of the preconditioned Helmholtz operator turns out to be

$$K_{\alpha=0} = \frac{|\lambda_{\max}|}{|\lambda_{\min}|} = \begin{cases} \dfrac{1}{1 - \left(\frac{k}{k+\epsilon}\right)^2} & \text{if } k < \sqrt{2}\pi, \\ \dfrac{(\frac{k}{\pi})^2 - 1}{1 - \left(\frac{k}{k+\epsilon}\right)^2} & \text{if } k > \sqrt{2}\pi. \end{cases} \tag{5.36}$$

  Neglecting higher order terms, (5.36) can be written as

$$K_{\alpha=0} = \begin{cases} \dfrac{k + 2\epsilon}{2\epsilon} & \text{if } k < \sqrt{2}\pi, \\ \dfrac{k(k^2 - \pi^2)}{2\epsilon\pi^2} & \text{if } k > \sqrt{2}\pi. \end{cases} \tag{5.37}$$

- Secondly, we assume $\epsilon < 0$. Then the condition number of the preconditioned Helmholtz operator turns out to be

$$K_{\alpha=0} = \frac{|\lambda_{\max}|}{|\lambda_{\min}|} = \begin{cases} \dfrac{1}{\left(\frac{k}{k+\epsilon}\right)^2 - 1} & \text{if } k < \sqrt{2}\pi, \\ \dfrac{(\frac{k}{\pi})^2 - 1}{\left(\frac{k}{k+\epsilon}\right)^2 - 1} & \text{if } k > \sqrt{2}\pi. \end{cases} \tag{5.38}$$

  Neglecting higher order terms, (5.36) can be written as

$$K_{\alpha=0} = \begin{cases} \dfrac{-(k + 2\epsilon)}{2\epsilon} & \text{if } k < \sqrt{2}\pi, \\ \dfrac{k(\pi^2 - k^2)}{2\epsilon\pi^2} & \text{if } k > \sqrt{2}\pi. \end{cases} \tag{5.39}$$

Comparing $K_{\alpha=0}$ and $K_{\alpha=1}$

In this subsection we compare $K_{\alpha=0,1}$ in order to decide which α is the best to choose. Again, one has to distinguish two cases: ϵ to be positive and negative.

- $\epsilon > 0$.

  Case 1
  Consider $K_{\alpha=0} = \frac{k+2\epsilon}{2\epsilon}$ with $k < \sqrt{2}\pi$. We find easily

  $$K_{\alpha=0} = \frac{k+2\epsilon}{2\epsilon} < \frac{k}{\epsilon} = K_{\alpha=1}, \tag{5.40}$$

  neglecting again higher order terms. In other words, if $k < \sqrt{2}\pi$ then $\alpha = 0$ leads to an optimal condition number.

  Case 2
  Consider $K_{\alpha=0} = \frac{k(k^2-\pi^2)}{2\epsilon\pi^2}$ with $k > \sqrt{2}\pi$. The inequality

  $$K_{\alpha=0} = \frac{k(k^2-\pi^2)}{2\epsilon\pi^2} < \frac{k}{\epsilon} = K_{\alpha=1}, \tag{5.41}$$

  only holds if $k < \sqrt{3}\pi$. In other words, if $k \in (\sqrt{2}\pi, \sqrt{3}\pi)$ then $\alpha = 0$ leads to an optimal condition number.

- $\epsilon < 0$.

  Case 1
  Consider $K_{\alpha=0} = \frac{-(k+2\epsilon)}{2\epsilon}$ with $k < \sqrt{2}\pi$. We find easily that

  $$K_{\alpha=0} = \frac{-(k+2\epsilon)}{2\epsilon} < \frac{k}{|\epsilon|} = K_{\alpha=1}, \tag{5.42}$$

  holds only if $k > 2\epsilon$. In other words, if $k \in (0, \sqrt{2}\pi)$ then $\alpha = 0$ leads to an optimal condition number since $k > 0$.

  Case 2
  Consider $K_{\alpha=0} = \frac{k(\pi^2-k^2)}{2\epsilon\pi^2}$ with $k > \sqrt{2}\pi$. The inequality

  $$K_{\alpha=0} = \frac{k(\pi^2-k^2)}{2\epsilon\pi^2} < \frac{k}{|\epsilon|} = K_{\alpha=1}, \tag{5.43}$$

  holds only if $k < \sqrt{3}\pi$. Thus, if $k < \sqrt{3}\pi$ then $\alpha = 0$ is optimal.

Conclusion

We conclude that

$$\alpha_{best} = \begin{cases} 0, & \text{if } k < \sqrt{3}\pi, \\ 1, & \text{if } k > \sqrt{3}\pi. \end{cases} \tag{5.44}$$

Observe that (5.44) is *independent* of the sign of ϵ.

5.6.2 Complex shifted Laplace preconditioner

The nice property of the real shifted Laplace operator, at least in one dimension, is that the eigenvalues have an upper bound. However, this property does not guarantee that the eigenvalues are favourably distributed. There is still the possibility that one or some eigenvalues (which are extremely small) can be very close to zero, which may be disadvantageous for iterative Krylov methods. We can improve the preconditioner by preserving the upper boundedness and at the same time shifting the minimum eigenvalue as far as possible from zero. Therefore we generalize α to be *complex*-valued.

We introduce a complex coefficient of the form $\alpha + i\beta$ and consider a more general complex-valued shifted Laplace operator

$$\mathcal{L}_{cp} = \frac{\mathrm{d}^2}{\mathrm{d}x^2} - (\alpha + i\beta)k^2 \quad , k > 0, \alpha \geq 0, \beta \in \mathbb{R}. \tag{5.45}$$

In the same way as in the previous section we find the eigenvalues

$$\lambda_n^c = \frac{k_n^2 - k^2}{k_n^2 + (\alpha + i\beta)k^2} = \frac{1 - (k/k_n)^2}{1 + (\alpha + i\beta)(k/k_n)^2}, \tag{5.46}$$

where $k_n = n\pi$ and $n = 1, 2, \ldots$. We can write expression (5.46) as

$$|\lambda_n^c|^2 = \frac{(k_n^2 - k^2)^2}{(k_n^2 + \alpha k^2)^2 + \beta^2 k^4}. \tag{5.47}$$

Evaluating $\lambda_{\max}$ and $\lambda_{\min}$ as in (5.24) and (5.25) leads to

$$|\lambda_{\max}^c|^2 = \max\left(\frac{1}{\alpha^2 + \beta^2}, 1\right), \quad |\lambda_{\min}^c|^2 = \frac{4}{(1+\alpha)^2 + \beta^2}\left(\frac{\epsilon}{k}\right)^2. \tag{5.48}$$

These results give the following condition numbers

$$|K_{\alpha,\beta}|^2 = \frac{|\lambda_{\max}|^2}{|\lambda_{\min}|^2} = \begin{cases} \dfrac{k^2}{4\epsilon^2}\left((1+\alpha)^2 + \beta^2\right) & \text{if } \alpha^2 + \beta^2 \geq 1, \\ \dfrac{k^2}{4\epsilon^2}\left(1 + \dfrac{1+2\alpha}{\alpha^2+\beta^2}\right) & \text{if } 0 < \alpha^2 + \beta^2 < 1. \end{cases} \tag{5.49}$$

Both expressions in (5.49) are monotonically increasing functions to both α and β, so it is not difficult to see that choosing $\alpha^2 + \beta^2 = 1$ leads to minimal $|K_{\alpha,\beta}|^2$ in both situations [1]. Therefore, $K_{\alpha,\beta}$ is minimal when we take $\alpha = 0$ implying $\beta = 1$, using (5.49). This combination gives the lowest possible condition number for the complex shifted-Laplace preconditioner for the 1-dimensional Helmholtz problem.

