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

Improving the linear solver used in the interactive wave model of a real-time ship simulator

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

ELWIN VAN 'T WOUT

Delft, the Netherlands
August 2009

MSc THESIS APPLIED MATHEMATICS

"Improving the linear solver used in the interactive wave model of a real-time ship simulator"

ELWIN VAN 'T WOUT

Delft University of Technology

Daily supervisor

Dr. ir. M. B. van Gijzen

Responsible professor

Prof. dr. ir. C. Vuik

Other thesis committee members

Dr. ir. H. X. Lin

Dr. ir. A. Ditzel

Dr. ir. A. van der Ploeg

August 2009

Delft, the Netherlands

Preface

This master's thesis has been written for the degree of Master of Science in Applied Mathematics at the faculty of Electrical Engineering, Mathematics and Computer Sciences of Delft University of Technology. The graduation is done at the department of Numerical Analysis. The actual research of the graduation project has been carried out at Maritime Research Institute Netherlands (MARIN) in Wageningen. MARIN is a company which serves the maritime sector with innovative products. To maintain their position, advanced hydrodynamic and nautical research is carried out. One of the research projects at MARIN is the development of a new wave model to be used in the real-time ship simulator. To improve the computational performance of the wave model, this graduation research has been performed.

I would like to thank MARIN for giving me the opportunity to perform my graduation project there. It was a pleasant time working at the department Maritime Software Group. Especially, I would like to thank my daily supervisors for their support during the project. Auke Ditzel helped me with understanding the wave model and gave valuable comments on the report. Auke van der Ploeg supported me during the research with his knowledge about linear solvers. My questions about implementational aspects were well answered by Anneke Sicherer. Gert Klopman introduced me to the wave model which he has developed.
I would also like to thank my supervisors at the TU Delft, Kees Vuik and Martin van Gijzen for their contribution to the project. In particular Martin van Gijzen for his excellent supervision of the graduation research. During my weekly visits to the TU Delft, we have had interesting discussions about this project and mathematical research in general.

Elwin van 't Wout,

Wageningen, August 2009.

Contents

1 Introduction

Background The actual research for this master's thesis has been carried out at the Maritime Research Institute Netherlands. MARIN serves the maritime industry with innovative products. One of the products MARIN supplies is a bridge simulator, which simulates the movements of a ship in a wave field. These real-time navigation simulators are used for research, consultancy and training purposes. The simulation technology has been developed in-house and is an ongoing process of research and development.
The current wave model is based on the Fourier theory, applied to a predefined wave spectrum and does only partially interact with objects. A new wave model is under development, which depends on the bathymetry, resulting in more realistic wave patterns. The influence between the motions of a ship and the waves is also modelled. To fullfill the requirement of real-time calculations, the computational methods need to have a large efficiency.
The linear solver in the wave model takes a considerable amount of computation time. Improving the performance of the linear solver will be the main topic of research in this graduation project. To investigate the properties of the linear system first a literature study about the underlying wave model has been performed. This results in a full description of the wave model and its numerical descretization. Then, the main properties of the implemented linear solvers have been derived. By implementing some improvements of these solvers and implementing the deflation method, a comparison of performance between the different methods will be presented.

Outline This thesis starts with the derivation of the recently developed wave model, called the variational Boussinesq model. In Chapter 2 the basic principles of this model are described. An important part in the derivation of the model is the insertion of a parameter for the vertical structure of the fluid flow, described in Chapter 3. To reduce the computational effort, in Chapter 4 a linearization of the model has been performed. The resulting set of equations is listed in Chapter 5, followed by the horizontal boundaries of the model in Chapter 6. To solve the differential equations for the wave field, the finite volume method has been used to approximate the solution. This spatial discretization as well as the numerical time integration has been described in Chapter 7. One of the model equations results in a linear system of equations. To solve this large system efficiently, the conjugate gradient method has been described in Chapter 8. Three different preconditioners are applied to this method and their properties have been listed in the next sections: diagonal scaling in Chapter 9, relaxed incomplete Cholesky in Chapter 10 and repeated red-black in Chapter 11. As explained in Chapter 12, the deflation method can be combined with the preconditioners, resulting in a better convergence. The linear solvers will be applied to the test problems presented in Chapter 13. In Chapter 14 the main results of the different methods at several test problems have been given. Conclusions of the graduation project will be given in Chapter 15. Some suggestions for future research will be explained in chapter 16. Several appendices for more background complete the thesis.

