Delft University of Technology
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft Institute of Applied Mathematics

Acceleration of the 2D Helmholtz model HARES

A thesis submitted to the
Delft Institute of Applied Mathematics
in partial fulfillment of the requirements

for the degree

MASTER OF SCIENCE
in
APPLIED MATHEMATICS

by

G.E.M. van de Sande
Delft, the Netherlands
May 2012

MSc THESIS APPLIED MATHEMATICS

"Acceleration of the 2D Helmholtz model HARES"

G.E.M. VAN DE SANDE

Delft University of Technology

Daily supervisor

Dr.ir. M.B. van Gijzen

Responsible professor

Prof.dr.ir. C. Vuik

Other thesis committee members

Dr. H.M. Schuttelaars

Dr. H. Talstra

May 2012

Delft, the Netherlands

Abstract

HARES is a model to determine the wave motion in harbours and at the shore. The effects of diffraction, reflection, refraction, shoaling and the dissipation caused by wave breaking and bottom friction are all taken into account in the calculation. The equation that includes these effects is the non-linear Mild-Slope equation. To discretize the non-linear Mild-Slope equation Picard iteration is performed and the spatial discretization is based on the Ritz-Galerkin finite element method. This discretization results in a linear system of equations which is iteratively solved with the Krylov subspace method Bi-CGSTAB preconditioned with the incomplete LU decomposition. The time to compute the solution is undesirably long for the current version of HARES. In this thesis we address several methods to reduce the computing time of this implementation. As an improvement we propose to implement a suitable stopping criterion for the non-linear loop, such that we only perform the necessary amount of iterations. We replace Picard iteration by inexact Picard iteration to reduce the number of matrix-vector products and hence the time to compute the non-linear solution. The iterative solution method is improved by changing the current implementation into IDR(s) preconditioned with a complex shifted Laplace preconditioner. These improvements result in a programme which is upto 58 times faster than the original implementation. Furthermore a direct method for solving the system of equations is tested, which gives very good results.

Preface

This thesis is the final project of the master Applied Mathematics at Delft University of Technology. I would like to thank my daily supervisor Martin van Gijzen for his guidance and input on my work for this project. I enjoyed his way of telling a story, not only about his research but also about his daily events. I would like to thank Harmen Talstra of Svašek Hydraulics for suggesting this project.

Furthermore I would like to thank my father Jan van de Sande for his input on my English writing, my boyfriend Jurriaan Versendaal for his support and patience, Linda Crapts, Frank Tabak and Jacob de Zoete for the nice conversations during the lunch breaks and Joost van Zwieten for the implementation of `MUMPS` and his thoughts on the project.

Utrecht, the Netherlands — Gemma van de Sande
May 2012

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The subject of this thesis is suggested by the company Svašek Hydraulics[1], a specialist consultant in coastal, harbour and river engineering. Advanced numerical models are used to determine water dynamics, i.e. currents and waves, and sediment transport caused by water dynamics. The majority of the models are developed by the employees of Svašek Hydraulics and mainly based on the finite element method. One of these models is HARES, which stands for HArbour RESonance. HARES calculates the wave penetration and is especially useful in harbour and breakwater studies. With the outcome of HARES the natural frequency of the studied domain can be determined and therefore also the sensitivity for resonance.

HARES incorporates the effects of diffraction, reflection, refraction and shoaling, which influences the shape of the water waves. The equation that includes these four effects is the two dimensional linear Mild-Slope equation. The latest addition to HARES is the inclusion of energy dissipation caused by bottom friction and wave breaking. This leads to a non-linear term in the Mild-Slope equation. The non-linear Mild-Slope equation is very similar to the damped Helmholtz equation. More advance numerical methods are necessary to obtain the wave motion in the considered area.

The current programme has the following structure: An outer loop is performed to deal with the non-linearity of the equation, which results in a sequence of linear systems of equations. In the inner loop we determine the solution of this system of equations. This procedure is repeated 25 times, without knowing whether convergence has been reached and whether the desired non-linear solution is obtained. In HARES Picard iteration is implemented to discretize the non-linear Mild-Slope equation and for the spatial discretization the Ritz-Galerkin finite element method is used. The resulting system of equations is solved with the Krylov subspace method Bi-CGSTAB preconditioned with the incomplete LU decomposition. For large domains, when the number of unknowns is large, the computing time becomes undesirably lengthy. The goal of this thesis is to accelerate HARES.

Research objectives & Outline of the report

The overall objective of this thesis is the acceleration of the model HARES, which is divided into several research objectives.

We start in chapter 2, *The Mild-Slope equation*, with the derivation of the linear and non-linear Mild-Slope equation and the boundary conditions when the considered area is a harbour.

The first research objective is to improve the solver for the non-linear Mild-Slope equation. In the current programme Picard iteration is used and 25 outer iterations are performed. In chapter 3, *Discretization of the non-linear Mild-Slope equation*, we present Picard iteration as used in HARES. The first improvement that we propose regarding the non-linear implementation, is to

[1] http://www.svasek.com/index.html

use a suitable stopping criterion for the non-linear outer loop, to determine when and whether the non-linear convergence is reached. To obtain a faster non-linear convergence we describe Newton's method which usually obtains quadratic convergence. Both Picard iteration and Newton's method result in a sequence of systems of equations, which need to be solved. To accelerate this procedure we present inexact Picard iteration and inexact Newton's method. For inexact Picard iteration, respectively inexact Newton's method, the computing time is reduced by relaxing the condition of solving the obtained system of equations exactly each outer iteration. We need a sequence of forcing terms for this relaxation, therefore we discuss several choices.

In chapter 4, *Spatial discretization of the Mild-Slope equation*, we describe the Ritz-Galerkin finite element method as used in HARES. We distinguish between the case when Picard iteration is applied as the discretization of the non-linear Mild-Slope equation or when Newton's method is used.

We describe in chapter 5, *Solving a system of equations*, the current method Bi-CGSTAB and the preconditioner incomplete LU decomposition. To improve the computing time we present the Krylov subspace method IDR(s) and the shifted Laplace preconditioner.The direct method `MUMPS` is also proposed, since a direct method can give good results for two dimensional problems.

There are elements of the IDR(s) algorithm and the shift in the shifted Laplace preconditioner that can be chosen freely. The second research objectives is how these elements should be chosen for the non-linear Mild-Slope equation. In chapter 6, *Bounds on the eigenvalue range*, we investigate how these elements can be chosen by determining a range on the eigenvalues of the preconditioned system of equations. Using this estimate an optimal shift can be determined for good convergence behaviour. Knowing a bound on the eigenvalue range, we can choose the coefficient ω in the IDR(S) algorithm based on Chebyshev polynomials for a disk in the complex plane.

In chapter 7, *Numerical experiments*, we present the performed experiments on the four test cases provided by Svašek Hydraulics, in order to answer the main research question. We investigate which implementation gives the best results regarding the computing time.

Finally in chapter 8, *Conclusion*, we give the conclusion of this thesis and do some recommendations for further research.

Chapter 2

The Mild-Slope equation

The four main physical phenomena in a harbour or in the neighbourhood of the shore which influence the shape of water waves are diffraction, reflection, refraction and shoaling. Diffraction is the bending and spreading of a wave around an edge of an object and reflection is the return of all or part of the wave energy due to a boundary. Diffraction and reflection occur due to the interaction of the waves with objects in the area and occur simultaneously. Refraction is the change in wave direction and wave length and shoaling is the change in wave height, both caused by a wave moving from deep to shallow water. The equation that combines these four effects is the (linear) Mild-Slope equation, assuming only gentle changes of the ocean bottom. The Mild-Slope equation has been derived by J.C.W. Berkhoff in 1976 in his dissertation at the Technical University of Delft. In section 2.1 we present the linear version of the the Mild-Slope equation, as derived by J.C.W. Berkhoff, and the boundary conditions for a harbour. The effects of bottom friction and wave breaking lead to the non-linear term in the (non-linear) Mild-Slope equation. In section 2.2 we discuss the non-linear Mild-Slope equation, derived by Dingemans (1997), and its boundary conditions.

2.1 The linear Mild-Slope equation

In this section we describe the linear Mild-Slope equation as presented by Berkhoff (1976). Section 2.1.1 contains the derivation of the linear Mild-Slope equation and in section 2.1.2 we discuss the boundary conditions for a harbour.

2.1.1 Derivation of the linear Mild-Slope equation

The derivation of the linear Mild-Slope equation starts with the assumption that it is valid to use the linearised small-amplitude wave equations. To be able to derive the linearised small-amplitude wave equations from the Navier-Stokes equations one has to make the following assumptions (Dingemans (1997)):

- Water is an ideal fluid, i.e., homogeneous, inviscid, irrotational and incompressible flow;
- The pressure at the free surface is constant and uniform;
- The wave slope $\epsilon_s = \frac{2\pi A}{L}$ is small, where A denotes the amplitude of the wave and L the wave length;
- The wave motion is harmonic in time;
- The surface tension and the Coriolis effect can be neglected.

Furthermore we assume that the changes in bottom topography are small or, as put into formula by Mei (1989),

$$\frac{\nabla h}{k_0 h} \ll 1,$$

where ∇h denotes the change in the ocean bottom, $h(x, y)$ the water height and k_0 the wave number. The stationary linearised equations for small-amplitude waves, the starting point for the derivation of the linear Mild-Slope equation, are given by Berkhoff (1976)

$$\frac{\partial^2 \Phi}{\partial x^2} + \frac{\partial^2 \Phi}{\partial y^2} + \frac{\partial^2 \Phi}{\partial z^2} = 0 \qquad ; \quad -h \leq z \leq 0, \tag{2.1.1}$$

$$g\frac{\partial \Phi}{\partial z} - \omega^2 \Phi = 0 \qquad ; \quad z = 0, \tag{2.1.2}$$

$$\frac{\partial \Phi}{\partial z} + \frac{\partial \Phi}{\partial x}\frac{\partial h}{\partial x} + \frac{\partial \Phi}{\partial y}\frac{\partial h}{\partial y} = 0 \qquad ; \quad z = -h. \tag{2.1.3}$$

Where $\Phi(x, y, z)$ denotes the velocity potential in the spatial coordinates x, y and z, g the gravitational acceleration and ω the wave frequency. The ocean bottom is denoted as $z = -h$ and the free surface with $z = 0$. Using the assumption of a gradual change in bottom topography, we can scale the horizontal coordinates x and y with $\sigma/\bar{h}$, i.e.

$$(\bar{x}, \bar{y}) = \frac{\sigma}{\bar{h}}(x, y),$$

with σ the mean slope over a distance D and $\bar{h}$ the average water depth in the domain. For the variation in water depth $\nabla h = (\partial h/\partial x,\ \partial h/\partial y)^T$ this scaling results in

$$\nabla h = \left(\frac{\partial h}{\partial x}, \frac{\partial h}{\partial y}\right)^T = \frac{\sigma}{\bar{h}}\left(\frac{\partial h}{\partial \bar{x}}, \frac{\partial h}{\partial \bar{y}}\right)^T = \frac{\sigma}{\bar{h}}\overline{\nabla} h \quad \text{with} \quad \overline{\nabla} = \left(\frac{\partial}{\partial \bar{x}}, \frac{\partial}{\partial \bar{y}}\right)^T.$$

The variables x, y, z and h are made dimensionless with the free surface characteristic $l_0 = g/\omega^2$, hence

$$(x', y', z', h') = \frac{1}{l_0}(x, y, z, h).$$

Substituting these dimensionless variables into equations (2.1.1) - (2.1.3), while omitting the primes, results in the dimensionless time-independent linearised equations for small-amplitude waves

$$\frac{\partial^2 \Phi}{\partial x^2} + \frac{\partial^2 \Phi}{\partial y^2} + \frac{\partial^2 \Phi}{\partial z^2} = 0 \qquad ; \quad -h \leq z \leq 0, \tag{2.1.4}$$

$$\frac{\partial \Phi}{\partial z} - \Phi = 0 \qquad ; \quad z = 0, \tag{2.1.5}$$

$$\frac{\partial \Phi}{\partial z} + \epsilon\left(\frac{\partial \Phi}{\partial x}\frac{\partial h}{\partial \bar{x}} + \frac{\partial \Phi}{\partial y}\frac{\partial h}{\partial \bar{y}}\right) = 0 \qquad ; \quad z = -h, \tag{2.1.6}$$

with $\epsilon = \sigma l_0/D$. We define the vertical structure $Z(h, z)$ of the wave motion as

$$Z(h, z) = \frac{\cosh(\kappa(h + z))}{\cosh(\kappa h)}. \tag{2.1.7}$$

A common assumption, Dingemans (1997), in the derivation of the Mild-Slope equation is that the vertical structure of the wave motion can be given beforehand by the function $Z(z, h)$, expression (2.1.7). The dependence of $Z(h, z)$ on the horizontal coordinates x and y is only weak through the presence $h(x, y)$. We can write the three-dimensional velocity potential $\Phi(x, y, z)$ as

$$\Phi(x, y, z) = Z(z, h)\varphi(x, y, z) = \frac{\cosh(\kappa(h + z))}{\cosh(\kappa h)}\varphi(x, y, z), \tag{2.1.8}$$

where κ, the dimensionless wave number, can be determined by the dimensionless dispersion relation $1 = \kappa \tanh(\kappa h)$. Substitution of expression (2.1.8) into equations (2.1.4) - (2.1.6) gives

$$Z\nabla^2\varphi + 2\nabla Z \cdot \nabla\varphi + \varphi\nabla^2 Z + \varphi\frac{\partial^2 Z}{\partial z^2} + 2\frac{\partial Z}{\partial z}\frac{\partial \varphi}{\partial z} + Z\frac{\partial^2\varphi}{\partial z^2} = 0 \quad ; \quad -h \le z \le 0, \tag{2.1.9}$$

$$\frac{\partial \varphi}{\partial z} = 0 \quad ; \quad z = 0, \tag{2.1.10}$$

$$Z\frac{\partial\varphi}{\partial z} + \epsilon\left[(\nabla Z \cdot \nabla\bar{h})\varphi + (\nabla\varphi \cdot \nabla\bar{h})Z\right] = 0 \quad ; \quad z = -h. \tag{2.1.11}$$

The derivatives of Z with respect to x and y are given by

$$\nabla Z = \left(\frac{\partial Z}{\partial h}\frac{\partial h}{\partial x}, \frac{\partial Z}{\partial h}\frac{\partial h}{\partial y}\right) = \epsilon\left(\frac{\partial Z}{\partial h}\frac{\partial h}{\partial \bar{x}}, \frac{\partial Z}{\partial h}\frac{\partial h}{\partial \bar{y}}\right) = \epsilon\frac{\partial Z}{\partial h}\overline{\nabla}h.$$

The second derivatives of Z with respect to x, y and z result in

$$\nabla^2 Z = \epsilon^2\left(\frac{\partial^2 Z}{\partial h^2}\overline{\nabla}h \cdot \overline{\nabla}h + \frac{\partial Z}{\partial h}\overline{\nabla}^2 h\right) \quad \text{and} \quad \frac{\partial^2 Z}{\partial z^2} = \kappa^2 Z.$$

Substitution of these expressions into equations (2.1.9) - (2.1.11) gives

$$\epsilon^2\left[\varphi\left(\frac{\partial^2 Z}{\partial h^2}\overline{\nabla}h \cdot \overline{\nabla}h + \frac{\partial Z}{\partial h}\overline{\nabla}^2 h\right)\right] + \epsilon\left[2\frac{\partial Z}{\partial h}\overline{\nabla}h \cdot \nabla\varphi\right]$$
$$+ Z\nabla^2\varphi + \kappa^2\varphi Z + 2\frac{\partial Z}{\partial z}\frac{\partial\varphi}{\partial z} + Z\frac{\partial^2\varphi}{\partial z^2} = 0 \quad ; \quad -h \le z \le 0, \tag{2.1.12}$$

$$\frac{\partial\varphi}{\partial z} = 0 \quad ; \quad z = 0, \tag{2.1.13}$$

$$\epsilon^2\left[\varphi\frac{\partial Z}{\partial h}\overline{\nabla}h \cdot \overline{\nabla}h\right] + \epsilon\left[\nabla\varphi \cdot \overline{\nabla}hZ\right] + Z\frac{\partial\varphi}{\partial z} = 0 \quad ; \quad z = -h. \tag{2.1.14}$$

Multiplying equation (2.1.12) with Z and integrating it over the total water depth, from $z = -h$ until $z = 0$, while using the boundary conditions (2.1.13) and (2.1.14), gives (see Appendix, A.1)

$$\epsilon^2\left[\int_{-h}^{0} z\left(\frac{\partial^2 Z}{\partial h^2}\overline{\nabla}h \cdot \overline{\nabla}h + \frac{\partial Z}{\partial h}\overline{\nabla}^2 h\right)\varphi\, dz\right] + \epsilon\left[\int_{-h}^{0}\frac{\partial Z^2}{\partial h}\overline{\nabla}h \cdot \nabla\varphi\, dz\right]$$

$$+ \int_{-h}^{0} Z^2\left(\nabla^2\varphi + \kappa^2\varphi\right)\, dz + \epsilon\left[Z^2\nabla\varphi \cdot \overline{\nabla}h\right]_{z=-h} + \epsilon^2\left[Z\varphi\frac{\partial Z}{\partial h}\overline{\nabla}h \cdot \overline{\nabla}h\right]_{z=-h} = 0. \tag{2.1.15}$$

The dependency of the velocity potential Φ with respect to the vertical coordinate z has largely been taken into account by the term $Z(h, z)$. This indicates that $\varphi(x, y, z)$ is only weakly depending on z. Therefore we can expand $\varphi(x, y, z)$ in a Taylor series with respect to the coordinate σz. The symmetry boundary condition (2.1.13), states that the expansion does not contain the term σz. This results in the following Taylor series

$$\varphi(x, y, z) = \varphi_0(x, y) + \sigma^2 z^2\varphi_1(x, y) + \sigma^3 z^3\varphi_2(x, y) + \ldots.$$

Substitution of this Taylor series into equation (2.1.15), while noting that φ_0 is independent of z, gives

$$\epsilon^2 \left[\varphi_0 \overline{\nabla} h \cdot \overline{\nabla} h \int_{-h}^{0} Z \frac{\partial^2 Z}{\partial h^2} \, dz + \varphi_0 \overline{\nabla}^2 h \int_{-h}^{0} \frac{1}{2} \frac{\partial Z^2}{\partial h} \, dz \right] + \epsilon \overline{\nabla} h \cdot \nabla \varphi_0 \int_{-h}^{0} \frac{\partial Z^2}{\partial h} \, dz$$

$$+ \left(\nabla^2 \varphi_0 + \kappa^2 \varphi_0 \right) \int_{-h}^{0} Z^2 \, dz + \epsilon \overline{\nabla} h \cdot \nabla \varphi_0 \, Z^2 \big|_{z=-h}$$

$$+ \epsilon^2 \varphi_0 \overline{\nabla} h \cdot \overline{\nabla} h \frac{1}{2} \frac{\partial Z^2}{\partial h} \bigg|_{z=-h} + \mathcal{O}\left(\epsilon^n \sigma^m\right) = 0,$$

with $n \geq 0$ and $m \geq 2$. Using the Leibniz integral rule (see Appendix A.2), the integral $\int_{-h}^{0} \frac{\partial Z^2}{\partial h} \, dz$ can be written as

$$\int_{-h}^{0} \frac{\partial}{\partial h} Z^2 \, dz = \frac{\partial}{\partial h} \int_{-h}^{0} Z^2 \, dz - Z^2 \big|_{z=-h} .$$

Neglecting the $\mathcal{O}(\epsilon^2)$ and $\mathcal{O}(\epsilon^n \sigma^m)$, $n \geq 0, m \geq 2$ terms gives

$$\epsilon \overline{\nabla} h \cdot \nabla \varphi_0 \left[\frac{\partial}{\partial h} \int_{-h}^{0} Z^2 \, dz - Z^2 \big|_{z=-h} \right] + \left(\nabla^2 \varphi_0 + \kappa^2 \varphi_0 \right) \int_{-h}^{0} Z^2 \, dz + \epsilon \overline{\nabla} h \cdot \nabla \varphi_0 \, Z^2 \big|_{z=-h} = 0.$$

Rewriting gives

$$\epsilon \overline{\nabla} h \cdot \nabla \varphi_0 \left[\frac{\partial}{\partial h} \int_{-h}^{0} Z^2 \, dz \right] + \left(\nabla^2 \varphi_0 + \kappa^2 \varphi_0 \right) \int_{-h}^{0} Z^2 \, dz = 0. \tag{2.1.16}$$

After some manipulations (see Appendix, A.1) equation (2.1.16) can be written as

$$\nabla \cdot \int_{-h}^{0} Z^2 \, dz \nabla \varphi_0 + \kappa^2 \varphi_0 \int_{-h}^{0} Z^2 \, dz = 0. \tag{2.1.17}$$

This equation includes the term

$$\int_{-h}^{0} Z^2 \, dz \quad \text{with} \quad Z(z,h) = \frac{\cosh(\kappa(z+h))}{\cosh(\kappa h)}, \tag{2.1.18}$$

which equals $\frac{n_0}{\kappa^2}$ (see Appendix A.3), or $\frac{n_0 \omega^2}{g k_0}$ in dimensional form. The coefficient n_0 denotes the relation between the wave speed $c = \omega / k_0$ and the group velocity c_g, i.e. $c_g = n_0 c$. Returning to the dimensional form and substituting the obtained expression for the integral of (2.1.18) into equation (2.1.17) gives the linear Mild-Slope equation.

$$\boxed{\begin{aligned} &\nabla \cdot \left(\frac{n_0 \omega^2}{g k_0^2} \nabla \varphi_0 \right) + k_0^2 \frac{n_0 \omega^2}{g k_0^2} \varphi_0 = 0, \quad \Longleftrightarrow \\ &\nabla \cdot \left(\frac{n_0}{k_0^2} \nabla \varphi_0 \right) + n_0 \varphi_0 = 0, \quad \Longleftrightarrow \\ &\nabla \cdot (c c_g \nabla \varphi_0) + \omega^2 \frac{c_g}{c} \varphi_0 = 0. \end{aligned}} \tag{2.1.19}$$

Equation (2.1.19) is known as the linear Mild-Slope equation which combines the effects of diffraction-reflection and refraction-shoaling.

2.1.2 Boundary conditions

We cannot solve the linear Mild-Slope equation without the appropriate boundary conditions for the considered domain. In a harbour there are two distinct boundaries, i.e.

- The open boundary (Γ_1), where a prescribed incoming wave from outside the domain and an outgoing wave from the interior are present. The open boundary is not a physical boundary, therefore approaching waves will completely pass it and no reflection occurs.
- The closed boundary (Γ_2), where complete reflection, or partial reflection or no reflection of the wave energy occurs.

The condition for the closed boundary is discussed first and then we present the condition for the open boundary.

Closed boundary
The situation at the closed boundary is sketched in figure 2.1.

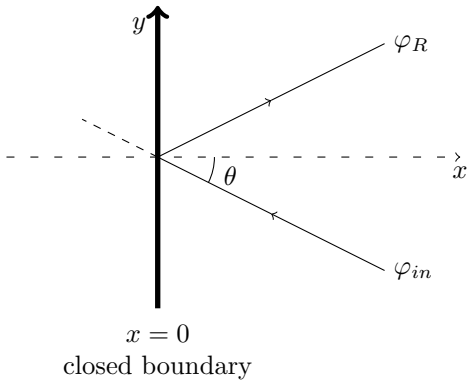


Figure 2.1: *An incoming wave at an angle θ to an element of the closed boundary, where φ_{in} denotes the incoming wave and φ_R the outgoing (reflected) wave.*

Suppose that the closed boundary is located at the axis $x = 0$. At this boundary there is the incoming wave φ_{in} at an angle θ and the outgoing reflected wave φ_R leaving at the same angle θ. The incoming wave is given by $\varphi_{in} = \tilde{\varphi} e^{-ik_x x} e^{-ik_y y}$ and the partially reflected wave by $\varphi_R = R\tilde{\varphi} e^{ik_x x} e^{-ik_y y}$, where $\tilde{\varphi}$ denotes the wave height, R the reflection coefficient ($0 \leq R \leq 1$) and $i = \sqrt{-1}$ the imaginary unit. If $R = 0$ there is no reflection and all the wave energy goes through the boundary, if $R = 1$ full reflection occurs and no wave energy goes through the boundary, and if $0 < R < 1$ partial reflection takes place. The wave which is present at the closed boundary is given by the summation of the waves φ_{in} and φ_R, i.e.

$$\varphi = \varphi_{in} + \varphi_R = \tilde{\varphi} \left(e^{-ik_x x} + R e^{ik_x x} \right) e^{-ik_y y}.$$

The amount of wave potential that crosses the boundary is determined by $\frac{\partial \varphi}{\partial n}$, or in this simplified case $\frac{\partial \varphi}{\partial x}$, which is given by

$$\frac{\partial \varphi}{\partial n} = -ik_x \tilde{\varphi} \left(e^{-ik_x x} - R e^{ik_x x} \right) e^{-ik_y y}.$$

At the closed boundary, where $x = 0$, we have the following two expressions

$$\left. \frac{\partial \varphi}{\partial n} \right|_{x=0} = -ik_x \tilde{\varphi} \left(1 - R\right) e^{-ik_y y} \qquad \text{and} \qquad \varphi|_{x=0} = \tilde{\varphi} \left(1 + R\right) e^{-ik_y y}. \tag{2.1.20}$$

Combining these two expressions gives the boundary condition at the closed boundary

$$\frac{\partial \varphi}{\partial n} = -ik_x \left(\frac{1 - R}{1 + R} \right) \varphi. \tag{2.1.21}$$

The term k_x depends on the angle θ of the incoming wave, i.e. $k_x = k_0 \cos(\theta)$, which is not known a priori. It is possible to approximation the $\cos(\theta)$ term with the first order Taylor series

$$\cos(\theta) = \sqrt{1 - \sin^2(\theta)} \approx 1 - \frac{1}{2}\sin^2(\theta) = 1 + \frac{1}{2}\frac{\partial^2\varphi/\partial s^2}{k_0^2\varphi},$$

where s denotes the direction parallel to the boundary. The expression for the $\sin(\theta)$ term, in the last step, is obtained using $k_y = k_0 \sin(\theta)$ and the second derivative of the right-hand-side of expression (2.1.20) with respect to y. The reflection coefficient R is an unknown parameter depending on many (non-linear) processes. However, its value can be determined experimentally by matching the outcome of the model to the measurements on the domain. These measurements will contain errors, hence the value of R will not be very accurate. Within this accuracy it is assumed that the first order Taylor series of the $\cos(\theta)$ term is sufficient. Substituting this approximation into equation (2.1.21) gives the boundary condition as used in HARES for the closed boundary

$$\frac{\partial\varphi}{\partial n} = -i\left(\frac{1-R}{1+R}\right)\left\{k_0\varphi + \frac{1}{2k_0}\frac{\partial^2\varphi}{\partial s^2}\right\}. \tag{2.1.22}$$

Open boundary
At the open boundary we have the sketched situation of figure 2.2

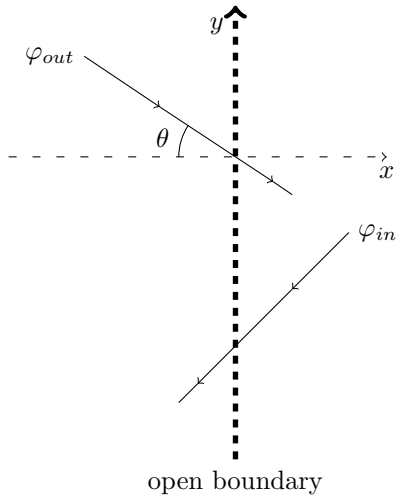


Figure 2.2: *An element of the open boundary with a prescribed incoming wave φ_{in} and an outgoing wave φ_{out} at an angle θ. Since no reflection occurs both the incoming and the outgoing wave cross the boundary without loss of wave energy.*

The condition at the open boundary is determined by a superposition of the influences of the incoming wave φ_{in} and the outgoing wave φ_{out}. The incoming wave φ_{in} is given by $\varphi_{in} = \tilde{\varphi}_{in}e^{-i\boldsymbol{k}\cdot\boldsymbol{x}}$. The contribution of the incoming wave at the boundary is then given by

$$\frac{\partial\varphi_{in}}{\partial n} = -i(\boldsymbol{k}\cdot\boldsymbol{n})\varphi_{in}.$$

The contribution of the outgoing wave is prescribed by equation (2.1.22) with $R = 0$, since no reflection occurs. Therefore we obtain

$$\frac{\partial\varphi_{out}}{\partial n} = -i\left\{k_0\varphi_{out} + \frac{1}{2k_0}\frac{\partial^2\varphi_{out}}{\partial s^2}\right\}.$$

The outgoing wave φ_{out} is equal to the difference between the wave present in the interior φ and the incoming wave φ_{in}, i.e. $\varphi_{out} = \varphi - \varphi_{in}$. Combining the contributions of the incoming and outgoing wave gives the boundary condition for the open boundary

$$\begin{aligned}\frac{\partial\varphi}{\partial n} &= \frac{\partial\varphi_{out}}{\partial n} + \frac{\partial\varphi_{in}}{\partial n} \\ &= -i\left\{k_0\varphi_{out} + \frac{1}{2k_0}\frac{\partial^2\varphi_{out}}{\partial s^2} + k_0(\boldsymbol{e}_{in}\cdot\boldsymbol{n})\varphi_{in}\right\} \\ &= -i\left\{k_0(\varphi-\varphi_{in}) + \frac{1}{2k_0}\left(\frac{\partial^2\varphi}{\partial s^2} - \frac{\partial^2\varphi_{in}}{\partial s^2}\right) + k_0(\boldsymbol{e}_{in}\cdot\boldsymbol{n})\varphi_{in}\right\} \end{aligned} \tag{2.1.23}$$

2.1.3 Summary of the linear Mild-Slope equation

In this section we derived the linear Mild-Slope equation, for convenience the resulting equations are repeated here. The linear Mild-Slope equation is given by

$$\nabla\cdot\left(\frac{n_0}{k_0^2}\nabla\varphi\right) + n_0\varphi = 0. \tag{2.1.24}$$

Where n_0 denotes the relation between the wave speed c and the group velocity c_g, i.e. $c_g = n_0 c$, k_0 the wave number and φ the unknown velocity potential.

At the closed boundary Γ_2 we use the following boundary condition

$$\frac{\partial\varphi}{\partial n} = -i\left(\frac{1-R}{1+R}\right)\left\{k_0\varphi + \frac{1}{2k_0}\frac{\partial^2\varphi}{\partial s^2}\right\}, \tag{2.1.25}$$

with R the reflection coefficient with $0 \le R \le 1$ and $i = \sqrt{-1}$. The boundary condition for the open boundary (Γ_1) is given by

$$\frac{\partial\varphi}{\partial n} = -i\left\{k_0(\varphi-\varphi_{in}) + \frac{1}{2k_0}\left(\frac{\partial^2\varphi}{\partial s^2} - \frac{\partial^2\varphi_{in}}{\partial s^2}\right) + k_0(\boldsymbol{e}_{in}\cdot\boldsymbol{n})\varphi_{in}\right\}, \tag{2.1.26}$$

with φ_{in} a prescribed incoming wave from outside the domain.

2.2 The non-linear Mild-Slope equation

In the derivation of the linear Mild-Slope equation it is assumed that no loss of wave energy occurs. However, this assumption does not hold if more effects, such as bottom friction and wave breaking, are taken into account. Including energy dissipation into the Mild-Slope equation leads to a non-linear term. In section 2.2.1 we discuss the non-linear Mild-Slope equation, in section 2.2.2 the coefficients for the loss of wave energy and finally in section 2.2.3 the boundary conditions for the non-linear Mild-Slope equation.

2.2.1 Derivation of the non-linear Mild-Slope equation

The non-linear Mild-Slope equation, as presented by Dingemans (1997), is given by

$$\nabla\cdot(cc_g\nabla\varphi) + \omega^2\frac{c_g}{c}\varphi - W\frac{\partial\varphi}{\partial t} = 0, \tag{2.2.1}$$

where $W$the dissipation of wave energy, represents the summation of the energy dissipation due to wave breaking and bottom friction. Note the relation between equation (2.1.24) and expression (2.2.1), the term $-W\partial\varphi/\partial t$ is added to equation (2.1.24) to obtain equation (2.2.1). Similarly, the non-liear Mild-Slope equation for the elevation of the free surface $\zeta(x,y,t)$ with $W(x,y,\zeta)$ is obtained by

$$\nabla\cdot(cc_g\nabla\zeta) + \omega^2\frac{c_g}{c}\zeta - W\frac{\partial\zeta}{\partial t} = 0. \tag{2.2.2}$$

The assumption of time harmonic wave motion still holds, hence the expression $\zeta = \mathbb{R}\left(\tilde{\zeta}e^{i\omega t}\right)$ is substituted into equation (2.2.2). This results in the time independent form of the non-linear Mild-Slope equation

$$\nabla \cdot \left(cc_g \nabla\tilde{\zeta}\right) + \omega^2 \frac{c_g}{c}\tilde{\zeta} - i\omega W\tilde{\zeta} = 0, \tag{2.2.3}$$

which can also be written as

$$\nabla \cdot \left(\frac{n_0}{k_0^2}\nabla\tilde{\zeta}\right) + \left(n_0 - \frac{iW}{\omega}\right)\tilde{\zeta} = 0. \tag{2.2.4}$$

Expression for p
Including the dissipation of wave energy does not only influence the Mild-Slope equation but also the expression for the general shape of the wave motion. The general shape of the wave motion for the linear Mild-Slope equation $\tilde{\zeta} = e^{-i\boldsymbol{k}\cdot\boldsymbol{x}}$ changes to $\tilde{\zeta} = e^{-\boldsymbol{p}\cdot\boldsymbol{x}}$ for the non-linear Mild-Slope equation. The coefficient p is denoted as the modified wave number due to the presence of energy dissipation. The expression for p is easily derived using the one dimensional version of equation (2.2.4). The one dimensional form, with constant coefficients, is given by

$$\frac{n_0}{k_0^2}\frac{\partial^2\tilde{\zeta}}{\partial x^2} + \left(n_0 - \frac{iW}{\omega}\right)\tilde{\zeta} = 0 \quad \Longleftrightarrow \quad \frac{\partial^2\tilde{\zeta}}{\partial x^2} + k_0^2\left(1 - \frac{iW}{\omega n_0}\right)\tilde{\zeta} = 0.$$

Substitution of the general solution $\tilde{\zeta} = \tilde{\zeta}_0 e^{px}$ into this equation results in the following expression for p

$$p = \pm ik_0\sqrt{1 - \frac{iW}{\omega n_0}}.$$

2.2.2 Dissipation of wave energy

Since shallow-water waves interact with the bottom, energy losses due to bottom friction will be present. The bottom topography is assumed to have a decreasing water depth towards the shore, which causes an increasing wave height. If no energy dissipation is present waves will become infinitely high. However, in reality this is not possible and waves will break after reaching a maximal height. Both effects lead to dissipation of wave energy. First we derive the expression for the amount of energy dissipated due to bottom friction, then the expression for the amount of energy dissipated due to wave breaking is discussed.

Bottom friction coefficient W_f

For the derivation of the bottom friction coefficient W_f we follow the approach as described by Visser (1984) and Dingemans (1997). The coefficient W_f is determined by dividing the dissipation of wave energy due to bottom friction $\langle D_f \rangle$ by the wave energy per unit length $\bar{E}$, i.e.

$$W_f = \frac{\langle D_f \rangle}{\bar{E}}.$$

The wave energy per unit length is given by

$$\bar{E} = \frac{\rho g H^2}{8},$$

with ρ the water density and $H = 2|\zeta(x, y, t)|$. We can determine the dissipation of wave energy in the bottom boundary layer with the following expression

$$\langle D_f \rangle = \left\langle \tau^b \cdot \boldsymbol{u}^b \right\rangle \quad \text{with} \quad \tau^b = c_f \rho \left|\boldsymbol{u}^b\right| \boldsymbol{u}^b, \tag{2.2.5}$$

where τ^b denotes the instantaneous stress exited by the water on the bottom, $\langle \cdot \rangle$ the mean over one wave period, c_f the bottom friction coefficient and $\boldsymbol{u}^b$ the horizontal velocity near the bottom. Visser (1984) states that linear wave theory for the horizontal velocity $\boldsymbol{u}^b$ can be used, which yields that $\boldsymbol{u}^b$ can be expressed as

$$\boldsymbol{u}^b = \frac{a\omega}{\sinh(k_0 h)} \cos(\omega t),$$

with $a(x, y, t) = |\zeta(x, y, t)|$. Substituting this expression for $\boldsymbol{u}_b$ into equation (2.2.5) gives

$$\langle D_f \rangle = \left\langle c_f \rho \left| \frac{a\omega \cos(\omega t)}{\sinh(k_0 h)} \right| \left(\frac{a\omega \cos(\omega t)}{\sinh(k_0 h)} \right)^2 \right\rangle = c_f \rho \left(\frac{a\omega}{\sinh(k_0 h)} \right)^3 \left\langle |\cos(\omega t)|^3 \right\rangle.$$

With $\left\langle |\cos(\omega t)|^3 \right\rangle = \frac{4}{3\pi}$ this results in

$$\langle D_f \rangle = \frac{4}{3\pi} c_f \rho \frac{a^3 \omega^3}{\sinh^3(k_0 h)}.$$

Hence we obtain the following expression for the bottom friction coefficient W_f

$$W_f = \frac{\langle D_f \rangle}{\bar{E}} = \frac{8}{3\pi} c_f \frac{a\omega^3}{\sinh^3(kh)}.$$

Wave breaking coefficient W_b

To determine an expression for the wave breaking coefficient W_b we follow the derivation of Battjes and Janssen (1978). Similarly as for the bottom friction coefficient, we obtain the wave breaking coefficient W_b by dividing the wave energy dissipation due to wave breaking D_b by the wave energy per unit length $\bar{E}$. This results in

$$W_b = \frac{D_b}{\bar{E}}.$$

The characteristic breaker height, as described by Southgate (1993), is defined as

$$\gamma_s = \left(\frac{H}{h} \right)_{\text{b,shallow}} \qquad \text{and} \qquad \gamma_d = \left(\frac{H}{L} \right)_{\text{b,deep}},$$

with H the wave height, h the water depth and L the wave length. Shallow-water waves break due to limitations on the water depth while for deep-water waves the steepness of the wave is crucial. The maximal value for γ_d is by default 0.14. The well-known Miche's criterion, Battjes and Janssen (1978), for the maximal possible wave height H_m is given by

$$H_m = \frac{2\pi\gamma_d}{k} \tanh\left(\frac{\gamma_s}{2\pi\gamma_d} k_0 h \right) = \frac{0.88}{k} \tanh\left(\frac{\gamma_s}{0.88} k_0 h \right).$$

The probability that at a given point a height is associated with a breaking or broken wave is specified by Q_b and can be determined from

$$\frac{1 - Q_b}{\ln Q_b} = -\left(\frac{H_{rms}}{H_m} \right)^2,$$

where H_{rms} denotes the root mean square wave height. The power dissipation in the bore of the wave per unit span is given by

$$D_b' = \frac{1}{4} \rho g (h_2 - h_1)^3 \sqrt{\frac{g(h_1 + h_2)}{2h_1 h_2}},$$

with h_1 and h_2 the water heights at both sides of the bore. We make the following estimates

$$h_2 - h_1 \approx H \quad \text{and} \quad \sqrt{\frac{g(h_1+h_2)}{2h_1h_2}} \approx \sqrt{\frac{g}{h}}.$$

Which results in

$$D_b' \sim \frac{1}{4}\rho g H^3 \sqrt{\frac{g}{h}}.$$

If the waves are periodic with frequency $f = \frac{1}{T}$ and wave period T, the average power dissipation per unit length for shallow-water waves in the breaking process is given by

$$D_b = \frac{D_b'}{L} = \frac{fD_b'}{c} \sim \frac{fD_b'}{\sqrt{gh}} = \frac{1}{4} f\rho g \frac{H^3}{h}. \tag{2.2.6}$$

For shallow water the ratio H/h is approximately 1, hence $D_b \sim \frac{1}{4} f\rho g H^2$. For random waves the expected value of the dissipated power per unit area is of interest. Hence we need to multiply equation (2.2.6) by Q_b and H set to H_m. This results in

$$D_b = \frac{\alpha}{4} f\rho g Q_b H_m^2.$$

The wave breaking coefficient W_b is then determined by

$$W_b = \frac{D_b}{\bar{E}} = \frac{2\alpha}{T} Q_b \frac{H_m^2}{H_{rms}^2} = \frac{2\alpha}{T} Q_b \frac{H_m^2}{4a^2}.$$

2.2.3 Boundary conditions

The boundary conditions for the non-linear Mild-Slope equation are very similar to the boundary conditions of the linear Mild-Slope equation. The only difference is that the incoming wave is now prescribed by $\varphi_{in} = \tilde{\varphi} e^{-i\hat{p}_x x} e^{-i\hat{p}_y y}$ and the complete derivation can be repeated using this form of the incoming wave. For the closed boundary we obtain the following boundary condition

$$\frac{\partial \tilde{\zeta}}{\partial n} = -i \left(\frac{1-R}{1+R} \right) \left\{ \hat{p}\tilde{\zeta} + \frac{1}{2\hat{p}} \frac{\partial^2 \tilde{\zeta}}{\partial s^2} \right\}, \tag{2.2.7}$$

and for the open boundary this results in

$$\frac{\partial \tilde{\zeta}}{\partial n} = -i \left\{ \hat{p}(\tilde{\zeta} - \tilde{\zeta}_{in}) + \frac{1}{2\hat{p}} \left(\frac{\partial^2 \tilde{\zeta}}{\partial s^2} - \frac{\partial^2 \tilde{\zeta}_{in}}{\partial s^2} \right) + \hat{p}(\boldsymbol{e}_{in} \cdot \boldsymbol{n}) \tilde{\zeta}_{in} \right\}, \tag{2.2.8}$$

with $\hat{p} = k_0 \sqrt{1 - \frac{iW}{\omega n_0}}$ the modified wave number.