[1] In the second expression one can verify that there is no other circle giving K^2 lower that that on the circle with radius one. This can be seen, e.g., by introducing condition $\alpha^2 + \beta^2 = 1 + \epsilon_1, \epsilon_1 \geq 0$ and proving $\epsilon_1 = 0$ is optimal.

5.6.3 Comparing real and complex α

We analyze the spectral properties of the discrete formulation of the one-dimensional Helmholtz equation with operator defined in (5.19). Suppose that this equation is discretized. Then we obtain the linear system $\mathbf{Ap} = \mathbf{f}$. Now we can write

$$(\mathbf{B} - k^2\mathbf{I})\mathbf{p} = \mathbf{f}, \tag{5.50}$$

where $\mathbf{B}$ is the *negative* Laplace component and $k^2\mathbf{I}$ the additional diagonal term so that $\mathbf{A} = \mathbf{B} - k^2\mathbf{I}$.

In this analysis we use only Dirichlet or Neumann conditions at the boundaries in order to keep the matrix $\mathbf{A}$ real-valued. We precondition (5.50) using $\mathbf{M}_{\alpha,\beta} = \mathbf{B} + (\alpha + i\beta)k^2\mathbf{I}$, constructed with the same boundary conditions as for $\mathbf{A}$. This gives

$$\left(\mathbf{B} + (\alpha + i\beta)k^2\mathbf{I}\right)^{-1}(\mathbf{B} - k^2\mathbf{I})\mathbf{p} = \left(\mathbf{B} + (\alpha + i\beta)k^2\mathbf{I}\right)^{-1}\mathbf{f}, \tag{5.51}$$

The generalized eigenvalue problem of (5.51) is accordingly

$$(\mathbf{B} - k^2\mathbf{I})\phi_v = \lambda_v\left(\mathbf{B} + (\alpha + i\beta)k^2\mathbf{I}\right)\phi_v, \tag{5.52}$$

where ϕ_v is the corresponding eigenvector of λ_v for $v = 1, 2, ..., N$ with N the dimension of $\mathbf{A}$.

Both systems (5.51) and (5.52) are indefinite if k^2 is larger than the smallest eigenvalue of B, which can be concluded by using (5.46). In such a case, the convergence of the method is difficult to estimate. Therefore, the subsequent analysis will be based on the normal equations formulation of the preconditioned matrix system as in Laird [15], because then the systems are positive definite and leads to a simpler convergence estimate.

Since B is positive definite, we can denote the eigenvalues of $\mathbf{B}$ as $0 < \mu_1 \leq \mu_2 \leq \ldots \leq \mu_N$. Observe first that

$$\lambda(\mathbf{A}^H\mathbf{A}) = (\mu_j - k^2)^2, \quad j = 1, 2, \ldots, N. \tag{5.53}$$

Denote further $Q_{\alpha+\beta i} = (\mathbf{M}_{\alpha,\beta}^{-1}\mathbf{A})^H(\mathbf{M}_{\alpha,\beta}^{-1}\mathbf{A})$. Then we find easily the following expressions for the eigenvalues of the preconditioners:

$$\begin{aligned}
\lambda_j(\mathbf{Q}_0) &= \left(\tfrac{\mu_j - k^2}{\mu_j}\right)^2 &&= \left(1 - \tfrac{k^2}{\mu_j}\right)^2, && \text{Bayliss;}\\
\lambda_j(\mathbf{Q}_1) &= \left(\tfrac{\mu_j - k^2}{\mu_j + k^2}\right)^2 &&= \left(1 - \tfrac{2k^2}{\mu_j + k^2}\right)^2, && \text{Laird;}\\
\lambda_j(\mathbf{Q}_i) &= \left(\tfrac{\mu_j - k^2}{\mu_j + ik^2}\right)\overline{\left(\tfrac{\mu_j - k^2}{\mu_j + ik^2}\right)} &&= \left(1 - \tfrac{2\mu_j k^2}{\mu_j^2 + k^4}\right)^2, && \text{Erlangga.}
\end{aligned}$$

for $j = 1, 2, \ldots, N$.

We distinguish two cases: $k < \mu_1$ and $\mu_1 \leq k^2 \leq \mu_N$. There is no need to examine the third case $k^2 > \mu_N$, because this case has no physical and numerical meaning and therefore it happens rarely in applications.

Case 1: $k < \mu_1$

In this case we have a *positive definite* matrix $\mathbf{A}$. After some analysis (see **Appendix B.1**), the following inequalities are derived:

$$\lambda_{\min}(\mathbf{Q}_0) > \lambda_{\min}(\mathbf{Q}_1) \quad \text{and} \quad \lambda_{\min}(\mathbf{Q}_0) > \lambda_{\min}(\mathbf{Q}_i). \tag{5.54}$$

Moreover, the following limit can easily be obtained:

$$\lim_{\mu_N \to \infty} \lambda_{\max}(\mathbf{Q}_\gamma) = 1, \tag{5.55}$$

for all $\gamma = 0, 1, i$.

The convergence of CGNR is well described by the condition number of $\mathbf{Q}_\gamma$, i.e., $K(\mathbf{Q}_\gamma) = \frac{|\lambda_{\max}|}{|\lambda_{\min}|} = \frac{\lambda_{\max}}{\lambda_{\min}}$. Using relations (5.54) and (5.55), we conclude that the Bayliss preconditioner $\mathbf{M}_{0,0}$ converges faster than the other choices. This is the same result as seen in section 5.6.1.

Case 2: $\mu_1 \leq k^2 \leq \mu_N$

In this case we have an *indefinite* matrix $\mathbf{A}$.

Firstly, we compare the Laird and Erlangga preconditioner. One find

$$\lim_{k \to \infty} \lambda_{\max}(\mathbf{Q}_1) = \lim_{k \to \infty} \lambda_{\max}(\mathbf{Q}_0) = 1, \tag{5.56}$$

(see section 5.3 of [7]), so the largest eigenvalues of both preconditioners are bounded by 1. In order to compare these preconditioners we have to consider the smallest eigenvalue in more detail. Assuming that $\lambda_{\min} \approx 0$ implies that $\mu_j \approx k^2$ (this can be seen to solve $\lambda_j(\mathbf{Q}_1) = 0$ or $\lambda_j(\mathbf{Q}_i) = 0$), i.e., there is an $j = 1, 2, \ldots, N$ such that $\mu_j = k^2 + \epsilon$ with a small $\epsilon \in \mathbb{R}$. After substituting this relation into the expressions of the smallest eigenvalues and neglecting higher order terms (see section 5.3 of [7]), we find

$$\lambda_{\min}(\mathbf{Q}_1) = \frac{\epsilon^2}{4k^4} < \frac{\epsilon^2}{2k^4} = \lambda_{\min}(\mathbf{Q}_i) \tag{5.57}$$

This implies immediately that

$$K(\mathbf{Q}_i) = \frac{2k^4}{\epsilon^2} < \frac{4k^4}{\epsilon^2} = K(\mathbf{Q}_1) \tag{5.58}$$

The conclusion is that we expect that the Erlangga preconditioner $\mathbf{M}_{0,1}$ is *better* than the Laird preconditioner $\mathbf{M}_{1,0}$.