2 The variational Boussinesq model

The recently developed wave model for use in the real-time ship simulator is based on a variational Boussinesq model, as described in [21, 43, 44]. In this chapter the main theory behind this model will be presented. First an expression of the pressure will be derived from the Euler equations. The basic idea of the model is to minimize the pressure in the whole fluid. The velocity and water level which the pressure is minimal for, satisfy a system of partial differential equations. These equations will be used in the next chapters.
Starting point of the derivation of the model equations are the *Euler equations*

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\,\mathbf{u} \;=\; -\frac{1}{\rho}\nabla p - \mathbf{g}, \tag{2.1}$$

which are valid for motions of ideal fluids and are a special case of the Navier-Stokes equations, for inviscid incompressible flow [22]. The fluid velocity is given by $\mathbf{u}$, ρ denotes the mass density, p the pressure and $\mathbf{g}$ the gravitation. In Section 2.1 we will show that for irrotational flows these equations can be rewritten as

$$\frac{\partial \phi}{\partial t} + \frac{1}{2}(\nabla\phi)^2 + \frac{p}{\rho} + gz \;=\; 0, \tag{2.2}$$

the instationary *Bernoulli equation*, with ϕ the velocity potential ($\mathbf{u} = \nabla\phi$) and $(\nabla\phi)^2 = (\frac{\partial\phi}{\partial x})^2 + (\frac{\partial\phi}{\partial y})^2 + (\frac{\partial\phi}{\partial z})^2$. By integrating the pressure p over the whole domain we get the total pressure

$$\mathcal{P}(\phi,\zeta) \;:=\; \iint \int_{-h}^{\zeta} \left(\frac{\partial \phi}{\partial t} + \frac{1}{2}(\nabla\phi)^2 + gz \right) dz\; dx\, dy. \tag{2.3}$$

The vertical domain boundaries are the bottom ($z = -h$) and the water level ($z = \zeta$). The reference level $z = 0$ is around the mean water level. The horizontal boundaries are not prescribed yet, this will be done in Chapter 6.
The basic idea of the variational Boussinesq model is to minimize the total pressure $\mathcal{P}$. Hence to find functions ϕ and ζ satisfying

$$\min_{\phi,\zeta} \int \mathcal{P}\; dt, \tag{2.4}$$

which is called *Luke's variational formulation* [24].
For functions, Fermat's principle states that a zero derivative is a necessary (but not sufficient) condition for a minimum. Similarly, a zero variation is a necessary condition for a minimal functional [44]. Therefore, we have to search for functions ϕ and ζ such that

$$\delta_\phi \int \mathcal{P}\; dt \;=\; 0 \quad \text{and} \quad \delta_\zeta \int \mathcal{P}\; dt \;=\; 0, \tag{2.5}$$

with δ_ϕ and δ_ζ denoting the first variation with respect to ϕ and ζ. These variations will be calculated in Section 2.2.
Because the vertical structure of the flow is often known, the velocity potential can be written as a series expansion in predefined vertical shape functions f_m:

$$\phi(x,y,z,t) \;=\; \varphi(x,y,t) + \sum_{m=1}^{M} f_m(z,\zeta)\,\psi_m(x,y,t). \tag{2.6}$$