2.2.4 Summary of the non-linear Mild-Slope equation

In this section we described the non-linear Mild-Slope equation as presented by Dingemans (1997). In the non-linear Mild-Slope equation not only the effects of diffraction-reflection and refraction-shoaling are included, but also energy dissipation caused by wave breaking and bottom friction is present. Including energy dissipation leads to a non-linear term in the Mild-Slope equation. The non-linear Mild-Slope is given by equation (2.2.4)

$$\nabla \cdot \left(\frac{n_0}{k_0^2} \nabla \tilde{\zeta} \right) + \left(n_0 - \frac{iW}{\omega} \right) \tilde{\zeta} = 0, \tag{2.2.9}$$

with n_0 the relation between the wave speed c and the group velocity c_g, i.e. $c_g = n_0 c$, k_0 the wave number, W the coefficient for the energy dissipation, ω the wave frequency, $i = \sqrt{-1}$ the

imaginary unit and $\tilde{\zeta}(x, y)$ the unknown elevation of the free surface. The energy source $W(x, y, \tilde{\zeta})$ is given by

$$W(\tilde{\zeta}) = W_f + W_b = \frac{8}{3\pi} c_f \frac{|\tilde{\zeta}|\omega^3}{\sinh^3(k_0 h)} + \frac{2\alpha}{T} Q_b \frac{H_m^2}{4|\tilde{\zeta}|^2}. \tag{2.2.10}$$

Equation 2.2.9 is a non-linear equation since the energy dissipation coefficient W depends on the elevation of the free surface $\tilde{\zeta}$.

The condition for the open boundary (Γ_1) with an incoming wave is given by

$$\frac{\partial \tilde{\zeta}}{\partial n} = -i \left\{ \hat{p}(\tilde{\zeta} - \tilde{\zeta}_{in}) + \frac{1}{2\hat{p}} \left(\frac{\partial^2 \tilde{\zeta}}{\partial s^2} - \frac{\partial^2 \tilde{\zeta}_{in}}{\partial s^2} \right) + \hat{p}(\boldsymbol{e}_{in} \cdot \boldsymbol{n}) \tilde{\zeta}_{in} \right\}, \tag{2.2.11}$$

and for the closed boundary (Γ_2) we obtain the boundary condition

$$\frac{\partial \tilde{\zeta}}{\partial n} = -i \left(\frac{1 - R}{1 + R} \right) \left\{ \hat{p}\tilde{\zeta} + \frac{1}{2\hat{p}} \frac{\partial^2 \tilde{\zeta}}{\partial s^2} \right\}. \tag{2.2.12}$$

With the modified wave number $\hat{p}$

$$\hat{p} = k_0 \sqrt{1 - \frac{iW}{\omega n_0}}. \tag{2.2.13}$$

Chapter 3

Discretization of the non-linear Mild-Slope equation

The first step in the discretization of the non-linear Mild-Slope equation is to determine a suitable method to deal with the non-linearity of the Mild-Slope equation. In this chapter we discuss two methods to discretize a non-linear equation. The two most frequently used methods are Picard iteration and Newton's method. In the current version of HARES Picard iteration is implemented. In section 3.1 we discuss Picard iteration and in section 3.2 Newton's method. The implementation of Picard iteration is usually much easier than Newton's method, however, Picard iteration converges linearly while Newton's method can obtain quadratic convergence.

3.1 Picard iteration

Picard iteration, as implemented in HARES, uses the previous solution $\tilde{\zeta}^{k-1}$ to compute a value for the energy dissipation term $W(x, y, \tilde{\zeta})$. The next iterate $\tilde{\zeta}^k$ is then determined with the following set of equations for the non-linear Mild-Slope equation

$$\nabla \cdot \left(\frac{n_0}{k_0^2} \nabla \tilde{\zeta}^k \right) + \left(n_0 - \frac{iW(\tilde{\zeta}^{k-1})}{\omega} \right) \tilde{\zeta}^k = 0 \qquad \text{on } \Omega, \tag{3.1.1}$$

$$\frac{\partial \tilde{\zeta}^k}{\partial n} = -i \left\{ \hat{p}(\tilde{\zeta}^k - \tilde{\zeta}_{in}) + \frac{1}{2\hat{p}} \left(\frac{\partial^2 \tilde{\zeta}^k}{\partial s^2} - \frac{\partial^2 \tilde{\zeta}_{in}}{\partial s^2} \right) + \hat{p}(\boldsymbol{e}_{in} \cdot \boldsymbol{n}) \tilde{\zeta}_{in} \right\} \qquad \text{on } \Gamma_1, \tag{3.1.2}$$

$$\frac{\partial \tilde{\zeta}^k}{\partial n} = -i \left(\frac{1-R}{1+R} \right) \left\{ \hat{p} \tilde{\zeta}^k + \frac{1}{2\hat{p}} \frac{\partial^2 \tilde{\zeta}^k}{\partial s^2} \right\} \qquad \text{on } \Gamma_2. \tag{3.1.3}$$

Spatial discretization of equation (3.1.1), with corresponding boundary conditions (3.1.2) and (3.1.3), results after discretization in a system of linear equations $\boldsymbol{S}\boldsymbol{\zeta}^k = \boldsymbol{b}$ (see section 4.1). The matrix $\boldsymbol{S}$ depends on the previous solution $\boldsymbol{\zeta}^{k-1}$. In the first iteration of HARES the non-linear contribution of $W(\boldsymbol{\zeta}^0)$ is set equal to zero. However, the initial guess could also be used to determine a value for $W(\boldsymbol{\zeta}^0)$. The algorithm for Picard iteration is given by algorithm 3.1

Picard iteration

1. Determine a value for $W(\boldsymbol{\zeta}^0)$
2. **for** $k = 1, 2, \ldots$ until convergence **do**
3. Solve $\boldsymbol{\zeta}^k$ from $\boldsymbol{S}(\boldsymbol{\zeta}^{k-1})\boldsymbol{\zeta}^k = \boldsymbol{b}$
4. $k = k + 1$
5. **end**

Alg. 3.1: *Picard iteration*

The weak formulation of the system of equations (3.1.1) - (3.1.3) is written as a function $F(\tilde{\zeta}) = 0$, see section 3.5. Using Picard iteration, the discretization of the weak formulation results in a system of equations $\boldsymbol{S}(\boldsymbol{\zeta}^{k-1})\boldsymbol{\zeta}^k = \boldsymbol{b}$. Therefore, the non-linear residual of Picard iteration is given by

$$F(\boldsymbol{\zeta}^k) = \boldsymbol{S}(\boldsymbol{\zeta}^k)\boldsymbol{\zeta}^k - \boldsymbol{b}.$$

The non-linear residual is needed to determine whether the method converges to the solution of the non-linear equation. Using Picard iteration we solve in each iteration the system of equations $\boldsymbol{S}\boldsymbol{\zeta}^k = \boldsymbol{b}$ exactly. However, this is not always necessary, especially in the first couple of iterations when the iterative solution is far away from the desired non-linear solution. Hence we want to relax the condition of finding for each iteration the exact solution with the following stopping criterion

$$\|\boldsymbol{S}(\boldsymbol{\zeta}^{k-1})\boldsymbol{\zeta}^k - \boldsymbol{b}\| \leq \eta_k \|\boldsymbol{b}\|,$$

where $\eta_k \in [0, 1)$ is denoted as the forcing term. The obtained method, when using such a stopping criterion, is denoted as inexact Picard iteration. The forcing term should have the following properties: Far away from the solution, i.e. $\boldsymbol{S}(\boldsymbol{\zeta}^k)\boldsymbol{\zeta}^k - b \gg 0$, or when the change in solution is large, i.e. $\|\boldsymbol{\zeta}^k - \boldsymbol{\zeta}^{k-1}\| \gg 0$, η_k should be large and close to the solution η_k should be small. The algorithm for inexact Picard iteration is given by Alg. 3.2.

Inexact Picard iteration

1. Determine a value for $W(\boldsymbol{\zeta}^0)$
2. **for** $k = 1, 2, \ldots$ until convergence **do**
3. Find some $\eta_k \in [0, 1)$ and $\boldsymbol{\zeta}^k$ such that
4. $\|\boldsymbol{S}(\boldsymbol{\zeta}^{k-1})\boldsymbol{\zeta}^k - \boldsymbol{b}\| \leq \eta_k \|\boldsymbol{b}\|$
5. $k = k + 1$
6. **end**

Alg. 3.2: *Inexact Picard iteration*

There are many ways of choosing the forcing sequence η_k, in section 3.3 some examples are discussed.

3.2 Newton's method

Application of Newton's method to the equation $F(\tilde{\zeta}) = 0$ is based on its first order Taylor expansion. The Taylor series of $F(\tilde{\zeta} + \epsilon)$ around $\tilde{\zeta} + \epsilon$ is given by

$$F(\tilde{\zeta} + \epsilon) = F(\tilde{\zeta}) + F'(\tilde{\zeta})\epsilon + \text{ higher order terms.} \tag{3.2.1}$$

Keeping the first order terms and setting $F(\tilde{\zeta}+\epsilon)$ equal to zero gives

$$F(\tilde{\zeta}) + F'(\tilde{\zeta})\epsilon = 0.$$

Now let ϵ denote the difference between two successive solutions, i.e. $\epsilon = \tilde{\zeta}^{k+1} - \tilde{\zeta}^{k}$. We use the known iterate $\tilde{\zeta}^k$ to obtain a value for $F(\tilde{\zeta})$ and $F'(\tilde{\zeta})$, hence Newton's method is given by

$$F(\tilde{\zeta}^k) + F'(\tilde{\zeta}^k)(\tilde{\zeta}^{k+1} - \tilde{\zeta}^k) = 0. \tag{3.2.2}$$

The non-linear residual of Newton's method is determined by $F(\tilde{\zeta}^k)$. The spatial discretization of equation (3.2.2) results in a system of equations. The discretized solution $\boldsymbol{\zeta}^{k+1}$ can be obtained by first solving the system

$$F'(\boldsymbol{\zeta}^k)\delta\boldsymbol{\zeta}^k = -F(\boldsymbol{\zeta}^k), \tag{3.2.3}$$

with $\delta\boldsymbol{\zeta}^k = \boldsymbol{\zeta}^{k+1} - \boldsymbol{\zeta}^k$ and then performing the computation $\boldsymbol{\zeta}^{k+1} = \boldsymbol{\zeta}^k + \delta\boldsymbol{\zeta}^k$. Equation (3.2.3) can be solved exactly at each step, but as in the case of Picard iteration, the first couple of iterations this condition can be relaxed. The approximated Newton step $\delta\boldsymbol{\zeta}^k$ has to satisfy the inexact Newton criterion

$$\|F(\boldsymbol{\zeta}^k) + F'(\boldsymbol{\zeta}^k)\delta\boldsymbol{\zeta}^k\| \leq \eta_k \|F(\boldsymbol{\zeta}^k)\|, \tag{3.2.4}$$

where η_k is denoted as the forcing term (see section 3.3) which satisfies $\eta_k \in [0, 1)$. The inexact Newton's method is given by algorithm 3.3, see Dembo et al. (1982).

Inexact Newton's method

1. Let $\boldsymbol{\zeta}^0$ be given.
2. **for** $k = 1, 2, \ldots$ until convergence **do**
3. Find some $\eta_k \in [0, 1)$ and $\delta\boldsymbol{\zeta}^k$ that satisfy
4. $$\left\| F(\boldsymbol{\zeta}^k) + F'(\boldsymbol{\zeta}^k)\delta\boldsymbol{\zeta}^k \right\| \leq \eta_k \left\| F(\boldsymbol{\zeta}^k) \right\|$$
5. Set $\boldsymbol{\zeta}^{k+1} = \boldsymbol{\zeta}^k + \delta\boldsymbol{\zeta}^k$.
6. **end**

Alg. 3.3: *Inexact Newton's method*

3.3 Choosing the forcing sequence η_k

A lot of research has been done for a good choice of the forcing sequence η_k in inexact Picard iteration and inexact Newton's method. However, no optimal forcing term is known which gives the best results in every situation. Therefore we present five choices and implement these to test which one gives the fastest computing time to achieve the desired convergence for the implementation of the non-linear Mild-Slope equation.

An important property of the forcing term is to prevent oversolving, i.e. imposing an accurate linear solution to an inaccurate non-linear correction. Hence it may take a considerable amount of computing time to obtain the solution of the linear system, but the improvement in the non-linear residual is small.

Choice 1
The first forcing term, for Newton's method, is presented by Eisenstat and Walker (1996).

Choose $\eta_0 \in [0,1)$, the next forcing term is then given by

$$\eta_k = \frac{\left| \|F(\boldsymbol{\zeta}^k)\| - \|F(\boldsymbol{\zeta}^{k-1}) + F'(\boldsymbol{\zeta}^{k-1})\delta\boldsymbol{\zeta}^{k-1}\| \right|}{\|F(\boldsymbol{\zeta}^{k-1})\|}, \qquad k = 1, 2, \ldots. \tag{3.3.1}$$

This forcing term represents the relative difference between the non-linear residual $F(\boldsymbol{\zeta}^k)$ and its value based on the local linear model at the previous step. Making the translation from the Newton's method to Picard iteration gives the following expression for forcing term 1.
Choose $\eta_0 \in [0,1)$, the next forcing term is then given by

$$\eta_k = \frac{\left| \|F(\boldsymbol{\zeta}^k)\| - \|\boldsymbol{r}^{k-1}\| \right|}{\|F(\boldsymbol{\zeta}^{k-1})\|}, \qquad k = 1, 2, \ldots, \tag{3.3.2}$$

with $\boldsymbol{r}^{k-1} = \boldsymbol{S}(\boldsymbol{\zeta}^{k-1})\boldsymbol{\zeta}^k - \boldsymbol{b}$.

Choice 2
The second choice for the forcing sequence η_k is again presented by Eisenstat and Walker (1996). Choose $\eta_0 \in [0,1)$, the forcing sequence is obtained by

$$\eta_k = 0.5 \left(\frac{\|F(\boldsymbol{\zeta}^k)\|}{\|F(\boldsymbol{\zeta}^{k-1})\|} \right)^2, \qquad k = 1, 2, \ldots. \tag{3.3.3}$$

This forcing term represents the relative change in the non-linear residual within two iterations.

Choice 3
The third choice is presented by Dembo et al. (1982)

$$\eta_k = \min \left(\frac{1}{k+2}, \|F(\boldsymbol{\zeta}^k)\| \right), \qquad k = 1, 2, \ldots. \tag{3.3.4}$$

Using this forcing term, each iteration the system of equations is solved more accurately, due to the presence of the term $\frac{1}{k+2}$. In the first couple of iterations, when the non-linear residual $\|F(\tilde{\boldsymbol{\zeta}}^k)\|$ is large, the quotient $\frac{1}{k+2}$ will determine the value of the forcing term η_k, but when a significant drop of the non-linear residual is obtained $\|F(\tilde{\boldsymbol{\zeta}}^k)\|$ will determine the forcing term.

Choice 4
The fourth choice is presented in the work of Brown and Saad (1990), they propose the following forcing sequence η_k

$$\eta_k = \frac{1}{2^{k+1}}, \qquad k = 1, 2, \ldots. \tag{3.3.5}$$

This forcing sequence has no information about the non-linear residual incorporated and only depends on the number of outer iterations. Hence for a very slow convergence of the non-linear residual this may lead to oversolving.

Choice 5
Choose $\eta_0 \in [0,1)$, the fifth forcing sequence is given by

$$\eta_k = \frac{\|\boldsymbol{\zeta}^k - \boldsymbol{\zeta}^{k-1}\|}{\|\boldsymbol{\zeta}^0\|} \cdot \mathrm{TOL}, \qquad k = 1, 2, \ldots, \tag{3.3.6}$$

with $\mathrm{TOL} \in [0,1)$ and usually small. This forcing term has no information about the non-linear residual incorporated, but unlike all the other forcing terms depends only on the difference between two successive iterative solutions.

3.4 Stopping criterion for the non-linear loop

The current version of HARES does not have a stopping criterion for the non-linear outer loop implemented. The programme performs 25 outer iterations, without knowing whether the iterations converge or not. Therefore it is necessary to implement a suitable stop criterion, such that when convergence is reached the calculations are stopped. Two commonly used stopping criteria, see Knoll and Keyes (2004), are

$$\frac{\|F(\boldsymbol{\zeta}^k)\|}{\|F(\boldsymbol{\zeta}^0)\|} \leq \text{TOL}_{\text{residual}} \qquad \text{and} \qquad \frac{\|\boldsymbol{\zeta}^{k+1} - \boldsymbol{\zeta}^k\|}{\|\boldsymbol{\zeta}^k\|} \leq \text{TOL}_{\text{update}}$$

The first stopping criterion is based on a required drop in the norm of the non-linear residual and the second stopping criterion on a sufficiently small change between two successive solutions.

It can occur that for consecutive iterations the difference between the solutions is small, while the non-linear residual $F(\boldsymbol{\zeta}^k)$ is not close to zero yet. This means that if the second stopping criterion is used the outer iteration is terminated while the desired solution is not reached. Therefore we incorporate the first stopping criterion into HARES.

3.5 Weak formulation of the non-linear Mild-Slope equation

To be able to apply Newton's method and the Ritz-Galerkin finite element method to the non-linear Mild-Slope equation, its weak formulation is needed. In this section we determine the weak formulation of the non-linear Mild-Slope equation.

The weak formulation of the non-linear Mild-Slope equation is obtained by multiplying equation (2.2.9) by a test function $\eta(x, y)$ and integrating equation over the domain Ω

$$\int_\Omega \left\{ \nabla \cdot \left(\frac{n_0}{k_0^2} \nabla \tilde{\zeta} \right) + \left(n_0 - \frac{iW}{\omega} \right) \tilde{\zeta} \right\} \eta \; d\Omega = 0 \quad \forall \eta.$$

Application of Gauss divergence theorem results in

$$- \int_\Omega \frac{n_0}{k_0^2} \nabla \tilde{\zeta} \cdot \nabla \eta \; d\Omega + \int_\Omega \left(n_0 - \frac{iW}{\omega} \right) \tilde{\zeta} \eta \; d\Omega + \int_\Gamma \frac{n_0}{k_0^2} \eta \frac{\partial \tilde{\zeta}}{\partial n} \; d\Gamma = 0 \quad \forall \eta, \tag{3.5.1}$$

where the boundary of the domain Ω is denoted by Γ. The boundary Γ consist of a part Γ_1, for the open boundary, and a part Γ_2, for closed boundary. Substitution of the boundary conditions (2.2.11) and (2.2.12) into equation (3.5.1) gives

$$\int_\Omega \left\{ \left(n_0 - \frac{iW}{\omega} \right) \tilde{\zeta} \eta - \frac{n_0}{k_0^2} \nabla \tilde{\zeta} \cdot \nabla \eta \right\} \; d\Omega - i \int_{\Gamma_2} \frac{n_0}{k_0^2} \left(\frac{1 - R}{1 + R} \right) \left\{ \hat{p} \tilde{\zeta} + \frac{1}{2\hat{p}} \frac{\partial^2 \tilde{\zeta}}{\partial s^2} \right\} \eta \; d\Gamma$$

$$- i \int_{\Gamma_1} \left\{ \frac{n_0}{k_0^2} \left(\hat{p} \tilde{\zeta}_{in} (\boldsymbol{e}_{in} \cdot \boldsymbol{n}) + \hat{p} (\tilde{\zeta} - \tilde{\zeta}_{in}) + \frac{1}{2\hat{p}} \left(\frac{\partial^2 \tilde{\zeta}}{\partial s^2} - \frac{\partial^2 \tilde{\zeta}_{in}}{\partial s^2} \right) \right) \right\} \eta \; d\Gamma = 0 \quad \forall \eta. \tag{3.5.2}$$

Both boundary conditions contain the term $\frac{1}{2\hat{p}} \frac{\partial^2 \tilde{\zeta}}{\partial s^2} \eta$, integration of this expression by parts gives

$$\frac{1}{2\hat{p}} \int_{\Gamma_i} \frac{\partial^2 \tilde{\zeta}}{\partial s^2} \eta \; d\Gamma = \frac{1}{2\hat{p}} \left\{ \eta \frac{\partial \tilde{\zeta}}{\partial s} \bigg|_{\Gamma_i} - \int_{\Gamma_i} \frac{\partial \eta}{\partial s} \frac{\partial \tilde{\zeta}}{\partial s} \; d\Gamma \right\}.$$

Where it is assumed that

$$\frac{1}{2\hat{p}}\left\{\eta\frac{\partial\tilde{\zeta}}{\partial s}\bigg|_{\Gamma_1}+\eta\frac{\partial\tilde{\zeta}}{\partial s}\bigg|_{\Gamma_2}\right\}=0$$

since $\Gamma=\Gamma_1+\Gamma_2$ is a closed curve. The term $\frac{1}{2\hat{p}}\frac{\partial^2\tilde{\zeta}_{in}}{\partial s^2}$ is only present at the open boundary, integration by parts leads to the term $\frac{1}{2\hat{p}}\eta\frac{\partial\tilde{\zeta}_{in}}{\partial s}\Big|_{\Gamma_1}$. In the implementation in HARES it is assumed that this term equals zero, since it only gives a contribution at the two ends of the open boundary. Applying integration by parts to equation (3.5.2) gives the weak formulation of the non-linear Mild-Slope equation

$$\int_\Omega\left\{\left(n_0-\frac{iW}{\omega}\right)\tilde{\zeta}\eta-\frac{n_0}{k_0^2}\nabla\tilde{\zeta}\cdot\nabla\eta\right\}\,d\Omega-i\int_{\Gamma_2}\frac{n_0}{k_0^2}\left(\frac{1-R}{1+R}\right)\left\{\hat{p}\tilde{\zeta}\eta-\frac{1}{2\hat{p}}\frac{\partial\tilde{\zeta}}{\partial s}\frac{\partial\eta}{\partial s}\right\}\,d\Gamma$$

$$-i\int_{\Gamma_1}\frac{n_0}{k_0^2}\left\{\hat{p}\tilde{\zeta}_{in}\left(\boldsymbol{e}_{in}\cdot\boldsymbol{n}\right)\eta+\hat{p}(\tilde{\zeta}-\tilde{\zeta}_{in})\eta-\frac{1}{2\hat{p}}\left(\frac{\partial\tilde{\zeta}}{\partial s}\frac{\partial\eta}{\partial s}-\frac{\partial\tilde{\zeta}_{in}}{\partial s}\frac{\partial\eta}{\partial s}\right)\right\}\,d\Gamma=0. \tag{3.5.3}$$

If Picard iteration is used to discretize the non-linear Mild-Slope equation, equation (3.1.1) is multiplied by a test function $\eta(x,y)$ and integrated over the domain Ω. The computations as described in this section can be repeated, using the boundary conditions (3.1.2) and (3.1.3), to obtain the weak formulation for Picard iteration.

3.6 Newton's method for the non-linear Mild-Slope equation

The weak formulation, expression (3.5.3), of the non-linear Mild-Slope equation is of the form $F(\tilde{\zeta})=0$, hence Newton's method can be applied to it. Newton's method is given by

$$F'(\tilde{\zeta}^k)\delta\zeta^k=-F(\tilde{\zeta}^k), \tag{3.6.1}$$

with $\delta\zeta^k=\tilde{\zeta}^{k+1}-\tilde{\zeta}^k$. We approximate the term $F'(\tilde{\zeta}^k)\delta\zeta^k$ by

$$F'(\tilde{\zeta}^k)\delta\zeta^k=\lim_{\epsilon\to0}\frac{F(\tilde{\zeta^k}+\epsilon\delta\zeta^k)-F(\tilde{\zeta^k})}{\epsilon}. \tag{3.6.2}$$

Since $F(\tilde{\zeta}^k)=F(\tilde{\zeta}^k,\tilde{\zeta}_x^k,\tilde{\zeta}_y^k,\tilde{\zeta}_s^k)$, equation (3.6.2) can be written as (while omitting the superscript k),

$$F'(\tilde{\zeta})\delta\zeta=\lim_{\epsilon\to0}\frac{F\left(\tilde{\zeta}+\epsilon\delta\zeta,\tilde{\zeta}_x+\epsilon(\delta\zeta)_x,\tilde{\zeta}_y+\epsilon(\delta\zeta)_y,\tilde{\zeta}_s+\epsilon(\delta\zeta)_s\right)-F\left(\tilde{\zeta},\tilde{\zeta}_x,\tilde{\zeta}_y,\tilde{\zeta}_s\right)}{\epsilon}. \tag{3.6.3}$$

With the following manipulation, subtracting and adding the same terms, the right-hand-side of equation (3.6.3) can be written as

$$\lim_{\epsilon\to0}\frac{1}{\epsilon}\Big\{F\left(\tilde{\zeta}+\epsilon\delta\zeta,\tilde{\zeta}_x+\epsilon(\delta\zeta)_x,\tilde{\zeta}_y+\epsilon(\delta\zeta)_y,\tilde{\zeta}_s+\epsilon(\delta\zeta)_s\right)-F\left(\tilde{\zeta},\tilde{\zeta}_x+\epsilon(\delta\zeta)_x,\tilde{\zeta}_y+\epsilon(\delta\zeta)_y,\tilde{\zeta}_s+\epsilon(\delta\zeta)_s\right)$$

$$+F\left(\tilde{\zeta},\tilde{\zeta}_x+\epsilon(\delta\zeta)_x,\tilde{\zeta}_y+\epsilon(\delta\zeta)_y,\tilde{\zeta}_s+\epsilon(\delta\zeta)_s\right)-F\left(\tilde{\zeta},\tilde{\zeta}_x,\tilde{\zeta}_y+\epsilon(\delta\zeta)_y,\tilde{\zeta}_s+\epsilon(\delta\zeta)_s\right)$$

$$+F\left(\tilde{\zeta},\tilde{\zeta}_x,\tilde{\zeta}_y+\epsilon(\delta\zeta)_y,\tilde{\zeta}_s+\epsilon(\delta\zeta)_s\right)-F\left(\tilde{\zeta},\tilde{\zeta}_x,\tilde{\zeta}_y,\tilde{\zeta}_s+\epsilon(\delta\zeta)_s\right)$$

$$+F\left(\tilde{\zeta},\tilde{\zeta}_x,\tilde{\zeta}_y,\tilde{\zeta}_s+\epsilon(\delta\zeta)_s\right)-F\left(\tilde{\zeta},\tilde{\zeta}_x,\tilde{\zeta}_y,\tilde{\zeta}_s\right)\Big\}$$

Taking the limit $\epsilon \to 0$ results in

$$F'(\tilde{\zeta})\delta\zeta = \frac{\partial F}{\partial \tilde{\zeta}}\delta\zeta + \frac{\partial F}{\partial \tilde{\zeta}_x}\delta\zeta_x + \frac{\partial F}{\partial \tilde{\zeta}_y}\delta\zeta_y + \frac{\partial F}{\partial \tilde{\zeta}_s}\delta\zeta_s.$$

Hence equation (3.6.1) can be written as, including the superscript k,

$$\frac{\partial F}{\partial \tilde{\zeta}^k}\delta\zeta^k + \frac{\partial F}{\partial \tilde{\zeta}_x^k}(\delta\zeta)_x^k + \frac{\partial F}{\partial \tilde{\zeta}_y^k}(\delta\zeta)_y^k + \frac{\partial F}{\partial \tilde{\zeta}_s^k}(\delta\zeta)_s^k = -F(\tilde{\zeta}^k). \tag{3.6.4}$$

The weak formulation of the Mild-Slope equation, equation (3.5.3), is used to determine an expression for equation (3.6.4). Each individual term of equation (3.6.4) is given below

$$\begin{aligned}
\frac{\partial F}{\partial \tilde{\zeta}^k}\delta\zeta^k \quad &= \int_\Omega \left\{ \left(n_0 - \frac{iW(\tilde{\zeta}^k)}{\omega} \right) \eta\delta\zeta^k + \frac{i}{\omega}\frac{\partial W(\tilde{\zeta}^k)}{\partial \tilde{\zeta}^k}\tilde{\zeta}^k\eta\delta\zeta^k \right\} \, d\Omega \\
&\quad - i\int_{\Gamma_2} \frac{n_0}{k_0^2}\left(\frac{1-R}{1+R}\right) \left\{ \hat{p}\eta\delta\zeta^k \; + \frac{\partial \hat{p}}{\partial \tilde{\zeta}^k}\tilde{\zeta}^k\eta\delta\zeta^k - \frac{1}{2\frac{\partial \hat{p}}{\partial \tilde{\zeta}^k}}\frac{\partial \tilde{\zeta}^k}{\partial s}\frac{\partial \eta}{\partial s} \right\} d\Gamma \\
&\quad - i\int_{\Gamma_1} \frac{n_0}{k_0^2} \left\{ \hat{p}\eta\delta\zeta^k + \frac{\partial \hat{p}}{\partial \tilde{\zeta}^k}\tilde{\zeta}^k\eta\delta\zeta^k - \frac{1}{2\frac{\partial \hat{p}}{\partial \tilde{\zeta}^k}}\frac{\partial \tilde{\zeta}^k}{\partial s}\frac{\partial \eta}{\partial s} \right\} \, d\Gamma,
\end{aligned} \tag{3.6.5}$$

$$\frac{\partial F}{\partial \tilde{\zeta}_x^k}(\delta\zeta)_x^k \quad = -\int_\Omega \frac{n_0}{k_0^2}\eta_x(\delta\zeta)_x^k \, d\Omega, \tag{3.6.6}$$

$$\frac{\partial F}{\partial \tilde{\zeta}_y^k}(\delta\zeta)_y^k \quad = -\int_\Omega \frac{n_0}{k_0^2}\eta_y(\delta\zeta)_y^k \, d\Omega, \tag{3.6.7}$$

$$\frac{\partial F}{\partial \tilde{\zeta}_s^k}(\delta\zeta)_s^k \quad = i\int_{\Gamma_1} \frac{n_0}{k_0^2}\left(\frac{1-R}{1+R}\right)\frac{1}{2\hat{p}}\eta_s(\delta\zeta)_s^k \, d\Gamma + i\int_{\Gamma_1}\frac{n_0}{k_0^2}\frac{1}{2\hat{p}}\eta_s(\delta\zeta)_s^k \, d\Gamma. \tag{3.6.8}$$

Equation (3.6.5) contains the derivatives $\frac{\partial W}{\partial \tilde{\zeta}^k}$ and $\frac{\partial \hat{p}}{\partial \tilde{\zeta}^k}$, but $W(x, y, z\tilde{eta}^k)$ and therefore also $\hat{p}$ depend on the absolute value of $\tilde{\zeta}^k$, see expression (2.2.10). However, the complex derivative $\frac{\partial |\tilde{\zeta}|}{\partial \tilde{\zeta}}$ does not exist and thus the derivatives $\frac{\partial W}{\partial \tilde{\zeta}^k}$ and $\frac{\partial \hat{p}}{\partial \tilde{\zeta}^k}$ do not exist. Therefore we cannot include these terms into the calculations. This makes it doubtful whether Newton's method will give the quadratic convergence for the non-linear Mild-Slope equation.

Substituting expressions (3.6.5) - (3.6.8), without the derivatives of W and $\hat{p}$ with respect to $\tilde{\zeta}^k$, into equation (3.6.4) and using that $F(\tilde{\zeta}^k)$ equals the weak derivative gives Newton's method for the non-linear Mild-Slope equation.

$$\begin{aligned}
&\int_\Omega \left\{ \left(n_0 - \frac{iW(\tilde{\zeta}^k)}{\omega} \right) \eta\delta\zeta^k - \frac{n_0}{k_0^2}\nabla\eta\cdot\nabla\delta\zeta^k \right\} \, d\Omega - i\int_{\Gamma_2}\frac{n_0}{k_0^2}\left(\frac{1-R}{1+R}\right)\left\{ \hat{p}\eta\delta\zeta^k - \frac{1}{2\hat{p}}\frac{\partial\eta}{\partial s}\frac{\partial\delta\zeta^k}{\partial s} \right\} \, d\Gamma \\
&- i\int_{\Gamma_1}\frac{n_0}{k_0^2}\left\{ \hat{p}\eta\delta\zeta^k - \frac{1}{2\hat{p}}\frac{\partial\eta}{\partial s}\frac{\partial\delta\zeta^k}{\partial s} \right\} \, d\Gamma = -\int_\Omega \left\{ \left(n_0 - \frac{iW(\tilde{\zeta}^k)}{\omega} \right)\tilde{\zeta}^k\eta - \frac{n_0}{k_0^2}\nabla\tilde{\zeta}^k\cdot\nabla\eta \right\} \, d\Omega \\
&+ i\int_{\Gamma_1}\frac{n_0}{k_0^2}\left\{ \hat{p}\tilde{\zeta}_{in}\left(\boldsymbol{e}_{in}\cdot\boldsymbol{n}\right)\eta + \hat{p}(\tilde{\zeta}^k - \tilde{\zeta}_{in})\eta - \frac{1}{2\hat{p}}\left\{ \frac{\partial\tilde{\zeta}^k}{\partial s}\frac{\partial\eta}{\partial s} - \frac{\partial\tilde{\zeta}_{in}}{\partial s}\frac{\partial\eta}{\partial s} \right\} \right\} \, d\Gamma \\
&+ i\int_{\Gamma_2}\frac{n_0}{k_0^2}\left(\frac{1-R}{1+R}\right)\left\{ \hat{p}\tilde{\zeta}^k\eta - \frac{1}{2\hat{p}}\frac{\partial\tilde{\zeta}^k}{\partial s}\frac{\partial\eta}{\partial s} \right\} \, d\Gamma.
\end{aligned} \tag{3.6.9}$$

3.7 Summary of the discretization of the non-linear Mild-Slope equation

In this chapter we presented Picard iteration and Newton's method to discretize the non-linear Mild-Slope equation. We proposed a stopping criterion for the non-linear outer iteration to determine when and whether the non-linear solution is obtained. To accelerate Picard iteration and Newton's method we described inexact Picard iteration and inexact Newton's method. A forcing sequence η_k is necessary for an inexact method, therefore we presented five possibilities for this. We determined the weak formulation of the non-linear Mild-Slope equation to be able to apply Newton's method to it. Finally we applied Newton's method to the non-linear Mild-Slope equation.

Chapter 4

Spatial discretization of the Mild-Slope equation

For the spatial discretization of the non-linear Mild-Slope equation we use the method of finite elements. Berkhoff (1976) states that this is a good discretization method to determine the solution of the Mild-Slope equation in a harbour for the following two reasons;

- It has an easy way of representing boundaries of an arbitrary shape.
- It is possible to use small elements in areas where a strong variation of the solution can be expected and large elements in other areas.

4.1 Ritz-Galerkin Finite Element Method

The derivation of the finite element integrals is based on Ritz-Galerkin method as described by Zienkewicz (1971). The first step is to determine the weak formulation of the considered equation, depending on the used method to discretize the non-linear equation (e.g. Picard iteration or Newton's method). The unknown solution $\tilde{\zeta}(x, y)$ is approximated by a finite linear combination of basis functions, i.e.

$$\tilde{\zeta}(x, y) \approx \sum_{j=1}^{n_e} \zeta_j \psi_j(x, y). \tag{4.1.1}$$

We substitute this approximation into the equation which is obtained after using Picard iteration or Newton's method. The test function $\eta(x, y)$, also present in this equation, is approximated by one basis function $\psi_m(x, y)$, i.e.

$$\eta(x, y) = \psi_m(x, y). \tag{4.1.2}$$

The domain is divided into triangular elements, where the expression for the basis function ψ_m depends on the shape of the element. In HARES linear triangular elements are used, whereby the basis functions have the following properties:

- $\psi_j(x, y)$ is linear per triangle $\quad j = 1, 2, 3,$
- $\psi_j(x_m, y_m) = \delta_{jm} \qquad j, m = 1, 2, 3,$

with δ_{jm} the Kronecker delta. To be able to substitute the basis function into the weak formulation, it is necessary to have an explicit expression per element. The linear polynomial is defined by

$$\psi_j(x, y) = \alpha_j + \beta_j x + \gamma_j y. \tag{4.1.3}$$

Where the coefficients α_j, β_j and γ_j need to satisfy the following system of equations

$$\begin{bmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{bmatrix} \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 \\ \beta_1 & \beta_2 & \beta_3 \\ \gamma_1 & \gamma_2 & \gamma_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

This system of equations has a solution if the determinant Δ of the coefficient matrix (first matrix) never equals zero. The determinant is given by

$$\Delta = \begin{vmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{vmatrix},$$

where $|\Delta|$ is twice the area of a linear triangle.

Application of the Ritz-Galerkin finite element method to the weak formulation leads to a minimization over the unknown parameters ζ_j. This results in a linear system of equations that we need to solve efficiently. For more details on solving a linear system of equations see chapter 5.

4.2 Finite element method combined with Picard iteration

The weak formulation of the Mild-Slope equation, when using Picard iteration, is given by (see section 3.5)

$$\int_\Omega \left\{ \left(n_0 - \frac{iW(\tilde{\zeta}^{k-1})}{\omega} \right) \tilde{\zeta}^k \eta - \frac{n_0}{k_0^2} \nabla \tilde{\zeta}^k \cdot \nabla \eta \right\} - i \int_{\Gamma_2} \frac{n_0}{k_0^2} \left(\frac{1-R}{1+R} \right) \left\{ \hat{p} \tilde{\zeta}^k \eta - \frac{1}{2\hat{p}} \frac{\partial \tilde{\zeta}^k}{\partial s} \frac{\partial \eta}{\partial s} \right\} \, d\Gamma$$

$$- i \int_{\Gamma_1} \frac{n_0}{k_0^2} \left\{ \hat{p} \tilde{\zeta}_{in} (\boldsymbol{e}_{in} \cdot \boldsymbol{n}) \eta + \hat{p} (\tilde{\zeta}^k - \tilde{\zeta}_{in}) \eta - \frac{1}{2\hat{p}} \left(\frac{\partial \tilde{\zeta}^k}{\partial s} \frac{\partial \eta}{\partial s} - \frac{\partial \tilde{\zeta}_{in}}{\partial s} \frac{\partial \eta}{\partial s} \right) \right\} \, d\Gamma = 0. \tag{4.2.1}$$

Note that the modified wave number $\hat{p}$ also depends on the previous solution $\tilde{\zeta}^{k-1}$. Equation (4.2.1) is linear with respect to $\tilde{\zeta}^k(x, y)$, hence we can apply the Ritz-Galerkin finite element method to it. The elevation of the free surface $\tilde{\zeta}^k(x, y)$ is approximated by a finite linear combination of basis functions and the test function $\eta(x, y)$ by one basis function, see expressions (4.1.1) and (4.1.2). Substitution into the weak formulation (4.2.1) results in

$$\sum_{j=1}^{n_{el}} \zeta_j \left\{ \int_\Omega \left[\left(n_0 - \frac{iW(\tilde{\zeta}^{k-1})}{\omega} \right) \psi_j \psi_m - \frac{n_0}{k_0^2} \nabla \psi_j \cdot \nabla \psi_m \right] \, d\Omega \right\}$$

$$- i \sum_{j=1}^{n_{bel1}} \zeta_j \left\{ \int_{\Gamma_1} \frac{n_0}{k_0^2} \left(\hat{p} \psi_j \psi_m - \frac{1}{2\hat{p}} \frac{\partial \psi_j}{\partial s} \frac{\psi_m}{\partial s} \right) \, d\Gamma \right\}$$

$$- i \sum_{j=1}^{n_{bel2}} \zeta_j \left\{ \int_{\Gamma_2} \frac{n_0}{k_0^2} \left(\frac{1-R}{1+R} \right) \left(\hat{p} \psi_j \psi_m - \frac{1}{2\hat{p}} \frac{\partial \psi_j}{\partial s} \frac{\partial \psi_m}{\partial s} \right) \, d\Gamma \right\}$$

$$= i \int_{\Gamma_1} \frac{n_0}{k_0^2} \left(\hat{p} \tilde{\zeta}_{in} (\boldsymbol{e}_{in} \cdot \boldsymbol{n}) \psi_m - \hat{p} \tilde{\zeta}_{in} \psi_m + \frac{1}{2\hat{p}} \frac{\partial \tilde{\zeta}_{in}}{\partial s} \frac{\partial \psi_m}{\partial s} \right) \, d\Gamma. \tag{4.2.2}$$

Since only the parameters ζ_j are unknown, we can write expression (4.2.2) as the matrix-vector notation $\boldsymbol{S\zeta} = \boldsymbol{b}$. Note that the only contribution to the right-hand-side $\boldsymbol{b}$ is given by the incoming

wave $\tilde{\zeta}_{in}$. The domain Ω is divided into n_{el} internal elements and the boundary Γ into n_{bel} boundary elements, with $n_e = n_{el} + n_{bel}$. On each element the coefficients n_0, k_0, ω, $W(\tilde{\zeta}^{k-1})$, R and $\hat{p}$ are considered as constants. As mentioned in section 4.1, in HARES the internal elements are shaped as linear triangles and the boundary elements as linear line segments. For the boundary we make the distinction between the open boundary Γ_1 and the closed boundary Γ_2. The open boundary is divided into n_{bel1} elements and the closed boundary into n_{bel2} elements. The global matrix $\boldsymbol{S}$ and global vector $\boldsymbol{b}$ are determined by the summation over all the contributions from the internal and boundary elements. This results in

$$\boldsymbol{S}_{jm} = \sum_{l=1}^{n_{el}} S_{jm}^{e_l} + \sum_{l=1}^{n_{bel1}} S_{jm}^{be_{l1}} + \sum_{l=1}^{n_{bel2}} S_{jm}^{be_{l2}}, \tag{4.2.3}$$

$$\boldsymbol{b}_{m} = \sum_{l=1}^{n_{el}} b_{m}^{e_l} + \sum_{l=1}^{n_{bel1}} b_{m}^{be_{l1}} + \sum_{l=1}^{n_{bel2}} b_{m}^{be_{l2}}. \tag{4.2.4}$$

The integrals are determined exactly using Gaussian integration.