Secondly, we compare Bayliss and Erlangga preconditioner. We find

$$\lim_{k \to \infty} \lambda_{\max}(\mathbf{Q}_0) = \lim_{k \to \infty} \left(1 - \frac{k^2}{\mu_j}\right)^2 = \infty \tag{5.59}$$

Therefore, $\lambda_{\max}(\mathbf{Q}_0)$ can become very large, while we have seen that $\lambda_{\max}(\mathbf{Q}_i)$ is bounded by 1. On the other hand, we can assume $\lambda_{\min}(\mathbf{Q}_0) \approx \lambda_{\min}(\mathbf{Q}_i) \approx 0$ for the most values of k. Therefore, we obtain $K(\mathbf{Q}_i) < K(\mathbf{Q}_0)$ which means that Erlangga preconditioner $\mathbf{M}_{0,1}$ is the best one to use in this case.

Indeed, it has been shown in numerical experiments (see e.g. section 5 of [8]) that the *complex shifted Laplace operator* leads to the most effective preconditioning matrix within this class of preconditioners.

5.7 Separation - of - Variables (SOV)

In Plessix & Mulder [17] an iterative solver has been considered with a preconditioner based on the separation-of-variables (SOV) technique and applied to examples derived from the discretized Helmholtz boundary value problem $\mathbf{Ap} = \mathbf{f}$. This technique leads to a direct solver when the wavenumber k is constant. When the wavenumber is separable, meaning that it can be expressed as a sum of two terms, one depending on one coordinate x and the second depending on the other coordinates, the SOV would have been exact if it were *not* for the absorbing boundary conditions. For a general wavenumber distribution, the technique can be used as a preconditioner. The idea is to replace the wavenumber k by a separable wavenumber and use this approximation to build the preconditioner. The Bi-CGSTAB iterative method is applied to solve the preconditioned system.

5.7.1 Separation-of-variables method

The matrix $\mathbf{A}$ (with dimensions $MNL \times MNL$) is the sum of a matrix corresponding to the (negative) Laplacian operator with some boundary conditions and a diagonal matrix $\mathbf{K}$ containing the square values of the wavenumber. Except for boundary conditions, the separation-of-variables technique can be applied to solve the Laplacian operator. *Approximations* $\overline{\gamma}$ for the absorbing boundary conditions are made (see section 2 and 3 of [17]), such that $\mathbf{A}$ can be rewritten as

$$\mathbf{A} = \mathbf{A}_x \otimes \mathbf{I}_{yz} + \mathbf{I}_x \otimes \mathbf{A}_{yz} - \mathbf{K}, \tag{5.60}$$

where $\otimes$ is the Kronecker product and with $\mathbf{I}_{yz}$ to be the $NL \times NL$ identity matrix, $\mathbf{I}_x$ to be the $M \times M$ identity matrix,

$$\begin{cases} \mathbf{A}_x(m,m) & = & \dfrac{2}{\Delta x^2} + \begin{cases} \overline{\gamma}_x^{\min} & \text{if } m = 1, \\ \overline{\gamma}_x^{\max} & \text{if } m = M, \\ 0 & \text{if otherwise,} \end{cases} \\ \mathbf{A}_x(m+1,m) & = & \mathbf{A}_x(m,m+1) = \dfrac{-1}{\Delta x^2}, \end{cases} \tag{5.61}$$

for $m = 1, \ldots, M$ and finally

$$\begin{cases} \mathbf{A}_{yz}(q,q) & = & \dfrac{2}{\Delta y^2} + \dfrac{2}{\Delta z^2} + \begin{cases} \overline{\gamma}_y^{\min} & \text{if } n = 1, \\ \overline{\gamma}_y^{\max} & \text{if } n = N, \\ \overline{\gamma}_z^{\min} & \text{if } l = 1, \\ \overline{\gamma}_z^{\max} & \text{if } l = L, \\ 0 & \text{if otherwise,} \end{cases} \\ \mathbf{A}_{yz}(q+1,q) & = & \mathbf{A}_x(q,q+1) = \dfrac{-1}{\Delta y^2}, \\ \mathbf{A}_{yz}(q+N,q) & = & \mathbf{A}_x(q,q+N) = \dfrac{-1}{\Delta z^2}, \end{cases} \tag{5.62}$$

for $q = (l-1)N + n$ where $l = 1, \ldots, L$ and $n = 1, \ldots, N$.

Generally $\mathbf{K}$ prevents us from using SOV. However, we can decompose the square of the wavenumber into

$$k^2(x,y,z) = k_x^2(x) + k_{yz}^2(y,z) + \tilde{k}^2(x,y,z), \tag{5.63}$$

such that

$$\begin{cases} \int \tilde{k}^2(x,y,z)\ \mathrm{d}x & = & 0 & \forall y, z, \\ \int \tilde{k}^2(x,y,z)\ \mathrm{d}y\ \mathrm{d}z & = & 0 & \forall x, \\ \int k_{x,y}^2(y,z)\ \mathrm{d}y\ \mathrm{d}z & = & 0. \end{cases} \tag{5.64}$$

This decomposition is unique, see Appendix A of [17]. Hence, the matrix $\mathbf{K}$ becomes

$$\mathbf{K} = \mathbf{K}_x \otimes \mathbf{I}_{yz} + \mathbf{I}_x \otimes \mathbf{K}_{yz} + \widetilde{\mathbf{K}}, \tag{5.65}$$

leading to

$$\mathbf{A} = (\mathbf{A}_x - \mathbf{K}_x) \otimes \mathbf{I}_{yz} + \mathbf{I}_x \otimes (\mathbf{A}_{yz} - \mathbf{K}_{yz}) - \widetilde{\mathbf{K}}. \tag{5.66}$$

The separation-of-variables technique consists of replacing the obtained 3-dimensional problem of size MNL by M 2-dimensional problems of size NL or in replacing the 2-dimensional problem of size MN by M 1-dimensional problems of size N. This involves eigenvector and eigenvalue decomposition of $\mathbf{A}_x - \mathbf{K}_x$, i.e.,

$$\mathbf{W}_L^H (\mathbf{A}_x - \mathbf{K}_x) \mathbf{W}_R = \Lambda, \tag{5.67}$$

with $\mathbf{W}_L$ and $\mathbf{W}_R$ the matrices of, respectively, the left and right eigenvectors of $\mathbf{A}_x - \mathbf{K}_x$ and Λ the corresponding diagonal eigenvalue matrix.

Multiplying $\mathbf{A}$ by $\mathbf{W}_L^H \otimes \mathbf{I}_{yz}$ on the left and $\mathbf{W}_R \otimes \mathbf{I}_{yz}$ on the right gives us matrix $\mathbf{B}$, i.e.,

$$\begin{array}{rcl} \mathbf{B} & = & (\mathbf{W}_L^H \otimes \mathbf{I}_{yz}) \mathbf{A} (\mathbf{W}_R \otimes \mathbf{I}_{yz}) \\ & = & \Lambda \otimes \mathbf{I}_{yz} + \mathbf{I}_x \otimes (\mathbf{A}_{yz} - \mathbf{K}_{yz}) - (\mathbf{W}_L^H \otimes \mathbf{I}_{yz}) \widetilde{\mathbf{K}} (\mathbf{W}_R \otimes \mathbf{I}_{yz}) \end{array} \tag{5.68}$$

where we have used the fact that $\mathbf{W}_L^H \mathbf{W}_R = \mathbf{I}_x$ and as an immediate consequence $(\mathbf{W}_L^H \otimes \mathbf{I}_{yz})(\mathbf{W}_R \otimes \mathbf{I}_{yz}) = (\mathbf{W}_L^H \mathbf{W}_R) \otimes \mathbf{I}_{yz} = \mathbf{I}$, with $\mathbf{I}$ the MNL by MNL identity matrix.