This expansion reduces the 3D-model to a 2D-model and is an important step in deriving the variational Boussinesq model. In Chapter 3 we will elaborate more on this expansion.
Now, the variable ϕ is written in terms of φ and ψ_m. Together with ζ these are the basic variables of the variational Boussinesq model. In Section 2.3 the minimization problem (2.4) will be simplified by introducing the Hamiltonian $\mathcal{H}$ to

$$\min_{\varphi,\zeta} \int \left(\iint \varphi \frac{\partial \zeta}{\partial t} \, dx \, dy - \mathcal{H}(\varphi, \zeta) \right) dt. \tag{2.7}$$

With this expression, the basic equations of the variational Boussinesq model can be derived. They read

$$\frac{\partial \zeta}{\partial t} - \delta_\varphi H = 0, \tag{2.8a}$$

$$\frac{\partial \varphi}{\partial t} + \delta_\zeta H = 0, \tag{2.8b}$$

$$\delta_{\psi_m} H = 0. \tag{2.8c}$$

This set of equations is called the *Hamiltonian description* [21]. The first equation is the continuity equation, the second is similar to Bernoulli's equation and the third leads to an elliptic partial differential equation in ψ_m.

2.1 Pressure functional

In this section an expression for the pressure term will be derived for irrotational flows (see [13, 34]). This is done by rewritting the Euler equations (2.1), with gravitation the only external force being considered. The Euler equations are a system of three coupled partial differential equations for the velocity. By introducing a potential, which is a scalar function, the system can be reduced to one partial differential equation. This equation is used to derive the total pressure $\mathcal{P}$ as given in Equation (2.3).
The *velocity potential* $\phi(x, y, z, t)$ is implicitly defined by

$$\nabla \phi := \mathbf{u}. \tag{2.9}$$

In order to define this correctly, one has to assume irrotational flow (see [13]):

$$\nabla \times \mathbf{u} = \mathbf{0}. \tag{2.10}$$

Substituting the velocity potential (2.9), the Euler equations (2.1) can be written in a conservation form [47]. The first term of the Euler equations can be rewritten as $\frac{\partial \nabla \phi}{\partial t} = \nabla \frac{\partial \phi}{\partial t}$, because of the smoothness of ϕ. Using a product rule[1] for vectors, we find $(\mathbf{u}\cdot\nabla)\mathbf{u} = \nabla(\frac{1}{2}(\nabla\phi)^2)$. The mass density ρ is assumed to be constant over the whole domain, implying *incompressible flow* [47]. Hence $\frac{1}{\rho}\nabla p = \nabla(\frac{p}{\rho})$. The gravitation can be written as $\mathbf{g} = \nabla(gz)$. With these identities, the Euler equations become

$$\nabla \left(\frac{\partial \phi}{\partial t} + \frac{1}{2}(\nabla\phi)^2 + \frac{p}{\rho} + gz \right) = \mathbf{0}. \tag{2.11}$$

[1] A vector dot product is given by [1] $\nabla(A \cdot B) = (A \cdot \nabla)B + (B \cdot \nabla)A + A \times (\nabla \times B) + B \times (\nabla \times A)$. For $A = \mathbf{u}$ and $B = \mathbf{u}$ this is equivalent with $\frac{1}{2}\nabla(\mathbf{u} \cdot \mathbf{u}) = (\mathbf{u} \cdot \nabla)\mathbf{u} + \mathbf{u} \times (\nabla \times \mathbf{u})$. Using irrotational flow (2.10) and the velocity potential (2.9) we now have $(\mathbf{u} \cdot \nabla)\mathbf{u} = \frac{1}{2}\nabla((\nabla\phi)^2)$.

Because all three spatial derivatives are zero, the function depends only on time:

$$\frac{\partial \phi}{\partial t} + \frac{1}{2}(\nabla \phi)^2 + \frac{p}{\rho} + gz \quad = \quad f(t). \tag{2.12}$$

For the choice[2] $f(t) = 0$ this is the Bernoulli equation (2.2). With $P := -\frac{p}{\rho}$ we have

$$P(\phi, z) \quad = \quad \frac{\partial \phi}{\partial t} + \frac{1}{2}(\nabla \phi)^2 + gz. \tag{2.13}$$

This is equal to the integrand in Equation (2.3), which gives an expression for the total pressure $\mathcal{P}$ [34].