4.2.1 Internal elements

The contribution of the internal elements is determined by the integral over the domain Ω in expression (4.2.2). Which is given by

$$S_{jm} = \int_{\Omega} \left\{ \left(n_0 - \frac{iW(\tilde{\zeta}^{k-1})}{\omega} \right) \psi_j \psi_m - \frac{n_0}{k_0^2} \nabla\psi_j \cdot \nabla\psi_m \right\} \, d\Omega \quad \text{and} \quad b_m = 0$$

Using the assumption of constant coefficients on each element and the expression for the linear basis functions (4.1.3) gives for each internal element

$$S_{jm}^{e_l} = \left(n_0 - \frac{iW(\tilde{\zeta}^{k-1})}{\omega} \right) \int_{e_l} \psi_j \psi_m \, d\Omega - \frac{n_0}{k_0^2} (\beta_j \beta_m + \gamma_j \gamma_m) \int_{e_l} 1 \, d\Omega \quad \text{and} \quad b_m^{e_l} = 0. \tag{4.2.5}$$

4.2.2 Boundary elements on the open boundary

On the open boundary Γ_1 we obtain the following equations

$$\begin{aligned} S_{jm} &= -i \int_{\Gamma 1} \frac{n_0}{k_0^2} \left(\hat{p} \psi_j \psi_m - \frac{1}{2\hat{p}} \frac{\partial \psi_j}{\partial s} \frac{\partial \psi_m}{\partial s} \right) \, d\Gamma, \\ b_m &= i \int_{\Gamma_1} \frac{n_0}{k_0^2} \left(\hat{p} \tilde{\zeta}_{in} (\boldsymbol{e}_{in} \cdot \boldsymbol{n}) \psi_m - \hat{p} \tilde{\zeta}_{in} \psi_m + \frac{1}{2\hat{p}} \frac{\partial \tilde{\zeta}_{in}}{\partial s} \frac{\partial \psi_m}{\partial s} \right) \, d\Gamma. \end{aligned}$$

Inserting the basis function (4.1.3) into the equations for S_{jm} and b_m gives the following integrals for the element matrix and vector for an element on the open boundary.

$$S_{jm}^{be_{l1}} = -i \frac{n_0}{k_0^2} \left\{ \hat{p} \int_{be_{l1}} \psi_j \psi_m \, d\Gamma - \frac{1}{2\hat{p}} \int_{be_{l1}} \frac{\partial \psi_j}{\partial s} \frac{\partial \psi_m}{\partial s} \, d\Gamma, \right\} \tag{4.2.6}$$

$$b_m^{be_{l1}} = i \frac{n_0}{k_0^2} \left\{ \hat{p} \int_{be_{l1}} \left\{ \tilde{\zeta}_{in} (\boldsymbol{e}_{in} \cdot \boldsymbol{n}) - \tilde{\zeta}_{in} \right\} \psi_m \, d\Gamma + \frac{1}{2\hat{p}} \int_{be_{l1}} \frac{\partial \tilde{\zeta}_{in}}{\partial s} \frac{\partial \psi_m}{\partial s} \, d\Gamma \right\}. \tag{4.2.7}$$

4.2.3 Boundary elements on the closed boundary

For the elements on the closed boundary Γ_2 we find the following expressions for S_{jm} and b_m

$$S_{jm} = -i \int_{\Gamma_2} \frac{n_0}{k_0^2} \left(\frac{1-R}{1+R} \right) \left(\hat{p} \psi_j \psi_m - \frac{1}{2\hat{p}} \frac{\partial \psi_j}{\partial s} \frac{\partial \psi_m}{\partial s} \right) \, d\Gamma \quad \text{and} \quad b_m = 0.$$

Which results in the following contribution per element to the global matrix $\boldsymbol{S}$ and global vector $\boldsymbol{b}$,

$$S_{jm}^{be_{l2}} = -i \frac{n_0}{k_0^2} \left(\frac{1-R}{1+R} \right) \left\{ \hat{p} \int_{be_{l2}} \psi_m \psi_j \, d\Gamma - \frac{1}{2\hat{p}} \int_{be_{l2}} \frac{\partial \psi_j}{\partial s} \frac{\partial \psi_m}{\partial s} \, d\Gamma \right\} \quad \text{and} \quad b_m^{be_{l2}} = 0. \tag{4.2.8}$$

4.3 Finite element method combined with Newton's method

Application of Newton's method to the weak formulation of the non-linear Mild-Slope equation results in expression (3.6.9), which for convenience is repeated here.

$$\begin{aligned}
&\int_\Omega \left\{ \left(n_0 - \frac{iW(\tilde{\zeta}^k)}{\omega} \right) \eta \delta\zeta^k - \frac{n_0}{k_0^2} \nabla\eta \cdot \nabla\delta\zeta^k \right\} \, d\Omega - i \int_{\Gamma_2} \frac{n_0}{k_0^2} \left(\frac{1-R}{1+R} \right) \left\{ \hat{p}\eta\delta\zeta^k - \frac{1}{2\hat{p}} \frac{\partial\eta}{\partial s} \frac{\partial\delta\zeta^k}{\partial s} \right\} \, d\Gamma \\
&- i \int_{\Gamma_1} \frac{n_0}{k_0^2} \left\{ \hat{p}\eta\delta\zeta^k - \frac{1}{2\hat{p}} \frac{\partial\eta}{\partial s} \frac{\partial\delta\zeta^k}{\partial s} \right\} \, d\Gamma = - \int_\Omega \left\{ \left(n_0 - \frac{iW(\tilde{\zeta}^k)}{\omega} \right) \tilde{\zeta}^k \eta - \frac{n_0}{k_0^2} \nabla\tilde{\zeta}^k \cdot \nabla\eta \right\} \, d\Omega \\
&+ i \int_{\Gamma_1} \frac{n_0}{k_0^2} \left\{ \hat{p}\tilde{\zeta}_{in} \left(\boldsymbol{e}_{in} \cdot \boldsymbol{n} \right) \eta + \hat{p}(\tilde{\zeta}^k - \tilde{\zeta}_{in})\eta - \frac{1}{2\hat{p}} \left\{ \frac{\partial\tilde{\zeta}^k}{\partial s} \frac{\partial\eta}{\partial s} - \frac{\partial\tilde{\zeta}_{in}}{\partial s} \frac{\partial\eta}{\partial s} \right\} \right\} \, d\Gamma \\
&+ i \int_{\Gamma_2} \frac{n_0}{k_0^2} \left(\frac{1-R}{1+R} \right) \left\{ \hat{p}\tilde{\zeta}^k\eta - \frac{1}{2\hat{p}} \frac{\partial\tilde{\zeta}^k}{\partial s} \frac{\partial\eta}{\partial s} \right\} \, d\Gamma.
\end{aligned} \tag{4.3.1}$$

Equation (4.3.1) is a linear equation with respect to the update $\delta\zeta^k$, given by $\delta\zeta^k = \tilde{\zeta}^{k+1} - \tilde{\zeta}^k$. Using the update $\delta\zeta^k$ we can determine the next iterative solution $\tilde{\zeta}^{k+1}$. The Ritz-Galerkin finite element method is applied to equation (4.3.1). Hence the update $\delta\zeta^k(x,y)$ is approximated by a finite linear combination of basis functions $\psi_j(x,y)$ and the test function $\eta(x,y)$ by the basis function $\psi_m(x,y)$. This results in the following discretized version of equation (4.3.1)

$$\sum_{j=1}^{n_{el}} \zeta_j^k \int_\Omega \left\{ \left(n_0 - \frac{iW(\tilde{\zeta}^k)}{\omega} \right) \psi_j \psi_m - \frac{n_0}{k_0^2} \nabla\psi_j \cdot \nabla\psi_m \right\} \, d\Omega$$

$$- i \sum_{j=1}^{n_{bel2}} \zeta_j^k \int_{\Gamma_2} \frac{n_0}{k_0^2} \left(\frac{1-R}{1+R} \right) \left\{ \hat{p}\psi_j\psi_n - \frac{1}{2\hat{p}} \frac{\partial\psi_j}{\partial s} \frac{\partial\psi_n}{\partial s} \right\} \, d\Gamma$$

$$- i \sum_{j=1}^{n_{bel1}} \zeta_j^k \int_{\Gamma_1} \frac{n_0}{k_0^2} \left\{ \hat{p}\psi_j\psi_m - \frac{1}{2\hat{p}} \frac{\partial\psi_j}{\partial s} \frac{\partial\psi_m}{\partial s} \right\} \, d\Gamma$$

$$= - \int_\Omega \left\{ \left(n_0 - \frac{iW(\tilde{\zeta}^k)}{\omega} \right) \tilde{\zeta}^k \psi_m - \frac{n_0}{k_0^2} \nabla\tilde{\zeta}^k \cdot \nabla\psi_m \right\} \, d\Omega$$

$$+ i \int_{\Gamma_1} \frac{n_0}{k_0^2} \left\{ \hat{p}\tilde{\zeta}_{in} \left(\boldsymbol{e}_{in} \cdot \boldsymbol{n} \right) \psi + \hat{p}(\tilde{\zeta}^k - \tilde{\zeta}_{in})\psi_m - \frac{1}{2\hat{p}} \left\{ \frac{\partial\tilde{\zeta}^k}{\partial s} \frac{\partial\psi_m}{\partial s} - \frac{\partial\tilde{\zeta}_{in}}{\partial s} \frac{\partial\psi_m}{\partial s} \right\} \right\} \, d\Gamma$$

$$+ i \int_{\Gamma_2} \frac{n_0}{k_0^2} \left(\frac{1-R}{1+R} \right) \left\{ \hat{p}\tilde{\zeta}^k\psi_m - \frac{1}{2\hat{p}} \frac{\partial\tilde{\zeta}^k}{\partial s} \frac{\partial\psi_m}{\partial s} \right\} \, d\Gamma. \tag{4.3.2}$$

Note that ζ_j^k are the unknown coefficients to determine the update $\delta\boldsymbol{\zeta}^k$ and $\tilde{\zeta}^k$ the previous solution. As in the case for Picard iteration, expression (4.3.2) can be written in matrix-vector notation $\boldsymbol{S\zeta} = \boldsymbol{b}$. Note that the contribution to the global matrix $\boldsymbol{S}$ is exactly the same as in the case of Picard iteration. However, Picard iteration determines the next solution directly while for Newton's method only the update to the previous solution is obtained. Different from Picard iteration is that the contribution to the right-hand-side $\boldsymbol{b}$ is not only due to the incoming wave $\tilde{\boldsymbol{\zeta}}_{in}$ but also to the previous solution $\tilde{\boldsymbol{\zeta}}^k$. The interior domain Ω is divided into n_{el} triangular elements and the open boundary, respectively the closed boundary, into n_{bel1}, resp. n_{bel2}, linear line segments, with $n_e = n_{el}+n_{bel1}+n_{bel2}$. The global matrix $\boldsymbol{S}$ and global vector $\boldsymbol{b}$ are determined by the summation over all the elements, given by expressions (4.2.3) and (4.2.4).

4.3.1 Internal elements

Using Newton's method and the linear basis functions gives the following integrals for the internal elements, see also (4.3.2). Again we assume that the coefficients n_0, $W(\tilde{\zeta}^k)$, ω and k_0 are constant on each element.

$$S_{jm}^{e_l} = \left(n_0 - \frac{iW(\tilde{\zeta}^k)}{\omega} \right) \int_{e_l} \psi_j\psi_m \, d\Omega - \frac{n_0}{k_0^2} \left(\beta_j\beta_m + \gamma_j\gamma_m \right) \int_{e_l} 1 \, d\Omega, \tag{4.3.3}$$

and

$$b_m^{e_l} = - \left(n_0 - \frac{iW(\tilde{\zeta}^k)}{\omega} \right) \int_{e_l} \tilde{\zeta}^k\psi_m \, d\Omega + \frac{n_0}{k_0^2} \int_{e_l} \nabla\tilde{\zeta}^k \cdot \nabla\psi_m \, d\Omega. \tag{4.3.4}$$

The term $\tilde{\zeta}^k$ is known and can be written as

$$\tilde{\zeta}^k = \sum_{q=1}^{3} \zeta_q^k \psi_q(x_q, y_q).$$

Substituting this into equation (4.3.4) gives

$$b_m^{e_l} = -\left(n_0 - \frac{iW(\tilde{\zeta}^k)}{\omega}\right)\sum_{q=1}^{3}\zeta_q^k \int_{e_l} \psi_q\psi_m \; d\Omega + \frac{n_0}{k_0^2}\sum_{q=1}^{3}\zeta_q^k(x_q,y_q)(\beta_q\beta_m+\gamma_q\gamma_m)\int_{e_l} 1 \; d\Omega. \quad (4.3.5)$$

4.3.2 Boundary elements on the open boundary

For an element on the open boundary Γ_1 we obtain the following expression for the element matrix $S_{jm}^{be_{l1}}$

$$S_{jm}^{be_{l1}} = -i\frac{n_0}{k_0^2}\left\{\hat{p}\int_{be_{l1}} \psi_j\psi_m \; d\Gamma - \frac{1}{2\hat{p}}\int_{be_{l1}} \frac{\partial\psi_j}{\partial s}\frac{\partial\psi_m}{\partial s} \; d\Gamma\right\}, \quad (4.3.6)$$

and for the element vector $b_m^{be_{l1}}$

$$\begin{aligned} b_m^{be_{l1}} =& i\frac{n_0}{k_0}\left\{\hat{p}\int_{be_{l1}} \left\{\tilde{\zeta}_{in}(\boldsymbol{e}_{in}\cdot\boldsymbol{n})\psi_m - \tilde{\zeta}_{in}\psi_m\right\} d\Gamma + \frac{1}{2\hat{p}}\int_{be_{l1}} \frac{\partial\tilde{\zeta}_{in}}{\partial s}\frac{\partial\psi_m}{\partial s} \; d\Gamma\right. \\ &\left. +\hat{p}\int_{be_{l1}} \tilde{\zeta}^k\psi_m \; d\Gamma - \frac{1}{2\hat{p}}\int_{be_{l1}} \frac{\partial\tilde{\zeta}^k}{\partial s}\frac{\partial\psi_m}{\partial s} \; d\Gamma\right\}. \end{aligned} \quad (4.3.7)$$

4.3.3 Boundary elements on the closed boundary

The contribution to the global matrix $\boldsymbol{S}$ and the global vector $\boldsymbol{b}$ for an element on the closed boundary Γ_2 is given by

$$S_{jm}^{be_{l2}} = -i\frac{n_0}{k_0^2}\left(\frac{1-R}{1+R}\right)\left\{\hat{p}\int_{be_{l2}} \psi_j\psi_m \; d\Gamma - \frac{1}{2\hat{p}}\int_{be_{l2}} \frac{\partial\psi_j}{\partial s}\frac{\partial\psi_m}{\partial s} \; d\Gamma\right\}, \quad (4.3.8)$$

and

$$b_m^{be_{l2}} = i\frac{n_0}{k_0^2}\left(\frac{1-R}{1+R}\right)\left\{\hat{p}\int_{be_{l2}} \tilde{\zeta}^k\psi_m \; d\Gamma - \frac{1}{2\hat{p}}\int_{be_{l2}} \frac{\partial\tilde{\zeta}^k}{\partial s}\frac{\partial\psi_m}{\partial s} \; d\Gamma\right\}. \quad (4.3.9)$$

4.4 Summary of the spatial discretization

In this chapter we presented the spatial discretization of the non-linear Mild-Slope equation, when Picard iteration or Newton's method has been used for the non-linear discretization. The spatial discretization results in the system of equations $\boldsymbol{S\zeta} = \boldsymbol{b}$. We used the Ritz-Galerkin finite element method where we divided the domain into triangular elements with piecewise linear basis functions. After the discretization we obtained for each element an integral for the element matrix S_{jm} and the element vector b_m. These integrals are determined by Gaussian integration. Using the element matrices and vector we can determine the contribution to the global matrix $\boldsymbol{S}$ and the global vector $\boldsymbol{b}$. Both Newton's method and Picard iteration combined with the Ritz-Galerkin finite element method result in the same global matrix $\boldsymbol{S}$, but in a different global vector $\boldsymbol{b}$. A suitable method to solve the system of equation depends on the properties of matrix $\boldsymbol{S}$. In chapter 6 we present the element matrices for an internal element and an element on the boundary. There we conclude that matrix $\boldsymbol{S}$ is a symmetric, non-Hermitian, sparse matrix.

Chapter 5

Solving a system of equations

Application of the Ritz-Galerkin finite element method to the non-linear Mild-Slope equation leads to a system of equations $\boldsymbol{S\zeta} = \boldsymbol{b}$, as described in chapter 4,. In chapter 6 we discuss that $\boldsymbol{S} \in \mathbb{C}^{N\times N}$ is a symmetric, non-Hermitian, sparse matrix. There are various methods to solve a system of equations; two major classes of solution methods are the direct methods and the iterative methods. In section 5.1 we discuss the direct methods briefly, with a focus on `MUMPS`, a state-of-the-art direct method. In section 5.2 we describe the iterative methods, especially the Krylov subspace methods. To accelerate the iterative methods it is possible to apply a preconditioner to the system of equations, in section 5.3 we present the incomplete LU decomposition and the Shifted Laplace preconditioner.

The current version of HARES uses the Krylov subspace method Bi-CGSTAB and the incomplete LU decomposition as a preconditioner.

For convenience the notation $\boldsymbol{Ax} = \boldsymbol{b}$ is used throughout this chapter, with $\boldsymbol{A}$ an $N \times N$-matrix and $\boldsymbol{x}$ and $\boldsymbol{b}$ vectors of dimension N.

5.1 Direct methods

A direct method is conceptually a very straightforward method. The LU factorization of matrix $\boldsymbol{A}$ is computed and the right-hand-side $\boldsymbol{b}$ is multiplied by it. Hence we obtain the solution $\boldsymbol{x}$ of the system of equations in one iteration by

$$\boldsymbol{x} = \boldsymbol{U}^{-1}\boldsymbol{L}^{-1}\boldsymbol{b},$$

with $\boldsymbol{L}$ a lower triangular matrix and $\boldsymbol{U}$ an upper triangular matrix. If the dimension of matrix $\boldsymbol{A}$ becomes larger, the process of computing the complete LU decomposition gets (mostly undesirably) lengthy. However, nowadays software packages exist, which smartly deal with computing the LU factorization resulting in a short computing time. Especially for two dimensional problems, the state-of-the-art direct solvers are on average faster than a well-preconditioned iterative method. An example of a fast direct method is `MUMPS` - MUltifrontal Massively Parallel Solver, see Amestoy et al. (2001) and Amestoy et al. (2006).

5.1.1 MUltifrontal Massively Parallel Solver - `MUMPS`

`MUMPS` is a package for solving a system of linear equations of the form $\boldsymbol{Ax} = \boldsymbol{b}$. Matrix $\boldsymbol{A}$ is a square sparse matrix, which can either be non-symmetric, symmetric positive definite, or general symmetric. Matrix $\boldsymbol{A}$ obtained after the non-linear and spatial discretization of the Mild-Slope equation is a complex non-Hermitian matrix, hence `MUMPS` handles it as a non-symmetric matrix. The computation of the LU factorization of matrix $\boldsymbol{A}$ is performed in the following three main steps:

1. *Analysis*
   During the analysis phase a preprocessing is performed, which include an ordering based on the symmetrized pattern $\boldsymbol{A} + \boldsymbol{A}^T$ and a symbolic factorization. After analysis, the preprocessed matrix $\boldsymbol{A_{pre}}$ is obtained by the following factorization

   $$\boldsymbol{A_{pre}} = \boldsymbol{P}\boldsymbol{D_r}\boldsymbol{A}\boldsymbol{Q_c}\boldsymbol{D_c}\boldsymbol{P}^T.$$

   Where $\boldsymbol{P}$ is a permutation matrix, $\boldsymbol{Q_c}$ a column permutation and $\boldsymbol{D_r}$ and $\boldsymbol{D_c}$ diagonal matrices for (respectively row and column) scaling.

2. *Factorization*
   A direct factorization $\boldsymbol{A_{pre}} = \boldsymbol{LU}$ is computed, where $\boldsymbol{L}$ is a lower triangular and $\boldsymbol{U}$ an upper triangular matrix. During the factorization multiple fronts are processed simultaneously, this approach is called a multifrontal approach. After factorization, the matrices $\boldsymbol{L}$ and $\boldsymbol{U}$ are kept distributed and used at the solution phase.

3. *Solution*
   The solution of $\boldsymbol{LU}\boldsymbol{x_{pre}} = \boldsymbol{b_{pre}}$ is obtained, where $\boldsymbol{x_{pre}}$ and $\boldsymbol{b_{pre}}$ are the transformed solution $\boldsymbol{x}$ and right-hand-side $\boldsymbol{b}$. Firstly a forward elimination step

   $$\boldsymbol{L}\boldsymbol{y} = \boldsymbol{b_{pre}}$$

   is performed, which is followed by a backward elimination step

   $$\boldsymbol{U}\boldsymbol{x_{pre}} = \boldsymbol{y}.$$

   The obtained solution $\boldsymbol{x_{pre}}$ is then post-processed to receive the solution $\boldsymbol{x}$.

`MUMPS` is written in Fortran 90 and available in a sequential and parallel version.

5.2 Iterative methods

There are three main classes of iterative methods, i.e. the basic iterative methods, the Krylov subspace methods and the multigrid methods. Contrary to a direct method, the solution of the system of equations is obtained by performing several iterations. Iterative methods are a good alternative to the direct method in one of the following cases; not many iterations are required, matrix $\boldsymbol{A}$ is sparse or has a special structure, or a good initial guess for the solution $\boldsymbol{x}$ is available. In this section we will limit ourselves to the Krylov subspace methods, which give good results for general matrices.

The measure for the correctness of the iterate $\boldsymbol{x}^i$ is the residual, which is given by $\boldsymbol{r}^i = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^i$.

Krylov subspace methods
Krylov subspace methods are based on the idea that the solution of the system of equations can be approximated by a polynomial in $\boldsymbol{A}$, i.e.

$$\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b} \approx P(\boldsymbol{A})\boldsymbol{b}.$$

If a good initial guess $\boldsymbol{x}^0$ is available, then the equivalent system of equations $\boldsymbol{A}(\boldsymbol{x} - \boldsymbol{x}^0) = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^0 = \boldsymbol{r}^0$ can be solved. The method starts with a polynomial of degree zero and every iteration, to increase the accuracy, a degree is added to this polynomial

$$\boldsymbol{x}^i = \boldsymbol{x}^0 + P^{i-1}(\boldsymbol{A})\boldsymbol{r}^0.$$

The iterates $\boldsymbol{x}^i$ are contained in the subspace $\boldsymbol{x}^0 + \mathcal{K}^i(\boldsymbol{A}; \boldsymbol{r}^0)$, where $\mathcal{K}^i(\boldsymbol{A}; \boldsymbol{r}^0)$ is denoted as the Krylov subspace of dimension i

$$\mathcal{K}^i(\boldsymbol{A}; \boldsymbol{r}^0) = \text{span}\left\{\boldsymbol{r}^0, \boldsymbol{A}\boldsymbol{r}^0, \ldots, \boldsymbol{A}^{i-1}\boldsymbol{r}^0\right\}.$$

The ideal Krylov subspace method satisfies the following properties:

- The error, $\boldsymbol{x} - \boldsymbol{x}^i$, is minimal in the some norm, which is denoted as the optimality property;
- Short recurrences, only the results of some foregoing steps are necessary to compute the next iterate.

However, it is shown that for a general matrix $\boldsymbol{A}$, e.g. $\boldsymbol{A}$ is not symmetric (Hermitian) positive definite, these properties cannot all be satisfied. The Krylov subspace methods Bi-CGSTAB and IDR(s) are presented here, which are both based on the short recurrences property. The work by Sonneveld and van Gijzen (2008) shows that the IDR(s) algorithm outperforms Bi-CGSTAB for a three-dimensional Helmholtz problem. Therefore, IDR(s) is proposed as an acceleration of the currently implemented Bi-CGSTAB algorithm.

5.2.1 Bi-CGSTAB

The Bi-CGSTAB method is in 1992 derived by van der Vorst as an improvement of the Conjugate Gradient Squared (CG-S) method presented by Sonneveld (1989). CG-S was on its turn an improvement of the Bi-Conjugate Gradient (Bi-CG) method.

Bi-CG
Bi-CG uses a basis $\boldsymbol{r}_0, \ldots \boldsymbol{r}_{i-1}$ which is constructed for $\mathcal{K}_i(\boldsymbol{A}; \boldsymbol{r}_0)$ such that $\boldsymbol{r}_j \perp span\{\boldsymbol{r}_0^*, \ldots, \boldsymbol{r}_{j-1}^*\}$ with $(j \leq i)$ and $\boldsymbol{r}_0^*, \ldots, \boldsymbol{r}_{i-1}^*$ forms a basis for $\mathcal{K}_i(\boldsymbol{A}^T; \boldsymbol{r}_0^*)$ such that $\boldsymbol{r}_j^* \perp span\{\boldsymbol{r}_0, \ldots, \boldsymbol{r}_{j-1}\}$ with $(j \leq i)$. The Bi-CG method is derived by Fletcher (1976) where he assumes that the residuals $\boldsymbol{r}_i$ and $\boldsymbol{r}_i^*$ can be written as

$$\boldsymbol{r}_i^{Bi-CG} = P_i(\boldsymbol{A})\boldsymbol{r}_0 \quad \text{and} \quad \boldsymbol{r}_i^{*Bi-CG} = P_i(\boldsymbol{A}^T)\boldsymbol{r}_0^*,$$

where $P_i(\boldsymbol{A})$ is a polynomial of degree at most i. The bi-orthogonality of the residuals $\boldsymbol{r}_i$ and $\boldsymbol{r}_j^*$ can also be written as

$$\left(P_i(\boldsymbol{A})\boldsymbol{r}_0, P_j(\boldsymbol{A}^T)\boldsymbol{r}_0^*\right) = 0 \quad \text{for } j < i. \tag{5.2.1}$$

With this expression $\boldsymbol{r}_i$ as well as $\boldsymbol{r}_j^*$ need to be constructed, therefore the matrix-vector products with both $\boldsymbol{A}$ and $\boldsymbol{A}^T$ need to be determined.

CG-S
The CG-S method is introduced by Sonneveld (1989), where he used that equation (5.2.1) are written as

$$\left(P_i(\boldsymbol{A})\boldsymbol{r}_0, P_j(\boldsymbol{A}^T)\boldsymbol{r}_0^*\right) = (P_j(\boldsymbol{A})P_i(\boldsymbol{A})\boldsymbol{r}_0, \boldsymbol{r}_0^*) = 0 \quad \text{for } j < i.$$

CG-S constructs residuals that can be written as

$$\boldsymbol{r}_i^{CG-S} = P_i(\boldsymbol{A})\boldsymbol{r}_i^{Bi-CG} = P_i^2(\boldsymbol{A})\boldsymbol{r}_0.$$

The benefit of the CG-S algorithm is that $\boldsymbol{r}_j^*$ does not need to be formed. Hence we do not have to compute the matrix-vector products with $\boldsymbol{A}^T$, but we do need twice the amount of matrix-vector products with $\boldsymbol{A}$. The downside of this method is that its convergence behaviour is not very smooth and the square in the residual polynomial may lead to a build-up of rounding errors.

Bi-CGSTAB
To improve the convergence behaviour of the CG-S method van der Vorst (1992) derived the more smoothly converging variant of CG-S called Bi-CGSTAB. The Bi-CGSTAB residual is written as

$$\boldsymbol{r}_i^{Bi-CGSTAB} = Q_j(\boldsymbol{A})\boldsymbol{r}_i^{Bi-CG} = Q_j(\boldsymbol{A})P_j(\boldsymbol{A})\boldsymbol{r}_0,$$

with

$$Q_j(\boldsymbol{A}) = (I - \omega_1\boldsymbol{A})(I - \omega_2\boldsymbol{A}) \ldots (I - \omega_j\boldsymbol{A}). \tag{5.2.2}$$

Bi-CGSTAB

$\boldsymbol{x}_0$ is an initial guess; $\boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0$;
$\boldsymbol{r}_0^*$ is an arbitrary vector, such that $(\boldsymbol{r}_0, \boldsymbol{r}_0^*) \neq 0$, e.g. $\boldsymbol{r}_0^* = \boldsymbol{r}_0$;
$\rho_0 = \alpha = \omega_0 = 1$;
$\boldsymbol{v}_0 = \boldsymbol{p}_0 = 0$;
for $i = 1, 2, \ldots$ until convergence **do**
    $\rho_i = (\boldsymbol{r}_0^*, \boldsymbol{r}_{i-1})$; $\beta = (\rho_i/\rho_{i-1})(\alpha/\omega_{i-1})$;
    $\boldsymbol{p}_i = \boldsymbol{r}_{i-1} + \beta(\boldsymbol{p}_{i-1} - \omega_{i-1}\boldsymbol{v}_{i-1})$;
    $\boldsymbol{v}_i = \boldsymbol{A}\boldsymbol{p}_i$;
    $\alpha = \rho_i/(\boldsymbol{r}_0^*, \boldsymbol{v}_i)$;
    $\boldsymbol{s} = \boldsymbol{r}_{i-1} - \alpha\boldsymbol{v}_{i-1}$;
    $\boldsymbol{t} = \boldsymbol{A}\boldsymbol{s}$;
    $\omega_i = (\boldsymbol{t}, \boldsymbol{s})/(\boldsymbol{t}, \boldsymbol{t})$;
    $\boldsymbol{x}_i = \boldsymbol{x}_{i-1} + \alpha\boldsymbol{p}_i + \omega_i\boldsymbol{s}$;
    if $\boldsymbol{x}_i$ is accurate enough then quit
    $\boldsymbol{r}_i = \boldsymbol{s} - \omega_i\boldsymbol{t}$;
end

Alg. 5.1: *Bi-CGSTAB algorithm.*

The coefficients ω_j in the polynomial $Q_j(\boldsymbol{A})$ are chosen such that the residual is minimized. The Bi-CGSTAB algorithm, see van der Vorst (1992), is given by algorithm 5.1.

Bi-CGSTAB is a finite method, which means that after at most n iterations the exact solution $\boldsymbol{x}$ in finite precision arithmetic is obtained. As mentioned earlier a common choice for the coefficients ω_j is to minimize the residual $\boldsymbol{r}_i$. However, other choices for ω_j are possible and this choice should depend on the specific problem that is solved. There are three cases where the Bi-CGSTAB algorithm breaks down. It can happen when $\rho_i = (\boldsymbol{r}_0^*, \boldsymbol{r}_{i-1}) = 0$ with $\boldsymbol{r}_{i-1} \neq 0$, when $(\boldsymbol{r}_0^*, \boldsymbol{v}_i) = 0$ or when $(\boldsymbol{t}, \boldsymbol{s}) = 0$.

5.2.2 IDR(s)

The IDR(s) method, see Sonneveld and van Gijzen (2008), is a Krylov subspace type method for which the residuals $\boldsymbol{r}_n = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_n$ are in the Krylov subspace $\mathcal{K}^n(\boldsymbol{A}; \boldsymbol{r}_0)$. The residuals of a Krylov method satisfy the following recursion method

$$\boldsymbol{r}_{n+1} = \boldsymbol{r}_n - \alpha\boldsymbol{A}\boldsymbol{v}_n - \sum_{l=1}^{\hat{l}} \gamma_l \Delta\boldsymbol{r}_{n-l},$$

where $\boldsymbol{v}_n$ is any computable vector in $\mathcal{K}^n(\boldsymbol{A}; \boldsymbol{r}_0) \setminus \mathcal{K}^{n-1}(\boldsymbol{A}; \boldsymbol{r}_0)$ and $\Delta\boldsymbol{r}_n = \boldsymbol{r}_{n+1} - \boldsymbol{r}_n$. The IDR($s$) method is based Theorem 1.

Theorem 1. ***(IDR theorem)*** *Let $\boldsymbol{A}$ be any matrix in $\mathbb{C}^{N\times N}$, let $\boldsymbol{v}_0$ be any non-zero vector in $\mathbb{C}^N$, and let $\mathcal{G}_0$ be the full Krylov space $\mathcal{K}^N(\boldsymbol{A}, \boldsymbol{v}_0)$. Let $\mathcal{S}$ denote any (proper) subspace of $\mathbb{C}^N$ such that $\mathcal{S}$ and $\mathcal{G}_0$ do not share a non-trivial invariant subspace of $\boldsymbol{A}$, and define the sequence $\mathcal{G}_j$, $j = 1, 2, \ldots$ as*

$$\mathcal{G}_j = (\boldsymbol{I} - \omega_j\boldsymbol{A})(\mathcal{G}_{j-1} \cap \mathcal{S}),$$

where the ω_j's are non-zero scalars. Then

(i) $\mathcal{G}_j \subset \mathcal{G}_{j-1}$ for all $\mathcal{G}_{j-1} \neq \{\boldsymbol{0}\}$, $j > 0$.

(ii) $\mathcal{G}_j = \{\boldsymbol{0}\}$ for some $j \leq N$.

Proof. See Sonneveld and van Gijzen (2008). □

The IDR(s) algorithm is based on generating residuals $\boldsymbol{r}_n$ which are forced to be in the subspace $\mathcal{G}_j$, where j is non-decreasing with increasing n. According to Theorem 1 ultimately $\boldsymbol{r}_n \in \mathcal{G}_M = \{\boldsymbol{0}\}$ with $M \leq N$. The residual $\boldsymbol{r}_{n+1}$ is in the space $\mathcal{G}_{j+1}$ if

$$\boldsymbol{r}_{n+1} = (\boldsymbol{I} - \omega_{j+1}\boldsymbol{A})\boldsymbol{v}_n, \tag{5.2.3}$$

with $\boldsymbol{v}_n \in \mathcal{G}_j \cap \mathcal{S}$. The main problem becomes finding $\boldsymbol{v}_n$. The following expression is chosen for $\boldsymbol{v}_n$

$$\boldsymbol{v}_n = \boldsymbol{r}_n - \sum_{l=1}^{s} \gamma_l \Delta \boldsymbol{r}_{n-l}.$$

Let there be a $N \times s$ matrix $\boldsymbol{P} = (\boldsymbol{p}_1\ \boldsymbol{p}_2\ \ldots\ \boldsymbol{p}_s)$ such that $\mathcal{S}$ is the left null-space of $\boldsymbol{P}$, i.e. $\mathcal{S} = \mathcal{N}\left(\boldsymbol{P}^H\right)$. Since $\boldsymbol{v}_n$ is contained in $\mathcal{S}$ it also holds that

$$\boldsymbol{P}^H \boldsymbol{v}_n = \boldsymbol{P}^H (\boldsymbol{r}_n - \boldsymbol{c}\Delta\boldsymbol{R}_n) = 0 \quad \Rightarrow \quad \boldsymbol{P}^H \Delta\boldsymbol{R}_n \boldsymbol{c} = \boldsymbol{P}^H \boldsymbol{r}_n,$$

with $\Delta\boldsymbol{R}_n = (\Delta\boldsymbol{r}_{n-1}\ \Delta\boldsymbol{r}_{n-2}\ \ldots\ \Delta\boldsymbol{r}_{n-s})$ and $\boldsymbol{c} \in \mathbb{C}^s$ contains the coefficients γ_l. We obtain an $s \times s$ linear system for the coefficients γ_l which generally is uniquely solvable. Vector $\boldsymbol{c}$ can be determined and hence we are able to compute $\boldsymbol{v}_n$ and $\boldsymbol{r}_{n+1} \in \mathcal{G}_{j+1}$. By Theorem 1 it follows that $\mathcal{G}_{j+1} \subset \mathcal{G}_j$, therefore it also holds that $\boldsymbol{r}_{n+1} \in \mathcal{G}_j$. Using $\boldsymbol{r}_{n+1}$ we can compute $\Delta\boldsymbol{R}_{n+1}$, $\boldsymbol{v}_{n+1} \in (\mathcal{G}_j \cap \mathcal{S})$ and hence $\boldsymbol{r}_{n+2} \in \mathcal{G}_{j+1}$. This needs to repeated $s+1$ times such that all the elements of $\Delta\boldsymbol{R}_n$ are in $\mathcal{G}_{j+1}$. Then we have that $\boldsymbol{v}_{n+(s+1)} \in (\mathcal{G}_{j+1} \cap \mathcal{S})$ and therefore the computed residual $\boldsymbol{r}_{n+(s+1)}$ is contained in the subspace $\mathcal{G}_{j+2}$ of $\mathcal{G}_{j+1}$.

The IDR(s) algorithm is given in algorithm 5.2. From the algorithm we can see that for the first residual in $\mathcal{G}_{j+1}$ the coefficient ω can be chosen freely. But for the computations of the other residuals in $\mathcal{G}_{j+1}$ the value of ω cannot be changed. In the presented algorithm we use a minimization of the norm of $\boldsymbol{r}_{n+1}$ for the computation of ω. Also matrix $\boldsymbol{P}$ can be chosen freely in the beginning of the algorithm.

In the current algorithm the initialization is done using a simple Krylov method. However, as long as the $\Delta\boldsymbol{x}_i$, $i = 0, \ldots, s-1$ are in the complete Krylov subspace, they can be chosen freely. Contrary to what is presented in algorithm 5.2, the choice of the coefficient ω in the current implementation of IDR(s) is not based on minimizing the residual, but on a strategy proposed by Sleijpen and van der Vorst (1995).

The IDR theorem states that dimension reduction takes place but not how large this reduction is. Theorem 2 describes the rate of dimension reduction.

Theorem 2. ***(Extended IDR theorem)*** *Let $\boldsymbol{A}$ be any vector in $\mathbb{C}^{N\times N}$, let $\boldsymbol{p}_1$, $\boldsymbol{p}_2$, …, $\boldsymbol{p}_s \in \mathbb{C}^N$ be linearly independent, let $\boldsymbol{P} = (\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_s)$, let $\mathcal{G}_0 = \mathcal{K}^N(\boldsymbol{A}; \boldsymbol{r}_0)$ be the full Krylov space corresponding to $\boldsymbol{A}$ and the vector $\boldsymbol{r}_0$, and let the sequence of spaces $\{\mathcal{G}_j, j = 1, 2, \ldots\}$ be defined by*

$$\mathcal{G}_j = (\boldsymbol{I} - \omega_j \boldsymbol{A}) \left(\mathcal{G}_{j-1} \cap \mathcal{N}\left(\boldsymbol{P}^H\right)\right),$$

where ω_j are non-zero numbers, such that $\boldsymbol{I} - \omega_j\boldsymbol{A}$ is non-singular. Let $\dim(\mathcal{G}_j) = d_j$; then the sequence $\{d_j, j = 0, 1, 2, \ldots\}$ is monotonically non-increasing and satisfies

$$0 \leq d_j - d_{j+1} \leq d_{j-1} - d_j \leq s.$$

Proof. See Sonneveld and van Gijzen (2008). □

According to the extended IDR theorem the dimension reduction per step is between 0 and s. If the dimension reduction is precisely s it is called the generic case, otherwise the non-generic case. The extended IDR theorem leads to Corollary 1 for the generic case.