If we introduce a permutation matrix $\mathbf{P}$ such that the non-zero elements are

$$\mathbf{P}\,(i + (j-1)M,\ j + (i-1)NL) = 1 \quad \forall i \in [1, M],\ \forall j \in [1, NL], \tag{5.69}$$

and if we further define $\widetilde{\widetilde{\mathbf{K}}}$ by

$$\widetilde{\widetilde{\mathbf{K}}} = \mathbf{P}^T(\mathbf{W}_L^H \otimes \mathbf{I}_{yz})\widetilde{\mathbf{K}}(\mathbf{W}_R \otimes \mathbf{I}_{yz})\mathbf{P}. \tag{5.70}$$

then the following simple relation is obtained:

$$\mathbf{P}^T\mathbf{B}\ \mathbf{P} = \mathbf{D} + \widetilde{\widetilde{\mathbf{K}}}, \tag{5.71}$$

with $\mathbf{D}$ a block diagonal matrix consisting of M blocks. Each block $\mathbf{D}_m$ of $\mathbf{D}$ is equal to

$$\mathbf{D}_m = \lambda_m \mathbf{I}_{yz} + \mathbf{A}_{yz} - \mathbf{K}_{yz}. \tag{5.72}$$

where λ_m are the corresponding eigenvalues of $\mathbf{D}_m$ and are elements of Λ.

Now, if we assume constant k, implying $\widetilde{\widetilde{\mathbf{K}}} = 0$, and use the earlier defined approximations $\overline{\gamma}$ for the absorbing conditions, the SOV method gives the following solution:

$$\mathbf{D}\tilde{\mathbf{p}} = \tilde{\mathbf{f}}, \tag{5.73}$$

with new variables

$$\begin{cases} \tilde{\mathbf{p}} &= \mathbf{P}^T(\mathbf{W}_L^H \otimes \mathbf{I}_{yz})\mathbf{p} \\ \tilde{\mathbf{f}} &= \mathbf{P}^T(\mathbf{W}_L^H \otimes \mathbf{I}_{yz})\mathbf{f} \end{cases} \tag{5.74}$$

The linear system (5.73) is easy to solve, because $\mathbf{D}$ is *block-diagonal*. Furthermore, since $\tilde{\mathbf{p}}$ and $\tilde{\mathbf{f}}$ comprise actually of M separate blocks of size NL, the overall problem also reduces to solving M 2-dimensional problems of size NL, which is clearly more efficient then solving one 3-dimensional problem of size MNL. This can be done in the following way. Let us decompose $\tilde{\mathbf{p}}$ and $\tilde{\mathbf{f}}$ in M blocks $\tilde{\mathbf{p}}_m$ and $\tilde{\mathbf{f}}_m$ of size NL. Then the solution can be obtained by solving the M systems

$$(\lambda_m \mathbf{I}_{yz} + \mathbf{A}_{yz} - \mathbf{K}_{yz})\tilde{\mathbf{p}}_m = \tilde{\mathbf{f}}_m \tag{5.75}$$

using expression (5.72).

Notice that even though the final cost for computing the solution is lo, one should anticipate the overhead cost involving the computation of M orthonormal eigenvectors and eigenvalues of the system $\mathbf{A}_x - \mathbf{K}_x$ which leads to matrix $\mathbf{D}$.

5.7.2 Preconditioned system

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 $\widetilde{\mathbf{K}} = 0$. Extension to inhomogeneous media can not be done in the same way, because of this restriction. However, we can use the linear system (5.73) as the preconditioner of the original system $\mathbf{Ap} = \mathbf{f}$ and solve the preconditioned linear system iteratively. This preconditioned system is given as $\mathbf{M}^{-1}\mathbf{Ap} = \mathbf{M}^{-1}\mathbf{f}$ where

$$\mathbf{M}^{-1} = (\mathbf{W}_R \otimes \mathbf{I}_{yz})\mathbf{P}\mathbf{D}^{-1}\mathbf{P}^T(\mathbf{W}_L^H \otimes \mathbf{I}_{yz}) \tag{5.76}$$

with the help of expressions (5.68) and (5.71).

Since $\mathbf{M}$ can be considered as an approximate system of $\mathbf{A}$, $\mathbf{M}^{-1}\mathbf{A}$ is close to the identity matrix $\mathbf{I}$, which should be efficient to solve. This is proved for very low frequency cases and relatively smooth media, especially when the wavenumber varies only in one dimension (i.e., with one-dimensional models), see sections 5 of [17]. Only a few iterations are needed to obtain convergence of the iterative scheme in one-dimensional cases. However, the degree of accuracy of this approximate depends largely on how k varies in the medium. The SOV method *fails* in numerical examples with more complex models, like small wedge and Marmousi, in cases where we have *largely* varying k or large frequencies, which are of practical interest.

Although the convergence is faster in a smoother background, the method also fails for large frequencies ($\omega \geq 30$) in this case.

In section 7 of [17] there has been concluded that numerically examples suggest that it is not possible to find a better decomposition of the wavenumber that would improve the convergence rate of this approach.

5.8 Analytic ILU (AILU)

Gander & Nataf [10, 11] have derived a new block ILU preconditioner for linear systems stemming from symmetric positive definite elliptic partial differential equations. The new preconditioner AILU is able to improve the asymptotic convergence behavior in contrast to other ILU preconditioners like ILU(0) and ILU(tol). Instead of finding a proper preconditioner for the discrete Helmholtz problems, the preconditioner is determined from an analytical factorization of the *continuous* differential operator.

5.8.1 Analytic parabolic factorization

The main idea of AILU preconditioner is based on the parabolic factorization of an elliptic operator (L) (like the Helmholtz operator) into a form

$$\mathcal{L} = (\partial_x + \Lambda_1) \circ (\partial_x - \Lambda_2) \tag{5.77}$$

where Λ_1 and Λ_2 are positive opereators and thus the first factor represents a parabolic operator acting in the *positive* x-direction and the second one a parabolic operator acting in the *negative* x-direction.

In the sequel we restrict ourselves for the analysis tot the case of the Helmholtz-operator $\mathcal{L} = -\Delta - k^2$ where Δ denotes the Laplacian in two dimensions and $k \geq 0$. The analysis in three dimensions is similar.

We denote the Fourier transform $\hat{f}(\vartheta)$ of $f(y) : \mathbb{R} \to \mathbb{R}$ by

$$\hat{f}(\vartheta) = \mathcal{F}_y(f)(\vartheta) := \int_{-\infty}^{\infty} e^{-i\vartheta y} f(y)\ \mathrm{d}y, \tag{5.78}$$

and the inverse Fourier transform of $\hat{f}(\vartheta)$ by

$$f(y) = \mathcal{F}_y^{-1}(\hat{f})(y) := \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{i\vartheta y} \hat{f}(y)\ \mathrm{d}k, \tag{5.79}$$

We obtain the following lemma.

Lemma 1 *The 2-dimensional linear operator $\mathcal{L} = -\Delta - k^2$ admits the continuous parabolic factorization*

$$-\Delta - k^2 = -(\partial_x + \Lambda_1) \circ (\partial_x - \Lambda_2) \tag{5.80}$$

where $\Lambda_1 = \Lambda_2 = \mathcal{F}_y^{-1}\left(\sqrt{\vartheta^2 - k^2}\ \right)$ are pseudo-differential operators in y.

In the proof of the above lemma we have used the fact that $\mathcal{F}_y\left(f^{(n)}\right) = (ik)^n \mathcal{F}_y(f)$ and therefore $\mathcal{F}_y(-\Delta - k^2) = -\partial_{xx} + \vartheta^2 - k^2$.