2.2 Differential equations resulting from the variation principle

The basic problem of the variational Boussinesq model is given by the variational formulation (2.4). A necessary condition for the minimal value is that the first variation to ϕ and ζ are zero. In this section the variations of the total pressure will be calculated. Equating these first variations to zero will give a partial differential equation with boundary conditions at the surface and bottom. The solution of this boundary value problem minimizes the total pressure.
The derivation starts with calculating the variations. The first variation with respect to ϕ has to be zero, i.e.,

$$\begin{aligned} 0 \quad &= \quad \delta_\phi \int \iint \int_{-h}^{\zeta} P(\phi)\ dz\ dx\, dy\ dt \\ &= \quad \int \iint \int_{-h}^{\zeta} \left(\frac{\partial (\delta_\phi(\phi))}{\partial t} + \delta_\phi \Big(\frac{1}{2}(\nabla \phi)^2 \Big) \right) dz\ dx\, dy\ dt. \end{aligned} \tag{2.14}$$

For this derivation use has been made of some elementary rules in variational calculus (see Appendix C). In order to derive the correct form of this equation, observe the following two equations.
With Leibniz's rule[3], we have

$$\int_{-h}^{\zeta} \frac{\partial (\delta_\phi(\phi))}{\partial t}\ dz \quad = \quad \frac{\partial}{\partial t} \int_{-h}^{\zeta} \delta\phi\ dz - \delta\phi|_{z=\zeta} \frac{\partial \zeta}{\partial t} + \delta\phi|_{z=-h} \frac{\partial (-h)}{\partial t}. \tag{2.15}$$

Applying Green's theorem[4] ($c = \delta\phi$, $\mathbf{u} = \nabla\phi$) gives

$$\begin{aligned} \iint \int_{-h}^{\zeta} \delta_\phi \Big(\frac{1}{2}(\nabla \phi)^2 \Big)\ dz\ dx\, dy \quad &= \quad \iiint_{\Omega} \nabla\phi \cdot \nabla(\delta\phi)\ dz\, dx\, dy \\ &= \quad \iint_{\Gamma} \nabla\phi \cdot \mathbf{n}\ \delta\phi\ dx\, dy - \iiint_{\Omega} (\nabla^2 \phi)\ \delta\phi\ dz\, dx\, dy \end{aligned} \tag{2.16}$$

with $\mathbf{n}$ denoting the unit outward normal. The domain is given by $\Omega = \mathbb{R}^2 \times [-h, \zeta]$, with the boundary $\Gamma = \{\mathbb{R}^3 | z = -h, z = \zeta\}$. In the horizontal plane boundaries have not been specified

[2] This choice is allowed because if one defines a new velocity potential $\phi' := \phi - \int f dt$, then $\frac{\partial \phi'}{\partial t} + \frac{1}{2}(\nabla \phi')^2 + \frac{p}{\rho} + gz = 0$ and $\nabla \phi' = \nabla \phi$, so the same velocity profile [5].
[3] $\frac{d}{dx} \int_a^b f ds = \int_a^b \frac{df}{dx} ds + f(b) \frac{db}{dx} - f(a) \frac{da}{dx}$
[4] $\int_\Omega c\ \nabla \cdot \mathbf{u}\ d\Omega = -\int_\Omega \nabla c \cdot \mathbf{u}\ d\Omega + \oint_\Gamma c\ \mathbf{u} \cdot \mathbf{n}\ d\Gamma$

yet. For the computational model, these boundaries will be specified later in Chapter 6. By substituting Equations (2.15) and (2.16) into Equation (2.14) we get