IDR(s)

1. **Require:** $\boldsymbol{A} \in \mathbb{C}^{N\times N}$; $\boldsymbol{x}_0, \boldsymbol{b} \in \mathbb{C}^N$; $\boldsymbol{P} \in \mathbb{C}^{N\times s}$; $TOL \in (0,1)$; $MAXIT > 0$
2. **Ensure:** $\boldsymbol{x}_n$ such that $\|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_n\| \leq TOL$
3. {*Initialization.*}
4. Calculate $\boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0$;
5. {*Apply s minimum norm steps, to build enough vectors in $\mathcal{G}_0$*}
6. **for** $n = 0, 1, \ldots, s-1$ **do**
7. $\boldsymbol{v} = \boldsymbol{A}\boldsymbol{r}_n$; Select ω;
8. $\Delta\boldsymbol{x}_n = \omega\boldsymbol{r}_n$; $\Delta\boldsymbol{r}_n = -\omega\boldsymbol{v}$;
9. $\boldsymbol{r}_{n+1} = \boldsymbol{r}_n + \Delta\boldsymbol{r}_n$; $\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \Delta\boldsymbol{x}_n$;
10. **end**
11. $\Delta\boldsymbol{R}_{n+1} = (\Delta\boldsymbol{r}_n \ldots \Delta\boldsymbol{r}_0)$; $\Delta\boldsymbol{X}_{n+1} = (\Delta\boldsymbol{x}_n \ldots \Delta\boldsymbol{x}_0)$;
12. {*Building $\mathcal{G}_j$ spaces for $j = 1, 2, 3, \ldots$*}
13. $n = s$
14. {*Loop over $\mathcal{G}_j$ spaces*}
15. **while** $\|\boldsymbol{r}_n\| > TOL$ and $n < MAXIT$ **do**
16. {*Loop inside $\mathcal{G}_j$ space*}
17. **for** $k = 0, 1, \ldots, s$ **do**
18. Solve $\boldsymbol{c}$ from $\boldsymbol{P}^H\Delta\boldsymbol{R}_n\boldsymbol{c} = \boldsymbol{P}^H\boldsymbol{r}_n$;
19. $\boldsymbol{v} = \boldsymbol{r}_n - \Delta\boldsymbol{R}_n\boldsymbol{c}$;
20. **if** $k = 0$ **do**
21. {*Entering $\mathcal{G}_{j+1}$*}
22. $\boldsymbol{t} = \boldsymbol{A}\boldsymbol{v}$;
23. Select ω;
24. $\Delta\boldsymbol{r}_n = -\Delta\boldsymbol{R}_n\boldsymbol{c} - \omega\boldsymbol{t}$;
25. $\Delta\boldsymbol{x}_n = -\Delta\boldsymbol{X}_n\boldsymbol{c} + \omega\boldsymbol{v}$;
26. **else**
27. {*Subsequent vectors in $\mathcal{G}_{j+1}$*}
28. $\Delta\boldsymbol{x}_n = -\Delta\boldsymbol{X}_n\boldsymbol{c} + \omega\boldsymbol{v}$;
29. $\Delta\boldsymbol{r}_n = -\boldsymbol{A}\Delta\boldsymbol{x}_n$;
30. **end**
31. $\boldsymbol{r}_{n+1} = \boldsymbol{r}_n + \Delta\boldsymbol{r}_n$;
32. $\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \Delta\boldsymbol{x}_n$;
33. $n = n + 1$;
34. $\Delta\boldsymbol{R}_n = (\Delta\boldsymbol{r}_{n-1} \ldots \Delta\boldsymbol{r}_{n-s})$;
35. $\Delta\boldsymbol{X}_n = (\Delta\boldsymbol{x}_{n-1} \ldots \Delta\boldsymbol{x}_{n-s})$;
36. **end**
37. **end**

Alg. 5.2: *IDR(s) algorithm*

Corollary 1. *In the generic case IDR(s) requires at most $N + \frac{N}{s}$ matrix-vector multiplications to compute the exact solution in finite precision arithmetic.*

IDR(s)-*biortho*

It is possible to make some adjustments to the IDR(s) algorithm as presented in algorithm 5.2. The residual of equation (5.2.3) can also be written as

$$\boldsymbol{r}_{n+1} = \boldsymbol{r}_n - \omega_{j+1}\boldsymbol{A}\boldsymbol{v}_n - \boldsymbol{G}_n\boldsymbol{c} \quad \text{with} \quad \boldsymbol{G}_n = \Delta\boldsymbol{R}_n.$$

The corresponding recursion for the iterate is obtained by multiplying the equation above by $\boldsymbol{A}^{-1}$, hence

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \omega_{j+1}\boldsymbol{v}_n + \boldsymbol{U}_n\boldsymbol{c} \quad \text{with} \quad \boldsymbol{U}_n = A^{-1}\boldsymbol{G}_n = \Delta\boldsymbol{X}_n.$$

For $\boldsymbol{u}_{n+1} = \boldsymbol{x}_{n+2} - \boldsymbol{x}_{n+1}$ and $\boldsymbol{g}_{n+1} = -(\boldsymbol{r}_{n+2} - \boldsymbol{r}_{n+1})$ we find the following iterates

$$\boldsymbol{u}_{n+1} = \omega_{j+1}\boldsymbol{v}_{n+1} + \boldsymbol{U}_{n+1}\boldsymbol{c} \quad \text{and} \quad \boldsymbol{g}_{n+1} = \boldsymbol{A}\boldsymbol{u}_{n+1}.$$

The next residual and iterate can be determined by

$$\boldsymbol{r}_{n+k+1} = \boldsymbol{r}_{n+k} - \boldsymbol{g}_{n+k} \quad \text{and} \quad \boldsymbol{x}_{n+k+1} = \boldsymbol{x}_{n+k} + \boldsymbol{u}_{n+k},$$

with $\boldsymbol{r}_{n+k+1}, \boldsymbol{r}_{n+k}, \boldsymbol{g}_{n+k} \in \mathcal{G}_{j+1}$. To compute a new residual in $\mathcal{G}_{j+1}$ we could also use a more general linear combination of vectors in $\mathcal{G}_{j+1}$

$$\boldsymbol{r}_{n+k+1} = \boldsymbol{r}_{n+k} - \sum_{i=1}^{k} \beta_i \boldsymbol{g}_{n+i}.$$

And for the vector $\boldsymbol{g}_{n+k}$ we can use

$$\boldsymbol{g}_{n+k} = \bar{\boldsymbol{g}} - \sum_{i=1}^{k-1} \alpha_i \boldsymbol{g}_{n+i} \quad \text{with} \quad \bar{\boldsymbol{g}} = -(\boldsymbol{r}_{n+k+1} - \boldsymbol{r}_{n+k}) = -\Delta\boldsymbol{r}_{n+k}.$$

The values of α_i and β_i are chosen such that intermediate residuals and $\boldsymbol{g}_{n+k}$ have desirable properties. Analogous $\boldsymbol{x}_{n+k+1}$ and $\boldsymbol{u}_{n+k}$ can be determined with a linear combination using the same parameters α_i and β_i. In IDR(s)-*biortho* α_i is chosen such that the vector $\boldsymbol{g}_{n+k}$ is orthogonal to $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_{k-1}$ and β_i such that the intermediate residual $\boldsymbol{r}_{n+k+1}$ is orthogonal to $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_k$.

5.3 Preconditioners

Applying an iterative method directly to a system of equations $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ may lead to convergence which is not fast enough. Therefore one can apply a preconditioner $\boldsymbol{K}$ to the system of equations, such that the preconditioned system of equations has better properties than the original system of equations. Due to the better properties of the preconditioned system less iterations are needed for a good approximation of the solution $\boldsymbol{x}$. As described by van der Vorst (1992) a good preconditioner $\boldsymbol{K}$ satisfies the following properties

- $\boldsymbol{K}$ is a good approximation to $\boldsymbol{A}$ in some sense.
- The cost of the construction of $\boldsymbol{K}$ is not prohibitive.
- The system $\boldsymbol{K}\boldsymbol{y} = \boldsymbol{z}$ is much easier to solve than the original system.

One can use left preconditioning

$$\boldsymbol{K}^{-1}\boldsymbol{A}\boldsymbol{x} = \boldsymbol{K}^{-1}\boldsymbol{b},$$

right preconditioning

$$\boldsymbol{A}\boldsymbol{K}^{-1}\boldsymbol{y} = \boldsymbol{b} \quad \text{with} \quad \boldsymbol{y} = \boldsymbol{K}\boldsymbol{x},$$

or a combination of both.

In this section we present the incomplete LU decomposition and the Shifted Laplace preconditioner. Erlangga et al. (2004) showed for a Helmholtz problem that replacing the incomplete LU decomposition of $\boldsymbol{A}$ as a preconditioner by the the incomplete LU decomposition of some shifted Helmholtz matrix $\boldsymbol{K}$ leads to less iterations to reach convergence. This shifted matrix $\boldsymbol{K}$ is denoted as the shifted Laplace preconditioner.

5.3.1 Incomplete LU decomposition

In HARES the incomplete LU factorization (ILU) is used as a preconditioner of matrix $\boldsymbol{S}$, ILU is a variant of Gaussian elimination where some elements in the LU factorization are discarded. Gaussian elimination of a matrix results in a $\boldsymbol{LU}$ factorization, where $\boldsymbol{L}$ is a lower triangular matrix and $\boldsymbol{U}$ an upper triangular matrix. The LU decomposition without fill-in (ILU(0)) preserves the zero pattern of $\boldsymbol{A}$ in the matrices $\boldsymbol{L}$ and $\boldsymbol{U}$, i.e. if $a_{i,j} = 0$ $(1 \leq i, j \leq N)$ for a certain combination (i, j) then $u_{i,j} = l_{i,j} = 0$ and if $a_{i,j} \neq 0$ then $u_{i,j} \neq 0$ and $l_{i,j} \neq 0$. The diagonal of $\boldsymbol{L}$ is set equal to one, i.e. $l_{i,i} = 1$, and the diagonal of $\boldsymbol{U}$ is determined in the ILU(0) algorithm. In general it is impossible to match $\boldsymbol{A}$ with $\boldsymbol{LU}$ when $\boldsymbol{L}$ and $\boldsymbol{U}$ have the same zero-pattern as $\boldsymbol{A}$. The extra elements of $\boldsymbol{LU}$ are called the fill-in elements. Matrix $\boldsymbol{A}$ can be written as

$$\boldsymbol{A} = \boldsymbol{LU} - \boldsymbol{R},$$

where matrix $\boldsymbol{R}$ is the residual of the factorization. The residual matrix $\boldsymbol{R}$ contains the fill-in elements and the ILU(0) preconditioner is given by $\boldsymbol{K} = \boldsymbol{LU}$.

The ILU preconditioner $\boldsymbol{K} = \boldsymbol{LU}$ is available in factored form, therefore in general the preconditioner is applied as a combination of left and right preconditioning. The preconditioned system of $\boldsymbol{Ax} = \boldsymbol{b}$ is given by

$$\boldsymbol{L}^{-1}\boldsymbol{A}\boldsymbol{U}^{-1}\boldsymbol{y} = \boldsymbol{L}^{-1}\boldsymbol{b} \quad \text{with} \quad \boldsymbol{y} = \boldsymbol{U}\boldsymbol{x}.$$

Denote the set of non-zero elements of matrix $\boldsymbol{A}$ as $NZ(\boldsymbol{A})$, i.e. the set of pairs (i, j), $1 \leq i, j \leq N$ such that $a_{i,j} \neq 0$. The incomplete factorization of matrix $\boldsymbol{A}$ is determined such that the elements of $\boldsymbol{A} - \boldsymbol{LU}$ are zero in the elements of $NZ(\boldsymbol{A})$. The algorithm of the incomplete LU factorization is given in algorithm 5.3.

ILU(0)

```
1.    for i = 1, 2, ..., n do
2.       for k = 1, ..., i − 1 and for (i, k) ∈ NZ(A) do
3.          a_ik = a_ik / a_kk;
4.          for j = k + 1, ..., n and for (i, j) ∈ NZ(A) do
5.             a_ij = a_ij − a_ik a_kj;
7.          end
8.       end
9.    end
```

Alg. 5.3: *Incomplete LU factorization algorithm.*

5.3.2 Shifted Laplace preconditioner

Erlangga et al. (2004) presented the shifted Laplace preconditioner as a suitable preconditioner for the Helmholtz equation. The standard form of the Helmholtz equation is given by

$$-\Delta u - k^2 u = f,$$

on a certain domain Ω and with suitable boundary conditions. With $\Delta = \partial^2/\partial x^2 + \partial^2/\partial y^2$, the Laplace operator, and k the wave number. Discretization of this Helmholtz equation leads to the system of equations of the form

$$\boldsymbol{Ax} = (\boldsymbol{L} - k^2\boldsymbol{M} + \boldsymbol{C})\boldsymbol{x} = \boldsymbol{b}.$$

Matrix $\boldsymbol{L}$ is the discretization of $-\Delta u$, $\boldsymbol{C}$ corresponds to the boundary conditions and $\boldsymbol{M}$ the mass matrix. The shifted Laplace preconditioner $\boldsymbol{K}$ is obtained by the discretization of the shifted Helmholtz equation

$$-\Delta u + \xi^2 u = f.$$

Where ξ^2 is denoted as the shift parameter and can be chosen freely, both real and complex. This results in the following matrix

$$\boldsymbol{K} = \boldsymbol{L} + \xi^2 \boldsymbol{M} + \boldsymbol{C}.$$

The preconditioned system of equations is now given by

$$(\boldsymbol{L} + \xi^2 \boldsymbol{M} + \boldsymbol{C})^{-1}(\boldsymbol{L} - k^2 \boldsymbol{M} + \boldsymbol{C})\boldsymbol{x} = (\boldsymbol{L} + \xi^2 \boldsymbol{M} + \boldsymbol{C})^{-1}\boldsymbol{b}.$$

The preconditioner $\boldsymbol{K}$ has the same sparsity pattern as global matrix $\boldsymbol{A}$, hence computing the complete LU decomposition can be very expensive. Therefore its incomplete LU decomposition is often determined, which can be easily computed.

The shifted Laplace preconditioner for the Mild-Slope equation

The non-linear Mild-Slope equation is very similar to the damped Helmholtz equation, which suggests that the shifted Laplace preconditioner is a suitable choice for the Mild-Slope equation. Application of the Ritz-Galerkin finite element method to the non-linear Mild-Slope equation results in the linear system of equations $\boldsymbol{S\zeta} = \boldsymbol{b}$, see chapter 4. The contribution of an internal element to the global matrix $\boldsymbol{S}$ is given by the following element matrix

$$\boldsymbol{S}_i^e = -\frac{n_0}{k_0^2}\boldsymbol{L}^e + \left(n_0 - \frac{iW}{\omega}\right)\boldsymbol{M}^e.$$

For details on matrices $\boldsymbol{L}^e$ and $\boldsymbol{M}^e$ we refer to chapter 6. Note that the coefficients n_0, k_0 and W are not constant on the domain. Matrices $\boldsymbol{L}^e$ and $\boldsymbol{M}^e$ have a different value per element, since their expressions depend on the coefficients of the basis functions in the finite element method. The contribution of a boundary element to the global matrix $\boldsymbol{S}$ is given by

$$\boldsymbol{S}_b^e = -\frac{in_0}{k_0^2}\left(\frac{1-R}{1+R}\right)\boldsymbol{C}^e.$$

Since the coefficients in the Mild-Slope equation are not constant, it does not seem reasonable to choose a constant shift for all the elements. Hence we propose the following shifted element matrix for the internal elements

$$\boldsymbol{K}_i^e = -\frac{n_0}{k_0^2}\boldsymbol{L}^e - \xi^2 \boldsymbol{M}^e.$$

The contribution from the boundary elements to the preconditioner can, for instance, be computed by

$$\boldsymbol{K}_b^e = \boldsymbol{S}_b^e = -\frac{in_0}{k_0^2}\left(\frac{1-R}{1+R}\right)\boldsymbol{C}^e.$$

However, a different choice can also be made. From these shifted element matrices $\boldsymbol{K}_i^e$ and $\boldsymbol{K}_b^e$ we can compute the global preconditioner $\boldsymbol{K}$.

5.4 Summary of the methods to solve the system of equations

In this chapter we presented three methods for solving the linear system of equations $\boldsymbol{S\zeta} = \boldsymbol{b}$ obtained after the discretization of the non-linear Mild-Slope equation. The direct method `MUMPS`, the Krylov subspace methods Bi-CGSTAB and IDR(s) and the incomplete LU decomposition and the shifted Laplace preconditioner has been discussed. The current implementation of HARES uses Bi-CGSTAB preconditioned with the incomplete LU decomposition. To improve the computational time for solving the system of equations we propose the direct method `MUMPS` and the Krylov subspace method IDR(s) preconditioned with the shifted Laplace preconditioner.

Chapter 6

Bounds on the eigenvalue range using element-by-element estimates

In chapter 5 we proposed the Krylov subspace method IDR(s) combined with the shifted Laplace preconditioner as an improvement of the current implementation Bi-CGSTAB preconditioned with the incomplete LU decomposition.

The element shift coefficient ξ^2 in the shifted Laplace preconditioner, see section 5.3.2, can be chosen freely. We would like to choose it such that the fastest convergence for the iterative method is obtained. Estimating the eigenvalues of the preconditioned system $\boldsymbol{K}^{-1}\boldsymbol{S}$ is necessary to be able to say something about the convergence. A bound on the eigenvalues is not only interesting for the convergence of the iterative method. But we might also be able to smartly choose the coefficient ω in the IDR(s) algorithm using Chebyshev polynomials.

In this chapter we determine a bound on the eigenvalues of the preconditioned system, using an element-by-element technique as presented by Loghin et al. (2006). In section 6.1 we present the element matrices that are obtained using the Ritz-Galerkin finite element method and Gaussian integration. In sections 6.2 and 6.3 we describe the theory needed to compute a bound for the eigenvalue estimate. Section 6.4 discusses the eigenvalue estimate for the non-linear Mild-Slope equation using the shifted Laplace preconditioner. In section 6.5 we present the theory of choosing the coefficient ω based on Chebyshev polynomials.

6.1 Element matrices

6.1.1 Internal elements

Applying the Ritz-Galerkin finite element method to the non-linear Mild-Slope equation and using exact integration gives the following element matrix for an internal element, see equation (4.2.5) in section 4.2.1 and equation (4.3.3) in section 4.3.1.

$$\boldsymbol{S}_i^e = -\frac{n_0}{k_0^2}\boldsymbol{L}^e + \left(n_0 - \frac{iW}{\omega}\right)\boldsymbol{M}^e, \tag{6.1.1}$$

with $n_0 > 0$, $k_0 > 0$ and

$$\boldsymbol{L}^e = \frac{|\Delta|}{2}\begin{bmatrix} \beta_1^2+\gamma_1^2 & \beta_1\beta_2+\gamma_1\gamma_2 & \beta_1\beta_3+\gamma_1\gamma_3 \\ \beta_1\beta_2+\gamma_1\gamma_2 & \beta_2^2+\gamma_2^2 & \beta_2\beta_3+\gamma_2\gamma_3 \\ \beta_1\beta_3+\gamma_1\gamma_3 & \beta_2\beta_3+\gamma_2\gamma_3 & \beta_3^2+\gamma_3^2 \end{bmatrix} \quad \text{and} \quad \boldsymbol{M}^e = \frac{|\Delta|}{2}\begin{bmatrix} \frac{1}{6} & \frac{1}{12} & \frac{1}{12} \\ \frac{1}{12} & \frac{1}{6} & \frac{1}{12} \\ \frac{1}{12} & \frac{1}{12} & \frac{1}{6} \end{bmatrix}.$$

Where $\frac{|\Delta|}{2}$ denotes the size of the triangular element and β_j and γ_j are the coefficients of basis function ψ_j $(j = 1, 2, 3)$. These coefficients are given by

$$\begin{aligned} \beta_1 &= \frac{1}{\Delta}(y_2 - y_3), \qquad \beta_2 = \frac{1}{\Delta}(y_3 - y_1), \qquad \beta_3 = \frac{1}{\Delta}(y_1 - y_2) \\ \gamma_1 &= \frac{1}{\Delta}(x_3 - x_2), \qquad \gamma_2 = \frac{1}{\Delta}(x_1 - x_3), \qquad \gamma_3 = \frac{1}{\Delta}(x_2 - x_1) \end{aligned} \tag{6.1.2}$$

With these expressions for $\boldsymbol{L}^e$ and $\boldsymbol{M}^e$ it is easily checked that $\boldsymbol{S}_i^e$ is a symmetric but not an Hermitian matrix.

6.1.2 Boundary elements

For the boundary elements (both the closed and open boundary) we get the following expression for the element matrices

$$\boldsymbol{S}_b^e = -\frac{in_0}{k_0^2}\left(\frac{1-R}{1+R}\right)\boldsymbol{C}^e, \tag{6.1.3}$$

with $R = 0$ for the elements on the open boundary and $0 \leq R \leq 1$ for the elements on the closed boundary. Element matrix $\boldsymbol{C}^e$ is given by

$$\boldsymbol{C}^e = k_0\sqrt{1 - \frac{iW}{\omega n_0}}\|\Delta x\|_2 \begin{bmatrix} \frac{1}{3} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{3} \end{bmatrix} - \frac{1}{2k_0\sqrt{1 - \frac{iW}{\omega n_0}}\|\Delta x\|_2} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}.$$

The term $\sqrt{1 - \frac{iW}{\omega n_0}}$ can also be written as

$$\sqrt{1 - \frac{iW}{\omega n_0}} = \sqrt{\frac{\sqrt{1 + \left(\frac{W}{\omega n_0}\right)^2} + 1}{2}} - i\sqrt{\frac{\sqrt{1 + \left(\frac{W}{\omega n_0}\right)^2} - 1}{2}} = a - ib, \tag{6.1.4}$$

with $a, b \in \mathbb{R}$. Rewriting the expression for $\boldsymbol{C}^e$ gives

$$\boldsymbol{C}^e = k_0(a - ib)\|\Delta x\|_2 \begin{bmatrix} \frac{1}{3} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{3} \end{bmatrix} - \frac{a + ib}{2k_0(a^2 + b^2)\|\Delta x\|_2} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}. \tag{6.1.5}$$

6.2 Field of values of a matrix

The spectrum of a matrix $\boldsymbol{A}$ is contained in its field of values. Hence knowing a bound on the field of values automatically results in a bound on the spectrum. The location of the eigenvalues of a (preconditioned) matrix is important for the choice of numerical method and we can determine an upper bound for the needed iterations until convergence.

Let $\boldsymbol{A}$ be a general square matrix of order N. The *field of values* of matrix $\boldsymbol{A}$ is defined as

$$FOV(\boldsymbol{A}) = \left\{ \frac{\boldsymbol{x}^H \boldsymbol{A}\boldsymbol{x}}{\boldsymbol{x}^H \boldsymbol{x}},\ \boldsymbol{x} \in \mathbb{C}^N,\ \boldsymbol{x} \neq 0 \right\}. \tag{6.2.1}$$

The *generalized field of values* for a matrix pair $\boldsymbol{A}$, $\boldsymbol{B}$ with $\boldsymbol{B}$ non-singular is given by

$$FOV(\boldsymbol{A}, \boldsymbol{B}) = \left\{ \frac{\boldsymbol{x}^H \boldsymbol{A}\boldsymbol{x}}{\boldsymbol{x}^H \boldsymbol{B}\boldsymbol{x}},\ \boldsymbol{x} \in \mathbb{C}^N,\ \boldsymbol{x} \neq 0 \right\}. \tag{6.2.2}$$

The set of eigenvalues $\lambda^{\boldsymbol{A},\boldsymbol{B}}$ of the generalized problem $\boldsymbol{A}\boldsymbol{x} = \lambda\boldsymbol{B}\boldsymbol{x}$ is contained in the field of values $FOV(\boldsymbol{A}, \boldsymbol{B})$, which follows by taking $\boldsymbol{x}$ to be an eigenvector of the generalized problem. A bound on the generalized field of values of the matrix pair $\boldsymbol{A}$, $\boldsymbol{B}$ is also a bound on the spectrum of the generalized problem. When the preconditioner $\boldsymbol{B}$ is Hermitian positive definite (or Hermitian

negative definite) its Cholesky decomposition exists such that $\boldsymbol{B} = \boldsymbol{C}\boldsymbol{C}^H$. The generalized field of values $FOV(\boldsymbol{A}, \boldsymbol{B})$ can then be written as

$$FOV(\boldsymbol{A}, \boldsymbol{B}) = FOV(\boldsymbol{C}^{-1}\boldsymbol{A}\boldsymbol{C}^{-H}).$$

The generalized Rayleigh quotient $R^{\boldsymbol{A},\boldsymbol{B}}$ for $\boldsymbol{A}$ Hermitian and $\boldsymbol{B}$ Hermitian positive definite (negative definite) is defined as

$$R^{\boldsymbol{A},\boldsymbol{B}}(\boldsymbol{x}) = \frac{\boldsymbol{x}^H\boldsymbol{A}\boldsymbol{x}}{\boldsymbol{x}^H\boldsymbol{B}\boldsymbol{x}}, \quad \forall \boldsymbol{x} \neq 0,$$

with the property

$$\lambda_{\min}^{\boldsymbol{A},\boldsymbol{B}} \leq R^{\boldsymbol{A},\boldsymbol{B}} \leq \lambda_{\max}^{\boldsymbol{A},\boldsymbol{B}}, \quad \forall \boldsymbol{x} \neq 0.$$

Using the relation between the Rayleigh quotient and the field of values we have

$$\lambda_{\min}^{\boldsymbol{A},\boldsymbol{B}} \leq z \leq \lambda_{\max}^{\boldsymbol{A},\boldsymbol{B}} \quad \forall\ z \in FOV(\boldsymbol{A}, \boldsymbol{B}),$$

if $\boldsymbol{A}$ is Hermitian and $\boldsymbol{B}$ Hermitian positive definite.

6.3 Bound on the field of values

Applying the finite element method to the Mild-Slope equation results in a dense element matrix $\boldsymbol{A}^e \in \mathbb{C}^{n^e \times n^e}$ of element e which has to be mapped into the global matrix $\boldsymbol{A} \in \mathbb{C}^{N \times N}$. Let $\boldsymbol{N}^e$ denote an $n^e \times N$ boolean matrix that maps the global vector $\boldsymbol{x}$ of variables into the vector $\boldsymbol{x}^e$ of variables associated with element e, i.e.

$$\boldsymbol{x}^e = \boldsymbol{N}^e\boldsymbol{x}. \tag{6.3.1}$$

The global matrix $\boldsymbol{A}$ can be determined with the following mapping

$$\boldsymbol{A} = \sum_{e=1}^{n_e} \boldsymbol{N}^{e^T}\boldsymbol{A}^e\boldsymbol{N}^e, \tag{6.3.2}$$

where n_e denotes the number of elements. Using expressions (6.3.1) and (6.3.2) we get

$$\boldsymbol{x}^H\boldsymbol{A}\boldsymbol{x} = \boldsymbol{x}^H\left(\sum_{e=1}^{n_e} \boldsymbol{N}^{e^T}\boldsymbol{A}^e\boldsymbol{N}^e\right)\boldsymbol{x} = \sum_{e=1}^{n_e} \boldsymbol{x}^{e^H}\boldsymbol{A}^e\boldsymbol{x}^e. \tag{6.3.3}$$

To be able to determine a bound on the field of values we need two theorems presented by Loghin et al. (2006). Theorem 3 gives the relation between the largest (resp. smallest) eigenvalue of all the preconditioned element matrices and the largest (resp. smallest) eigenvalue of the global generalized eigenvalue problem.

Theorem 3. *Let $\boldsymbol{A}^e$, $e = 1, 2, \ldots, n_e$ be Hermitian and $\boldsymbol{B}^e$, $e = 1, \ldots, n_e$ be Hermitian positive definite element matrices and let $\boldsymbol{A}$ and $\boldsymbol{B}$ be the global matrices that are assembled from these element matrices. Let ω be the smallest eigenvalue of all element matrix pairs $\boldsymbol{A}^e$, $\boldsymbol{B}^e$, i.e.*

$$\omega = \min_e \lambda_{\min}^{\boldsymbol{A}^e,\boldsymbol{B}^e},$$

and let Ω be the largest eigenvalue, i.e.

$$\Omega = \max_e \lambda_{\max}^{\boldsymbol{A}^e,\boldsymbol{B}^e}.$$

Then the following bounds hold for the eigenvalues $\lambda^{\boldsymbol{A},\boldsymbol{B}}$ of the global eigenvalue problem $\boldsymbol{A}\boldsymbol{x} = \lambda\boldsymbol{B}\boldsymbol{x}$:

$$\omega \leq \lambda^{\boldsymbol{A},\boldsymbol{B}} \leq \Omega.$$

Proof. By the Rayleigh quotient property for element matrices we have

$$\omega \leq \frac{\boldsymbol{x}^{e^H}\boldsymbol{A}^e\boldsymbol{x}^e}{\boldsymbol{x}^{e^H}\boldsymbol{B}^e\boldsymbol{x}^e} \leq \Omega \qquad \forall\ e,\ \boldsymbol{x}^e \neq 0,$$

and hence also

$$\omega \boldsymbol{x}^{e^H}\boldsymbol{B}^e\boldsymbol{x}^e \leq \boldsymbol{x}^{e^H}\boldsymbol{A}^e\boldsymbol{x}^e \leq \Omega \boldsymbol{x}^{e^H}\boldsymbol{B}^e\boldsymbol{x}^e, \qquad \forall\ e,\ \boldsymbol{x}^e. \tag{6.3.4}$$

This bounds holds for any $\boldsymbol{x}^e$ (even $\boldsymbol{x}^e = 0$), hence also for element vectors generated from any global vector $\boldsymbol{x}$ through $\boldsymbol{x}^e = \boldsymbol{N}^e\boldsymbol{x}$. Substituting this into expression (6.3.4) gives

$$\omega \boldsymbol{x}^H\boldsymbol{N}^{e^T}\boldsymbol{B}^e\boldsymbol{N}^e\boldsymbol{x} \leq \boldsymbol{x}^H\boldsymbol{N}^{e^T}\boldsymbol{A}^e\boldsymbol{N}^e\boldsymbol{x} \leq \Omega \boldsymbol{x}^H\boldsymbol{N}^{e^T}\boldsymbol{B}^e\boldsymbol{N}^e\boldsymbol{x} \qquad \forall\ e,\ \boldsymbol{x}.$$

Applying the assembly operator (6.3.3) gives

$$\omega \boldsymbol{x}^H\boldsymbol{B}\boldsymbol{x} \leq \boldsymbol{x}^H\boldsymbol{A}\boldsymbol{x} \leq \Omega \boldsymbol{x}^H\boldsymbol{B}\boldsymbol{x} \qquad \forall\ \boldsymbol{x}.$$

Since it is assumed that $\boldsymbol{B}$ is positive definite we can divide by $\boldsymbol{x}^H\boldsymbol{B}\boldsymbol{x}$, which results in

$$\omega \leq \frac{\boldsymbol{x}^H\boldsymbol{A}\boldsymbol{x}}{\boldsymbol{x}^H\boldsymbol{B}\boldsymbol{x}} \leq \Omega \qquad \forall\ \boldsymbol{x} \neq 0.$$

By the Rayleigh quotient property for global matrices we get the desired result. □

Theorem 4 gives a bound on the generalized eigenvalue problem when the element matrices are possibly non-Hermitian.

Theorem 4. *Let $\boldsymbol{A}^e$, $e = 1, \ldots, n_e$ be (possibly non-Hermitian) element matrices and $\boldsymbol{B}^e$, $e = 1, \ldots, n_e$ be Hermitian positive definite element matrices and let $\boldsymbol{A}$ and $\boldsymbol{B}$ be the global matrices that are assembled from these element matrices. Let ω_R (resp. ω_I) be the smallest eigenvalue of all the element matrix pairs $\Re(\boldsymbol{A}^e), \boldsymbol{B}^e$, (resp. $\Im(\boldsymbol{A}^e), \boldsymbol{B}^e$), i.e.*

$$\omega_R = \min_e \lambda_{\min}^{\Re(\boldsymbol{A}^e),\boldsymbol{B}^e}, \qquad \omega_I = \min_e \lambda_{\min}^{\Im(\boldsymbol{A}^e),\boldsymbol{B}^e},$$

and let Ω_R (resp. Ω_I) be the largest eigenvalue

$$\Omega_R = \max_e \lambda_{\max}^{\Re(\boldsymbol{A}^e),\boldsymbol{B}^e}, \qquad \Omega_I = \max_e \lambda_{\max}^{\Im(\boldsymbol{A}^e),\boldsymbol{B}^e}.$$

Then the following bounds hold for $z \in FOV(\boldsymbol{A}, \boldsymbol{B})$:

$$\omega_R \leq Re(z) \leq \Omega_R,$$
$$\omega_I \leq Im(z) \leq \Omega_I.$$

Proof. Any non-Hermitian matrix $\boldsymbol{A}^e$ can be split into two Hermitian matrices, i.e.

$$\boldsymbol{A}^e = \frac{1}{2}\left(\boldsymbol{A}^e + \boldsymbol{A}^{e^H}\right) + i\frac{1}{2i}\left(\boldsymbol{A}^e - \boldsymbol{A}^{e^H}\right) = \Re(\boldsymbol{A}^e) + i\Im(\boldsymbol{A}^e).$$

For the generalized field of values we obtain

$$FOV(\boldsymbol{A}^e, \boldsymbol{B}^e) = FOV(\Re(\boldsymbol{A}^e) + i\Im(\boldsymbol{A}^e), \boldsymbol{B}^e).$$

Since $\Re(\boldsymbol{A}^e)$, $\Im(\boldsymbol{A}^e)$ and $\boldsymbol{B}^e$ are Hermitian matrices, it holds that both $FOV(\Re(\boldsymbol{A}^e), \boldsymbol{B}^e)$ and $FOV(\Im(\boldsymbol{A}^e), \boldsymbol{B}^e)$ are real. Therefore the following projection property holds

$$\mathrm{Re}(FOV(\boldsymbol{A}^e, \boldsymbol{B}^e)) = FOV(\Re(\boldsymbol{A}^e), \boldsymbol{B}^e),$$
$$\mathrm{Im}(FOV(\boldsymbol{A}^e, \boldsymbol{B}^e)) = FOV(\Im(\boldsymbol{A}^e), \boldsymbol{B}^e).$$

Applying theorem 3 to the generalized eigenvalue problems $\Re(\boldsymbol{A}^e), \boldsymbol{B}^e$ and $\Im(\boldsymbol{A}^e), \boldsymbol{B}^e$ gives the desired result. □

6.4 Eigenvalue estimate for the Mild-Slope equation

Using the theory presented in section 6.3 we are able to determine a bound on the field of values of the Mild-Slope equation. In this section we propose the preconditioners for an internal and a boundary element. Using the element-by-element technique presented in Theorem 4, we determine a bound on the eigenvalues of the generalized eigenvalue problem.

6.4.1 Internal elements

Preconditioner

The shifted Laplace preconditioner, presented by Erlangga et al. (2004), gives good results for both the Helmholtz and the damped Helmholtz equation. The Mild-Slope equation with energy dissipation has the same properties as the damped Helmholtz equation. Therefore it seems a reasonable choice to apply the shifted Laplace preconditioner to the Mild-Slope equation.

The shifted Laplace preconditioner for the non-linear Mild-Slope equation is of the form

$$\boldsymbol{K}_i^e = -\frac{n_0}{k_0^2}\boldsymbol{L}^e - \xi^2\boldsymbol{M}^e = -\left(\frac{n_0}{k_0^2}\boldsymbol{L}^e + \xi^2\boldsymbol{M}^e\right), \tag{6.4.1}$$

with $\xi \in \mathbb{R}$. Comparing expression (6.1.1) with expression (6.4.1) we see that only the coefficient in front of the mass matrix $\boldsymbol{M}^e$ is changed, which is denoted as the shift ξ^2. To apply Theorem 4 to the shifted Laplace preconditioner (6.4.1) it has to be an Hermitian positive definite (or negative definite) matrix. Matrices $\boldsymbol{L}^e$ and $\boldsymbol{M}^e$ are symmetric and contain only real elements, therefore is matrix $\boldsymbol{K}_i^e$ symmetric and Hermitian. It rests to show that matrix $\boldsymbol{K}_i^e$ is positive (or negative) definite.

For the matrix $\frac{n_0}{k_0^2}\boldsymbol{L}^e + \xi^2\boldsymbol{M}^e$ to be positive definite, it must hold that

$$\begin{aligned}
& \boldsymbol{x}^T\left(\frac{n_0}{k_0^2}\boldsymbol{L}^e + \xi^2\boldsymbol{M}^e\right)\boldsymbol{x} > 0, \\
\Rightarrow\quad & \frac{n_0}{k_0^2}\boldsymbol{x}^T\boldsymbol{L}^e\boldsymbol{x} + \xi^2\boldsymbol{x}^T\boldsymbol{M}^e\boldsymbol{x} > 0, \\
\Rightarrow\quad & \boldsymbol{x}^T\boldsymbol{L}^e\boldsymbol{x} \geq 0 \quad \text{and} \quad \boldsymbol{x}^T\boldsymbol{M}^e\boldsymbol{x} > 0, \quad \forall \boldsymbol{x} \neq 0,\ \boldsymbol{x} \in \mathbb{R}^3.
\end{aligned}$$

Matrix $\boldsymbol{M}^e$ has the eigenvalues $\lambda_1 = \frac{1}{12}$, $\lambda_2 = \frac{1}{12}$ and $\lambda_3 = \frac{1}{3}$. Hence it is a symmetric positive definite matrix. We continue with demonstrating that matrix $\boldsymbol{L}^e$ is positive semi-definite, i.e. its eigenvalues are larger than or equal to zero. This can be done using the Gerschgorin circle theorem, see Theorem 5.

Theorem 5. *(**Gerschgorin circle theorem**) Let $\boldsymbol{A}$ be an (real or complex) $n \times n$-matrix let r_i denote the sum of the absolute values of the off-diagonal entries in the ith row of $\boldsymbol{A}$; i.e. $r_i = \sum_{j\neq i}|a_{ij}|$. Then every eigenvalue of $\boldsymbol{A}$ is contained within a Gerschgorin disk with center a_{ii} and radius r_i.*

Proof. See Gerschgorin (1931). □

Applying Gerschgorin circle theorem to matrix $\boldsymbol{L}^e$ we obtain three circles with center $\beta_i^2 + \gamma_i^2$ and radius $|\beta_i\beta_j + \gamma_i\gamma_i| + |\beta_i\beta_k + \gamma_i\gamma_k|$ with $i, j, k = 1, 2, 3$ and $j \neq k \neq i$. The center of each circle is located in the first quadrant, hence it is necessary to show that $a_{ii} - r_i \geq 0$ for $\boldsymbol{L}^e$ to be positive semi-definite. We do this by the following computation

$$\begin{aligned}
a_{ii} - r_i &= \beta_i^2 + \gamma_i^2 - (|\beta_i\beta_j + \gamma_i\gamma_j| + |\beta_i\beta_k + \gamma_i\gamma_k|) \\
&\geq \beta_i^2 + \gamma_i^2 - |\beta_i\beta_j + \beta_i\beta_k + \gamma_i\gamma_j + \gamma_i\gamma_k| \\
&= \beta_i^2 + \gamma_i^2 - |-\beta_i^2 - \gamma_i^2| \\
&= \beta_i^2 + \gamma_i^2 - |-1||\beta_i^2 + \gamma_i^2| \\
&= \beta_i^2 + \gamma_i^2 - (\beta_i^2 + \gamma_i^2) = 0.
\end{aligned}$$

Concluding $a_{ii} - r_i \geq 0$, thus matrix $\boldsymbol{L}^e$ is positive semi-definite. Hence it is shown that $\frac{n_0}{k_0^2}\boldsymbol{L}^e + \xi^2\boldsymbol{M}^e$ is symmetric positive definite and thus the shifted Laplace preconditioner $\boldsymbol{K}_i^e$ is symmetric (Hermitian) negative definite.