To relate this parabolic factorization to the exact block LU decomposition of the discrete matrix operator, we discretize the x-direction of $-\Delta - k^2$ and compute the analytic factorization of (5.80) for the *semi*-discrete operator $-\Delta_h - k^2$ with

$$\Delta_h = D_x^- D_x^+ + \partial_{yy}, \tag{5.81}$$

where for a vector $\mathbf{u}$ the difference operator

$$D_x^-(\mathbf{u})_i := \frac{u_i - u_{i-1}}{h} \quad \text{and} \quad D_x^+(\mathbf{u})_i := \frac{u_{i+1} - u_i}{h} \tag{5.82}$$

represent the discrete derivatives on a given structured mesh. Then we get the following lemma (see section 2 of [10] for the proof).

Lemma 2 *The semi-discrete operator $\Delta_h = D_x^- D_x^+ + \partial_{yy}$ admits the following semi-discrete parabolic factorization in the Fourier transformed domain:*

$$\mathcal{F}_y(-\Delta - k^2) = -\left(D_x^- + \left(\tau h - \frac{1}{h}\right)\right) \frac{1}{h^2 \tau} \left(D_x^+ - \left(\tau h - \frac{1}{h}\right)\right) \tag{5.83}$$

where the pesudo-differential operator τ has the symbol

$$\tau = \frac{1}{h^2} + \frac{\Psi}{2} + \frac{1}{2h}\sqrt{\Psi^2 + 4\Psi}, \tag{5.84}$$

with $\Psi := \vartheta^2 - k^2$.

Note that as we take the limit for $h \to 0$ in (5.83) we recover again the continuous parabolic factorization (5.80), since the middle term disappears in the limit ($h^2\tau \to 1$ as $h \to 0$). For discrete problems, it is however important to include the middle factor, which was not the case in previous work on continuous parabolic factorizations.

Furthermore, one can show that (5.83) corresponds to the exact block-LU decomposition of the fully discrete matrix operator with the following block structure

$$\mathbf{A} = \begin{bmatrix} \mathbf{D}_1 & \mathbf{L}_{1,2} & & \\ \mathbf{L}_{2,1} & \mathbf{D}_2 & \ddots & \\ & \ddots & \ddots & \mathbf{L}_{n-1,n} \\ & & \mathbf{L}_{n,n-1} & \mathbf{D}_n \end{bmatrix} \tag{5.85}$$

where $\mathbf{A}$ satisfies $\mathbf{Au} = \mathbf{f}$, which is obtained by discretizing the Helmholtz operator. See section 2 of [11] for the proof.

5.8.2 AILU preconditioner

One could use directly the parabolic factorization given in (5.83) to solve the original problem $\mathbf{Au} = \mathbf{f}$. Instead of solving the linear system, one would have to solve two lower-dimensional parabolic problems: one in the positive and one in the negative x-direction, corresponding to a forward and a backward solve of the exact block LU decomposition. This is however not advisable since the parabolic factorization contains *non-local* operators in y-direction corresponding to dense subblocks $\mathbf{T}_i$ in the block LU decomposition (see lemma 2.3 of [11]). Therefore, we *approximate* the parabolic factorization by *local* operators and use the factorization as a preconditioner corresponding to a new type of ILU preconditioner we call Analytic ILU (AILU). We replace the non-local operator τ in (5.83) by a local approximation of the form

$$\tau_{app} = \tau = \frac{1}{h^2} + \frac{\Psi}{2} + \frac{1}{2h}\sqrt{\alpha + \beta\vartheta^2}, \tag{5.86}$$

with $\alpha, \beta \in \mathbb{C}, \mathrm{Re}(\beta) > 0$ and Ψ as in lemma 2. This leads to a classical linear second order parabolic problem, since k^2 corresponds to a second derivative in y [2]. Since we have the analytic parabolic factorization, we can use the parameters α and β to optimize the performance of the AILU preconditioner. We insert the approximation τ_{app} into the factorization (5.83) and obtain the operator resulting from the approximate factorization of the original operator $\mathcal{F}_y\ (-\Delta_h - k^2) = \Psi - D_x^- D_x^+$ in the form

$$\mathcal{L}_{app} = -D^{-1}D^+ + \tau_{app} + \frac{1}{\tau_{app}h^4} - \frac{2}{h^2}.\textbf{[navragen]} \tag{5.87}$$

[2] Note that we did not include a first order term because the underling problem is symmetric.

The complex numbers α and β are to be chosen such that $\mathcal{L}_{app}^{-1}\,\mathcal{L}$ is as close as possible to the identity, except for a few frequencies which will be taken into account by the Krylov method, see section 4 of [11]. We find after some calculations that we have to minimize

$$\rho(\vartheta) := \left| 1 - 2\frac{(2 - k^2h^2 + \alpha h + h(h+\beta)\vartheta^2)\Psi}{(\alpha - k^2h + (\beta + h)\vartheta^2)^2} \right|. \tag{5.88}$$

Notice that in a numerical setting, the frequency parameter ϑ is bounded by $\vartheta_{\min}$ and $\vartheta_{\max}$ which can easily be determined, see section 3 of both [10, 11]. There can also be found more detail analysis including the convergence properties of the AILU.

In numerical experiments on the Helmholtz problem (section 4 of [10]), one has found interesting computational performance with the AILU preconditioner, where the QMR algorithm has been used (see section 5.9 (QMR method) of [6] for an explanation of the method). A significant improvement is shown for high frequency cases, whenever QMR with standard ILU-preconditioners fails to converge. In such cases QMR with AILU converges to the solution. In terms of operation count required for computing the preconditioner, AILU is comparable to ILU(0).

Chapter 6

Summary & Outlook

In this Chapter we give a short summary of this report and we give our plans towards further research in the coming six months.

6.1 Short summary

We have derived the Helmholtz boundary value problem (HBVP) in Chapter 2, which consists of the Helmholtz equation with absorbing boundary conditions. The Helmholtz equation has been deduced from the wave equation.

Numerical experiments in one-dimension were done in Chapter 3 to get more experience with HBVP. Analytical and numerical solutions of some testcases, for constant and variable wavenumber k, are worked out and compared. In numerical methods we have seen that inhomogenity of k can be implemented in several ways. We have also done some spectral-analysis as preparation to iterative methods and preconditioners.

Chapter 4 deals with this Krylov iterative methods and their algorithms. For sparse and large linear systems direct solvers are not attractive, while iterative methods could lead to satisfactory solutions. Important methods are CGNR, Bi-CGSTAB and GMRES, which has been discussed in more detail. We have seen that these methods are only efficient if they are preconditioned. Therefore, we summed up a few preconditioners in Chapter 5, which can be used to deal with HBVP. Many preconditioners (like separation-of-variables and Made) fail when increasing the frequency, while we are interested in large frequencies.

Investigations on good preconditioners for Helmholtz problems are pursued. An example is AILU, which has been compared with another ILU-preconditioners in 2-dimensional numerical experiments. The results are positive. However, so far no results are reported for general 3-dimensional inhomogeneous problems.

Another example of a preconditioner is based on separation-of-variables

(SOV). For 2-dimensional test cases with simple inhomogenity, the linear system can be solved with a convergence rate nearly independent of the domain and mesh size. However, the preconditioner breaks down for complex inhomogeneous media.

Recently, the complex shifted Laplace preconditioner has been proposed as a good one for solving HBVP. The method is being improved to get an efficient method with this preconditioner.