$$\begin{aligned}
&\int \iint \left(\frac{\partial}{\partial t} \int_{-h}^{\zeta} \delta\phi \; dz - \delta\phi|_{z=\zeta} \frac{\partial \zeta}{\partial t} - \delta\phi|_{z=-h} \frac{\partial h}{\partial t} \right) \; dx\, dy \; dt \\
+ &\int \left(\iint \left(- \int_{-h}^{\zeta} (\nabla^2 \phi)\, \delta\phi \; dz \right) \; dx\, dy + \iint_{\Gamma} (\nabla\phi \cdot \mathbf{n})\, \delta\phi \; dx\, dy \right) dt \quad = \\
&\int \frac{\partial}{\partial t} \left(\iiint_{\Omega} \delta\phi \; dx\, dy\, dz \right) dt - \int \iiint_{\Omega} (\nabla^2 \phi)\, \delta\phi \; dx\, dy\, dz \; dt \\
&- \int \iint_{z=\zeta} \frac{\partial \zeta}{\partial t} \delta\phi \; dx\, dy \; dt + \int \iint_{z=\zeta} (\nabla\phi \cdot \mathbf{n})\, \delta\phi \; dx\, dy \; dt \\
&- \int \iint_{z=-h} \frac{\partial h}{\partial t} \delta\phi \; dx\, dy \; dt + \int \iint_{z=-h} (\nabla\phi \cdot \mathbf{n})\, \delta\phi \; dx\, dy \; dt \quad = \quad 0. \qquad (2.17)
\end{aligned}$$

The integral containing the time derivative vanishes because the variations w.r.t. ϕ and ζ are zero at the start and end point in time [24]. For arbitrary variations $\delta\phi$, Equation (2.17) is satisfied when

$$\begin{aligned}
\Delta\phi &= 0, \qquad \text{for } -h < z < \zeta; && (2.18a)\\
\frac{\partial \zeta}{\partial t} &= \nabla\phi \cdot \mathbf{n}_s, \qquad \text{for } z = \zeta; && (2.18b)\\
\frac{\partial h}{\partial t} &= \nabla\phi \cdot \mathbf{n}_b, \qquad \text{for } z = -h; && (2.18c)
\end{aligned}$$

with $\mathbf{n}_s$ and $\mathbf{n}_b$ denoting the unit outward normals at the surface and bottom, respectively. They read

$$\begin{aligned}
\mathbf{n}_s &= \frac{1}{\sqrt{(\nabla\zeta)^2 + 1}} \begin{bmatrix} -\frac{\partial \zeta}{\partial x} & -\frac{\partial \zeta}{\partial y} & 1 \end{bmatrix}^T, && (2.19a)\\
\mathbf{n}_b &= \frac{1}{\sqrt{(\nabla h)^2 + 1}} \begin{bmatrix} -\frac{\partial h}{\partial x} & -\frac{\partial h}{\partial y} & -1 \end{bmatrix}^T. && (2.19b)
\end{aligned}$$

Written out, Equations (2.18) read

$$\begin{aligned}
\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} &= 0, && (2.20a)\\
\sqrt{(\nabla\zeta)^2 + 1}\, \frac{\partial \zeta}{\partial t} &= -u \frac{\partial \zeta}{\partial x} - v \frac{\partial \zeta}{\partial y} + w \qquad \text{for } z = \zeta, && (2.20b)\\
\sqrt{(\nabla h)^2 + 1}\, \frac{\partial h}{\partial t} &= -u \frac{\partial h}{\partial x} - v \frac{\partial h}{\partial y} - w \qquad \text{for } z = -h. && (2.20c)
\end{aligned}$$

The first equation is the *continuity equation*, which relates to mass conservation for incompressible flow [29]. The second equation states that no particles can leave the water surface and is called the *kinematic boundary condition*. The third equation resembles the impermeability of the sea bottom.