Eigenvalue estimate

Using the two theorems presented in section 6.3 we are able to determine a bound on the eigenvalues of the generalized eigenvalue problem $\boldsymbol{S}\boldsymbol{x} = \lambda^{\boldsymbol{S},\boldsymbol{K}}\boldsymbol{K}\boldsymbol{x}$. This bound, however, depends on the choice of the shift parameter ξ^2. Smartly choosing this shift can improve the rate of convergence of the system of equations. Knowing the influence of the shift on the field of values, it is possible to determine an optimal value for the shift ξ^2. Hence we need to determine the eigenvalues of the generalized problem $\boldsymbol{S}_i^e\boldsymbol{x}^e = \lambda^{\boldsymbol{S}_i^e,\boldsymbol{K}_i^e}\boldsymbol{K}_i^e\boldsymbol{x}^e$. As noted in section 6.1.1 matrix $\boldsymbol{S}_i^e$ is not an Hermitian matrix, but it can be written in a Hermitian real and imaginary part, i.e. $\boldsymbol{S}_i^e = \Re(\boldsymbol{S}_i^e) + i\Im(\boldsymbol{S}_i^e)$. For details see the proof of Theorem 4). The expressions for $\Re(\boldsymbol{S}_i^e)$ and $\Im(\boldsymbol{S}_i^e)$ are given by

$$\Re(\boldsymbol{S}_i^e) = -\frac{n_0}{k_0^2}\boldsymbol{L}^e + n_0\boldsymbol{M}^e \quad \text{and} \quad \Im(\boldsymbol{S}_i^e) = -\frac{W}{\omega}\boldsymbol{M}^e.$$

Hence we need to solve the following two element eigenvalue problems

$$\left(-\frac{n_0}{k_0^2}\boldsymbol{L}^e + n_0\boldsymbol{M}^e\right)\boldsymbol{x}^e = \lambda^{\Re(\boldsymbol{S}_i^e),\boldsymbol{K}_i^e}\left(-\frac{n_0}{k_0^2}\boldsymbol{L}^e - \xi^2\boldsymbol{M}^e\right)\boldsymbol{x}^e, \tag{6.4.2}$$

and

$$-\frac{W}{\omega}\boldsymbol{M}^e\boldsymbol{x}^e = \lambda^{\Im(\boldsymbol{S}_i^e),\boldsymbol{K}_i^e}\left(-\frac{n_0}{k_0^2}\boldsymbol{L}^e - \xi^2\boldsymbol{M}^e\right)\boldsymbol{x}^e. \tag{6.4.3}$$

To determine the eigenvalues of the generalized eigenvalue problem (6.4.2) we look at its Rayleigh quotient, i.e.

$$\begin{aligned}\frac{\boldsymbol{x}^{e^T}\Re(\boldsymbol{S}_i^e)\boldsymbol{x}^e}{\boldsymbol{x}^{e^T}\boldsymbol{K}_i^e\boldsymbol{x}^e} &= \frac{\boldsymbol{x}^{e^T}\boldsymbol{K}_i^e\boldsymbol{x}^e}{\boldsymbol{x}^{e^T}\boldsymbol{K}_i^e\boldsymbol{x}^e} + \frac{\boldsymbol{x}^{e^T}\left(\Re(\boldsymbol{S}_i^e) - \boldsymbol{K}_i^e\right)\boldsymbol{x}^e}{\boldsymbol{x}^{e^T}\boldsymbol{K}_i^e\boldsymbol{x}^e} \\ &= 1 + \left(n_0 + \xi^2\right)\frac{\boldsymbol{x}^{e^T}\boldsymbol{M}^e\boldsymbol{x}^e}{\boldsymbol{x}^{e^T}\boldsymbol{K}_i^e\boldsymbol{x}^e}.\end{aligned}$$

Hence only the eigenvalues of the generalized problem $\boldsymbol{M}^e\boldsymbol{x}^e = \lambda^{\boldsymbol{M}^e,\boldsymbol{K}_i^e}\boldsymbol{K}_i^e\boldsymbol{x}^e$ need to be determined to obtain the eigenvalues of both the element eigenvalue problems (6.4.2) and (6.4.3). The largest and smallest eigenvalue of the generalized eigenvalue problem are given by (see Appendix B)

$$\lambda_{\max}^{\boldsymbol{M}^e,\boldsymbol{K}_i^e} = \frac{\Delta^2 k_0^2\left(-6n_0\alpha - \xi^2 k_0^2 + 12n_0\sqrt{\frac{\alpha^2}{4} - \frac{3}{\Delta^2}}\right)}{\xi^4 k_0^4\Delta^2 + 432n_0^2 + 12n_0\xi^2 k_0^2\Delta^2\alpha} \quad \text{and} \quad \lambda_{\min}^{\boldsymbol{M}^e,\boldsymbol{K}_i^e} = -\frac{1}{\xi^2}, \tag{6.4.4}$$

with $\alpha = \beta_1^2 + \beta_2^2 + \beta_3^2 + \gamma_1^2 + \gamma_2^2 + \gamma_3^2$. Since matrices $\boldsymbol{M}^e$ and $\boldsymbol{K}_i^e$ are both symmetric positive (resp. negative) definite the eigenvalues of the generalized eigenvalue problem are real. Hence it must hold that $\frac{\alpha^2}{4} - \frac{3}{\Delta^2} > 0$. We obtain the following bounds for the eigenvalues $\lambda_{\max}^{\boldsymbol{M}^e,\boldsymbol{K}_i^e}$ and $\lambda_{min}^{\boldsymbol{M}^e,\boldsymbol{K}_i^e}$

$$-1 < \lambda_{\max}^{\boldsymbol{M}^e,\boldsymbol{K}_i^e} < 0 \qquad \text{and} \qquad -\infty < \lambda_{\min}^{\boldsymbol{M}^e,\boldsymbol{K}_i^e} < 0. \tag{6.4.5}$$

Eigenvalues of equation (6.4.2)
Using the expression found in equation (6.4.4) the minimal and maximal eigenvalue of the gener-

alized eigenvalue problem (6.4.2) are given by

$$\begin{aligned}\lambda_{\max}^{\Re(\boldsymbol{S}_i^e),\boldsymbol{K}_i^e} &= 1 + (n_0 + \xi^2)\lambda_{\max}^{\boldsymbol{M}^e,\boldsymbol{K}_i^e} \\ &= 1 + (n_0 + \xi^2)\frac{\Delta^2 k_0^2\left(-6n_0\alpha - \xi^2 k_0^2 + 12n_0\sqrt{\frac{\alpha^2}{4} - \frac{3}{\Delta}}\right)}{\xi^4 k_0^4\Delta^2 + 432n_0^2 + 12n_0\xi^2 k_0^2\Delta^2\alpha}\end{aligned} \tag{6.4.6}$$

$$\begin{aligned}\lambda_{\min}^{\Re(\boldsymbol{S}_i^e),\boldsymbol{K}_i^e} &= 1 + (n_0 + \xi^2)\lambda_{\min}^{\boldsymbol{M}^e,\boldsymbol{K}_i^e} \\ &= -\frac{n_0}{\xi^2}\end{aligned} \tag{6.4.7}$$

Using the bounds given in expression (6.4.5), we obtain the following bounds for the eigenvalues of $\Re(\boldsymbol{S}_i^e)\boldsymbol{x}_e = \lambda^{\Re(\boldsymbol{S}_i^e),\boldsymbol{K}_i^e}\boldsymbol{K}_i^e\boldsymbol{x}_e$

$$0 < \lambda_{\max}^{\Re(\boldsymbol{S}_i^e),\boldsymbol{K}_i^e} < 1 \qquad \text{and} \qquad -\infty < \lambda_{\min}^{\Re(\boldsymbol{S}_i^e),\boldsymbol{K}_i^e} < 0. \tag{6.4.8}$$

These bounds show that the preconditioned version of $\Re(\boldsymbol{S}_i^e)$ is indefinite, since the minimal and maximal eigenvalues do not have the same sign.

Eigenvalues of equation (6.4.3)
The minimal and maximal eigenvalue of the generalized eigenvalue problem (6.4.3) are given by

$$\begin{aligned}\lambda_{\max}^{\Im(\boldsymbol{S}_i^e),\boldsymbol{K}_i^e} &= -\frac{W}{\omega}\lambda_{\min}^{\boldsymbol{M}^e,\boldsymbol{K}_i^e} \\ &= \frac{W}{\omega\xi^2}\end{aligned} \tag{6.4.9}$$

$$\begin{aligned}\lambda_{\min}^{\Im(\boldsymbol{S}_i^e),\boldsymbol{K}_i^e} &= -\frac{W}{\omega}\lambda_{\max}^{\boldsymbol{M}^e,\boldsymbol{K}_i^e} \\ &= -\frac{W}{\omega}\frac{\Delta^2 k_0^2\left(-6n_0\alpha - \xi^2 k_0^2 + 12n_0\sqrt{\frac{\alpha^2}{4} - \frac{3}{\Delta}}\right)}{\xi^4 k_0^4\Delta^2 + 432n_0^2 + 12n_0\xi^2 k_0^2\Delta^2\alpha}\end{aligned} \tag{6.4.10}$$

Again the bound of these eigenvalues can be determined using the bounds found in expression (6.4.5), hence we obtain

$$0 < \lambda_{\max}^{\Im(\boldsymbol{S}_i^e),\boldsymbol{K}_i^e} < \infty \qquad \text{and} \qquad 0 < \lambda_{\min}^{\Im(\boldsymbol{S}_i^e),\boldsymbol{K}_i^e} < \frac{W}{\omega}. \tag{6.4.11}$$

Both eigenvalues are larger than zero, hence the preconditioned matrix $\Im(\boldsymbol{S}_i^e)$ is positive definite.

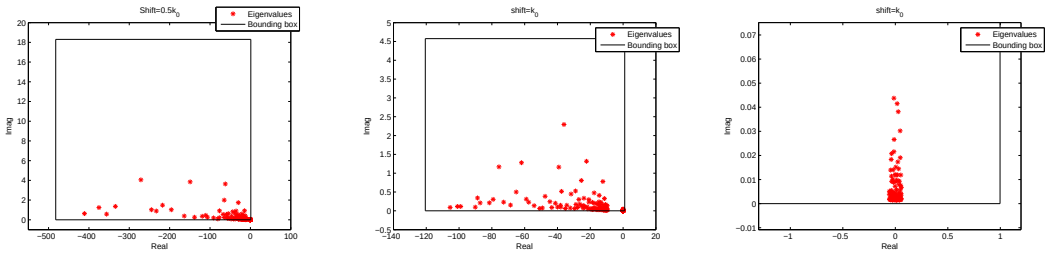
Note that the coefficients W, n_0, Δ, k_0 and α differ for every element, hence it is not directly clear for which elements these eigenvalues are minimal or maximal. But since all coefficients are known for each element, it is easily verified numerically.

To check whether the theory holds we determine the bounding box and the eigenvalues of the the internal elements. We use the test case *"The harbour of Scheveningen"*, see section 7.1. For the shift ξ in the shifted Laplace preconditioner we choose $\xi = 0.5k_0$ and $\xi = k_0$. Using these values we obtain the following values for $\omega_R = \min_e \lambda_{\min}^{\Re(\boldsymbol{S}^e),\boldsymbol{K}^e}$, $\omega_I = \min_e \lambda_{\min}^{\Im(\boldsymbol{S}^e),\boldsymbol{K}^e}$, $\Omega_R = \max_e \lambda_{\max}^{\Re(\boldsymbol{S}^e),\boldsymbol{K}^e}$ and $\Omega_I = \max \lambda_{\max}^{\Im(\boldsymbol{S}^e),\boldsymbol{K}^e}$ presented in table 6.1.

Shift $\xi = 0.5k_0$		Shift $\xi = k_0$	
ω_R	-482.2996	ω_R	-120.5749
Ω_R	0.9979	Ω_R	0.9979
ω_I	7.9933e-7	ω_I	7.9931e-7
Ω_I	18.2997	Ω_I	4.5749

Table 6.1: *The values of the eigenvalues that determine the boundaries of the bounding box for two different values of the shift parameter ξ. This is for the internal elements only.*

Table 6.1 shows that the values for ω_R and Ω_I differ significantly for different values of the shift ξ. While the values of Ω_R and ω_I almost do not change. It is interesting to check whether the actual eigenvalues of the generalized eigenvalue problem $\boldsymbol{Sx} = \lambda \boldsymbol{Kx}$ are located inside the bounding box as the theory describes. Therefore we determine the 100 largest and smallest eigenvalues, the results are shown in figure 6.1.



(a) Shift $\xi = 0.5k_0$ (b) Shift $\xi = k_0$ (c) Shift $\xi = k_0$, zoomed in

Figure 6.1: *The bounding box and the corresponding eigenvalues for the internal elements for two values of the shift parameter ξ.*

Figure 6.1 shows that the eigenvalues of the matrix $\boldsymbol{S}$, assembled from the element matrices $\boldsymbol{S}_i^e$, preconditioned with matrix $\boldsymbol{K}$, assembled from the element preconditioner matrices $\boldsymbol{K}_i^e$, are located inside the bounding box.

6.4.2 Boundary elements

Preconditioner

The boundary conditions are taken into account in the shifted Laplace preconditioner. It is common to use the same contribution of the boundary conditions in the shifted Laplace matrix $\boldsymbol{K}$ as in the global matrix $\boldsymbol{S}$. However, this is not necessary and the contribution for the boundary conditions can be chosen freely. In our case we need it to be Hermitian positive (negative) definite. We like to preserve the properties of the preconditioned internal element matrices. That is the positive definiteness of the imaginary part of the element matrix and the bounds of the eigenvalues.

Combining the expressions (6.1.5) and (6.1.3) we easily see that matrix $\boldsymbol{S}_b^e$ is not an Hermitian matrix, hence we spilt it into its real $\Re(\boldsymbol{S}_b^e)$ and imaginary $\Im(\boldsymbol{S}_b^e)$ part.

$$\Re(\boldsymbol{S}_b^e) = -\frac{n_0}{k_0^2}\left(\frac{1-R}{1+R}\right) b \left\{ k_0 \|\Delta x\|_2 \begin{bmatrix} \frac{1}{3} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{3} \end{bmatrix} + \frac{1}{2k_0(a^2+b^2)\|\Delta x\|_2} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \right\}, \quad (6.4.12)$$

and

$$\Im(\boldsymbol{S}_b^e) = -\frac{n_0}{k_0^2}\left(\frac{1-R}{1+R}\right) a \left\{ k_0 \|\Delta x\|_2 \begin{bmatrix} \frac{1}{3} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{3} \end{bmatrix} - \frac{1}{2k_0(a^2+b^2)\|\Delta x\|_2} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \right\}.$$

It is easily verified that $\Re(\boldsymbol{S}_b^e)$ is symmetric negative definite since the first matrix is positive definite (eigenvalues $\frac{1}{6}$ and $\frac{1}{2}$) and the second matrix is positive semi-definite (eigenvalues 0 and 2). The expression for $\Im(\boldsymbol{S}_b^e)$ is symmetric, however it is in general indefinite. For the internal elements we have the opposite situation, i.e. $\Re(\boldsymbol{S}_i^e)$ is indefinite and $\Im(\boldsymbol{S}_i^e)$ negative definite. Therefore we like to choose a preconditioner which switches the properties of $\Re(\boldsymbol{S}_b^e)$ and $\Im(\boldsymbol{S}_b^e)$. We propose the following preconditioner

$$\begin{aligned}\boldsymbol{K}_b^e &= -i\left(-\frac{n_0}{k_0^2}\left(\frac{1-R}{1+R}\right)a\left\{k_0\|\Delta x\|_2\begin{bmatrix}\frac{1}{3} & \frac{1}{6}\\ \frac{1}{6} & \frac{1}{3}\end{bmatrix}+\frac{1}{2k_0(a^2+b^2)\|\Delta x\|_2}\begin{bmatrix}1 & -1\\ -1 & 1\end{bmatrix}\right\}\right),\\ &= -i\cdot\frac{a}{b}\cdot\Re(\boldsymbol{S}_b^e).\end{aligned} \tag{6.4.13}$$

This results in the following generalized eigenvalue problem

$$\begin{aligned}\boldsymbol{S}_b^e\boldsymbol{x}^e &= \lambda^{\boldsymbol{S}_b^e,\boldsymbol{K}_b^e}\boldsymbol{K}_b^e\boldsymbol{x}^e,\\ &= -i\lambda^{\boldsymbol{S}_b^e,\boldsymbol{K}_b^e}\cdot\frac{a}{b}\cdot\Re(\boldsymbol{S}_b^e)\boldsymbol{x}^e,\\ i\boldsymbol{S}_b^e\boldsymbol{x}^e &= \lambda^{\boldsymbol{S}_b^e,\boldsymbol{K}_b^e}\cdot\frac{a}{b}\cdot\Re(\boldsymbol{S}_b^e)\boldsymbol{x}^e.\end{aligned} \tag{6.4.14}$$

Now we can apply the theory presented in section 6.3 since matrix $\Re(\boldsymbol{S}_b^e)$ is symmetric negative definite.

Eigenvalue estimate

$\boldsymbol{S}_{b,i}^e = i\boldsymbol{S}_b^e$ is not Hermitian, hence we determine the expressions for $\Re(\boldsymbol{S}_b^e)$ and $\Im(\boldsymbol{S}_b^e)$. This gives

$$\Re(\boldsymbol{S}_{b,i}^e) = \frac{n_0}{k_0^2}\left(\frac{1-R}{1+R}\right)a\left\{k_0\|\Delta x\|_2\begin{bmatrix}\frac{1}{3} & \frac{1}{6}\\ \frac{1}{6} & \frac{1}{3}\end{bmatrix}-\frac{1}{2k_0(a^2+b^2)\|\Delta x\|_2}\begin{bmatrix}1 & -1\\ -1 & 1\end{bmatrix}\right\},$$

and

$$\Im(\boldsymbol{S}_{b,i}^e) = -\frac{n_0}{k_0^2}\left(\frac{1-R}{1+R}\right)b\left\{k_0\|\Delta x\|_2\begin{bmatrix}\frac{1}{3} & \frac{1}{6}\\ \frac{1}{6} & \frac{1}{3}\end{bmatrix}+\frac{1}{2k_0(a^2+b^2)\|\Delta x\|_2}\begin{bmatrix}1 & -1\\ -1 & 1\end{bmatrix}\right\}.$$

We need to determine the eigenvalues of the following two generalized eigenvalue problems

$$\Re(_{b,i}^e)\boldsymbol{x}^e = \lambda^{\Re(\boldsymbol{S}_{b,i}^e),\boldsymbol{K}_b^e}\cdot\frac{a}{b}\cdot\Re(\boldsymbol{S}_b^e)\boldsymbol{x}^e, \tag{6.4.15}$$

and

$$\Im(\boldsymbol{S}_{b,i}^e)\boldsymbol{x}^e = \lambda^{\Im(\boldsymbol{S}_{b,i}^e,\boldsymbol{K}_b^e}\cdot\frac{a}{b}\cdot\Re(\boldsymbol{S}_b^e)\boldsymbol{x}^e. \tag{6.4.16}$$

Due to the resemblance of the preconditioner and the matrices $\Re(\boldsymbol{S}_{b,i}^e)$ and $\Im(\boldsymbol{S}_{b,i}^e)$ the eigenvalues are easily determined. For expression (6.4.15) we obtain

$$\lambda_{\max}^{\Re(\boldsymbol{S}_{b,i}^e),\boldsymbol{K}_i^e} = -\frac{(a^2+b^2)k_0^2\|\Delta x\|_2^2-6}{(a^2+b^2)k_0^2\|\Delta x\|_2^2+6} \quad\text{and}\quad \lambda_{\min}^{\Re(\boldsymbol{S}_{b,i}^e),\boldsymbol{K}_b^e} = -1.$$

For $\lambda_{\max}^{\Re(\boldsymbol{S}_{b,i}^e),\boldsymbol{K}_i^e}$ this expression results in the following bound

$$-1 < \lambda_{\max}^{\Re(\boldsymbol{S}_{b,i}^e),\boldsymbol{K}_i^e} < 1.$$

However, with the possible values of the coefficients k_0, a, b and $\|\Delta x\|_2$ it holds that $\lambda_{\max}^{\Re(\boldsymbol{S}_{b,i}^e),\boldsymbol{K}_i^e} > 0$. These bounds for the eigenvalues of the preconditioner matrix $\Re(\boldsymbol{S}_p^e)$ indicate that indeed it is an indefinite matrix.

The eigenvalues of expression (6.4.16) are given by

$$\lambda_{\max}^{\Im(\boldsymbol{S}_{b,i}^e),\boldsymbol{K}_b^e} = \lambda_{\min}^{\Im(\boldsymbol{S}_{b,i}^e),\boldsymbol{K}_b^e} = \frac{b}{a}.$$

To derive a bound on this eigenvalue we look at the expression for a and b, see equation (6.1.4). It holds that $a, b > 0$ and $b < a$, hence we obtain the following bound

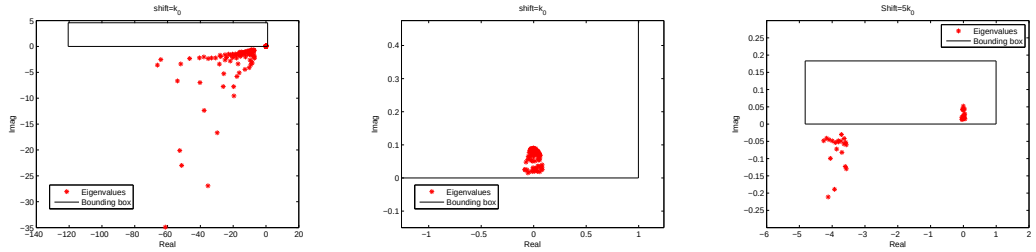
$$0 < \lambda_{\max}^{\Im(\boldsymbol{S}_{b,i}^e),\boldsymbol{K}_b^e}, \lambda_{\min}^{\Im(\boldsymbol{S}_{b,i}^e),\boldsymbol{K}_b^e} < 1.$$

Using this preconditioner for the boundary elements and the shifted Laplace preconditioner for the internal elements, the theory of section 6.3 seems applicable. Even though the actual preconditioner $\boldsymbol{K}_b^e$ is not an Hermitian matrix. However, computing the bounding box, see table 6.2, and the eigenvalues, see figure 6.2, shows that this is not the case. The smallest eigenvalues are located inside the bounding box, but the largest eigenvalues are not included in the bounding box. This indicates that the theory is only applicable when the preconditioner is an Hermitian matrix at the start of the computation and not, after some manipulations, becomes Hermitian.

Shift $\xi = k_0$		Shift $\xi = 5k_0$	
ω_R	-120.5749	ω_R	-4.8230
Ω_R	0.9979	Ω_R	0.9975
ω_I	7.9931e-7	ω_I	7.9892e-7
Ω_I	4.5749	Ω_I	0.1830

Table 6.2: *The values of the eigenvalues that determine the boundaries of the bounding box for two different values of the shift parameter ξ. This is the combination of both the internal and boundary elements.*

Note that the bounds on the eigenvalues are completely determined by the eigenvalues of the internal elements, see table 6.1 for $\xi = k_0$. This indicates that if this preconditioner is used for the boundary elements, the bound on the eigenvalues of the boundary elements lies inside the bound on the eigenvalues of the internal elements.



(a) Shift $\xi = k_0$ (b) Shift $\xi = k_0$, zoomed in (c) Shift $\xi = 5k_0$

Figure 6.2: *The bounding box and the corresponding eigenvalues for both the preconditioned internal elements and boundary elements for two values of the shift parameter ξ.*

This result indicates that it is not possible to obtain either a positive definite $\Re(\boldsymbol{S})$ or a positive definite $\Im(\boldsymbol{S})$ after preconditioning.

Hermitian preconditioner $\boldsymbol{K}_b^e$

To determine whether the theory holds if the preconditioner of the boundary elements is Hermitian positive definite, we choose the preconditioner $\boldsymbol{K}_b^e$ to be of the same form as $\Re(\boldsymbol{S}_b^e)$, see expression

6.4.12. We propose the following preconditioner

$$\boldsymbol{K}_b^e = \frac{a}{b}\Re(\boldsymbol{S}_b^e), \tag{6.4.17}$$

which clearly is Hermitian negative definite. The eigenvalues of the generalized eigenvalue problem $\Re(\boldsymbol{S}_b^e)\boldsymbol{x}^e = \lambda^{\Re(\boldsymbol{S}_b^e),\boldsymbol{K}_b^e}\boldsymbol{K}_b^e\boldsymbol{x}^e$ are given by

$$\lambda_{\min}^{\Re(\boldsymbol{S}_b^e),\boldsymbol{K}_b^e} = \lambda_{\max}^{\Re(\boldsymbol{S}_b^e),\boldsymbol{K}_b^e} = \frac{a}{b}.$$

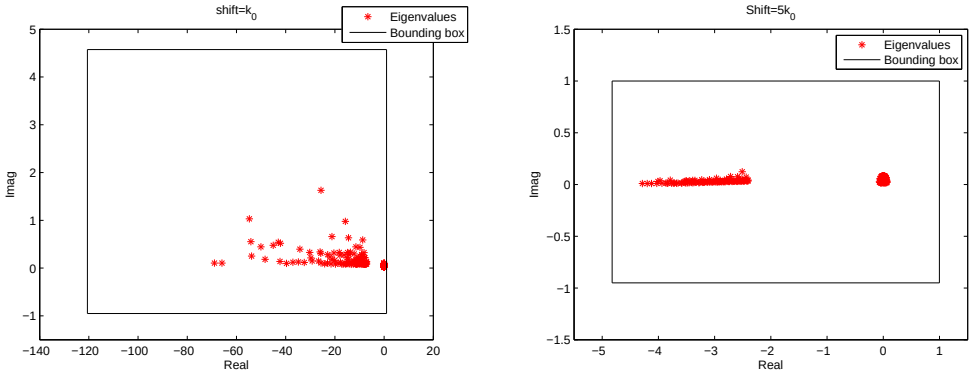
The eigenvalues of the generalized eigenvalue problem $\Im(\boldsymbol{S}_b^e)\boldsymbol{x}^e = \lambda^{\Im(\boldsymbol{S}_b^e),\boldsymbol{K}_b^e}\boldsymbol{K}_b^e\boldsymbol{x}^e$ are given by

$$\lambda_{\min}^{\Im(\boldsymbol{S}_b^e),\boldsymbol{K}_b^e} = \frac{(a^2+b^2)k_0^2\|\Delta x\|_2^2 - 6}{(a^2+b^2)k_0^2\|\Delta x\|_2^2 + 6} \quad \text{and} \quad \lambda_{\max}^{\Im(\boldsymbol{S}_b^e),\boldsymbol{K}_b^e} = 1.$$

Table 6.3 shows the values of the bounding box for the shifts $\xi = k_0$ and $\xi = 5k_0$. Figure 6.3 shows the location of the eigenvalues, both the internal and boundary elements, and the bounding box for this choice of preconditioner.

Shift $\xi = k_0$		Shift $\xi = 5k_0$	
ω_R	-120.5749	ω_R	-4.8230
Ω_R	0.9979	Ω_R	0.9975
ω_I	-0.9500	ω_I	-0.9500
Ω_I	4.5749	Ω_I	1.0000

Table 6.3: *The extreme values of the eigenvalues that determine the boundaries of the bounding box for two different values of the shift parameter ξ. For both the internal and boundary elements.*



(a) Shift $\xi = k_0$ (b) Shift $\xi = 5k_0$

Figure 6.3: *The bounding box and the corresponding eigenvalues for both the preconditioned internal elements and boundary elements for two values of the shift parameter ξ.*

Figure 6.3 shows that the theory of section 6.3 does hold when the preconditioner $\boldsymbol{K}_b^e$ is Hermitian positive (negative) definite from the beginning of the computation. The value for ω_I in table 6.3 is now determined by the eigenvalues of the boundary elements, for Ω_I its value depends on the choice of the shift parameter ξ. For ξ sufficient large, i.e. when Ω_I from the internal elements becomes smaller than one, its value is determined by the eigenvalues of the boundary elements. However, since neither the $\Re(\boldsymbol{S})$ nor the $\Im(\boldsymbol{S})$ part is positive (or negative) definite it is not possible to determine an optimal value of the shift as presented by van Gijzen and Erlangga (2006).

6.5 Choosing the coefficient ω in IDR(s) using Chebyshev polynomials

Chebyshev polynomials are named after the Russian mathematician Pafnuty Lvovich Chebyshev (1821 - 1894) and are of great importance in approximation theory. Let B be an arbitrary, bounded and closed set in the complex plane, then the polynomial deviating least from zero on B is called the Chebyshev polynomial for B. The roots of the Chebyshev polynomial are called Chebyshev nodes and can be taken as points of interpolation. The nth Chebyshev polynomial is a polynomial of degree n with the leading coefficient equal to one.

The generalized problem of finding the nth Chebyshev polynomial for the set B is given by, see Smirnov and Lebedev (1968).
Among the polynomials $P_n(z) = z^n + c_1 z^{n-1} + \ldots + c_n$ ($n \geq 1$) find the polynomial whose maximum absolute value on the set B is minimal, that is if we set

$$\mu_n = \inf_{P_n(z)} \sup_{z \in B} |P_n(z)|$$

then we have to find a polynomial $\Pi_n(z) = z^n + c_1^{(0)} z^{n-1} + c_n^{(0)}$ for which the value of μ_n is attained, i.e. such that

$$\sup_{z \in B} |\Pi_n(z)| = \mu_n.$$

6.5.1 Chebyshev polynomial on a disk

Suppose that B is a disk in the complex plane, i.e. $|z| \leq R$, and we like to determine the nth Chebyshev polynomial on this region. Smirnov and Lebedev (1968) derived Theorem 6, which state that the nth Chebyshev polynomial on a disk $|z| \leq R$ is given by z^n.

Theorem 6. *For the polynomials $P_n(z) = z^n + c_1 z^{n-1} + \ldots + c_n$ of degree $n \leq 1$, we have*

$$\mu_n = \inf_{P_n(z)} \sup_{|z| \leq R} |P_n(z)| = R^n,$$

and μ_n is attained only for the polynomial $\Pi_n(z) = z^n$.

Proof. Write the polynomial $P_n(z)$ as

$$\begin{aligned} P_n(z) &= z^n + c_1 z^{n-1} + \ldots + c_n \\ &= z^n \left(1 + \frac{c_1}{z} + \ldots + \frac{c_n}{z^n}\right) \\ &= z^n \varphi(z). \end{aligned} \tag{6.5.1}$$

Note that the function $\varphi(z)$ is an analytic function on the region $|z| \geq R$. Applying twice the maximum principle (the maximum of a function in a domain is to be found on the boundary of that domain, otherwise the function is constant) the following result is obtained

$$\sup_{|z| \leq R} |P_n(z)| = \sup_{|z|=R} |P_n(z)| = \sup_{|z|=R} R^n |\varphi(z)| = R^n \sup_{|z| \geq R} |\varphi(z)| \geq R^n |\varphi(\infty)| = R^n.$$

The equality holds if and only if the function $\varphi(z)$ is a constant, i.e., if and only if $\varphi(z) = \varphi(\infty) = 1$. Substituting this result for $\varphi(z)$ into expression 6.5.1 gives the desired result for the nth Chebyshev polynomial on the disk $|z| \leq R$. □

Faber et al. (2010) generalise this result for a disk with center $\gamma \in \mathbb{C}$ in a complex plane. Hence we are looking for the nth Chebyshev polynomial in the region $|z - \gamma| \leq R$, which is given by $\Pi_n(z) = (z - \gamma)^n$. The proof that this is the nth Chebyshev polynomial of the disk $|z - \gamma| \leq R$

is analogous to the proof of Theorem 6 by substituting $(z-\gamma)^n$ for z^n. The Chebyshev nodes for this Chebyshev polynomial are given by $z = \gamma$.

We can apply this theory to minimize the polynomial $Q_j(\boldsymbol{A})$ in the residuals of IDR(s). The residuals of IDR(s) can be written as

$$\boldsymbol{r}_n = \Omega_j(\boldsymbol{A})\Psi_{n-j}(\boldsymbol{A})\boldsymbol{r}_0,$$

with $\Omega_j(t) = (1-\omega_j t)(1-\omega_{j-1}t)\dots(1-\omega_1 t)$ and $\Psi_{n-j}(\boldsymbol{A})$ such that $\boldsymbol{p}_k^H\Omega_l(\boldsymbol{A})\Psi_{n-j}(\boldsymbol{A})\boldsymbol{r}_0 = 0$ for $k = 1, 2, \dots, s$ and $l = 0, 1, \dots, j-1$. The roots t_j of $\Omega_j(t)$ are given by $t_j = 1/\omega_j$, hence choosing $\omega_j = 1/\gamma$ results in the polynomial $\Omega_j(t)$ with the same roots as the jth Chebyshev polynomial on a disk in the complex plane.

The residuals of IDR(s) are very similar to the residuals in Richardson iteration, see van der Vorst (2003). The Richardson residual are given by

$$\|\boldsymbol{r}^{k+1}\| = \|(\boldsymbol{I} - \omega\boldsymbol{A})\boldsymbol{r}^k\| \leq \|\boldsymbol{I} - \omega\boldsymbol{A}\|\|\boldsymbol{r}^k\|.$$

The method converges if $\|\boldsymbol{I} - \omega\boldsymbol{A}\| \leq 1$ or $|1 - \omega\lambda_j| \leq 1$, where λ_j is an eigenvalue of matrix $\boldsymbol{A}$. Suppose that the eigenvalues are contained in a disk in the complex plane with center γ and radius R_γ, i.e. we know that $\lambda_j \in (Re^{it} + \gamma)$ with $i = \sqrt{-1}$ and $t \in [0, 2\pi]$. We choose the coefficient $\omega = 1/\gamma$. Now we can make the following estimate

$$\left|1 - \frac{1}{\gamma}(Re^{it} + \gamma)\right| = \left|1 - \frac{R}{\gamma}e^{it} - 1\right| = \left|\frac{R}{\gamma}e^{it}\right| \leq \left|\frac{R}{\gamma}\right|.$$

We have convergence when $|R/\gamma| < 1$ or $R < |\gamma|$. This indicates that the origin cannot be contained in the eigenvalue range of matrix $\boldsymbol{A}$, since then we would have that $R > |\gamma|$.

We have implemented this choice for ω and determined the solution of the system of equations $\boldsymbol{S}\boldsymbol{\zeta} = \boldsymbol{b}$ using IDR(s). However, the results were not an improvement to the current implementation of the coefficient ω. This can be explained since the origin is contained in the eigenvalues estimate obtained is section 6.4.

6.6 Summary of the eigenvalue estimate

In this chapter we presented a method to determine a bounding box around the eigenvalues using an element-by-element technique. The bound on the eigenvalues of the global matrix $\boldsymbol{S}$ of the non-linear Mild-Slope equation preconditioned with the shifted Laplace preconditioner has been determined. Using this bound one can determine an optimal shift ξ in the shifted Laplace preconditioner. Unfortunately, the preconditioned matrix $\boldsymbol{S}$ does not have a positive (negative) definite real part or a positive (negative) definite imaginary part. Hence we were not able to determine an optimal shift for the non-linear Mild-Slope equation. However, we have obtained a bounding box around the eigenvalues. Using this bounding box we were able to test whether choosing the coefficient ω in the IDR(s) algorithm based on Chebyshev polynomials will improve the computational time to solve the system of equations.

Chapter 7

Numerical experiments

In the previous chapters we presented several methods to improve the current implementation of HARES. In this chapter we discuss the results of these changes. In every section we change one part of the implementation and compare this with the original implementation of HARES. In section 7.1 we discuss the properties of the four test cases on which these methods are tested. In section 7.2 we present the initial time measurement of the current implementation of HARES. For a non-linear problem it is important to implement a suitable stopping criterion, this is discussed in section 7.3. In the following two sections we describe a different choice for the dissipation of energy term $W(x, y, \tilde{\zeta})$ in the first outer iteration and the implementation of the modified wave number $\hat{p}$ in the boundary conditions. In the current version of HARES the modified wave number is not taken into account, instead the original wave number k_0 is used. In section 7.6 we replace the Krylov subspace method Bi-CGSTAB with IDR(s) and in section 7.7 the results for the shifted Laplace preconditioner are presented. As a possible improvement for the convergence of the non-linear residual we propose Newton's method, the results are discussed in section 7.8. We describe the results for the five different forcing sequences for inexact Picard iteration. in section 7.9. In section 7.10 we present the results of the direct method `MUMPS`. Finally is section 7.11 we give an overview of the obtained results and improvements.

In this chapter we present an overview of the results, in Appendix C.2 the results of all performed tests can be found. HARES is written in FORTRAN 90. The English notation for numbers is used throughout this chapter.

7.1 Test cases

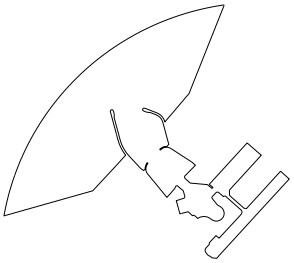
The numerical methods are tested on three harbours, i.e. the harbour of Schevenigen, the Maasvlakte and the harbour of Marsaxlokk in Malta, all provided by Svašek Hydraulics. The test case the Maasvlakte is divided into two sub-problems where the bottom topography differs. Hence we have four test cases to consider. Further on in this chapter we will use the shorter names Scheveningen, Maasvlakte A, Maasvlakte B and Malta. For each test case we present a satellite image of the harbour, the considered geometry, the initial guess of the incoming wave and the solution of the non-linear Mild-Slope equation as obtained by HARES. Details on the input files for these test cases can be found in Appendix C.1.

7.1.1 Harbour of Scheveningen

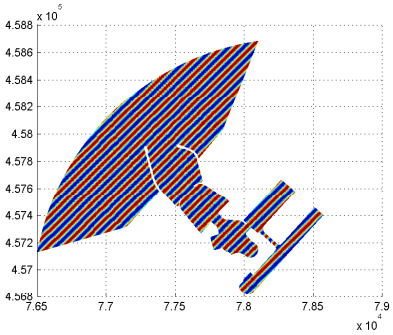
Figure 7.1 shows the harbour of Scheveningen. The incoming wave has a height of 2 meters, a period of 8 seconds and an incoming direction of 315°(based on the unit circle with a counter clockwise orientation).The maximal depth in the domain is 13.87 meters and the minimal depth of 5.00 meters. The domain is divided into 126,504 elements, with 2,213 boundary elements and

124,291 internal elements. The number of unknowns N is 63,253 and matrix $\boldsymbol{S} \in \mathbb{C}^{N \times N}$ has 438,339 non-zero elements.

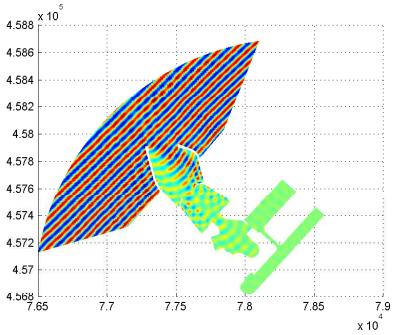
(a) The harbour of Scheveningen from Google™earth.



(b) The considered geometry of the harbour of Scheveningen.



(c) The initial guess for the harbour of Scheveningen.



(d) Solution for the harbour of Scheveningen obtained using HARES.

Figure 7.1: *Overview of the harbour of Scheveningen.*

7.1.2 Maasvlakte

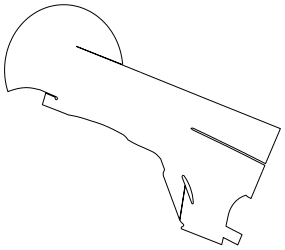
In figure 7.2 we see the Maasvlakte. Note that the considered geometry, figure 7.2(b), only contains the entrance of the Maasvlakte and not the whole area as shown in the satellite image 7.2(a). We make the distinction between the bottom topographies A and B, which results in two test cases. The difference between bottom topography A and bottom topography B is the interior depth at the embankment (northern boundary in figure 7.2(a)). In case B the interior depth at the embankment is much larger than for case A. A wave coming over the embankment, e.g. during a storm, behaves numerically oddly when the interior depth is small, therefore a larger depth is necessary. The wave motion in the entrance of the Maasvlakte is determined by combining the outcome of case A and case B.

For both bottom topographies the minimal and maximal depth is the same, namely a minimum of 3.00 meters and a maximum of 35.03 meters. The domain is divided into 347,224 elements with 2,967 boundary elements and 344,257 internal elements. This results in a system of equations with $N = 173,612$ unknowns, where matrix $\boldsymbol{S}$ has 1,209,350 non-zero elements.

Bottom topography A

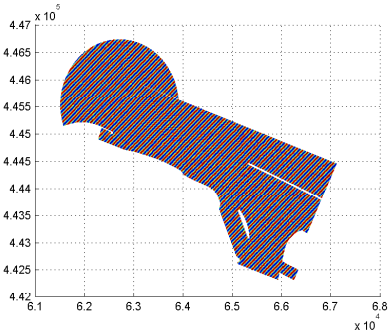
For bottom topography A there is an incoming wave with a height of 1.0 meter, a period of 10 seconds and an incoming direction of 313°. Figure 7.3 shows the initial guess and the non-linear solution for bottom topography A.

(a) The Maasvlakte from Google™earth.

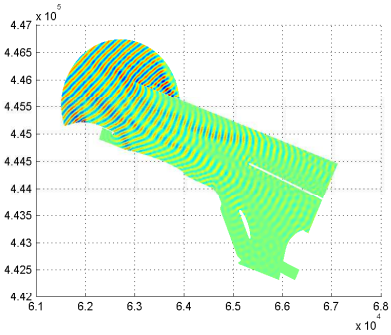


(b) The considered geometry for the Maasvlakte.

Figure 7.2: *Overview of the Maasvlakte.*



(a) The initial guess for the Maasvlakte - bottom topography A.

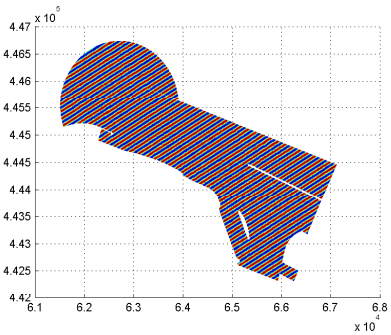


(b) The solution for the Maasvlakte - bottom topography A obtained using HARES.

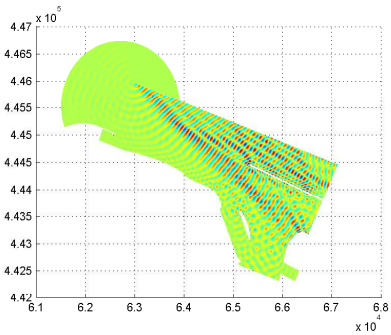
Figure 7.3: *Initial guess and solution for the Maasvlakte - bottom topography A.*

Bottom topography B

For bottom topography B we use a wave with incoming height 1.0 meter, period 10 seconds and an incoming direction of 300°. Figure 7.4 shows the initial guess and the wave motion for the Maasvlakte with bottom topography B. Note the difference between figures 7.3(b) and 7.4(b) while the initial guesses, figures 7.3(a) and 7.4(a), for both topographies are comparable. This is due to the increased interior water depth at the embankment in bottom topography B.



(a) The initial guess for the Maasvlakte - bottom topography B.



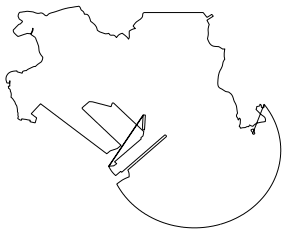
(b) The solution of the Maasvlakte - bottom topography B obtained using HARES.

Figure 7.4: *Initial guess and solution for the Maasvlakte - bottom topography B.*

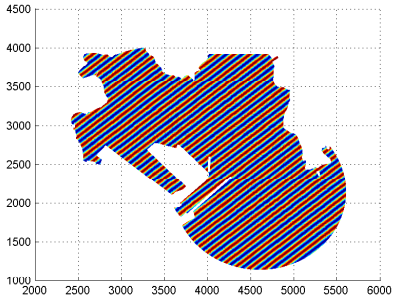
7.1.3 The harbour of Marsaxlokk

The harbour of Marsaxlokk is located in the south-east of the island Malta. In figure 7.5 we see the harbour of Marsaxlokk. The incoming wave is a wave with height 1.0 meter, period 9 seconds and incoming direction 130°. The maximal depth is 44.66 meter and the minimal depth 6.00 meter. The domain is divided into 340,848 elements with 2,910 boundary elements and 337,938 internal elements. Matrix $\boldsymbol{S}$ is of dimension $N = 170,423$ with 1,187,147 non-zero elements.