6.2 Future research

In further research we will restrict our domain to preconditioning of HBVP and try to find a good preconditioner which can be applied to solve the 3-dimensional case. Several ideas are available about the SoV and shifted Laplace preconditioners. We know that the complex shifted Laplace preconditioner is easy to implement and 'weakly' dependent on the wavenumber k, but it is still slowly converging. While separation-of-variables is a very efficient preconditioner for constant k, it fails for large and highly inhomogeneous values of k. Therefore, combining this two preconditioners may lead to a better preconditioner.

Plessix and Mulder [17] have given a possible (physical) explanation for the bad convergence of the separation-of-variables method. We like to investigate the cause of this bad convergence by using *eigenvalue analysis*. Hopefully this will also lead to ideas for a good combination with the shifted Laplace preconditioner.

We shall deal with target problems to test our preconditioners in future research, where *physical* parameters will be applied to imitate the practical situation as good as possible. In 2-dimensions, we use models with constant k first and subsequently layer and wedge models (see e.g. Figure 2 of [17] and Figure 5 of [6]), respectively. Later on, we shall use 3-dimensional models, if it is practicable.

Bibliography

[1] Achenbach, J.D., *Wave propagation in Elastic Solids*, North-Holland publishing company, 1973.

[2] Axelsson, O., *Iterative Solution Methods*, Cambridge University Press: Cambridge, 1994.

[3] Bayliss, A., Goldstein, C.I., Turkel, E., *An iterative method for Helmholtz equation*, J. Comput. Phys., 49:443-457, 1983.

[4] Boyce, W.E., DiPrima, R.C., *Elementary differential equations and boundary value problems*, 6th edition, Wiley & Sons, 1997.

[5] Colton, J., Kress, R., *Inverse acoustic and electromagnetic scatting theory*, Springer-Verlag, Berlin-Heidelberg, 1998.

[6] Erlangga, Y.A., *Some numerical aspects for solving sparse large linear systems derived from the Helmholtz equation*, report 02-12, TU Delft, 2002.

[7] Erlangga, Y.A., Vuik, C., Oosterlee, C.W., *On a class of preconditioners for solving the Helmholtz equation*, report 03-01, TU Delft, 2003.

[8] Erlangga, Y., Oosterlee, C.W., Vuik, C., *Shifted Laplace preconditioners for the Helmholtz equations*, report 03-18, TU Delft, 2003.

[9] Faber, V., Manteuffel, T., *Necessary and Sufficient Conditions for the Existence of a Conjugate Gradient Method*, SIAM, J. Numer. Anal. 21, 315-339, 1984.

[10] Gander, M.J., Nataf, F., *AILU for Helmholtz problems: a new preconditioner based on the analytic parabolic factorization*, J. Comp. Acoustics, 9:1499-1509, 2001.

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

[12] Heikkola, E., Rossi, T., Toivanen, J., *A parallel fictitious domain method for the three-dimensional Helmholtz equation*, No. B9, University of Jyväskylä, 2000.

[13] van Kan, J., Segal, A., *Numerieke methoden voor partiële differentiaalvergelijkingen*, Delftse uitgevers Maatschappij BV, 1993.

[14] Korving, C., Corstens, H.F.M., *Mechanica van continue media I*, lecture-notes, WI2090, TU Delft, 2000.

[15] Laird, A.L., *Preconditioned iterative solution of the 2nd Helmholtz equation*, 1st year's report, Oxford University, St. Hugh's College, Oxford, 2001.

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

[17] Mulder, W., Plessix, R.E., *Separation-of-variables as a preconditioner for an iterative Helmholtz solver*, Appl. Num. Math., 44(3): 385-400, 2003.

[18] Peters, A., Eiermann, M., Daniels, H., *Symmetric versus non-symmetric matrix techniques: a comparative study of two Galerkin-FE approaches for the advection dispersion equation*, IBM report 75.91.07, 1991.

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

[20] Shewchuk, J.R., *An Introduction to the Conjugate Gradient Method without the Agonizing Pain*, Edition 1,25, Carnegie Mellon University Pittsburgh, 1994.

[21] Sleijpen, G.L.G., Fokkema, D.R., *BiCGstab(l) for linear equations involving unsymmetric matrices with complex spectrum*. Electron. Trans. Numer. Anal.,1(Sept.):1132 , 1993.

[22] Sonneveld, P., *CGS: a fast Lanczos type solver for nonsymmetric linear systems*, SIAM J. Sci. Stat. Comp., 10 , 36-52. 53, 1989.

[23] Van der Sluis, A., *Conditioning, equilibration and pivoting in linear algebraic systems*, Numer. Math., 15:74-86, 1970.

[24] Tang, J., *Gauss-Seidel methods: iteratieve methodes ter oplossing van de Poission vergelijking*, report, wi4040, TU Delft, 2003.

[25] Voevodin, V., *The Problem of Non-Self-Adjoint Generalization of the Conjugate Gradient Method is Closed*, U.S.S.R. Comput. Maths. and Math. Phys. 23, 143-144, 1983.

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

[27] Van der Vorst, H.A., *Bi-CGSTAB: A Fast and Smoothly Converging Variant of Bi-CG for the Solution of Nonsymmetric Linear Systems*, SIAM, J. Sci. Statist. Comput. 13, 631-644, 1992.

[28] Vuik, C., *Numerical methods for large algebraic systems*, lecture-notes, WI4010, TU Delft, 2004.

[29] Vuik, C., *Numerieke methoden voor differentiaalvergelijkingen*, lecture-notes, wi2091/wi2092, TU Delft, 2000.

Appendix A

Matlab codes

In this appendix one can find the Matlab codes which have been used in Chapter 3.

A.1 Example 1

```
function p=Ex1(N1,k1,f1);

N=N1-2;
deltax=pi/N1;

k=k1*ones(N,1);
kreal=k1*ones(N+2,1);
f=f1*ones(N,1);

A=zeros(N,N);
hoofd=ones(N,1);
sub=ones(N-1,1);
A1=(1/deltax^2)*(-2*diag(hoofd)+diag(sub,1)+diag(sub,-1));
A2=diag(k)^2;
hoofdA3=zeros(N,1);
hoofdA3(1)=1/(deltax^2*(1+j * kreal(1) * deltax));
hoofdA3(N)=1/(deltax^2*(1+j * kreal(N+2) * deltax));
A3=diag(hoofdA3);
A=A1+A2+A3;

p=A\f;
p0=p(1)/(1+j*kreal(1)*deltax);
pLast=p(N)/(1+j*kreal(N+2)*deltax);
preal=zeros(N+2,1);
preal=real([p0; p; pLast]);
```

```
pim=zeros(N+2,1);
pim=imag([p0; p; pLast]);

count=0:pi/(N+1):pi;
subplot(2,1,1)
plot(count,preal)
ylabel('pressure p')
title('Example 1:  Real part of the solution (N=100)')
axis([0 pi 1.8 2.2])
subplot(2,1,2)
plot(count,pim)
hold on plot(count, 2*sin(count), 'r:')
xlabel('x-axis')
ylabel('pressure p')
title('Example 1:  Imaginary part of the solution (N=100)')
axis([0 pi -0.5 2.5])
```