Equations (2.20) result from requirement (2.14) of zero variation of $\mathcal{P}$ w.r.t ϕ. Similarly, the first variation of $\mathcal{P}$ w.r.t. ζ should be zero:

$$\begin{aligned}
\int\iint\left(\delta_\zeta\int_{-h}^{\zeta}\left(\frac{\partial\phi}{\partial t}+\frac{1}{2}(\nabla\phi)^2+gz\right)dz\right)\,dx\,dy\,dt &= 0,\\
\int\iint\left(\left(\frac{\partial\phi}{\partial t}+\frac{1}{2}(\nabla\phi)^2+gz\right)\Big|_{z=\zeta}\delta\zeta\right)\,dx\,dy\,dt &= 0,\\
\frac{\partial\phi}{\partial t}+\frac{1}{2}(\nabla\phi)^2+gz &= 0 \quad \text{for } z=\zeta. \qquad (2.21)
\end{aligned}$$

This is equal to $P(z{=}\zeta)=0$, so zero pressure at the sea surface. This is called the *dynamic free surface condition*. Note that the pressure at the surface is not really zero, but atmospheric [22]; so the pressure considered here is the pressure compared to the atmospheric pressure[5].

2.3 The Hamiltonian system for the surface potential

The basic quantities in the model as presented yet, are the velocity potential $\phi(x,y,z,t)$ and the surface elevation $\zeta(x,y,t)$. Note that ϕ depends on three spatial dimensions. Three dimensional problems are computationally considerably more difficult than two-dimensional problems. Therefore, the number of spatial dimensions of the model are reduced by writing the velocity potential as a series expansion in vertical shape functions f_m:

$$\phi(x,y,z,t) = \varphi(x,y,t)+\sum_{m=1}^{M} f_m(z)\psi_m(x,y,t), \qquad (2.22)$$

The vertical shape functions have to represent the vertical structure of fluid flows, which often has a characteristic shape. Two different models for the vertical shape functions will be explained in more detail in Chapter 3. In this section we will focus on the variable φ, so assume $M=0$.

The Lagrangian and the Hamiltonian are defined in this section in order to rewrite the pressure term in a more favorable form. Calculating the minimal value of the total pressure will lead to a system of two Hamiltonian equations.

The *surface velocity potential* φ, introduced in the series expansion (2.22), is defined by

$$\varphi(x,y,t) := \phi(x,\ y,\ z{=}\zeta(x,y,t),\ t). \qquad (2.23)$$

Note that the gradient of the surface potential $\nabla\varphi$ does not equal the velocity at the surface $(\nabla\phi)|_{z=\zeta}$, because $\nabla\varphi=\nabla\phi|_{z=\zeta}+\frac{\partial\phi}{\partial z}|_{z=\zeta}\nabla\zeta$.

In order to rewrite the total pressure $\mathcal{P}$ from Equation (2.3), let's consider the kinetic energy $\mathcal{K}(\varphi,\zeta)$ which is implicitly defined by (see [43])

$$\mathcal{K}(\varphi,\zeta) := \min_{\phi}\{K(\phi,\zeta)\mid \phi=\varphi \text{ at } z=\zeta\}, \qquad (2.24)$$

[5]The choice of $P=0$ at the surface can be justified by looking at the velocity potential $\phi' := \phi - p_a t$, with p_a the atmospheric pressure at the surface. Note that $\nabla\phi'=\nabla\phi$, so it does not change the velocity. Substituting ϕ' in (2.21), gives $\frac{\partial\phi}{\partial t}-p_a+\frac{1}{2}(\nabla\phi)^2+gz=0$, so $P(z=\zeta)=p_a$ as desired. Note that the atmospheric pressure p_a is assumed to be constant.

with

$$K(\phi,\zeta) \;:=\; \iint \left(\int_{-h}^{\zeta} \frac{1}{2}(\nabla\phi)^2 \, dz - \phi|_{z=-h} \frac{\partial h}{\partial t} \right) dx\, dy. \tag{2.25}$$

This constrained minimization problem is governed by the zero first variation of K w.r.t. ϕ, which can be reduced to

$$-\iiint_{\Omega} (\nabla^2\phi)\delta\phi \; dx\, dy\, dz + \iint_{\Gamma} (\nabla\phi\cdot\mathbf{n})\delta\phi \; dx\, dy - \iint_{z=-h} \frac{\partial h}{\partial t}\delta\phi \; dx\, dy \;=\; 0. \tag{2.26}$$