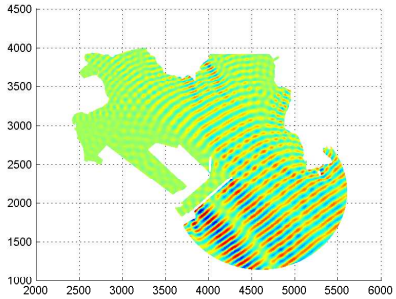
(a) The harbour of Marsaxlokk from Google™earth.



(b) The considered geometry of the harbour of Marsaxlokk.



(c) The initial guess for the harbour of Marsaxlokk.



(d) Solution of the harbour of Marsaxlokk obtained using HARES.

Figure 7.5: *Overview of the harbour of Marsaxlokk - Malta.*

7.2 Current implementation of HARES

The current implementation of HARES performs 25 outer iterations where the system of linear equations $\boldsymbol{S}\boldsymbol{\zeta} = \boldsymbol{b}$ is solved using Bi-CGSTAB preconditioned with the incomplete LU decomposition. In the first outer iteration the dissipation of wave energy $W(x, y, \tilde{\zeta})$ is set equal to zero. In the other outer loops the previous solution is used to compute a value for W. In the boundary conditions the wave number k_0 is used instead of the modified wave number $\hat{p}$. Table 7.1 gives the initial time measurement of the current implementation of HARES. The second column gives the total time in seconds for HARES to complete the calculations. The third column gives the number of outer iterations and the fourth column the total time in seconds needed to solve the system of equations. In the fifth column we find the time needed to build matrix $\boldsymbol{S}$ 25 times and in the last column the total amount of iterations performed for solving the system of equations 25 times.

Test case	Total time	# outer	Solve $\boldsymbol{S}\boldsymbol{\zeta} = \boldsymbol{b}$	Build $\boldsymbol{S}$	# matvecs
Scheveningen	994.81	25	934.17	58.26	60,206
Maasvlakte A	6700.50	25	6543.40	150.81	148,610
Maasvlakte B	6898.30	25	6741.20	150.35	156,066
Malta	5851.20	25	5697.50	146.78	132,222

Table 7.1: *Initial time measurement of the current implementation of HARES.*

Note that the total time presented in the second column is not completely determined by the sum of the time to solve the system of equations and to build matrix $\boldsymbol{S}$. At the start of the programme the mesh and the bottom topography is loaded into HARES to be able to make matrix $\boldsymbol{S}$ and determine the non-linear solution. This process takes some seconds, depending on the size of the mesh. Building matrix $\boldsymbol{S}$ per outer iterations costs around 2.3 seconds for the test case Scheveningen, 6.0 seconds for the test cases Maasvlakte A & B and 5.9 seconds for the test case Malta.

7.3 Stopping criterion for the outer loop

The current version of HARES does not contain a stopping criterion for the outer loop. For every geometry 25 outer iterations are performed without knowing when and whether convergence has been reached or not. In section 3.4 we describe the stopping criterion

$$\frac{\|F(\boldsymbol{\zeta}^k)\|_2}{\|F(\boldsymbol{\zeta}^0)\|_2} \leq \mathrm{TOL}_{\mathrm{residual}},$$

which is implemented in HARES to determine the required number of outer iterations. A suitable value for $\mathrm{TOL}_{\mathrm{residual}}$ depends on the geometry and the convergence behaviour. Table 7.2 contains some details on the convergence behaviour of the four test cases. The second column contains the initial non-linear residual $\|F(\boldsymbol{\zeta}^0)\|$, the third column the number of outer iterations until the non-linear residual is of the order $\mathcal{O}(10^{-9})$. The fourth column gives the relative decrease of the non-linear residual, after the number of outer iterations presented in the third column.

The only test case where the non-linear residual is not of the order $\mathcal{O}(10^{-9})$ after 25 outer iterations is Scheveningen. We obtain a relative decrease of the non-linear residual of 1.2094E-05. Therefore for the test case Scheveningen we choose the value of 10^{-5} for $\mathrm{TOL}_{\mathrm{residual}}$. For the other three test cases it is not reasonable to perform 25 outer iterations, since the non-linear residual is already very small after 10 iterations. For these test cases we use the tolerance of 10^{-6}, since using this value already gives a significant drop in the non-linear residual. These tolerance bounds are used

Test case	$\|F(\boldsymbol{\zeta}^0)\|_2$	# outer	Rel. decrease
Scheveningen	964.12	25	1.2094E-5
Maasvlakte A	2104.18	22	4.7524E-12
Maasvlakte B	2314.09	12	1.2964E-11
Malta	1584.47	17	6.3113E-12

Table 7.2: *Convergence behaviour of the test cases*

in all the experiments presented in this chapter. The results of implementing the stopping criterion for the non-linear residual are shown in table 7.3.

Test case	Total time	# outer	Solve $\boldsymbol{S\zeta} = \boldsymbol{b}$	Build $\boldsymbol{S}$	# matvecs
Scheveningen	1045.92	26	982.89	60.63	62,560
Maasvlakte A	2719.23	10	2654.50	60.16	60,490
Maasvlakte B	1776.98	6	1763.70	36.06	39,612
Malta	1916.61	8	1865.40	46.95	43,159

Table 7.3: *Implementing the stopping criterion for the outer loop.*

The results in table 7.1 and table 7.3 show that implementing the stopping criterion already leads to a significant improvement of the computational time for the test cases Maasvlakte A, Maasvlakte B and Malta. For Scheveningen we need one outer iteration more until we satisfy the stopping criterion.

7.4 Using the incoming wave for a value of $W(x, y, \tilde{\zeta})$

The current implementation uses the incoming wave only as an initial guess for iteratively solving the system of equations. However, we could also use it to compute a value for the dissipation of energy term $W(x, y, \tilde{\zeta})$ in the first outer loop. Table 7.4 gives the results of this implementation.

Test case	Total time	# outer	Solve $\boldsymbol{S\zeta} = \boldsymbol{b}$	Build $\boldsymbol{S}$	# matvecs
Scheveningen	868.98	22	818.52	48.23	53,002
Maasvlakte A	2729.67	10	2664.70	60.35	60,360
Maasvlakte B	1759.80	6	1719.60	36.14	39,096
Malta	1904.41	8	1852.90	47.24	42,786

Table 7.4: *Using the initial guess to determine a value for $W(x, y, \tilde{\zeta})$.*

For the test case Scheveningen the number of outer iterations has decreased from 26 to 22 and a decrease of almost 10,000 matrix-vector products for solving the system of equations. This is not caused by a decrease in the initial non-linear residual. Figure 7.6 shows the rate $\|F(\boldsymbol{\zeta}^{k+1})\|_2/\|F(\boldsymbol{\zeta}^k)\|_2$, when the initial guess is used to determine a value for W and when the initial value of W is set equal to zero. We see that the relative drop of the non-linear residual is much smoother when the incoming wave is used and therefore a more steady decrease in the non-linear residual is obtained.

The number of outer iterations did not decrease for the other test cases, however, we need some

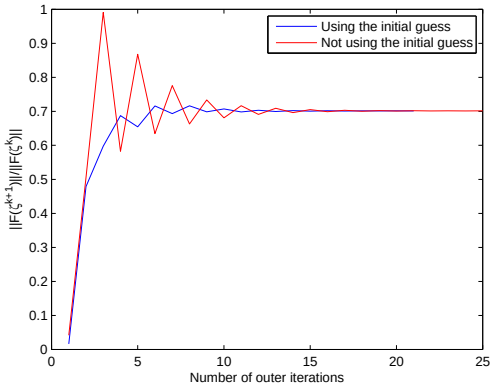


Figure 7.6: *The relative decrease in the non-linear residual $F(\boldsymbol{\zeta})$ for the test case Scheveningen.*

iterations less to solve $\boldsymbol{S\zeta} = \boldsymbol{b}$. From now on we will use the incoming wave to determine the value of $W(x, y, \tilde{\zeta})$ in the first outer loop.

7.5 The modified wave number $\hat{p}$

The current implementation of HARES does not take the modified wave number $\hat{p}$ in the boundary conditions into account. However, the modified wave number is needed to determine the wave motion accurately. Table 7.5 shows the results, regarding the computational time and the number of iterations, when the modified wave number $\hat{p}$ is implemented instead of the wave number k_0.

Test case	Total time	# outer	Solve $\boldsymbol{S\zeta} = \boldsymbol{b}$	Build $\boldsymbol{S}$	# matvecs
Scheveningen	849.31	21	800.78	46.29	50,944
Maasvlakte A	2640.63	10	2575.60	60.50	58,476
Maasvlakte B	1762.87	6	1722.50	36.31	39,061
Malta	1913.73	8	1862.20	47.26	42,886

Table 7.5: *Implementing the modified wave number $\hat{p}$.*

There is a slight decrease in the number of iterations needed to solve the linear systems of equations for all the test cases. More important is to know whether the solution changes and where this change occurs. The difference between the solution $\boldsymbol{\zeta}_{k_0}$ for the wave number k_0 and the solution $\boldsymbol{\zeta}_{\hat{p}}$ for the modified wave number $\hat{p}$ is given in table 7.6.

	Scheveningen	Maasvlakte A	Maasvlakte B	Malta
$\frac{\|\boldsymbol{\zeta}_{k_0} - \boldsymbol{\zeta}_{\hat{p}}\|_2}{\|\boldsymbol{\zeta}_{\hat{p}}\|_2}$	0.0031	0.0041	0.0010	5.3277E-4
$\|\boldsymbol{\zeta}_{k_0} - \boldsymbol{\zeta}_{\hat{p}}\|_\infty$	0.0429	0.0560	0.0039	0.0033

Table 7.6: *The change in the non-linear solution when the modified wave number $\hat{p}$ is included.*

For the current bottom topographies of the test cases the implementation of the modified wave number $\hat{p}$ does not significantly change the solution. The changes that did occur are present

around the boundary of the domains, since the modified wave number is present in the boundary conditions. The dissipation of wave energy is included in the modified wave number $\hat{p}$, see expression 2.2.13. Hence we expect that for a decreasing depth, the influence of the modified wave number increases. We changed the water level for the test case Maasvlakte to 1.00 meter resp. 2.00 meter, which results in a minimal depth of 1.00 meter resp. 2.00 meter.

	MSA - level 1.00 m	MSA - level 2.00 m	MSB - level 1.00 m	MSB - level 2.00 m
$\frac{\|\boldsymbol{\zeta}_{k_0}-\boldsymbol{\zeta}_{\hat{p}}\|_2}{\|\boldsymbol{\zeta}_{\hat{p}}\|_2}$	0.0028	0.0037	0.0022	0.0014
$\|\boldsymbol{\zeta}_{k_0}-\boldsymbol{\zeta}_{\hat{p}}\|_\infty$	0.0510	0.0640	0.0088	0.0058

Table 7.7: *The change in the solution of the Maasvlakte for a decreasing water depth. Where MSA stands voor Maasvlakte A and MSB for Maasvlakte B.*

The results in table 7.7 show that the decrease in water depth did not lead to a significant change in the solution between the two implementations compared to the water depth of 3.00 meters. It is possible that the minimal water depth of 1.00 meter is still not small enough, such that the dissipation of wave energy in the boundary conditions significantly influence the behaviour of the wave motion.

7.6 Bi-CGSTAB versus IDR(s)

In chapter 5 we have presented the Krylov subspace method IDR(s) as an improvement over Bi-CGSTAB. The behaviour of IDR(2), IDR(4) and IDR(8) preconditioned with the incomplete LU decomposition of matrix $\boldsymbol{S}$ is discussed in this section. Table 7.8 contains the computing time and the number of matrix-vector products for IDR(2), IDR(4) and IDR(8) applied to the four test cases.

Test case	Total time	# outer	Solving $\boldsymbol{S\zeta} = \boldsymbol{b}$	Build $\boldsymbol{S}$	# matvecs
	IDR(2)				
Scheveningen	263.78	21	215.31	46.27	13,839
Maasvlakte A	733.45	10	668.33	60.60	14,920
Maasvlakte B	624.13	6	583.66	36.38	13,013
Malta	607.77	8	566.23	47.27	12,496
	IDR(4)				
Scheveningen	269.78	21	221.35	46.18	12,823
Maasvlakte A	713.28	10	648.15	60.53	12,680
Maasvlakte B	590.63	6	550.15	36.29	10,750
Malta	617.26	8	565.65	47.30	11,276
	IDR(8)				
Scheveningen	312.74	21	264.19	46.33	12,370
Maasvlakte A	726.77	10	661.61	60.62	10,453
Maasvlakte B	603.58	6	562.30	37.18	8,911
Malta	728.05	8	676.52	47.27	10,729

Table 7.8: *Computing time for IDR(s) combined with the ILU(0) preconditioner*

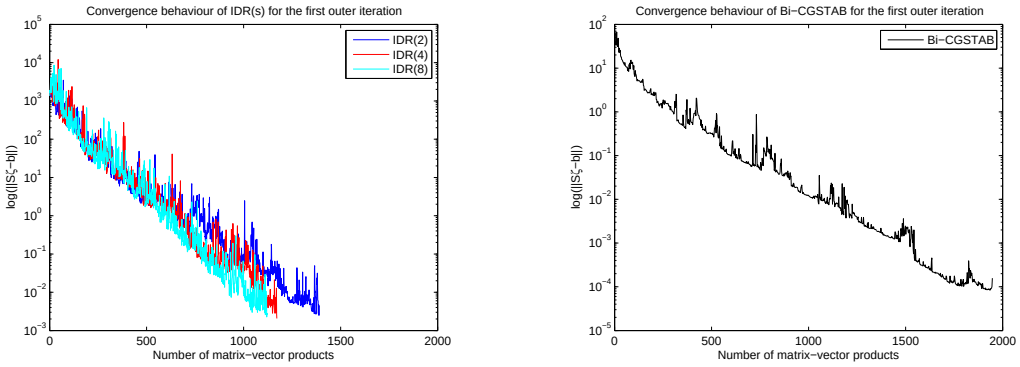
The results in table 7.8 show that when we increase the value of s the number of matrix-vector products decreases. However, the computing time to determine the solution of the system of

equation does not decrease. This indicates that for a higher value of s an iteration becomes more expensive. This is confirmed by the results in table 7.9, where also the time for half an iteration performed with Bi-CGSTAB is showed.

Test case	Bi-CGSTAB	IDR(2)	IDR(4)	IDR(8)
Scheveningen	0.0157	0.0153	0.0172	0.0212
Maasvlakte A & B	0.0440	0.0450	0.0515	0.0632
Malta	0.0434	0.0443	0.0502	0.0640

Table 7.9: *Computing time for one iteration of IDR(s) and half an iteration of Bi-CGSTAB.*

IDR(2) and IDR(4) give overall the best performance regarding the computing time for solving the systems of equations. Figure 7.7 shows the convergence behaviour of IDR(s) and Bi-CGSTAB in the first outer iteration of the test case Scheveningen. The initial residual for IDR(s) and Bi-CGSTAB differ since for IDR(s) right preconditioning is used, while for Bi-CGSTAB left preconditioning is used. However, comparing the convergence behaviour of IDR(s) and Bi-CGSTAB we conclude that the residual decreases faster for IDR(s) than for Bi-CGSTAB. Looking at figure 7.7(a) we see that after the first 500 matrix-vector product the difference between IDR(2), IDR(4) and IDR(8) is small. After more matrix-vector products the difference in convergence behaviour increases, where IDR(8) gives the fastest convergence.



(a) Convergence behaviour of IDR(s)

(b) Convergence behaviour of Bi-CGSTAB

Figure 7.7: *The convergence behaviour of IDR(s) and Bi-CGSTAB for the first outer iteration of the test case Scheveningen.*

Figure 7.8 shows the comparison between the number of matrix-vector products per outer iteration and the four iterative methods. A very noticeable difference between IDR(s) and Bi-CGSTAB is that the number of matrix-vector products decreases for IDR(s) and not for Bi-CGSTAB. This is caused by the stopping criterion that is used in Bi-CGSTAB. The Bi-CGSTAB algorithm in HARES uses the stopping criterion

$$\frac{\|\boldsymbol{b} - \boldsymbol{S}\boldsymbol{\zeta}_l^k\|_2}{\|\boldsymbol{b} - \boldsymbol{S}\boldsymbol{\zeta}_0^k\|_2} \leq \text{TOL}, \tag{7.6.1}$$

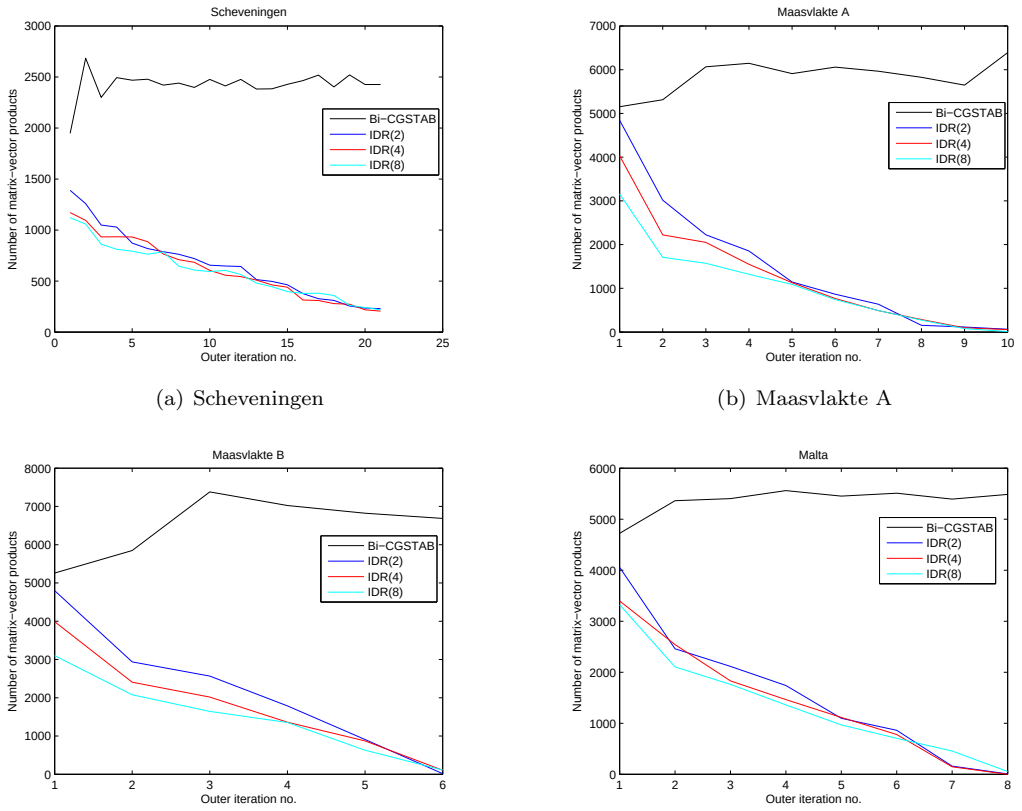
with $\boldsymbol{\zeta}_l^k$ the iterative solution after the lth iteration and kth outer iteration, and $\boldsymbol{\zeta}_0^k$ the initial guess. Every loop of solving the system of equations the stopping criterion has the initial value one and the residual has to decrease significantly before the Bi-CGSTAB algorithm converges.

However, for Picard iteration, as described in section 3.1, the non-linear residual is given by the denominator of stopping criterion (7.6.1). The non-linear residual decreases if the non-linear problem converges. This means that using this stopping criterion and a tolerance 10^{-6} each inner loop the residual, which for instance can already be of the order $\mathcal{O}(10^{-2})$, has to decrease a factor 10^6. Therefore, the algorithm has no benefit of the already small initial residual. The implemented stopping criterion in IDR(s) is given by

$$\frac{\|\boldsymbol{b} - \boldsymbol{S}\boldsymbol{\zeta}_l^k\|_2}{\|\boldsymbol{b}\|_2} \leq \text{TOL}. \tag{7.6.2}$$

Using Picard iteration, the right-hand-side $\boldsymbol{b}$ does not change each outer iteration. Therefore, this stopping criterion does benefit an already good initial residual.

For the smaller problem Scheveningen the difference in matrix-vector products between IDR(2), IDR(4) and IDR(8) is small, while for the larger problem Maasvlakte A & B the difference is for the first couple of outer iterations larger. Towards the last outer iterations, this difference decreases.



(a) Scheveningen

(b) Maasvlakte A

(c) Maasvlakte B

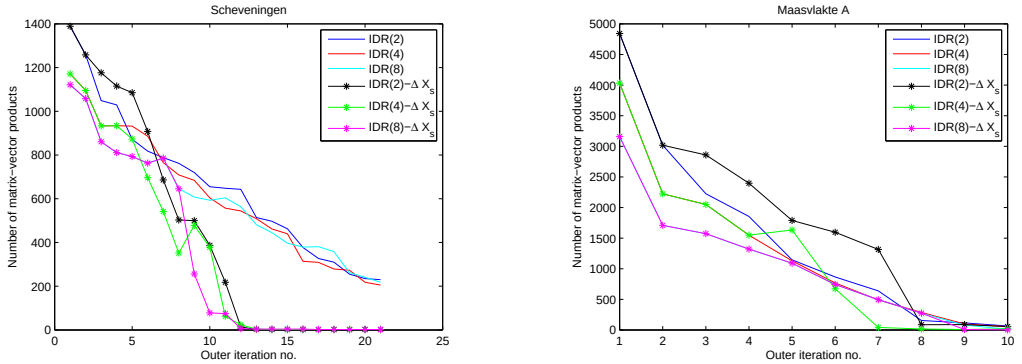
(d) Malta

Figure 7.8: *Number of matrix-vector products per outer iteration*

Note that there are several differences between the implementation of Bi-CGSTAB in HARES and the IDR(s) algorithm used for these results. The two differences are the stopping criterion and the way of preconditioning. In the literature study of this thesis, see van de Sande (2011), we have tested Bi-CGSTAB and IDR(s) without these differences. There we obtained that the results regarding the number of matrix-vector products and the computing time are significantly better for IDR(s).

7.6.1 Choosing the initial space $\mathcal{G}_0$

As mentioned is section 5.2.2 we can choose the initial vector $\Delta \boldsymbol{X}_s$ as long as its elements are in the complete subspace. After s outer loops we are able to choose the initial vector $\Delta \boldsymbol{X}_s$ such that it contains the iterative solutions of the s previous outer iterations. The results of choosing the initial vector this way is given by table 7.10 and figure 7.9.



(a) Scheveningen

(b) Maasvlakte A

Figure 7.9: *Matrix-vector products when the initial space $\Delta \boldsymbol{X}_s$ is chosen in IDR(s) for the test cases Scheveningen and Maasvlakte A*

Figure 7.9(a) shows that for the test case Scheveningen the number of matrix-vector products decrease faster when the initial space is chosen. During the first s outer iterations the number of matrix-vector products is exactly the same. Then for IDR(4) and IDR(8) the number of matrix-vector products decreases immediately after 4 resp. 8 outer iterations. However, for IDR(2) the number of matrix-vector products is larger when the initial space is chosen. In the first couple of outer iterations the change in the iterative solution is quite large. Using these previous iterative solutions to build the initial space $\Delta \boldsymbol{X}_s$ does not benefit the algorithm. When the relative change in solution is of the order $\mathcal{O}(10^{-2})$ it becomes beneficial. As of the seventh outer iteration the number of matrix-vector products is smaller when $\Delta \boldsymbol{X}_s$ is chosen. The decrease in matrix-vector products is much larger and therefore in the complete process the number of matrix-vector products decreases.

From table 7.10 and figure 7.9(b) we conclude that for the test case Maasvlakte A choosing the initial space does not reduce the number of matrix-vector products as much as for the test case Scheveningen. For IDR(2) the number of matrix-vector products has even increased compared to not choosing the initial space. For IDR(4) and IDR(8) it is beneficial, however, since only 10 outer iterations are needed the influence is not that large.

For the test cases Maasvlakte B and Malta the computational time is decreased for IDR(2) and IDR(4). Since only 6 resp. 8 outer iterations are needed for these test cases, it is not possible for IDR(8) to choose the initial space this way. Hence no change in the number of matrix-vector products has occurred.

Test case	Total time	# outer	Solving $\boldsymbol{S\zeta} = \boldsymbol{b}$	Build $\boldsymbol{S}$	# matvecs
	IDR(2) - $\Delta \boldsymbol{X}_s$				
Scheveningen	192.36	21	143.82	46.31	9,257
Maasvlakte A	872.29	10	807.11	60.60	18,040
Maasvlakte B	603.55	6	563.14	36.31	12,570
Malta	534.93	8	483.25	47.36	10696
	IDR(4) - $\Delta \boldsymbol{X}_s$				
Scheveningen	183.22	21	134.56	46.38	7,558
Maasvlakte A	705.07	10	639.85	60.59	12,241
Maasvlakte B	537.87	6	497.33	36.33	9,777
Malta	518.26	8	466.75	47.22	9,246
	IDR(8) -$\Delta \boldsymbol{X}_s$				
Scheveningen	203.13	21	154.67	46.23	7,280
Maasvlakte A	721.51	10	656.35	60.54	10,370
Maasvlakte B	603.03	6	562.60	36.28	8,911
Malta	730.41	8	671.05	55.09	10,729

Table 7.10: *Computational time for IDR(s) when the initial space $\Delta \boldsymbol{X}_s$ is chosen.*

7.7 Shifted Laplace preconditioner

In this section we consider four versions of the shifted Laplace preconditioner, all are approximated by the incomplete LU decomposition. Three versions are based on the shifted Laplace preconditioner as presented in section 5.3.2, i.e.

$$\boldsymbol{K} = \boldsymbol{L} + \xi^2 \boldsymbol{M} + \boldsymbol{C},$$

with the shifts

$$\xi_1^2 = i\left|n_0 - \frac{iW}{\omega}\right|, \quad \xi_2^2 = k_0^2 \quad \text{and} \quad \xi_3^2 = k_0^2 + i\left|n_0 - \frac{iW}{\omega}\right|.$$

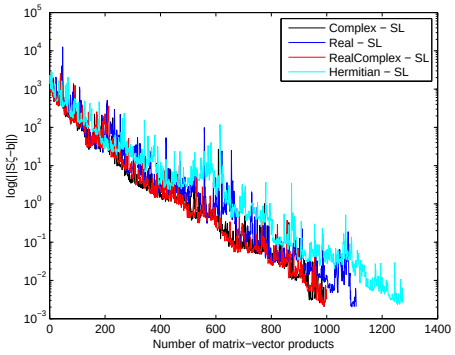
van Gijzen et al. (2007) proposed the complex shift ξ_1^2, the real shift $\xi_2^2 = k_0^2$ is derived based on the work of van Gijzen and Erlangga (2006) and the third shift is a combination of the first two shifts, presented by Erlangga et al. (2006). Note that for each element the value of the shift ξ^2 changes. The fourth version of the shifted Laplace preconditioner is based on the shifted Laplace preconditioner presented in section 6.4.2 with the Hermitian positive definite preconditioner for the boundary elements, expression (6.4.17). For the internal elements we use the shift $\xi_2^2 = k_0^2$. Table 7.11 shows the results regarding the computing time and the number of matrix-vector products for the four types of the shifted Laplace preconditioner on the test cases. Only the results for IDR(2) are shown.

When the shifted Laplace preconditioner is used a second matrix needs to be computed, hence the building time increases slightly. For one outer loop it takes approximately 2.3 seconds for the test case Scheveningen and 6.2 seconds for the other test cases. The complex shift and the real-complex shift give the best results and their computational time and the number of matrix-vector products is very similar. The fourth type of the shifted Laplace preconditioner performs the poorest. However, its results are considerably better than the case with only the incomplete LU decomposition of matrix $\boldsymbol{S}$. We conclude that for the shifted Laplace preconditioner it is good to fully include the contribution of the boundary conditions. The results in the upcoming sections include the shifted Laplace preconditioner with the complex shift.

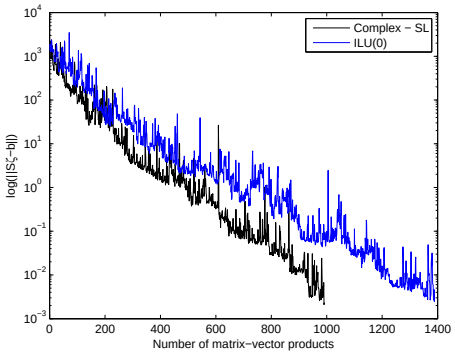
	Total time	# outer	Solve $\boldsymbol{S\zeta = b}$	Build $\boldsymbol{S}$ & $\boldsymbol{K}$	# matvecs
	Scheveningen				
Complex - SL	207.27	21	157.33	47.85	10,091
Real - SL	231.69	21	181.98	47.59	11,589
RealComplex - SL	209.07	22	156.89	50.05	10,181
Hermitian - SL	255.42	21	205.80	47.52	13,289
	Maasvlakte A				
Complex - SL	341.87	10	274.63	62.63	6,074
Real - SL	406.56	10	422.51	62.52	9,187
RealComplex - SL	336.81	10	269.68	62.56	6,023
Hermitian - SL	489.70	10	422.51	62.52	9,187
	Maasvlakte B				
Complex - SL	276.40	6	234.73	37.53	5,227
Real - SL	333.61	6	292.22	37.27	6,475
RealComplex - SL	277.68	6	235.92	37.63	5,255
Hermitian - SL	399.10	6	357.62	37.40	7,978
	Malta				
Complex - SL	401.26	8	348.13	48.84	7,901
Real - SL	460.68	8	407.64	48.68	8,937
RealComplex - SL	405.78	8	352.62	48.88	7,934
Hermitian - SL	474.47	8	421.35	48.72	9,417

Table 7.11: *Computational time for four types of shifted Laplace preconditioners for IDR(2).*

Figure 7.10 shows the convergence behaviour of the four types of the shifted Laplace preconditioner on the test case Scheveningen. We compare the incomplete LU decomposition of matrix $\boldsymbol{S}$ (blue line) with the incomplete LU decomposition of the shifted Laplace preconditioner (black line), both for the first outer iteration. The convergence behaviour of the shifted Laplace preconditioner with the complex shift and the real-complex shift is very similar. For the shifted Laplace preconditioners with only a real shift the convergence behaviour is less smooth than for the other two choices. This suggests that when the shift has a complex part the convergence is faster and smoother.



(a) Convergence behaviour of the four types of the shifted Laplace preconditioner



(b) Comparison between the ILU(0) and the shifted Laplace preconditioner

Figure 7.10: *The convergence behaviour of the four types of shifted Laplace preconditioners and a comparison with the incomplete LU decomposition for the test case Scheveningen and IDR(2).*

7.8 Newton's method

In section 3.2 we have presented Newton's method as a possible improvement to Picard iteration. However, it is not possible to include the derivatives of $W(x, y, \tilde{\zeta})$ and $\hat{p}$ with respect to $\tilde{\zeta}$ in Newton's method. The results for Newton's method with the IDR(2) and the shifted Laplace preconditioner are shown in table 7.12

Test case	Total time	# outer	Solve $\boldsymbol{S\zeta} = \boldsymbol{b}$	Build $\boldsymbol{S}$ & $\boldsymbol{K}$	# matvecs
Scheveningen	713.22	22	659.12	52.36	40,770
Maasvlakte A	1486.96	10	1417.60	63.64	30,465
Maasvlakte B	924.81	6	882.32	39.86	18,877
Malta	1559.95	8	1505.90	50.84	33,381

Table 7.12: *Computing time for Newton's method.*

The number of outer iterations did not decrease for any of the test cases. This indicates that the usually quadratic convergence property of Newton's method is not obtained. Hence not being able to include the derivatives gives the same convergence rate as Picard iteration. However, it takes much longer to reach the desired non-linear solution. In the case of Newton's method the implemented stopping criterion in IDR(s), expression (7.6.2), is not suitable. For Newton's method the non-linear residual, given by the right-hand-side $\boldsymbol{b}$, will decrease towards zero. Hence it is better to implement the stopping criterion (7.6.1) when Newton's method is used. The results for this stopping criterion can be found in table 7.13. Using this stopping criterion clearly improves the computing time, however the total time is about twice as high as for Picard iteration.

Test case	Total time	# outer	Solve $\boldsymbol{S\zeta} = \boldsymbol{b}$	Build $\boldsymbol{S}$ & $\boldsymbol{K}$	# matvecs
Scheveningen	432.51	22	376.20	51.92	24,283
Maasvlakte A	945.28	10	875.84	65.04	19,516
Maasvlakte B	596.91	6	553.41	39.11	12,353
Malta	931.22	8	876.13	50.69	19,722

Table 7.13: *The computing time for Newton's method with the modified stopping criterion.*

7.9 Inexact Picard iteration

In section 3.3 we have proposed five choices for the forcing sequence η_k in inexact Picard iteration. In this section we discuss the results for these choices. In the choices 1, 2 and 5 we have to choose the initial forcing term η_0. Two values of this initial forcing term have been tested, however, we only present the best results regarding the computing time here. The other results can be found in Appendix C.2. The forcing sequence based on the fifth choice also has a tolerance incorporated, which can be chosen freely. Two choices of this tolerance bound are tested.

We make the distinction between the two stopping criteria (7.6.1) and (7.6.2). Stopping criterion (7.6.1) is denoted as ST1 and stopping criterion (7.6.2) as ST2. We present the results of IDR(2) and IDR(2) - $\Delta \boldsymbol{X}_s$ both preconditioned with the shifted Laplace preconditioner with a complex shift. Where IDR(2) - $\Delta \boldsymbol{X}_s$ indicates that the initial space $\Delta \boldsymbol{X}_s$ is chosen as presented in section 7.6.1. At most 49 outer iterations are performed. If the non-linear solution has not been obtained after 49 outer iterations, only the number 49 is present in the third column. In case we did not obtain convergence for the test case Scheveningen, it also did not occur for the other three test cases. Therefore, these results are not present for the other test cases.

7.9.1 Choice 1

Choose $\eta_0 \in [0, 1)$, the next forcing term is given by

$$\eta_k = \frac{\left| \|F(\boldsymbol{\zeta}^k)\|_2 - \|\boldsymbol{r}^{k-1}\|_2 \right|}{\|F(\boldsymbol{\zeta}^{k-1})\|_2}, \qquad k = 1, 2, \ldots.$$

The results presented in table 7.14 are obtained using the initial forcing term $\eta_0 = 0.5$.

With stopping criterion (7.6.2) we did not obtain the non-linear solution after 49 outer iterations. The non-linear residual decreases slightly after the first outer iteration. The second forcing term is then already larger than the quotient $\boldsymbol{r}_0^2/\boldsymbol{b}$. With $\boldsymbol{r}_0^2$ the initial residual of the system of equations in the second outer loop. The IDR(2) algorithm does not perform any iterations, since the stopping criterion is already satisfied. This leads to no change in the non-linear residual, hence the next forcing term will also be quite large. This results in no convergence to the non-linear solution. Using stopping criterion (7.6.1) do lead to convergence, since then the relative residual will equal one at the start and the IDR(2) algorithm does perform some iterations.

The results in table 7.14 indicate that choosing the initial space $\Delta \boldsymbol{X}_s$ does not improve the computational time. Since the system of equations is solved less accurately using inexact Picard iteration, the difference between two successive iterative solutions is larger than for standard Picard iteration. Hence the algorithm can only benefit from a small range between the solutions after many outer iterations.

	Total time	# outer	Solve $\boldsymbol{S\zeta} = \boldsymbol{b}$	Build $\boldsymbol{S}$ & $\boldsymbol{K}$	# matvecs
	Scheveningen				
IDR(2) - ST1	66.22	17	25.81	38.62	1,431
IDR(2) - $\Delta \boldsymbol{X}_s$ - ST1	125.09	29	56.91	65.86	3,228
IDR(2) - ST2		49			
IDR(2) - $\Delta \boldsymbol{X}_s$ - ST2		49			
	Maasvlakte A				
IDR(2) - ST1	202.86	11	129.79	68.89	2,376
IDR(2) - $\Delta \boldsymbol{X}_s$ - ST1	339.86	12	249.96	85.62	4,787
	Maasvlakte B				
IDR(2) - ST1	264.25	9	165.78	94.01	3,066
IDR(2) - $\Delta \boldsymbol{X}_s$ - ST1	493.71	15	392.42	96.67	7,494
	Malta				
IDR(2) - ST1	234.04	11	162.57	67.29	3,104
IDR(2) - $\Delta \boldsymbol{X}_s$ - ST1	317.73	9	258.86	54.92	5,027

Table 7.14: *Computational time for the forcing sequence based on choice 1.*

7.9.2 Choice 2

Choose $\eta_0 \in [0, 1)$, the forcing sequence is obtained by

$$\eta_k = 0.5 \left(\frac{\|F(\boldsymbol{\zeta}^k)\|_2}{\|F(\boldsymbol{\zeta}^{k-1})\|_2} \right)^2, \qquad k = 1, 2, \ldots.$$

The initial forcing term $\eta_0 = 0.5$ is used, the results of choice 2 for the forcing sequence are presented in table 7.15.

As for choice 1 of the forcing sequence, the stopping criterion (7.6.2) did not result in the non-linear solution. The same kind of stagnation occurred, but for this choice all the forcing terms were equal to 0.5. Stopping criterion (7.6.1) did give good results regarding the convergence to the non-linear solution. Using this forcing sequence we needed less outer iterations, however, more matrix-vector products in the inner loop than for the first choice. Building matrix $\boldsymbol{S}$ costs quite some time, hence it is not directly necessary to decrease the number of matrix-vector products if more outer iterations are necessary to reach convergence.

Choosing the initial space $\Delta \boldsymbol{X}_s$ does not improve the computational time. However, the difference between IDR(2) and IDR(2) - $\Delta \boldsymbol{X}_s$ is smaller than for the forcing sequence based on choice 1. Each inner loop more matrix-vector products are performed, hence the system of equations is solved more accurately and the difference between two successive iterative solution decreases faster than for choice 1.

	Total time	# outer	Solve $\boldsymbol{S\zeta} = \boldsymbol{b}$	Build $\boldsymbol{S}$ & $\boldsymbol{K}$	# matvecs
	Scheveningen				
IDR(2) - ST1	76.80	17	36.73	38.47	2,368
IDR(2) - $\Delta \boldsymbol{X}_s$ - ST1	109.31	22	57.69	49.82	3,714
IDR(2) - ST2		49			
IDR(2) - $\Delta \boldsymbol{X}_s$ - ST2		49			
	Maasvlakte A				
IDR(2) - ST1	227.06	11	154.77	68.55	3,445
IDR(2) - $\Delta \boldsymbol{X}_s$ - ST1	331.96	11	259.66	68.54	5,794
	Maasvlakte B				
IDR(2) - ST1	304.63	8	251.47	49.76	5,590
IDR(2) - $\Delta \boldsymbol{X}_s$ - ST1	422.13	8	368.93	49.81	8,260
	Malta				
IDR(2) - ST1	355.61	8	304.00	48.22	6,879
IDR(2) - $\Delta \boldsymbol{X}_s$ - ST1	424.70	8	372.62	48.67	8,454

Table 7.15: *Computational time for the forcing sequence based on choice 2.*

7.9.3 Choice 3

The forcing sequence is given by

$$\eta_k = \min\left(\frac{1}{k+2}, \|F(\boldsymbol{\zeta}^k)\|_2\right), \qquad k = 1, 2, \ldots.$$

Table 7.16 present the results for the third choice of the forcing sequence. The results of IDR(2) - $\Delta \boldsymbol{X}_s$ are from now on left out, since this is not an improvement for inexact Picard iteration.

	Total time	# outer	Solve $\boldsymbol{S\zeta} = \boldsymbol{b}$	Build $\boldsymbol{S}$ & $\boldsymbol{K}$	# matvecs
	Scheveningen				
IDR(2) - ST1	118.05	22	65.90	50.36	4,244
IDR(2) - ST2		49			
	Maasvlakte A				
IDR(2) - ST1	260.07	11	187.21	69.07	4,174
	Maasvlakte B				
IDR(2) - ST1	243.59	9	183.61	56.48	4,085
	Malta				
IDR(2) - ST1	365.54	11	292.57	69.21	6,632

Table 7.16: *Computational time for the forcing sequence based on choice 3.*

Although we did not obtain convergence within 49 outer iterations using stopping criterion (7.6.2), the non-linear convergence did not stagnate as for the choices 1 and 2. Each outer iteration the IDR(2) algorithm performs so little matrix-vector products that the non-linear residual decreases very slowly. Such a slow convergence rate leads to many outer iterations where each time the

matrices $\boldsymbol{S}$ and $\boldsymbol{K}$ need to be build. Hence the total process will take a long time, even though the time for solving the system of equations is quite short.

The convergence using the third choice is slower than for the first and second choice. For the test cases Scheveningen and Maasvlakte A more matrix-vector product are needed. For the first couple of outer iterations the number of matrix-vector products is lower than for choices 1 and 2. But when the decrease in the non-linear residual is small the forcing terms prescribed by choices 1 and 2 are of the order $\mathcal{O}(10^{-1})$, while the forcing terms of choice 3 are of the order $\mathcal{O}(10^{-3})$. Hence in the last part of the computation the number of matrix-vector products is much higher than for choices 1 and 2. For the test cases Maasvlakte B and Malta more outer iterations are needed and less matrix-vector products. Until the seventeenth outer loop the forcing sequence is determined by the quotient $\frac{1}{k+2}$ for the test case Scheveningen. After seventeen outer iterations the non-linear residual $\|F(\boldsymbol{\zeta}^k)\|_2$ determines the forcing terms. For the other test cases the non-linear residual defines the forcing sequence as of the eight outer iteration.

7.9.4 Choice 4

The next forcing term is given by

$$\eta_k = \frac{1}{2^{k+1}}, \qquad k = 1, 2, \ldots.$$

Table 7.17 shows the results for the forcing sequence based on choice 4.