A.2 Example 2

```
function p=Ex2(N1,k1,f1);

N=N1-2;
deltax=pi/N1;

k=k1*ones(N,1);
kreal=k1*ones(N+2,1);
f=f1*ones(N,1);

A=zeros(N,N);
hoofd=ones(N,1);
sub=ones(N-1,1);
A1=(1/deltax^2)*(-2*diag(hoofd)+diag(sub,1)+diag(sub,-1));
A2=diag(k)^2;
hoofdA3=zeros(N,1);
hoofdA3(1)=1/(deltax^2*(1+j * kreal(1) * deltax));
hoofdA3(N)=1/(deltax^2*(1+j * kreal(N+2) * deltax));
A3=diag(hoofdA3);
A=A1+A2+A3;

p=A\f;
p0=p(1)/(1+j*kreal(1)*deltax);
pLast=p(N)/(1+j*kreal(N+2)*deltax);
preal=zeros(N+2,1);
preal=real([p0; p; pLast]);

count=0:pi/(N+1):pi;
plot(count,preal)
xlabel('x-axis')
hold on
plot(count, 0.25-0.25*cos(2*count), 'r:')
ylabel('pressure p')
title('Solution of example 2 (N=100)')
axis([0 pi 1.8 2.2])
```

A.3 Example 3

Method 1

```
function p=Ex3met1(N1,k1,k2,f1);

N=N1-2;
deltax=pi/N1;
Nh=0.5*(N-1);

f=f1*ones(N,1);
f(0.5*(N+1))=0;

A=zeros(N,N);
hoofd1=(-2/deltax^2+k1^2)*ones(Nh,1);
hoofd2=2;
hoofd3=(-2/deltax^2+k2^2)*ones(Nh,1);
hoofd=[hoofd1; hoofd2 ;hoofd3];
subonder=(1/deltax^2)*ones(N-1,1);
subonder(0.5*(N-1))=-1;
subboven=(1/deltax^2)*ones(N-1,1);
subboven(0.5*(N+1))=-1;
A1=(diag(hoofd)+diag(subboven,1)+diag(subonder,-1));
hoofdA2(1)=1/(deltax^2*(1+j * k1 * deltax));
hoofdA2(N)=1/(deltax^2*(1+j * k2 * deltax));
A2=diag(hoofdA2);
A=A1+A2;

p=A\f;
p0=p(1)/(1+j*kreal(1)*deltax);
pLast=p(N)/(1+j*kreal(N+2)*deltax);

preal=zeros(N+2,1);
preal=real([p0; p; pLast]);
pim=zeros(N+2,1);
pim=imag([p0; p; pLast]);

h=pi/N;
x1=[0:h:0.5*pi];
x2=[0.5*pi:h:pi];
p1r=9-(15/2)*cos(x1)-6*sin(x1);
p2r=1+(5/2)*cos(3*x2)-2*sin(3*x2);
x=[x1 x2];
prex=[p1r p2r]';
```

```
p1i=6*cos(x1) +1.5*sin(x1);
p2i=-2*cos(3*x2) - 1.5*sin (3*x2);
piex=[p1i p2i]';
count=0:pi/(N+1):pi;

subplot(2,2,1)
plot(count,preal)
hold on
plot(x, prex, 'r:')
ylabel('pressure p')
title('Real part (Method 1)')
axis([0 pi -3 5])

subplot(2,2,3)
plot(count,pim)
hold on
plot(x, piex, 'r:')
xlabel('x-axis')
ylabel('pressure p')
title('Imaginary part (Method 1)')
axis([0 pi -4 7])
```

Method 2

```
function p=Ex3met2(N1,k1,k2,f1);

N=N1-2;
deltax=pi/N1;
Nh=0.5*(N-1);

f=f1*ones(N,1);
f(0.5*(N+1))=0;

A=zeros(N,N);
hoofd1=(-2/deltax^2+k1^2)*ones(Nh,1);
hoofd2=-2/deltax^2+((k1+k2)/2)^2;
hoofd3=(-2/deltax^2+k2^2)*ones(Nh,1);
hoofd=[hoofd1; hoofd2 ;hoofd3];
subonder=(1/deltax^2)*ones(N-1,1);
subboven=(1/deltax^2)*ones(N-1,1);
A1=(diag(hoofd)+diag(subboven,1)+diag(subonder,-1));
hoofdA2(1)=1/(deltax^2*(1+j * k1 * deltax));
hoofdA2(N)=1/(deltax^2*(1+j * k2 * deltax));
```

```
A2=diag(hoofdA2);
A=A1+A2;

p=A\f;
p0=p(1)/(1+j*kreal(1)*deltax);
pLast=p(N)/(1+j*kreal(N+2)*deltax);

preal=zeros(N+2,1);
preal=real([p0; p; pLast]);
pim=zeros(N+2,1);
pim=imag([p0; p; pLast]);

h=pi/N;
x1=[0:h:0.5*pi];
x2=[0.5*pi:h:pi];
p1r=9-(15/2)*cos(x1)-6*sin(x1);
p2r=1+(5/2)*cos(3*x2)-2*sin(3*x2);
x=[x1 x2];
prex=[p1r p2r]';
p1i=6*cos(x1) +1.5*sin(x1);
p2i=-2*cos(3*x2) - 1.5*sin (3*x2);
piex=[p1i p2i]';
count=0:pi/(N+1):pi;

subplot(2,2,2)
plot(count,preal)
hold on
plot(x, prex, 'r:')
ylabel('pressure p')
title('Real part (Method 2)')
axis([0 pi -3 5])

subplot(2,2,4)
plot(count,pim)
hold on
plot(x, piex, 'r:')
xlabel('x-axis')
ylabel('pressure p')
title('Imaginary part (Method 2)')
axis([0 pi -4 7])
```

A.4 Example 4

Method 1

```
function p=Ex4met1(N1,k1,k2,f1);

N=N1-2;
deltax=pi/N1;
Nh=0.5*(N-1);

f=f1*ones(N,1);
f(0.5*(N+1))=0;

A=zeros(N,N);
hoofd1=(-2/deltax^2+k1^2)*ones(Nh,1);
hoofd2=2;
hoofd3=(-2/deltax^2+k2^2)*ones(Nh,1);
hoofd=[hoofd1; hoofd2 ;hoofd3];
subonder=(1/deltax^2)*ones(N-1,1);
subonder(0.5*(N-1))=-1;
subboven=(1/deltax^2)*ones(N-1,1);
subboven(0.5*(N+1))=-1;
A1=(diag(hoofd)+diag(subboven,1)+diag(subonder,-1));
hoofdA2(1)=1/(deltax^2*(1+j * k1 * deltax));
hoofdA2(N)=1/(deltax^2*(1+j * k2 * deltax));
A2=diag(hoofdA2);
A=A1+A2;

p=A\f;
p0=p(1)/(1+j*kreal(1)*deltax);
pLast=p(N)/(1+j*kreal(N+2)*deltax);

preal=zeros(N+2,1);
preal=real([p0; p; pLast]);
pim=zeros(N+2,1);
pim=imag([p0; p; pLast]);
count=0:pi/(N+1):pi;

subplot(2,2,1)
plot(count,preal)
ylabel('pressure p')
title('Real part (Method 1)')
axis([0 pi -12 2])
```

```
subplot(2,2,3)
plot(count,pim)
xlabel('x-axis')
ylabel('pressure p')
title('Imaginary part (Method 1)')
axis([0 pi -1 1])
```

Method 2

```
function p=Ex4met2(N1,k1,k2,f1);

N=N1-2;
deltax=pi/N1;
Nh=0.5*(N-1);

f=f1*ones(N,1);
f(0.5*(N+1))=0;

A=zeros(N,N);
hoofd1=(-2/deltax^2+k1^2)*ones(Nh,1);
hoofd2=-2/deltax^2+((k1+k2)/2)^2;
hoofd3=(-2/deltax^2+k2^2)*ones(Nh,1);
hoofd=[hoofd1; hoofd2 ;hoofd3];
subonder=(1/deltax^2)*ones(N-1,1);
subboven=(1/deltax^2)*ones(N-1,1);
A1=(diag(hoofd)+diag(subboven,1)+diag(subonder,-1));
hoofdA2(1)=1/(deltax^2*(1+j * k1 * deltax));
hoofdA2(N)=1/(deltax^2*(1+j * k2 * deltax));
A2=diag(hoofdA2);
A=A1+A2;

p=A\f;
p0=p(1)/(1+j*kreal(1)*deltax);
pLast=p(N)/(1+j*kreal(N+2)*deltax);

preal=zeros(N+2,1);
preal=real([p0; p; pLast]);
pim=zeros(N+2,1);
pim=imag([p0; p; pLast]);
count=0:pi/(N+1):pi;

subplot(2,2,2)
plot(count,preal)
```

```
ylabel('pressure p')
title('Real part (Method 2)')
axis([0 pi -12 2])

subplot(2,2,4)
plot(count,pim)
xlabel('x-axis')
ylabel('pressure p')
title('Imaginary part (Method 2)')
axis([0 pi -1 1])
```

A.5 Norm of error of example 1

Add-on in `Ex3met1(N1,k1,k2,f)` and `Ex3met2(N1,k1,k2,f)`

```
function normmet1=Ex3met1(N1,k1,k2,f1) /
function normmet2=Ex3met2(N1,k1,k2,f1);

normreal=norm(preal-prex);
normim=norm(pim-piex);
normmet2=[normreal normim];
```

Main program

```
function InputEx3(hoogsteN);

N=hoogsteN;
aantal=(N+5)/10;
realvec1=zeros(aantal,1);
imvec1=zeros(aantal,1);
realvec2=zeros(aantal,1);
imvec2=zeros(aantal,1);

for i=5:10:N
figure(1)
iecht=0.1*i+0.5;
pmet1=Ex3met1(i,1,3,9);
pmet2=Ex3met2(i,1,3,9);
realvec1(iecht,1)=pmet1(1);
imvec1(iecht,1)=pmet1(2);
realvec2(iecht,1)=pmet2(1);
imvec2(iecht,1)=pmet2(2);
end

i=5:10:N;
figure(2)
subplot(1,2,1)
plot(i,realvec1)
hold on
plot(i,realvec2,'--g' )
xlabel('N');
ylabel('norm of error');
title('Real part of the solution');
axis([5 N 0 6]);
legend('Method 1','Method 2')
```

```
subplot(1,2,2)
plot(i,imvec1)
hold on
plot(i,imvec2,'--g')
xlabel('N');
ylabel('norm of error');
title('Imaginary part of the solution');
axis([5 N 0 6]);
legend('Method 1','Method 2')
```

A.6 Eigenvalues of example 1

```
A=Ex1(25,1,2);

figure(4);
subplot(3,3,1);
eigA=eig(A);
realeigA=real(eigA)
imeigA=imag(eigA)
plot(realeigA,imeigA, 'rx' );
xlabel('real part');
ylabel('imaginary part');
title('N=25');
axis([min(realeigA) max(realeigA) min(imeigA) max(imeigA)]);

figure(2);
A1=Ex1(50,1,2);

figure(4);
subplot(3,3,2);
eigA1=eig(A1);
realeigA1=real(eigA1);
imeigA1=imag(eigA1);
plot(realeigA1,imeigA1, 'bo' );
xlabel('real part');
title('N=50');
axis([min(realeigA1) max(realeigA1) min(imeigA1) max(imeigA1)]);

figure(3);
A2=Ex1(100,1,2);

figure(4);
subplot(3,3,3);
eigA2=eig(A2);
realeigA2=real(eigA2);
imeigA2=imag(eigA2);
plot(realeigA2,imeigA2, 'g*' );
xlabel('real part');
title('N=100');
axis([min(realeigA2) max(realeigA2) min(imeigA2) max(imeigA2)]);

figure(4);
subplot(3,3,5);
plot(realeigA,imeigA, 'rx' );
```

```
xlabel('real part');
ylabel('imaginary part');
hold on;
plot(realeigA1,imeigA1, 'bo' );
hold on;
plot(realeigA2,imeigA2, 'g*' );
axis([min(realeigA2) max(realeigA2) min(imeigA2) max(imeigA2)]);
```

A.7 Comparing modified & original example 1

Modified system

```
function A=Ex1mod(N1,k1,f1);

N=N1-2;
deltax=pi/N1;

A=zeros(N,N);
hoofd=ones(N,1);
sub=ones(N-1,1);
A1=(1/deltax^2)*(-2*diag(hoofd)+diag(sub,1)+diag(sub,-1));
A2=diag(k)^2;
A=A1+A2;
```

Main program

```
A=Ex1(25,1,2);
eigA=eig(A);
realeigA=real(eigA);
imeigA=imag(eigA);

figure(3);
subplot(2,2,1);
plot(realeigA,imeigA, 'rx' );
xlabel('real part');
ylabel('imaginary part');
axis([min(realeigA) max(realeigA) min(imeigA) max(imeigA)]);
title('Eigenvalues of example 1');
subplot(2,2,3);
plot(realeigA,imeigA, 'rx' );
xlabel('real part');
ylabel('imaginary part');
A1=Ex1mod(25,1,2);
eigA1=eig(A1);
realeigA1=real(eigA1);
imeigA1=imag(eigA1);

figure(3);
subplot(2,2,2)
plot(realeigA1,imeigA1, 'bo' );
xlabel('real part');
ylabel('imaginary part');
```

```
axis([min(realeigA) max(realeigA) -0.1 0.1]);
title('Eigenvalues of modified example 1');
subplot(2,2,3);
hold on;
plot(realeigA1,imeigA1, 'bo' );
xlabel('real part');
ylabel('imaginary part');
axis([min(realeigA) max(realeigA) min(imeigA) 0.1]);
legend('Original example 1 (N = 25)','Modified example 1 (N = 25)')
```

Appendix B

Rest of appendices

After giving the Matlab codes in the previous appendices, the rest of the appendices can be found here.

B.1 Proof from Chapter 5

Denote $a = k^2$ and $b = \mu_1$ for simplicity, then we have $0 < a < b$. We shall prove $\lambda_{\min}(\mathbf{Q}_0) > \lambda_{\min}(\mathbf{Q}_1)$ and $\lambda_{\min}(\mathbf{Q}_0) > \lambda_{\min}(\mathbf{Q}_i)$ by using expressions (31)-(34) of [7].

Proof of $\lambda_{\min}(\mathbf{Q}_0) > \lambda_{\min}(\mathbf{Q}_1)$
We have assumed $0 < a < b$, so the following inequalities are equivalent to each other

$$\begin{aligned}
\frac{a}{b} &< 1, \\
\frac{1}{b} &< \frac{2}{a+b}, \\
1 - \frac{2a}{a+b} &< 1 - \frac{a}{b}, \\
\left(1 - \frac{2a}{a+b}\right)^2 &< \left(1 - \frac{a}{b}\right)^2, \\
\lambda_{\min}(\mathbf{Q}_1) &< \lambda_{\min}(\mathbf{Q}_0).
\end{aligned}$$

□

Proof of $\lambda_{\min}(\mathbf{Q}_0) > \lambda_{\min}(\mathbf{Q}_i)$
The following inequalities are equivalent:

$$\begin{aligned}
\frac{1}{a^2+b^2} &< \frac{1}{b^2}, \\
\frac{(b-a)^2}{a^2+b^2} &< \frac{(b-a)^2}{b^2}, \\
1-\frac{2ab}{a^2+b^2} &< \left(1-\frac{a}{b}\right)^2, \\
\lambda_{\min}(\mathbf{Q}_i) &< \lambda_{\min}(\mathbf{Q}_0).
\end{aligned}$$

□