	Total time	# outer	Solve $\boldsymbol{S\zeta} = \boldsymbol{b}$	Build $\boldsymbol{S}$ & $\boldsymbol{K}$	# matvecs
	Scheveningen				
IDR(2) - ST1	463.01	26	401.49	59.36	22,117
IDR(2) - ST2	71.22	19	25.88	43.35	1,651
	Maasvlakte A				
IDR(2) - ST1	355.47	11	282.46	68.90	5,418
IDR(2) - ST2	266.15	20	114.78	145.65	2,530
	Maasvlakte B				
IDR(2) - ST1	321.53	8	267.27	50.56	5,052
IDR(2) - ST2	271.96	20	140.96	125.39	3,115
	Malta				
IDR(2) - ST1	530.89	10	465.58	61.32	9,216
IDR(2) - ST2	269.54	20	141.52	122.34	3,169

Table 7.17: *Computational time for the forcing sequence based on choice 4.*

The first observation from table 7.17 is that the number of outer iterations is the same for the test cases Maasvlakte A, Maasvlakte B and Malta when the stopping criterion (7.6.2) is used. But when complete Picard iteration is applied the different test cases need a different amount of outer iterations. Using this forcing sequence, the non-linear residual decreases by the same rate for every test case. Hence only the size of the initial non-linear residual determines the number of outer iterations. The value of the forcing terms based on choice 4 decreases fast as k increases and thus the IDR(2) algorithm will perform enough matrix-vector products to converge to the non-linear solution within the maximal amount of outer iterations. For stopping criterion (7.6.1), the number of outer iterations corresponds to the results of Picard iteration regarding the number of outer iterations. Since the value of the forcing term decreases fast, the number of matrix-vector

products increases for each outer iteration. For the forcing sequence based on choice 4 the original stopping criterion in IDR(s), expression (7.6.2), gives the best results.

7.9.5 Choice 5

Choose $\eta_0 \in [0, 1)$, the forcing sequence is given by

$$\eta_k = \frac{\|\boldsymbol{\zeta}^k - \boldsymbol{\zeta}^{k-1}\|_2}{\|\boldsymbol{\zeta}^0\|_2} \cdot \text{TOL}, \qquad k = 1, 2, \ldots.$$

We need to choose an initial forcing term η_0 and a tolerance TOL for the forcing sequence based on choice 5. Since the convergence behaviour of the different test cases is different, there was not one unique choice which gave the best results. For the test case Scheveningen we use $\eta_0 = 0.5$ and $\text{TOL} = 10^{-2}$, for the test cases Maasvlakte A & B and Malta we use $\eta_0 = 0.1$ and $\text{TOL} = 10^{-2}$. The results for these values are given in table 7.18.

	Total time	# outer	Solve $\boldsymbol{S\zeta} = \boldsymbol{b}$	Build $\boldsymbol{S}$ & $\boldsymbol{K}$	# matvecs
	Scheveningen				
IDR(2) - ST1	450.80	26	389.52	59.30	24,426
IDR(2) - ST2	57.76	11	31.32	25.08	2,008
	Maasvlakte A				
IDR(2) - ST1	650.08	10	583.78	62.64	13,023
IDR(2) - ST2	188.98	11	116.24	68.93	2,592
	Maasvlakte B				
IDR(2) - ST1	464.17	7	417.04	43.85	9,321
IDR(2) - ST2	183.21	10	117.43	62.18	2,615
	Malta				
IDR(2) - ST1	696.04	8	643.68	48.96	14,598
IDR(2) - ST2	196.18	9	137.58	55.08	3,113

Table 7.18: *Computational time for the forcing sequence based on choice 5.*

The results for the test case Scheveningen in table 7.18 for stopping criterion (7.6.2) is very remarkable, the number of outer iterations is much lower than for all the other forcing sequences. The relative difference between the obtained non-linear solution and the solution of Picard iteration is of the order $\mathcal{O}(10^{-5})$, hence we do obtain the non-linear solution.

Stopping criterion (7.6.2) performs better than stopping criterion (7.6.1), since the value of the forcing terms decreases fast. The forcing sequence based on choice 5 gives overall the best performance of the presented forcing sequences.

7.10 The direct method MUMPS

The direct method MUMPS is available in a sequential version and a parallel version. We have implemented the sequential version, the results for the test cases are presented in table 7.19.
The time to solve the system of equation depends on its dimension, for the test case Scheveningen it takes around 1 second and for the other test cases more or less 4.3 seconds. The initialization for MUMPS has to be done only once, since the structure of matrix $\boldsymbol{S}$ does not change during the process. The time for the initialization is shown in the sixth column of table 7.19. Only having

Test case	Total time	# outer	Solve $\boldsymbol{S\zeta} = \boldsymbol{b}$	Build $\boldsymbol{S}$	Init. time
Scheveningen	69.48	21	21.36	46.13	0.62
Maasvlakte A	108.46	10	43.25	60.53	1.78
Maasvlakte B	66.84	6	25.95	36.31	1.78
Malta	90.38	8	37.11	46.67	1.80

Table 7.19: *Computational time for the sequential version of the direct method* `MUMPS`.

to perform the initialization once and not every inner loop is very beneficial, for the test case Scheveningen this results in a decrease of 60 percent and for the other test cases a decrease of 40 percent in the computational time for solving the system of equations.

7.11 Summary of the numerical experiments

In this chapter we presented the results of the improvements to the current version of HARES. Firstly we implemented a stopping criterion for the non-linear residual. Secondly Bi-CGSTAB preconditioned with the incomplete LU decomposition of matrix $\boldsymbol{S}$ is replaced by IDR(2) preconditioned with the incomplete LU decomposition of the shifted Laplace preconditioner. Thirdly inexact Picard iteration is applied as the last improvement to decrease the number of matrix-vector products and the computational time. In addition to the iterative solution methods we tested the direct method `MUMPS` for its performance. Figure 7.11 gives an overview of the decrease in computing time for the several improvements.

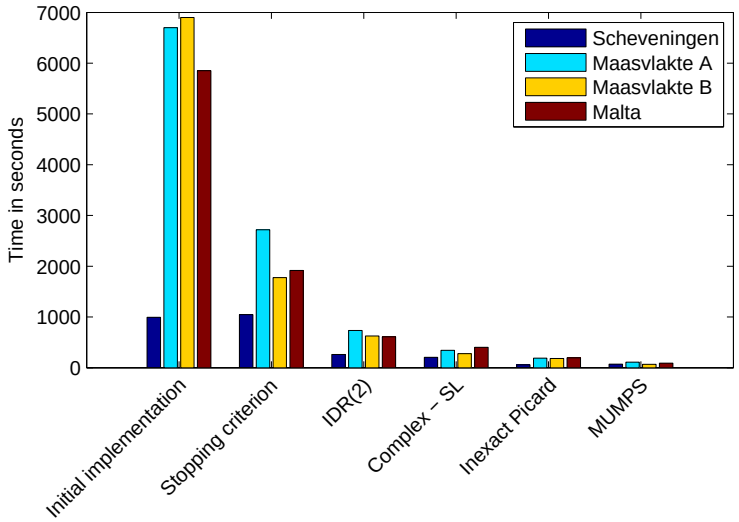


Figure 7.11: *Overview of the decrease in computing time.*

Figure 7.11 shows that we successfully achieved a considerable reduction of the computing time of HARES. Table 7.20 gives the percentage of the computing time and the matrix-vector products of the initial implementation needed for the improved iterative solution method and the direct method.

	Scheveningen	Maasvlakte A	Maasvlakte B	Malta
	Iterative solution method			
Computational time	5.8%	2.8%	2.7%	3.4%
# matrix-vector products	3.3%	1.7%	1.7%	2.4%
	Direct solution method			
Computational time	7.0%	1.6%	1.0%	1.5%

Table 7.20: *Percentage of the computing time and the matrix vector-products of the initial implementation needed in the improved version.*

Figure 7.12 gives an overview of the decrease in matrix-vector products for the several improvements.

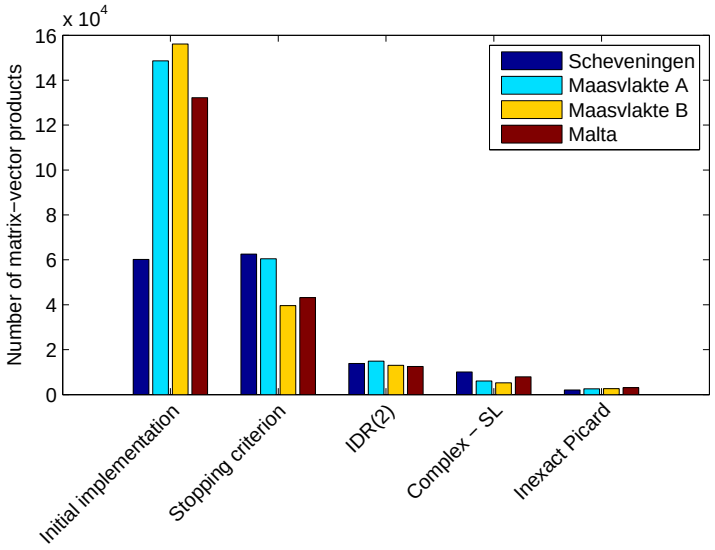


Figure 7.12: *Overview of the decrease in matrix-vector products.*

Chapter 8

Conclusion

In this thesis we have discussed the acceleration of the 2D Helmholtz model HARES. We proposed several methods to improve the current implementation. In this chapter we will give an overview of main results and give a recommendation to improve HARES. Finally we will give some suggestions for further research.

We started with the derivation of the linear and non-linear Mild-Slope equation to gain insight in the equations. In the non-linear Mild-Slope equation the effects of diffraction, reflection, refraction and shoaling are included and the energy dissipation caused by wave breaking and bottom friction is taken into account.

In chapter 3 we discussed the discretization of the non-linear Mild-Slope equation. We proposed a suitable stopping criterion for the non-linear outer loop, to determine how many outer iterations are needed to obtain the desired non-linear solution. Implementing this stopping criterion already lead to a significant drop in the computing time, since too many outer iterations are performed in the current implementation of HARES. Newton's method is presented, since this method generally has a faster convergence than Picard iteration. However, the non-linearity in the Mild-Slope equation is not handled better by Newton's method than by Picard iteration. The results regarding the number of matrix-vector products and the computing time are better for Picard iteration than for Newton's method. The computing time of Picard iteration is improved by implementing inexact Picard iteration. Five different forcing sequences are proposed, the forcing sequence based on the relative update of the non-linear solution gives overall the best results.

In chapter 5 we proposed to replace the current linear solver, Bi-CGSTAB preconditioned with the incomplete LU decomposition, by the Krylov subspace method IDR(s) preconditioned with the shifted Laplace preconditioner. This replacements lead to a significant drop in the number of matrix-vector products and the computational time. In chapter 6 we determined the bounding box around the eigenvalues of the preconditioned system of equations. However, since the origin is contained in this bounding box we were not able to determine an optimal shift for the shifted Laplace preconditioner. We implemented four different types of the shifted Laplace preconditioner, where the ones with a complex shift give the best results. For the IDR(s) algorithm we proposed a choice for the initial space $\Delta \boldsymbol{X}_s$ and the coefficient ω. The initial space $\Delta \boldsymbol{X}_s$ is build using the previous s iterative solutions. This gives good results if the difference between two successive solutions is not too large. In the first couple of outer iterations the update of the solution is still quite large, hence it is not recommended to use this approach. However, after several outer iterations the update of the non-linear solution becomes small. Choosing the initial space based on the previous iterative solution do gives good results. We proposed the choose the coefficient ω based on Chebyshev polynomials on a disk in the complex plane. Since the origin is contained in the bounding box of the eigenvalue estimate of the non-linear Mild-Slope equation, this choice for ω did not give good results.

The best results for an iterative solution method are obtained by using inexact Picard iteration.

The solution of the system of equations is approximated using IDR(2) preconditioned with the incomplete LU decomposition of the shifted Laplace preconditioner with a complex shift. This implementation is around 17 times faster for the small test case Scheveningen and 35 times faster for the large test problem Maasvlakte A & B. The number of matrix-vector products is reduced by a factor 30 for the small test cas and a factor 58 for the larger test cases.

In chapter 5 we presented the direct method `MUMPS`. For two dimensional problems the state-of-the-art direct methods give good results regarding the computational time. For the considered test problems we did obtain the fastest results using this direct solver. The non-linear residual decreases fast for the large test cases and not many outer iterations need to be performed. Using inexact Picard iteration the decrease of the non-linear residual is often slower. Therefore more outer iterations are necessary and thus also more computing time for building the matrix.

We propose to Svašek Hydraulics to use a direct method, e.g. `MUMPS`, to determine the solution of the system of equations for the cases considered in this thesis. If the dimension of the matrix is considerably larger the performance of `MUMPS` will become worse and for these we propose inexact Picard with a forcing sequence based on the fifth choice. The system of equations can then be solved using IDR(2) preconditioned with a shifted Laplace preconditioner with a complex shift.

Suggestions for further research

We propose the following issues for further research:

- We were not able to determine the derivatives of $|\tilde{\zeta}|$ with respect to $\tilde{\zeta}$ in Newton's method. Newton's method presented in section 3.2 is a Jacobian free method, where we approximate the Jacobian. Therefore we do not have to compute the LU decomposition of the Jacobian to obtain the next iteration. For the non-linear Mild-Slope equation this Jacobian free method is not suitable. To improve the non-linear convergence it is recommended to investigate a different implementation of Newton's method.

- We approximated the shifted Laplace preconditioner by its incomplete LU decomposition to be able to apply it to the system of equations. The LU factorization of the shifted Laplace preconditioner can be determined for example with the software package `MUMPS`. Preconditioning the system of equations with the LU factorization will presumably lead to less matrix-vector products. The current shift of the shifted Laplace preconditioner is based on a shift found in literature. It is possible that there is an other shift for the non-linear Mild-Slope equation which gives even faster convergence.

- The forcing sequence based on the fifth choice gives already a fast convergence. But for most test cases we do need more outer iterations to obtain the non-linear solution. A different forcing sequence which does not need many more matrix-vector products and less outer iterations can give even better results. Especially for larger problems, when a direct method can become slow, this can become very beneficial.

- In section 6.5 we proposed to choose the coefficient ω based on Chebyshev polynomials. Since the origin is contained in our estimate of the eigenvalue range we were not able to test whether this choice leads to less matrix-vector products than the choice proposed by Sleijpen and van der Vorst (1995).

- The state-of-the-art direct method `MUMPS` is also available in a parallel version. This parallel version will most likely be faster than the sequential version. In the current implementation the assembly of the global matrix is the most expensive part of the complete process. Making this assembly parallel will decrease the computational time even more.

Appendix A

Calculations for the derivation of the Mild-Slope equation

A.1 Left out steps in the derivation

Multiplication of equation (2.1.12) with Z and integrating it over the water depth, from $z = -h$ until $z = 0$, gives

$$\epsilon^2 \left[\int_{-h}^{0} Z \left(\frac{\partial^2 Z}{\partial h^2} \overline{\nabla} h \cdot \overline{\nabla} h + \frac{\partial Z}{\partial h} \overline{\nabla}^2 h \right) \varphi \, dz \right] + \epsilon \left[\int_{-h}^{0} \frac{\partial Z^2}{\partial h} \overline{\nabla} h \cdot \nabla \varphi \, dz \right]$$

$$+ \int_{-h}^{0} \left[Z^2 (\nabla^2 \varphi + \kappa^2 \varphi) + \frac{\partial Z^2}{\partial z} \frac{\partial \varphi}{\partial z} + Z^2 \frac{\partial^2 \varphi}{\partial z^2} \right] \, dz = 0, \tag{A.1.1}$$

where we use that $2Z \frac{\partial Z}{\partial h} = \frac{\partial Z^2}{\partial h}$. Applying partial integration to the last two terms on the second line gives

$$\int_{-h}^{0} \left[\frac{\partial Z^2}{\partial h} \frac{\partial \varphi}{\partial z} + Z^2 \frac{\partial^2 \varphi}{\partial z^2} \right] \, dz = Z^2 \frac{\partial \varphi}{\partial z} \Big|_{-h}^{0} - \int_{-h}^{0} Z^2 \frac{\partial^2 \varphi}{\partial z^2} \, dz + \int_{-h}^{0} Z^2 \frac{\partial^2 \varphi}{\partial z^2} \, dz = Z^2 \frac{\partial \varphi}{\partial z} \Big|_{-h}^{0}.$$

Application the boundary conditions (2.1.13) and (2.1.14) gives

$$\begin{aligned} Z^2 \frac{\partial \varphi}{\partial z} \Big|_{-h}^{0} &= Z^2 \frac{\partial \varphi}{\partial z} \Big|_{z=0} - Z^2 \frac{\partial \varphi}{\partial z} \Big|_{z=-h} \\ &= -Z \left(Z \frac{\partial \varphi}{\partial z} \right)_{z=-h} = Z \left[\epsilon^2 \left(\varphi \frac{\partial Z}{\partial h} \overline{\nabla} h \cdot \overline{\nabla} h \right) + \epsilon \left(\nabla \varphi \cdot \overline{\nabla} h Z \right) \right]_{z=-h}. \end{aligned} \tag{A.1.2}$$

Substitution of the obtained expression (A.1.2) into equation (A.1.1) results in

$$\epsilon^2 \left[\int_{-h}^{0} z \left(\frac{\partial^2 Z}{\partial h 2} \overline{\nabla} h \cdot \overline{\nabla} h + \frac{\partial Z}{\partial h} \overline{\nabla}^2 h \right) \varphi \, dz \right] + \epsilon \left[\int_{-h}^{0} \frac{\partial Z^2}{\partial h} \overline{\nabla} h \cdot \nabla \varphi \, dz \right]$$

$$\int_{-h}^{0} Z^2 \left(\nabla^2 \varphi + \kappa^2 \varphi \right) \, dz + \epsilon \left[Z^2 \nabla \varphi \cdot \overline{\nabla} h \right]_{z=-h} + \epsilon^2 \left[Z \varphi \frac{\partial Z}{\partial h} \overline{\nabla} h \cdot \overline{\nabla} h \right]_{z=-h} = 0.$$

It is not straightforward that equation (2.1.16) and expression (2.1.17) are equivalent. We start with equation (2.1.17) and do the manipulations to end up with equation (2.1.16).

$$
\begin{aligned}
&\nabla \cdot \int_{-h}^{0} Z^2 \; dz \nabla \varphi_0 + \kappa^2 \varphi_0 \int_{-h}^{0} Z^2 \; dz \\
&\quad = \int_{-h}^{0} Z^2 \; dz \nabla^2 \varphi_0 + \nabla \varphi_0 \nabla \cdot \int_{-h}^{0} Z^2 \; dz + \kappa^2 \varphi_0 \int_{-h}^{0} Z^2 \; dz, \\
&\quad = \int_{-h}^{0} Z^2 \; dz \nabla^2 \varphi_0 + \nabla \varphi_0 \left[\nabla h \; Z^2 \big|_{z=-h} + \int_{-h}^{0} \frac{\partial Z^2}{\partial h} \nabla h \; dz \right] + \kappa^2 \varphi_0 \int_{-h}^{0} Z^2 \; dz, \\
&\quad = \int_{-h}^{0} Z^2 \; dz \nabla^2 \varphi_0 + \nabla \varphi_0 \left[\nabla h \; Z^2 \big|_{z=-h} + \frac{\partial}{\partial h} \int_{-h}^{0} Z^2 \; dz \nabla h - \nabla h \; Z^2 \big|_{z=-h} \right] + \kappa^2 \varphi_0 \int_{-h}^{0} Z^2 \; dz, \\
&\quad = \int_{-h}^{0} Z^2 \; dz \nabla^2 \varphi_0 + \frac{\partial}{\partial h} \int_{-h}^{0} Z^2 \; dz \nabla h \cdot \nabla \varphi_0 + \kappa^2 \varphi_0 \int_{-h}^{)} Z^2 \; dz, \\
&\quad = \int_{-h}^{0} Z^2 \; dz \nabla^2 \varphi_0 + \epsilon \frac{\partial}{\partial h} \int_{-h}^{0} Z^2 \; dz \overline{\nabla} h \cdot \nabla \varphi_0 + \kappa^2 \varphi_0 \int_{-h}^{0} Z^2 \; dz, \\
&\quad = \left(\nabla^2 \varphi_0 + \kappa^2 \varphi_0 \right) \int_{-h}^{0} Z^2 \; dz + \epsilon \overline{\nabla} h \cdot \nabla \varphi_0 \frac{\partial}{\partial h} \int_{-h}^{0} Z^2 \; dz.
\end{aligned}
$$

We apply Leibniz rule to the term $\nabla \cdot \int_{-h}^{0} z^2 \; dz$ on the second line, which results in the expression on line three. Again using the Leibniz integral rule for the integral $\int_{-h}^{0} \frac{\partial Z^2}{\partial h} \nabla h \; dz$ on the third line gives the fourth line. The expression in the final line equals equation (2.1.16)

A.2 Leibniz integral rule for variable limits

Suppose that the limits of integration a and b and the integrand $f(x, \alpha)$ all are functions of the parameter α. Then we can apply Leibniz integral rule to obtain

$$
\frac{d}{d\alpha} \int_{a(\alpha)}^{b(\alpha)} f(x, \alpha) \; dx = \frac{db(\alpha)}{d\alpha} f(b(\alpha), \alpha) - \frac{da(\alpha)}{d\alpha} f(a(\alpha), \alpha) + \int_{a(\alpha)}^{b(\alpha)} \frac{\partial}{\partial \alpha} f(x, \alpha) \; dx.
$$

A.3 Integral $\int_{-h}^{0} Z^2 \; dz$

We want to determine and expression for the following integral

$$
\int_{-h}^{0} Z^2 \; dz = \int_{-h}^{0} \left(\frac{\cosh(\kappa(z+h))}{\cosh(\kappa h)} \right)^2 \; dz.
$$

With the following computation we derive the desired expression for the integral.

$$\begin{aligned}
\int_{-h}^{0}\left(\frac{\cosh(\kappa(z+h))}{\cosh(\kappa h)}\right)^2\ dz &= \frac{1}{\cosh^2(\kappa h)}\int_{-h}^{0}\frac{1}{2}\left[\cosh(2\kappa(z+h))+1\right]\ dz,\\
&= \frac{1}{\cosh^2(\kappa h)}\frac{1}{2}\left[\frac{1}{2\kappa}\sinh(2\kappa(z+h))+z\right]_{-h}^{0},\\
&= \frac{1}{\cosh^2(\kappa h)}\frac{1}{2}\left[\frac{\sinh(2\kappa h)}{2\kappa}+h\right],\\
&= \frac{1}{2}\left[\frac{\sinh(2\kappa h)}{2\kappa\cosh^2(\kappa h)}+\frac{h}{\cosh^2(\kappa h)}\right],\\
&= \frac{1}{2}\left[\frac{\sinh(\kappa h)}{\kappa\cosh(\kappa h)}+\frac{h}{\cosh^2(\kappa h)}\right],\\
&= \frac{1}{2}\left[\frac{\tanh(\kappa h)}{\kappa}+\frac{h}{\cosh^2(\kappa h)}\right],\\
&= \frac{1}{2\kappa}\left[\tanh(\kappa h)+\frac{\kappa h\sinh(\kappa h)}{\sinh(\kappa h)\cosh^2(\kappa h)}\right],\\
&= \frac{1}{2\kappa}\left[\tanh\kappa h)+\frac{2\kappa h\tanh(\kappa h)}{\sinh(2\kappa h)}\right],\\
&= \frac{\tanh(\kappa h)}{\kappa}\frac{1}{2}\left[1+\frac{2\kappa h}{\sinh(2\kappa h)}\right],\\
&= \frac{n_0}{\kappa^2}.
\end{aligned}$$

Where we use the dimensionless dispersion relation $1=\kappa\tanh(\kappa h)$ and $n_0=\frac{1}{2}\left(1+\frac{2\kappa h}{\sinh(2\kappa h)}\right)$.

Appendix B

Determining the eigenvalues of $\boldsymbol{M}^e\boldsymbol{x}^e = \lambda^{\boldsymbol{M}^e,\boldsymbol{P}_i^e}\boldsymbol{P}_i^e\boldsymbol{x}^e$

The eigenvalues of the generalized eigenvalue problem $\boldsymbol{M}^e\boldsymbol{x}^e = \lambda^{\boldsymbol{M}^e,\boldsymbol{P}_{int}}\boldsymbol{P}^e\boldsymbol{x}^e$ can be easily determined using Maple. We obtain the following three eigenvalues

$$\begin{aligned}
\lambda_1^e &= -\frac{1}{s^2},\\
\lambda_2^e &= \frac{\Delta^2 k_0^2\left(-6n_0\alpha - s^2k_0^2 + 12n_0\sqrt{\frac{\alpha^2}{4}-\frac{3}{\Delta^2}}\right)}{s^4k_0^4\Delta^2 + 432n_0^2 + 12n_0s^2k_0^2\Delta^2\alpha},\\
\lambda_3^e &= \frac{\Delta^2 k_0^2\left(-6n_0\alpha - s^2k_0^2 - 12n_0\sqrt{\frac{\alpha^2}{4}-\frac{3}{\Delta^2}}\right)}{s^4k_0^4\Delta^2 + 432n_0^2 + 12n_0s^2k_0^2\Delta^2\alpha}.
\end{aligned}$$

All the coefficients are larger than zero and its value depend on the actual location of the element in the harbour and $\frac{\alpha^2}{4} - \frac{3}{\Delta^2} > 0$ (since $\boldsymbol{M}^e$ and $\boldsymbol{P}_i^e$ are both Hermitian) it is easily seen that $\lambda_1^e, \lambda_3^e < 0$ and $\lambda_3^e < \lambda_2^e$. We would like to know whether there is an eigenvalue which is the largest (resp. smallest) for all the element matrices and does not depend on the choice of s whether one eigenvalue is larger (resp. smaller) than the other eigenvalue. We will show that $\lambda_1^e < \lambda_3^e$ regardless of the value of the coefficients and then it must always hold that the smallest eigenvalue is λ_1^e and the largest eigenvalue λ_2^e. We rewrite eigenvalue λ_3^e as

$$\lambda_3^e = -\frac{1}{s^2}\left\{\frac{\Delta^2 k_0^2\left(6n_0\alpha + s^2k_0^2 + 12n_0\sqrt{\frac{\alpha^2}{4}-\frac{3}{\Delta^2}}\right)}{s^2k_0^4\Delta^2 + 432\frac{n_0^2}{s^2} + 12n_0k_0^2\Delta^2\alpha}\right\}.$$

It rests to show that

$$\frac{\Delta^2 k_0^2\left(6n_0\alpha + s^2k_0^2 + 12n_0\sqrt{\frac{\alpha^2}{4}-\frac{3}{\Delta^2}}\right)}{s^2k_0^4\Delta^2 + 432\frac{n_0^2}{s^2} + 12n_0k_0^2\Delta^2\alpha} < 1.$$

The following upper bound for the numerator is determined

$$\begin{aligned}
6n_0k_0^2\Delta^2\alpha + s^2k_0^4\Delta^2 + 12n_0k_0^2\Delta^2\sqrt{\frac{\alpha^2}{4}-\frac{3}{\Delta^2}} &< 6n_0k_0^2\Delta^2\alpha + s^2k_0^4\Delta^2 + 12n_0k_0^2\Delta^2\sqrt{\frac{\alpha^2}{4}}\\
&= s^2k_0^4\Delta^2 + 12n_0k_0^2\Delta^2\alpha\\
&< s^2k_0^4\Delta^2 + 432\frac{n_0^2}{s^2} + 12n_0k_0^2\Delta^2\alpha.
\end{aligned}$$

Hence we have showed that $\lambda_1^e < \lambda_3^e$ and hence that for each element the minimal eigenvalue is given by λ_1^e and the maximal eigenvalue of λ_2^e. Using the same approximation one can also show that $\lambda_2^e < 0$, since

$$-\left(6n_0\alpha + s^2k_0^2 - 12n_0\sqrt{\frac{\alpha^2}{4} - \frac{3}{\Delta^2}}\right) < -s^2k_0^2 < 0.$$

Appendix C

Numerical results

C.1 Input files

For the computation of the numerical results we use the following input files.

C.1.1 Scheveningen

&general			
meshfname	=	'mesh_huidigesituatie_5m.out'	!*Mesh of the geometry*
depthfname	=	'bodem_huidigesituatie_5m.asc'	!*Depth profile of the geometry*
globaldepth	=	10	!*Global depth in the geometry*
depthoutside	=	10	!*Depth outside the geometry*
waterlevel	=	2	!*Waterlevel inside the geometry*
cf	=	0.01	!*Bottom friction coefficient*
gamma	=	0.73	!*Wave breaking coefficient (Miche)*
bottomfriction	=	.TRUE.	!*Switch bottom friction coefficient on/off*
wavebreaking	=	.TRUE.	!*Switch wave breaking impact on/off*
accelerate_iteration	=	.TRUE.	!*Switch iteration shortcut on/off*
&wave			
waveheights	=	2.0	!*Wave height of the incoming wave*
waveperiods	=	8.0	!*Wave period of the incoming wave*
wavedirections	=	315	!*Wave direction of the incoming wave*

C.1.2 Maasvlakte - Bottom topography A

&general		
meshfname	=	'mesh_8m.out'
depthfname	=	'bodem_A_8m_NAP0m.asc'
globaldepth	=	25
depthoutside	=	25
waterlevel	=	3.00
cf	=	0.01
gamma	=	0.73
bottomfriction	=	.TRUE.
wavebreaking	=	.TRUE.
accelerate_iteration	=	.TRUE.
&wave		
waveheights	=	1.0
waveperiods	=	10.0
wavedirections	=	313

C.1.3 Maasvlakte - Bottom topography B

&general

meshfname	=	'mesh_8m.out'
depthfname	=	'bodem_B_8m_NAP0m.asc'
globaldepth	=	17
depthoutside	=	17
waterlevel	=	3.00
cf	=	0.01
gamma	=	0.73
bottomfriction	=	.TRUE.
wavebreaking	=	.TRUE.
accelerate_iteration	=	.TRUE.

&wave

waveheights	=	1.0
waveperiods	=	10.0
wavedirections	=	300

C.1.4 Malta

&general

meshfname	=	'A_huidig_mesh_6m.out'
depthfname	=	'A_huidig_bodem_6m.asc'
globaldepth	=	30
depthoutside	=	30
waterlevel	=	0.0
cf	=	0.01
gamma	=	0.73
bottomfriction	=	.TRUE.
wavebreaking	=	.TRUE.
accelerate_iteration	=	.TRUE.

&wave

waveheights	=	1.0
waveperiods	=	9.0
wavedirections	=	130

C.2 Detailed numerical results

	Total time	# outer	Solve $\boldsymbol{S\zeta} = \boldsymbol{b}$	Build $\boldsymbol{S}$	# matvecs	Rel. error
Initial measurement - ILU(0)-Bi-CGSTAB						
Scheveningen	994.81	25	934.17	58.26	60206.00	5.6495E-06
Maasvlakte A	6700.50	25	6543.40	150.81	148610.00	3.6439E-07
Maasvlakte B	6898.30	25	6741.20	150.35	156066.00	1.9198E-07
Malta	5851.20	25	5697.50	146.78	132222.00	4.7628E-07
Stopping criterion outer loop - ILU(0) of matrix $\boldsymbol{S}$						
Wave number k_0		Scheveningen				
Bi-CGSTAB	1328.93	26	982.89	60.63	62560	1.7638E-06
IDR(2)	357.66	26	298.25	56.97	19746	1.9124E-05
IDR(4)	367.79	26	308.20	57.09	17657	1.8203E-05
IDR(8)	397.58	26	338.15	56.99	16672	1.8323E-05
		Maasvlakte A				
Bi-CGSTAB	2719.23	10	2654.50	60.16	60490	8.5397E-08
IDR(2)	818.75	11	747.94	66.16	16948	2.1647E-06
IDR(4)	788.65	11	717.85	66.14	14290	2.0517E-06
IDR(8)	780.28	10	715.58	60.11	11506	1.5047E-06
		Maasvlakte B				
Bi-CGSTAB	1776.98	6	1736.70	36.06	39612	1.1028E-07
IDR(2)	665.86	6	625,7791	36.01	14182	1.7702E-06
IDR(4)	611.49	6	571.36	36.01	11380	2.0461E-06
IDR(8)	616.74	6	576.64	36.04	9187	3.3404E-06
		Malta				
Bi-CGSTAB	1916.61	8	1865.40	46.95	43158	3.4016E-07
IDR(2)	661.81	8	610.64	46.90	13944	4.6274E-06
IDR(4)	672.71	8	621.51	46.91	12484	2.5872E-06
IDR(8)	757.88	8	706.69	46.91	11356	3.7568E-06
Modified wave number $\hat{p}$		Scheveningen				
Bi-CGSTAB	1060.91	26	1001.10	57.37	63508	3.0996E-06
IDR(2)	355.07	26	295.32	57.29	19199	6.0172E-06
IDR(4)	356.85	25	299.31	55.10	17199	2.5433E-05
IDR(8)	396.17	25	338.70	55.09	16058	2.1394E-05
		Maasvlakte A				
Bi-CGSTAB	2697.33	10	2632.30	60.44	59814	2.8858E-08
IDR(2)	746.64	10	681.65	60.47	15171	1.4231E-06
IDR(4)	736.53	10	671.47	60.47	13197	1.9008E-06
IDR(8)	745.29	10	680.25	60.49	10758	1.2828E-06
		Maasvlakte B				
Bi-CGSTAB	1776.48	6	1736.10	36.25	39358	1.1279E-07
IDR(2)	657.10	6	616.85	36.13	13796	3.5708E-06
IDR(4)	624.23	6	583.96	36.15	11477	3.5857E-06
IDR(8)	634.88	6	593.70	37.05	9404	3.1277E-06
		Malta				
Bi-CGSTAB	1932.36	8	1880.90	47.17	43340	3.3841E-07
IDR(2)	659.43	8	607.99	47.19	13709	6.0800E-06
IDR(4)	700.26	9	642.68	53.12	12708	2.7366E-06
IDR(8)	772.74	8	721.27	47.20	11553	3.3059E-06

	Total time	# outer	Solve $\boldsymbol{S\zeta = b}$	Build $\boldsymbol{S}$	# matvecs	Rel. error
Using the incoming wave to determine a value of $\boldsymbol{W(x, y, \tilde{\zeta})}$ - ILU(0) of matrix $\boldsymbol{S}$						
Wave number k_0		Scheveningen				
Bi-CGSTAB	868.98	22	818.52	48.23	53002	1.3259E-08
IDR(2)	261.40	22	211.10	48.05	14012	2.8084E-06
IDR(4)	286.41	22	228.06	56.07	13090	4.4698E-06
IDR(8)	308.34	21	259.96	46.16	12479	1.9123E-05
		Maasvlakte A				
Bi-CGSTAB	2729.67	10	2664.70	60.35	60360	1.3400E-09
IDR(2)	797.25	10	732.73	60.00	16492	2.3995E-06
IDR(4)	776.02	10	711.37	60.06	13933	1.7833E-06
IDR(8)	762.42	10	697.56	60.32	11064	1.2422E-06
		Maasvlakte B				
Bi-CGSTAB	1759.80	6	1719.60	36.14	39096	4.2010E-09
IDR(2)	624.64	6	584.50	36.07	13215	2.3223E-06
IDR(4)	584.60	6	544.24	36.20	10704	5.5131E-06
IDR(8)	590.55	6	550.34	36.12	8729	4.5855E-06
		Malta				
Bi-CGSTAB	1904.41	8	1852.90	47.24	42786	4.4012E-09
IDR(2)	614.17	8	562.82	47.04	12738	4.6963E-06
IDR(4)	596.65	8	545.30	47.02	10869	6.0172E-06
IDR(8)	708.54	8	657.17	47.09	10495	2.5247E-06
Modified wave number $\hat{p}$		Scheveningen				
Bi-CGSTAB	849.31	21	800.78	46.29	50944	1.3295E-08
IDR(2)	263.78	21	215.31	46.27	13839	3.8965E-06
IDR(4)	269.78	21	221.35	46.18	12823	5.7965E-06
IDR(8)	312.74	21	264.19	46.33	12370	4.9912E-06
		Maasvlakte A				
Bi-CGSTAB	2640.63	10	2575.60	60.50	58476	1.0541E-09
IDR(2)	733.45	10	668.33	60.60	14920	2.5974E-06
IDR(4)	713.28	10	648.15	60.53	12680	2.1453E-06
IDR(8)	726.77	10	661.61	60.62	10453	2.0457E-06
		Maasvlakte B				
Bi-CGSTAB	1762.87	6	1722.50	36.31	39016	4.2052E-09
IDR(2)	624.13	6	583.66	36.38	13013	4.2101E-06
IDR(4)	590.63	6	550.15	36.29	10750	2.7335E-06
IDR(8)	603.58	6	562.30	37.18	8911	3.4382E-06
		Malta				
Bi-CGSTAB	1913.73	8	1862.20	47.26	42886	4.6396E-09
IDR(2)	607.77	8	556.23	47.27	12496	6.2144E-06
IDR(4)	617.26	8	565.65	47.30	11276	4.4665E-06
IDR(8)	728.05	8	676.52	47.27	10729	2.9225E-06
Shifted Laplace preconditioner - complex shift - version 1						
Wave number k_0		Scheveningen				
Bi-CGSTAB	665.24	22	605.23	57.85	38326	1.3243E-08
IDR(2)	212.56	22	160.45	49.97	10324	5.5793E-06
IDR(4)	231.90	22	179.81	49.91	10332	6.4878E-06
IDR(8)	261.08	21	211.35	47.65	9961	1.7643E-05
		Maasvlakte A				
Bi-CGSTAB	999.97	10	923.55	71.81	20872	1.0486E-09
IDR(2)	378.18	11	304.82	68.57	6779	1.9571E-06
IDR(4)	401.71	11	328.25	68.66	6424	1.7439E-06

	Total time	# outer	Solve $\boldsymbol{S\zeta} = \boldsymbol{b}$	Build $\boldsymbol{S}$	# matvecs	Rel. error
IDR(8)	480.80	11	407.49	68.59	6413	1.2518E-06
		Maasvlakte B				
Bi-CGSTAB	694.81	6	653.28	37.42	14804	3.5749E-09
IDR(2)	279.78	6	238.24	37.40	5305	5.4257E-06
IDR(4)	311.59	6	269.92	37.47	5284	5.7802E-06
IDR(8)	370.98	6	329.42	37.42	5202	3.7939E-06
		Malta				
Bi-CGSTAB	1275.12	8	1222.10	48.66	28044	4.0707E-09
IDR(2)	398.14	8	344.63	49.20	7779	4.9593E-06
IDR(4)	464.17	8	411.02	48,7510	8166	4.9847E-06
IDR(8)	544.61	8	491.55	48.75	7855	3.6419E-06
Modified wave number $\hat{p}$		Scheveningen				
Bi-CGSTAB	629.44	21	579.52	47.84	36568	1.3275E-08
IDR(2)	207.27	21	157.33	47.85	10091	9.2227E-06
IDR(4)	228.54	21	178.52	47.87	10116	4.3449E-06
IDR(8)	263.75	21	213.94	47.71	9967	3.5246E-06
		Maasvlakte A				
Bi-CGSTAB	953.02	10	885.95	62.52	20172	6.2598E-10
IDR(2)	341.87	10	274.63	62.63	6074	2.0153E-06
IDR(4)	365.79	10	298.69	62.46	5858	2.1279E-06
IDR(8)	437.54	10	370.22	62.68	5839	1.8614E-06
		Maasvlakte B				
Bi-CGSTAB	707.19	6	665.63	37.42	14884	3.4499E-09
IDR(2)	276.40	6	234.73	37.53	5227	6.9247E-06
IDR(4)	300.73	6	259.09	37.54	5075	7.0272E-06
IDR(8)	371.51	6	324.78	42.55	5139	3.4382E-06
		Malta				
Bi-CGSTAB	1285.04	8	1232.10	48.64	28306	3.9840E-09
IDR(2)	401.26	8	348.13	48.84	7901	6.4564E-06
IDR(4)	466.33	8	413.11	48.86	8151	3.4616E-06
IDR(8)	544.75	8	491.51	48.92	7851	3.0825E-06
Shifted Laplace preconditioner - real shift - version 2						
Wave number k_0		Scheveningen				
Bi-CGSTAB	833.76	22	780.61	51.01	49444	1.3244E-08
IDR(2)	232.99	22	181.06	49.78	11648	6.9158E-06
IDR(4)	248.25	21	198.64	47.46	11330	2.0306E-05
IDR(8)	288.27	22	236.31	49.81	11131	4.4927E-06
		Maasvlakte A				
Bi-CGSTAB	1312.60	10	1245.40	62.53	28102	1.2043E-09
IDR(2)	452.56	11	379.49	68.30	8433	2.1450E-06
IDR(4)	472.65	10	405.82	62.10	7924	2.1719E-06
IDR(8)	554.71	10	487.42	62.63	7680	1.1643E-06
		Maasvlakte B				
Bi-CGSTAB	947.18	6	905.77	37.27	20490	3.5819E-09
IDR(2)	332.50	6	290.91	37.43	6468	7.0499E-06
IDR(4)	368.90	6	321.80	42.92	6289	5.7694E-06
IDR(8)	426.14	6	384.74	37.28	6054	2.9985E-06
		Malta				
Bi-CGSTAB	947.18	6	905.77	37.27	20490	3.5819E-09
IDR(2)	332.50	6	290.91	37.43	6468	7.0499E-06
IDR(4)	368.90	6	321.80	42.92	6289	5.7694E-06

	Total time	# outer	Solve $\boldsymbol{S\zeta} = \boldsymbol{b}$	Build $\boldsymbol{S}$	# matvecs	Rel. error
IDR(8)	426.14	6	384.74	37.28	6054	2.9985E-06
Modified wave number $\hat{p}$		SCHEVENINGEN				
Bi-CGSTAB	805.83	21	754.78	48.92	46952	1.3270E-08
IDR(2)	231.69	21	181.98	47.59	11589	5.6041E-06
IDR(4)	245.25	21	195.51	47.59	11141	4.6869E-06
IDR(8)	287.23	21	237.56	47.59	11159	5.8713E-06
		MAASVLAKTE A				
Bi-CGSTAB	1323.12	10	1256.20	62.24	28452	6.7107E-10
IDR(2)	406.56	10	339.58	62.32	7516	2.3816E-06
IDR(4)	440.52	10	373.68	62.15	7323	2.0159E-06
IDR(8)	513.52	11	437.28	71.49	6895	2.0339E-06
		MAASVLAKTE B				
Bi-CGSTAB	990.57	6	949.04	37.42	20878	3.4700E-09
IDR(2)	333.61	6	292.22	37.27	6475	5.3631E-06
IDR(4)	361.03	6	313.80	43.08	6133	5.0768E-06
IDR(8)	425.57	6	384.20	37.28	6064	3.0755E-06
		MALTA				
Bi-CGSTAB	1602.45	8	1549.30	48.79	35256	4.1848E-09
IDR(2)	460.68	8	407.64	48.68	8937	5.8951E-06
IDR(4)	493.05	8	439.86	48.74	8507	5.1772E-06
IDR(8)	572.37	8	519.23	48.71	8082	3.0421E-06
Shifted Laplace preconditioner - real-complex shift - version 3						
Wave number k_0		SCHEVENINGEN				
Bi-CGSTAB	662.35	22	610.31	49.89	38700	1.3248E-08
IDR(2)	210.47	22	158.62	49.71	10208	5.7735E-06
IDR(4)	232.10	22	180.17	49.74	10267	5.5825E-06
IDR(8)	262.68	21	213.25	47.33	9998	2.1246E-05
		MAASVLAKTE A				
Bi-CGSTAB	989.96	10	922.47	62.67	20834	1.0896E-09
IDR(2)	363.30	10	296.48	62.19	6589	2.3367E-06
IDR(4)	399.69	10	332.64	62,3679	6505	2.3288E-06
IDR(8)	469.17	10	402.06	62.48	6350	1.6898E-06
		MAASVLAKTE B				
Bi-CGSTAB	701.28	6	659.97	37.19	14944	3.5765E-09
IDR(2)	278.53	6	236.10	38.29	5256	5.9106E-06
IDR(4)	308.78	6	267.32	37.30	5229	5.9490E-06
IDR(8)	363.24	6	321.83	37.28	5084	4,0146E-06
		MALTA				
Bi-CGSTAB	1280.82	8	1227.90	48.57	28200	4.0216E-09
IDR(2)	403.14	8	350.31	48.51	7899	5.5008E-06
IDR(4)	457.25	8	404.37	48.52	8036	3.7228E-06
IDR(8)	536.83	8	484.01	48.49	7737	3.4592E-06
Modified wave number $\hat{p}$		SCHEVENINGEN				
Bi-CGSTAB	623.37	21	573.64	47.65	36526	1.3271E-08
IDR(2)	209.07	22	156.89	50.05	10181	1.8389E-05
IDR(4)	230.36	22	178.23	49.98	10218	2.3774E-05
IDR(8)	263.86	21	214.17	47.60	10036	6.0534E-06
		MAASVLAKTE A				
Bi-CGSTAB	965.11	10	897.61	62.81	20220	6.5787E-10
IDR(2)	336.81	10	269.68	62.56	6023	2.9615E-06
IDR(4)	377.31	11	300.32	72.35	5896	2.3760E-06

	Total time	# outer	Solve $\boldsymbol{S\zeta} = \boldsymbol{b}$	Build $\boldsymbol{S}$	# matvecs	Rel. error
IDR(8)	447.57	11	374.21	68.63	5900	1.1633E-06
		Maasvlakte B				
Bi-CGSTAB	698.58	6	656.57	37.89	14888	3.4348E-09
IDR(2)	277.68	6	235.92	37.63	5255	4.4068E-06
IDR(4)	307.84	6	266.12	37.60	5224	7.4859E-06
IDR(8)	371.55	6	329.89	37.56	5191	3.3739E-06
		Malta				
Bi-CGSTAB	1340.54	8	1287.20	49.06	28662	3.9741E-09
IDR(2)	405.79	8	352.62	48.88	7934	4.8923E-06
IDR(4)	450.90	8	397.76	48.84	7935	4.7053E-06
IDR(8)	544.65	8	491.56	48.83	7902	4.2181E-06
Shifted Laplace preconditioner - Hermitian preconditioner - version 4						
Modified wave number $\hat{p}$		Scheveningen				
Bi-CGSTAB	826.95	21	777.26	47.58	48340	1.3278E-08
IDR(2)	255.42	21	205.80	47.52	13289	6.9170E-06
IDR(4)	283.44	22	231.34	49.95	13304	1.9420E-05
IDR(8)	336.73	22	284.66	49.94	13265	1.8646E-05
		Maasvlakte A				
Bi-CGSTAB	1628.27	10	1561.10	62.54	34682	8.6523E-10
IDR(2)	489.70	10	422.51	62.52	9187	2.4835E-06
IDR(4)	513.96	10	446.93	62.32	8757	1.9060E-06
IDR(8)	597.98	10	530.90	62.43	8184	1.1387E-06
		Maasvlakte B				
Bi-CGSTAB	1137.70	6	1096.10	37.47	24820	3.5873E-09
IDR(2)	399.10	6	357.62	37.40	7978	4.3999E-06
IDR(4)	413.93	6	372.35	37.47	7287	2.5354E-06
IDR(8)	488.31	6	446.72	37.49	6810	3.5905E-06
		Malta				
Bi-CGSTAB	1574.13	8	1521.00	48.73	34038	4.1413E-09
IDR(2)	474.47	8	421.35	48.72	9417	5.5631E-06
IDR(4)	562.86	8	501.79	56.65	9802	3.2686E-06
IDR(8)	634.57	8	581.50	48.70	8975	2.8441E-06
Choosing the initial space $\boldsymbol{\Delta X_s}$ - ILU(0) of matrix $\boldsymbol{S}$						
Wave number k_0		Scheveningen				
IDR(2)	187.69	22	137.16	48.26	8844	3.3027E-06
IDR(4)	181.82	22	131.29	48.22	7536	5.2425E-06
IDR(8)	200.70	22	150.34	48.08	7126	3.0016E-06
		Maasvlakte A				
IDR(2)	967.03	10	902.31	60.18	20112	4.2603E-06
IDR(4)	703.95	10	639.16	60.20	12554	2.9561E-06
IDR(8)	743.24	10	678.46	60,2158	10730	2.0707E-06
		Maasvlakte B				
IDR(2)	602.39	6	562.18	36.09	12555	6.5738E-06
IDR(4)	541.23	6	500.91	36.18	9817	4.4160E-06
IDR(8)	592.33	6	552.11	36.09	8729	4.5855E-06
		Malta				
IDR(2)	522.12	8	470.81	47.02	10613	6.9438E-06
IDR(4)	508.30	8	456.95	47.03	9061	7.7771E-06
IDR(8)	709.60	8	656.85	48.46	10495	2.5247E-06

	Total time	# outer	Solve $\boldsymbol{S\zeta = b}$	Build $\boldsymbol{S}$	# matvecs	Rel. error
Modified wave number $\hat{p}$		Scheveningen				
IDR(2)	192.36	21	143.82	46.31	9257	4.5859E-06
IDR(4)	183.22	21	134.56	46.38	7558	4.4634E-06
IDR(8)	203.13	21	154.67	46.23	7280	3.6358E-06
		Maasvlakte A				
IDR(2)	872.29	10	807.11	60.60	18040	3.2239E-06
IDR(4)	705.07	10	639.85	60.59	12241	3.8050E-06
IDR(8)	721.51	10	656.35	60.54	10370	2.2797E-06
		Maasvlakte B				
IDR(2)	603.55	6	563.14	36.31	12570	7.9160E-06
IDR(4)	537.87	6	497.33	36.33	9777	5.4035E-06
IDR(8)	603.03	6	562.60	36.28	8911	4.8188E-06
		Malta				
IDR(2)	534.93	8	483.25	47.36	10696	5.9781E-06
IDR(4)	518.26	8	466.75	47.22	9246	4.2137E-06
IDR(8)	730.41	8	671.05	55.09	10729	2.9225E-06
Choosing the initial space $\boldsymbol{\Delta X_s}$ - Shifted Laplace preconditioner version 1						
Wave number k_0		Scheveningen				
IDR(2)	147.04	21	97.38	47.58	6279	2.4054E-05
IDR(4)	156.27	22	104.15	49.95	5977	5.6999E-06
IDR(8)	178.73	22	125.48	51.10	5938	3.9160E-06
		Maasvlakte A				
IDR(2)	416.86	10	349.89	62.32	7776	3.2060E-06
IDR(4)	365.47	10	298.31	62.48	5842	4.1673E-06
IDR(8)	465.16	10	398.31	62.22	6289	1.3521E-06
		Maasvlakte B				
IDR(2)	287.43	6	245.96	37.33	5485	4.6321E-06
IDR(4)	289.47	6	247.86	37.39	4859	3.4747E-06
IDR(8)	371.34	6	329.98	37.21	5202	3.7939E-06
		Malta				
IDR(2)	349.55	8	288.66	56.58	6513	4.4464E-06
IDR(4)	393.68	8	340.64	48.69	6757	2.8024E-06
IDR(8)	546.53	8	492.26	49.94	7855	3.6419E-06
Modified wave number $\hat{p}$		Scheveningen				
IDR(2)	148.49	21	98.58	47.81	6340	1.2000E-03
IDR(4)	156.47	21	106.59	47.72	6103	1.2000E-03
IDR(8)	178.42	21	128.69	47.62	6067	1.2000E-03
		Maasvlakte A				
IDR(2)	391.70	10	324.48	62.56	7207	1.0746E-05
IDR(4)	348.59	10	281.54	62.37	5500	1.0855E-05
IDR(8)	431.42	10	364.30	62.52	5744	9.5351E-06
		Maasvlakte B				
IDR(2)	288.53	6	246.88	37.49	5480	6.8395E-05
IDR(4)	288.40	6	246.78	37.47	4822	6.7106E-05
IDR(8)	367.56	6	326.02	37.43	5136	6.7178E-05
		Malta				
IDR(2)	334.78	8	274.16	56.30	6167	5.5228E-06
IDR(4)	381.14	8	328.07	48.72	6503	4.4813E-06
IDR(8)	550.68	8	497.61	48.75	7938	3.1253E-06

	Total time	# outer	Solve $\boldsymbol{S\zeta = b}$	Build $\boldsymbol{S}$	# matvecs	Rel. error
Newton's method - IDR(2) - Shifted Laplace preconditioner version 1						
Wave number k_0 ***- stopping criterion*** (7.6.1)						
Scheveningen	428.59	22	375.39	51.54	24264	1.2667E-05
Maasvlakte A	948.99	10	880.76	64.72	19579	8.2031E-06
Maasvlakte B	598.26	6	556.36	38.76	12367	5.0466E-06
Malta	920.93	8	867.19	50.50	19561	9.5559E-06
Wave number k_0 ***- stopping criterion*** (7.6.2)						
Scheveningen	684.91	22	631.40	51.83	40727	6.9342E-10
Maasvlakte A	1471.57	10	1402.90	65.04	30019	6.2598E-10
Maasvlakte B	943.01	6	900.50	39.35	18999	7.3955E-10
Malta	1612.74	8	1558.90	50.62	33811	7.4737E-10
Updated wave number $\hat{p}$ ***- stopping criterion*** (7.6.1)						
Scheveningen	432.51	22	376.20	51.92	24283	2.2884E-05
Maasvlakte A	945.28	10	875.84	65.04	19516	9.3442E-06
Maasvlakte B	596.91	6	553.41	39.11	12353	5.9935E-06
Malta	931.22	8	876.13	50.69	19722	8.0468E-06
Updated wave number $\hat{p}$ ***- stopping criterion*** (7.6.2)						
Scheveningen	713.22	22	659.12	52.36	40770	2.0203E-10
Maasvlakte A	1486.96	10	1417.60	65.64	30465	4.7759E-10
Maasvlakte B	924.81	6	882.32	39.36	18877	9.2263E-10
Malta	1559.95	8	1505.90	50.84	33381	8.6091E-10
Changed water level of the test case Maasvlakte - SL prec. version 1						
Wave number k_0		Maasvlakte A - water level 1.00 m				
IDR(2)	363.73	9	304.46	55.70	6774	-
IDR(4)	400.78	9	341.33	55.84	6686	-
		Maasvlakte A - water level 2.00 m				
IDR(2)	392.74	11	320.45	68.47	7140	-
IDR(4)	440.08	11	367.80	68.45	7201	-
		Maasvlakte B - water level 1.00 m				
IDR(2)	577.29	11	505.33	68.24	11278	-
IDR(4)	604.15	11	532.19	68.18	10427	-
		Maasvlakte B - water level 2.00 m				
IDR(2)	423.11	9	363.53	56.02	8122	-
IDR(4)	465.32	9	405.74	56.03	7953	-
Modified wavenumber $\hat{p}$		Maasvlakte A - waterlevel 1.00 m				
IDR(2)	369.78	9	309.24	56.02	6667	-
IDR(4)	400.64	9	338.73	57.42	6665	-
		Maasvlakte A - water level 2.00 m				
IDR(2)	381.23	11	307.84	68.66	6856	-
IDR(4)	416.13	11	341.87	69.49	6728	-
		Maasvlakte B - water level 1.00 m				
IDR(2)	581.57	11	498.35	78.53	11162	-
IDR(4)	594.63	11	521.45	68.47	10281	-
		Maasvlakte B - water level 2.00 m				
IDR(2)	421.58	9	360.84	56.31	8074	-
IDR(4)	460.26	9	399.63	56.16	7872	-
Inexact Picard - SL prec. version 1 - choice 1 - Updated wave number $\hat{p}$						
$\boldsymbol{\eta_0 = 0.1}$		Scheveningen				
Bi-CGSTAB	105.35	33	27.49	75.34	1500	3.0878E-05
IDR(2) - ST1	64.92	21	26.57	36.55	1439	3.2505E-05

	Total time	# outer	Solve $\boldsymbol{S\zeta = b}$	Build $\boldsymbol{S}$	# matvecs	Rel. error
IDR(2) - $\Delta \boldsymbol{X}_s$ - ST1	105.16	25	46.19	56.81	2625	3.2505E-05
IDR(2) - ST2	-	49	-	-	-	-
IDR(2) - $\Delta \boldsymbol{X}_s$ - ST2	-	49	-	-	-	-
$\boldsymbol{\eta_0 = 0.5}$						
Bi-CGSTAB	90.29	27	26.47	61.62	1498	3.6518E-05
IDR(2) - ST1	66.22	17	25.81	38.62	1431	2.5385E-05
IDR(2) - $\Delta \boldsymbol{X}_s$ - ST1	125.09	29	56.91	65.86	3228	2.5385E-05
IDR(2) - ST2	-	49	-	-	-	-
IDR(2) - $\Delta \boldsymbol{X}_s$ - ST2	-	49	-	-	-	-
$\boldsymbol{\eta_0 = 0.1}$		Maasvlakte A				
Bi-CGSTAB	247.38	15	148.57	94.13	2930	1.1898E-06
IDR(2) - ST1	220.65	11	147.35	69.00	2653	1.5988E-06
IDR(2) - $\Delta \boldsymbol{X}_s$ - ST1	358.41	14	266.22	87.57	4979	1.5988E-06
IDR(2) - ST2	-	49	-	-	-	-
IDR(2) - $\Delta \boldsymbol{X}_s$ - ST2	-	49	-	-	-	-
$\boldsymbol{\eta_0 = 0.5}$						
Bi-CGSTAB	280.48	20	150.00	125.00	2826	2.1177E-06
IDR(2) - ST1	202.86	11	129.79	68.89	2376	1.9451E-06
IDR(2) - $\Delta \boldsymbol{X}_s$ - ST1	339.86	12	249.96	85.62	4787	1.9451E-06
IDR(2) - ST2	-	49	-	-	-	-
IDR(2) - $\Delta \boldsymbol{X}_s$ - ST2	-	49	-	-	-	-
$\boldsymbol{\eta_0 = 0.1}$		Maasvlakte B				
Bi-CGSTAB	218.68	9	158.34	56.50	3144	7.0369E-07
IDR(2) - ST1	218.79	9	158.45	56.45	2965	3.0564E-07
IDR(2) - $\Delta \boldsymbol{X}_s$ - ST1	401.18	8	347.65	49.80	6528	3.0564E-07
IDR(2) - ST2	-	49	-	-	-	-
IDR(2) - $\Delta \boldsymbol{X}_s$ - ST2	-	49	-	-	-	-
$\boldsymbol{\eta_0 = 0.5}$						
Bi-CGSTAB	268.38	18	150.49	112.98	2988	3.4929E-06
IDR(2) - ST1	264.25	9	165.78	94.01	3066	7.5100E-07
IDR(2) - $\Delta \boldsymbol{X}_s$ - ST1	493.71	15	392.42	96.67	7494	7.5100E-07
IDR(2) - ST2	-	49	-	-	-	-
IDR(2) - $\Delta \boldsymbol{X}_s$ - ST2	-	49	-	-	-	-
$\boldsymbol{\eta_0 = 0.1}$		Malta				
Bi-CGSTAB	275.03	11	203.39	67.45	3990	1.6050E-06
IDR(2) - ST1	253.01	9	193.77	55.33	3664	8.1486E-07
IDR(2) - $\Delta \boldsymbol{X}_s$ - ST1	365.87	9	298.58	63.41	5620	8.1486E-07
IDR(2) - ST2	-	49	-	-	-	-
IDR(2) - $\Delta \boldsymbol{X}_s$ - ST2	-	49	-	-	-	-
$\boldsymbol{\eta_0 = 0.5}$						
Bi-CGSTAB	301.39	18	186.85	109.63	3830	1.7676E-06
IDR(2) - ST1	234.04	11	162.57	67.29	3104	2.2985E-06
IDR(2) - $\Delta \boldsymbol{X}_s$ - ST1	317.73	9	258.86	54.92	5027	2.2985E-06
IDR(2) - ST2	-	49	-	-	-	-
IDR(2) - $\Delta \boldsymbol{X}_s$ - ST2	-	49	-	-	-	-
Inexact Picard - SL prec. version 1 - choice 2 - Updated wave number $\hat{p}$						
$\boldsymbol{\eta_0 = 0.1}$		Scheveningen				
IDR(2) - ST1	89.93	19	45.22	43.04	2903	3.1000E-03
IDR(2) - $\Delta \boldsymbol{X}_s$ - ST1	108.32	21	59.03	47.54	3792	3.1307E-03
IDR(2) - ST2	-	49	-	-	-	-
IDR(2) - $\Delta \boldsymbol{X}_s$ - ST2	-	49	-	-	-	-

	Total time	# outer	Solve $\boldsymbol{S\zeta} = \boldsymbol{b}$	Build $\boldsymbol{S}$	# matvecs	Rel. error
$\boldsymbol{\eta_0 = 0.5}$						
IDR(2) - ST1	76.80	17	36.73	38.47	2368	3.1312E-03
IDR(2) - $\Delta \boldsymbol{X}_s$ - ST1	109.31	22	57.69	49.82	3714	3.1294E-03
IDR(2) - ST2	-	49	-	-	-	-
IDR(2) - $\Delta \boldsymbol{X}_s$ - ST2	-	49	-	-	-	-
$\boldsymbol{\eta_0 = 0.1}$		Maasvlakte A				
IDR(2) - ST1	253.40	11	180.98	68.63	4030	4.0610E-03
IDR(2) - $\Delta \boldsymbol{X}_s$ - ST1	347.13	10	281.16	62.32	6284	4.0615E-03
IDR(2) - ST2	-	49	-	-	-	-
IDR(2) - $\Delta \boldsymbol{X}_s$ - ST2	-	49	-	-	-	-
$\boldsymbol{\eta_0 = 0.5}$						
IDR(2) - ST1	227.06	11	154.77	68.55	3445	4.0610E-03
IDR(2) - $\Delta \boldsymbol{X}_s$ - ST1	331.96	11	259.66	68.54	5794	4.0610E-03
IDR(2) - ST2	-	49	-	-	-	-
IDR(2) - $\Delta \boldsymbol{X}_s$ - ST2	-	49	-	-	-	-
$\boldsymbol{\eta_0 = 0.1}$		Maasvlakte B				
IDR(2) - ST1	303.34	7	256.43	43.63	5693	1.0261E-03
IDR(2) - $\Delta \boldsymbol{X}_s$ - ST1	401.41	7	354.57	43.56	7914	1.0261E-03
IDR(2) - ST2	-	49	-	-	-	-
IDR(2) - $\Delta \boldsymbol{X}_s$ - ST2	-	49	-	-	-	-
$\boldsymbol{\eta_0 = 0.5}$						
IDR(2) - ST1	304.63	8	251.47	49.76	5590	1.0261E-03
IDR(2) - $\Delta \boldsymbol{X}_s$ - ST1	422.13	8	368.93	49.81	8260	1.0260E-03
IDR(2) - ST2	-	49	-	-	-	-
IDR(2) - $\Delta \boldsymbol{X}_s$ - ST2	-	49	-	-	-	-
$\boldsymbol{\eta_0 = 0.1}$		Malta				
IDR(2) - ST1	411.94	9	353.69	54.73	8004	5.3379E-04
IDR(2) - $\Delta \boldsymbol{X}_s$ - ST1	443.21	9	385.20	54.54	8728	5.3381E-04
IDR(2) - ST2	-	49	-	-	-	-
IDR(2) - $\Delta \boldsymbol{X}_s$ - ST2	-	49	-	-	-	-
$\boldsymbol{\eta_0 = 0.5}$						
IDR(2) - ST1	355.61	8	304.00	48.22	6879	5.3389E-04
IDR(2) - $\Delta \boldsymbol{X}_s$ - ST1	424.70	8	372.62	48.67	8454	5.3396E-04
IDR(2) - ST2	-	49	-	-	-	-
IDR(2) - $\Delta \boldsymbol{X}_s$ - ST2	-	49	-	-	-	-
Inexact Picard - SL prec. version 1 - choice 3 - Updated wave number $\hat{p}$						
		Scheveningen				
Bi-CGSTAB	115.50	20	67.83	45.62	4286	1.0378E-06
IDR(2) - ST1	118.05	22	65.90	50.36	4244	2.2106E-05
IDR(2) - $\Delta \boldsymbol{X}_s$ - ST1	141.09	26	80.22	58.89	5080	2.2106E-05
IDR(2) - ST2	-	49	-	-	-	-
IDR(2) - $\Delta \boldsymbol{X}_s$ - ST2	-	49	-	-	-	-
		Maasvlakte A				
Bi-CGSTAB	250.92	11	176.00	70.18	3988	2.8904E-07
IDR(2) - ST1	260.07	11	187.21	69.07	4174	1.0713E-07
IDR(2) - $\Delta \boldsymbol{X}_s$ - ST1	372.66	11	300.33	68.51	6588	1.0713E-07
IDR(2) - ST2	-	49	-	-	-	-
IDR(2) - $\Delta \boldsymbol{X}_s$ - ST2	-	49	-	-	-	-
		Maasvlakte B				
Bi-CGSTAB	303.29	10	236.36	62.38	5336	2.1782E-07
IDR(2) - ST1	243.59	9	183.61	56.48	4085	4.2383E-07

	Total time	# outer	Solve $\boldsymbol{S\zeta} = \boldsymbol{b}$	Build $\boldsymbol{S}$	# matvecs	Rel. error
IDR(2) - $\Delta \boldsymbol{X}_s$ - ST1	406.90	9	347.43	55.95	7612	4.2383E-07
IDR(2) - ST2	-	49	-	-	-	-
IDR(2) - $\Delta \boldsymbol{X}_s$ - ST2	-	49	-	-	-	-
		Malta				
Bi-CGSTAB	419.93	11	348.02	67.25	7962	9.7642E-07
IDR(2) - ST1	365.54	11	292.57	69.21	6632	1.7695E-07
IDR(2) - $\Delta \boldsymbol{X}_s$ - ST1	469.84	11	399.28	66.74	8889	1.7695E-07
IDR(2) - ST2	-	49	-	-	-	-
IDR(2) - $\Delta \boldsymbol{X}_s$ - ST2	-	49	-	-	-	-
Inexact Picard - SL prec. version 1 - choice 4 - Updated wave number $\hat{p}$						
		Scheveningen				
Bi-CGSTAB	487.01	25	427.61	57.15	27134	1.5833E-07
IDR(2) - ST1	463.01	26	401.49	59.36	22117	2.1212E-05
IDR(2) - $\Delta \boldsymbol{X}_s$ - ST1	146.03	25	87.22	56.70	4921	2.1212E-05
IDR(2) - ST2	71.22	19	25.88	43.35	1651	8.3543E-06
IDR(2) - $\Delta \boldsymbol{X}_s$ - ST2	110.41	21	60.47	47.86	3857	2.2267E-05
		Maasvlakte A				
Bi-CGSTAB	324.20	11	249.98	69.50	5654	2.5875E-08
IDR(2) - ST1	355.47	11	282.46	68.90	5418	2.4016E-08
IDR(2) - $\Delta \boldsymbol{X}_s$ - ST1	397.36	11	324.47	68.73	6118	2.4016E-08
IDR(2) - ST2	266.15	20	114.78	145.65	2530	9.9811E-06
IDR(2) - $\Delta \boldsymbol{X}_s$ - ST2	333.52	20	203.05	124.69	4503	7.6041E-06
		Maasvlakte B				
Bi-CGSTAB	301.15	8	246.62	50.21	5582	2.1232E-07
IDR(2) - ST1	321.53	8	267.27	50.56	5052	2.8671E-07
IDR(2) - $\Delta \boldsymbol{X}_s$ - ST1	499.94	9	439.84	56.25	8316	2.8671E-07
IDR(2) - ST2	271.96	20	140.96	125.39	3115	2.8141E-06
IDR(2) - $\Delta \boldsymbol{X}_s$ - ST2	533.23	20	402.25	125.33	8946	1.0378E-05
		Malta				
Bi-CGSTAB	575.67	10	508.07	63.04	11636	1.4418E-06
IDR(2) - ST1	530.89	10	465.58	61.32	9216	1.5142E-07
IDR(2) - $\Delta \boldsymbol{X}_s$ - ST1	446.46	10	381.60	60.82	7568	1.5142E-07
IDR(2) - ST2	269.54	20	141.52	122.34	3169	8.6399E-06
IDR(2) - $\Delta \boldsymbol{X}_s$ - ST2	366.76	20	238.87	122.12	5358	8.7180E-06
Inexact Picard - SL prec. version 1 - choice 5 - Updated wave number $\hat{p}$						
***TOL=1E-2* & $\eta_0 = 0.1$**		Scheveningen				
IDR(2) - ST1	431.66	25	369.12	60.65	23840	1.1382E-06
IDR(2) - ST2	144.77	26	83.41	59.40	5371	1.7119E-05
***TOL=1E-2* & $\eta_0 = 0.5$**						
IDR(2) - ST1	450.80	26	389.52	59.30	24426	-
IDR(2) - ST2	57.76	11	31.32	25.08	2008	1.2744E-05
***TOL=5E-2* & $\eta_0 = 0.1$**						
IDR(2) - ST1	380.70	25	319.93	58.83	20336	1.1683E-06
IDR(2) - ST2	90.90	19	39.39	49.51	2522	1.2744E-05
***TOL=5E-2* & $\eta_0 = 0.5$**						
IDR(2) - ST2	117.49	23	63.21	52.45	4076	2.0277E-05
***TOL=1E-2* & $\eta_0 = 0.1$**		Maasvlakte				
IDR(2) - ST1	650.08	10	583.78	62.64	13023	1.6892E-07
IDR(2) - ST2	188.98	11	116.24	68.93	2592	3.2225E-06
***TOL=1E-2* & $\eta_0 = 0.5$**						

	Total time	# outer	Solve $\boldsymbol{S\zeta} = \boldsymbol{b}$	Build $\boldsymbol{S}$	# matvecs	Rel. error
IDR(2) - ST2	192.66	11	120.06	68.83	2671	1.7453E-06
***TOL=5E-2* & $\eta_0 = 0.1$**						
IDR(2) - ST1	592.21	10	517.03	71.50	11531	1.6858E-07
IDR(2) - ST2	232.31	14	139.67	87.59	3095	4.5959E-06
***TOL=5E-2* & $\eta_0 = 0.5$**						
IDR(2) - ST2	249.90	13	164.43	81.47	3668	7.4704E-06
***TOL=1E-2* & $\eta_0 = 0.1$**		Maasvlakte B				
IDR(2) - ST1	464.17	7	417.04	43.85	9321	5.1961E-08
IDR(2) - ST2	183.21	10	117.43	62.17	2615	4.1483E-06
***TOL=1E-2* & $\eta_0 = 0.5$**						
IDR(2) - ST2	192.02	9	124.11	64.39	2762	5.6835E-06
***TOL=5E-2* & $\eta_0 = 0.1$**						
IDR(2) - ST1	395.16	7	347.96	43.85	7772	5.8035E-07
IDR(2) - ST2	246.00	14	153.10	87.93	3397	3.2028E-06
***TOL=5E-2* & $\eta_0 = 0.5$**						
IDR(2) - ST2	319.33	13	233.65	81.74	5213	6.1330E-06
***TOL=1E-2* & $\eta_0 = 0.1$**		Malta				
IDR(2) - ST1	696.04	8	643.68	48.96	14598	1.0394E-06
IDR(2) - ST2	196.18	9	137.58	55.08	3113	9.6129E-06
***TOL=1E-2* & $\eta_0 = 0.5$**						
IDR(2) - ST2	256.58	9	198.06	55.02	4477	1.0237E-05
***TOL=5E-2* & $\eta_0 = 0.1$**						
IDR(2) - ST1	488.61	7	442.52	42.81	10023	9.7220E-07
IDR(2) - ST2	357.87	16	247.26	105.35	5562	7.0021E-06
***TOL=5E-2* & $\eta_0 = 0.5$**						
IDR(2) - ST2	357.43	14	267.85	85.49	6058	5.0428E-06

	Total time	# outer	Solve $\boldsymbol{S\zeta} = \boldsymbol{b}$	Build $\boldsymbol{S}$	Init. time	Rel. error
Direct method `MUMPS`						
Wave number k_0						
Scheveningen	72.59	22	22.47	48.30	0.62	-
Maasvlakte A	108.81	10	43.27	60.84	1.78	-
Maasvlakte B	66.73	6	25.97	36.20	1.78	-
Malta	88.83	8	37.14	47.05	1.80	-
Modified wave number $\hat{p}$						
Scheveningen	69.48	21	21.36	46.31	0.62	-
Maasvlakte A	108.46	10	43.25	60.53	1.78	-
Maasvlakte B	66.84	6	25.95	36.31	1.78	-
Malta	90.38	8	37.11	48.67	1.80	-

Geometry

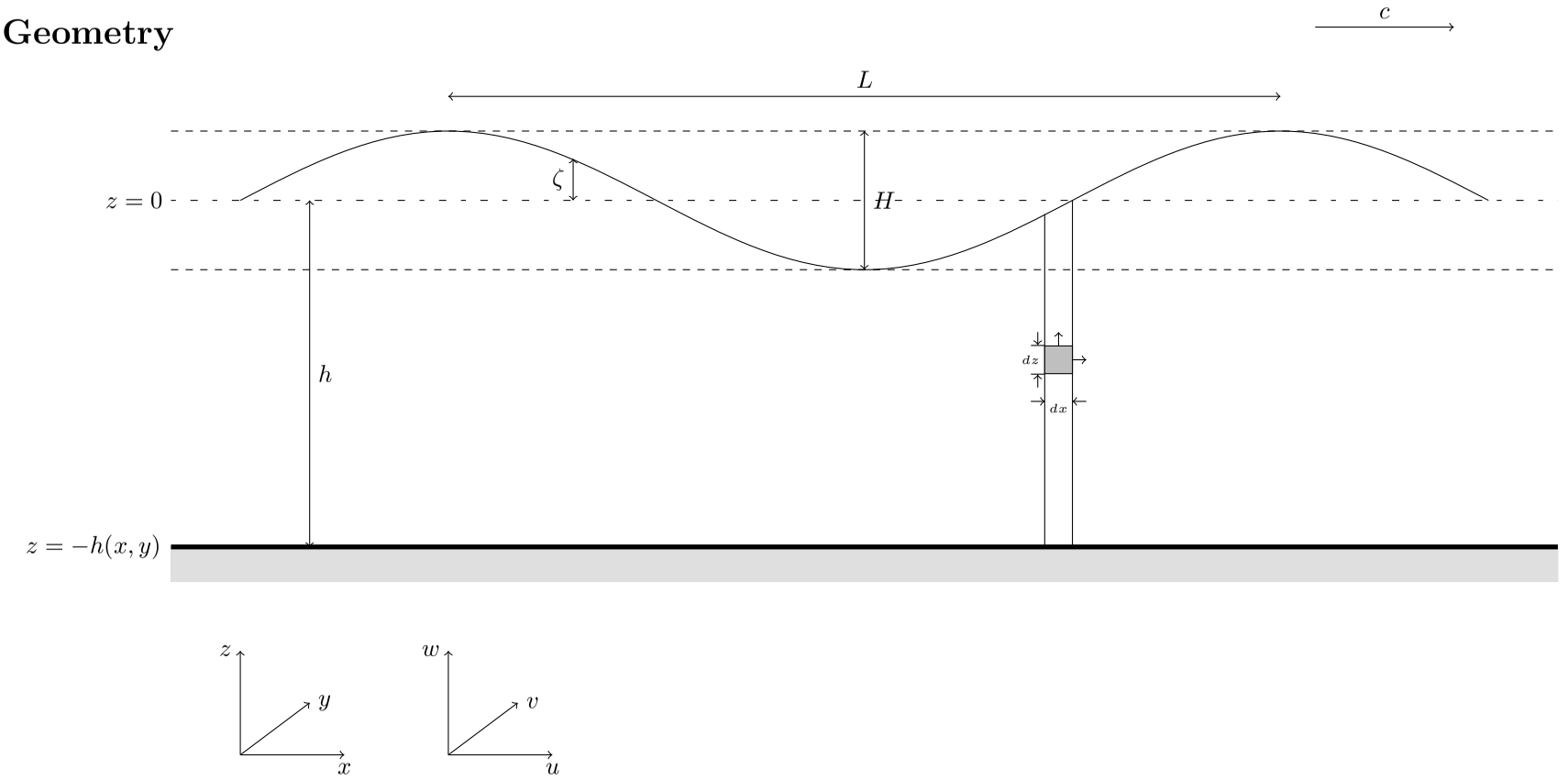


Figure C.1: *The geometry in a vertical section with at $z = -h(x, y)$ the ocean bottom and at $z = 0$ the free surface.*

$\zeta(x, y, t)$	Elevation of the free surface;
$h(x, y)$	Water height from the ocean bottom to the free surface;
$H(x, y)$	Wave height, equals twice the amplitude of the wave;
$L(x, y)$	Wave length, the distance between to successive wave crests;
$c(x, y)$	Wave celerity, depends on the wave frequency and the local wave number.

Bibliography

P. R. Amestoy, I. S. Duff, J. Koster, and J.-Y. L'Excellent. A Fully Asynchronous Multifrontal Solver Using Distributed Dynamic Scheduling. *SIAM Journal on Matrix Analysis and Applications*, 23(1):15–41, 2001.

P. R. Amestoy, A. Guermouche, J.-Y. L'Excellent, and S. Pralet. Hybrid scheduling for the parallel solution of linear systems. *Parallel Computing*, 32(2):136–156, 2006.

J.A. Battjes and J.P.F.M. Janssen. Energy Loss and Set-Up Due to Breaking of Random Waves. *Proc. 16th Int. Conf. on Coastal Engineering*, 1978.

J.C.W. Berkhoff. *Mathematical Models for Simple Harmonic Linear Water Wave Models; Wave refraction and Diffraction*. PhD thesis, Technical University of Delft, 1976.

P.N. Brown and Y. Saad. Hybrid Krylov Methods for Nonlinear Systems of Equations. *SIAM J. on Scientific Computing*, 11:450–481, 1990.

R.S. Dembo, S.C. Eisenstat, and T. Steihaug. Inexact Newton Methods. *SIAM Journal on Numerical Analysis*, 19(2), 1982.

M.W. Dingemans. *Water Wave Propagation over Uneven Bottoms, Part 1 - Linear Wave Propagation*. World Scientific, first edition, 1997.

B.J.O. Eikema and B.C. van Prooijen. HARES - Numerical model for the determination of wave penetration in harbour basins. *Svasek-rapport t.b.v. validatie HARES*, 2005.

S.C. Eisenstat and H.F. Walker. Choosing the Forcing Terms in an Inexact Newton Method. *SIAM Journal for Scientific Computing*, 17(1):16–32, 1996.

Y.A. Erlangga, C. Vuik, and C.W. Oosterlee. On a class of preconditioners for solving the Helmholtz equation. *Applied Numerical Mathematics*, 50:409–425, 2004.

Y.A. Erlangga, C.W. Oosterlee, and C. Vuik. A novel multigrid based preconditioner for heterogeneous Helmholtz problems. *SIAM J. Sci. Comput.*, 27:1471–1492, 2006.

V. Faber, J. Liesen, and P. Tichý. On Chebyshev Polynomials of Matrices. *SIAM Journal on Matrix Analysis and Applications*, 31:2205–2221, 2010.

R. Fletcher. Conjugate gradient methods for indefinite systems. *Lecture Notes in Math*, 506:73–89, 1976.

S. Gerschgorin. Über die Abgrenzung der Eigenwerte einer Matrix. *Izv. Akad. Nauk. USSR Otd. Fiz.-Mat. Nauk*, 7:749–754, 1931.

I. Holland and K. Bell. *Finite Element Methods in Stress Analysis*. Tapir, Trondheim, Norway, 1969.

D.A. Knoll and D.E. Keyes. Jacobian-free Newton-Krylov methods: A Survey of Approaches and Applications. *Journal of Computational Physics*, 193:357–397, 2004.

D. Loghin, M.B. van Gijzen, and E. Jonkers. Bounds on the eigenvalue range and on the Field of Values of non-hermitian and indefinite finite element matrices. *Journal of Computational and Applied Mathematics*, 189:304–323, 2006.

C.C. Mei. *The Applied Dynamics of Ocean Surface Waves.* World Scientific, 1989.

J.A. Meijerink and H.A. van der Vorst. An Iterative Solution Method for Linear Systems of Which the Coefficient Matrix is a Symmetric m-Matrix. *Math. of Comput.*, 31:148–162, 1977.

Y. Saad. *Iterative Methods for Sparse Linear Systems.* SIAM, second edition, 2003.

G.L.G. Sleijpen and H.A. van der Vorst. Maintaining convergence properties of Bi-CGstab methods in finite precision arithmetic. *Numerical Algorithms*, 10:203–223, 1995.

V.I. Smirnov and N.A. Lebedev. *Functions of a Complex Variable: Constructive theory.* Scripta Technica Ltd - ILIFFE BOOKS Ltd London (in Russian), 1968.

P. Sonneveld. CGS, A Fast Lanczos-Type Solver for Nonsymmetric systems. *SIAM J. Sci. Statist. Comput.*, 20:36–52, 1989.

P. Sonneveld and M.B. van Gijzen. IDR(s): A family of simple and fast algorithms for solving large nonsymmetric systems of linear equations. *SIAM J. Sci. Comput.*, 31(2):1035–1062, 2008.

H.N. Southgate. Review of Wave Breaking in Shallow Water. *Society for Underwater Technology, Wave Kinematics and Environmental Forces*, Volume 29, 1993.

G.E.M. van de Sande. The mild-slope equation and its numerical implementation. Master's thesis, Delft University of Technology, 2011.

H.A. van der Vorst. Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. *SIAM J. Sci. Statist. Comput.*, Volume 13:631 – 644, 1992.

H.A. van der Vorst. *Iterative Krylov Methods for Large Linear Systems.* Cambridge University Press, first edition, 2003.

M.B. van Gijzen and Y.A. Erlangga. Convergence bounds for preconditioned GMRES using element-by-element estimates of the field of values. In *Proceedings of the ECCOMAS CFD 2006 Conference*, 2006.

M.B. van Gijzen and P. Sonneveld. Algorithm 913: An Elegant IDR(s) Variant that Efficiently Exploits Bi-orthogonality Properties. *ACM Transactions of Mathematical Software*, 38(1):5:1–5:19, 2011.

M.B. van Gijzen, Y.A. Erlangga, and C. Vuik. Spectral analysis of the discrete Helmholtz operator preconditioned with a shifted Laplacian. *SIAM J. Sci. Comput.*, 29(5), 2007.

J. van Kan, A. Segal, and F. Vermolen. *Numerical methods in Scientific Computing.* VSSD, improved edition, 2008.

S. van Veldhuizen. *Efficient numerical methods for the instationary solution of laminar reacting gas flow problems.* PhD thesis, Technical University Delft, 2009.

P.J. Visser. A Mathematical Model of Uniform Longshore Currents and the Comparison with Laboratory Data. *Communications on hydraulics 84-2*, 1984.

O.C. Zienkewicz. *Finite element method in Engineering Science.* MC Graw-Hill, 1971.