The constraint of the minimization problem (2.24) yields the essential boundary condition $\phi = \varphi$ at $z = \zeta$; so $\delta\phi|_{z=\zeta} = 0$. Hence, the equations

$$\begin{aligned}
\Delta\phi &= 0, && \text{for } -h < z < \zeta; && \text{(2.27a)}\\
\phi &= \varphi, && \text{for } z = \zeta; && \text{(2.27b)}\\
\frac{\partial h}{\partial t} &= \nabla\phi\cdot\mathbf{n}_b, && \text{for } z = -h; && \text{(2.27c)}
\end{aligned}$$

satisfy the kinetic minimization problem (2.24). Observe that this system is similar to Equation (2.18).
With the introduction of the kinetic energy (2.25) the total pressure $\mathcal{P}(\phi,\zeta)$ from (2.3) can be written as

$$\begin{aligned}
\mathcal{P}(\phi,\zeta) &= \iint \int_{-h}^{\zeta} \frac{\partial\phi}{\partial t} \, dz \; dx\, dy + \iint \left(\int_{-h}^{\zeta} \frac{1}{2}(\nabla\phi)^2 \, dz + \int_{-h}^{\zeta} gz \, dz \right) dx\, dy\\
&= \iint \left(\frac{\partial}{\partial t} \int_{-h}^{\zeta} \phi \; dz - \phi|_{z=\zeta}\frac{\partial\zeta}{\partial t} - \phi|_{z=-h}\frac{\partial h}{\partial t} \right) dx\, dy\\
&\quad + \iint \left(\int_{-h}^{\zeta} \frac{1}{2}(\nabla\phi)^2 \, dz + \left[\frac{1}{2}gz^2\right]_{z=-h}^{\zeta} \right) dx\, dy\\
&= -\iint \left(\phi|_{z=\zeta}\frac{\partial\zeta}{\partial t} - \int_{-h}^{\zeta} \frac{1}{2}(\nabla\phi)^2 \, dz + \phi|_{z=-h}\frac{\partial h}{\partial t} - \frac{1}{2}g(\zeta^2 - h^2) \right) dx\, dy + \frac{\partial}{\partial t} \iint \int_{-h}^{\zeta} \phi \; dz \; dx\, dy\\
&= -\left(\iint \phi|_{z=\zeta}\frac{\partial\zeta}{\partial t} \; dx\, dy - K(\phi,\zeta) - \iint \frac{1}{2}g(\zeta^2 - h^2) dx\, dy \right) + \frac{\partial}{\partial t} \iint \int_{-h}^{\zeta} \phi \; dz \; dx\, dy. \qquad \text{(2.28)}
\end{aligned}$$

The last integral, containing the time derivative of ϕ, vanishes because the variation w.r.t. ϕ and ζ are zero at the end points of the time interval, as in Equation (2.17).
The model uses the minimal value of $\mathcal{P}$, which can be rewritten as

$$\min_{\phi,\zeta} \int \mathcal{P}(\phi,\zeta) \; dt \;=\; \min_{\varphi,\zeta} \int \left(-\iint \varphi\frac{\partial\zeta}{\partial t} \; dx\, dy - \mathcal{H}(\varphi,\zeta) \right) dt, \tag{2.29}$$

with the Hamiltonian $\mathcal{H}$ defined by

$$\mathcal{H}(\varphi,\zeta) \;:=\; \mathcal{K}(\varphi,\zeta) + \iint \frac{1}{2}g(\zeta^2 - h^2) dx\, dy. \tag{2.30}$$

Note that the component $\frac{1}{2}gh^2$ can be omitted, because the variations w.r.t. φ and ζ are zero.

Now, with the definition of the Hamiltonian (2.30) and the kinetic energy (2.25), we have reduced the variational principle (2.4) for ϕ and ζ to a variational principle in φ and ζ:

$$\min_{\varphi,\zeta} \int \left(\iint \varphi \frac{\partial \zeta}{\partial t} \, dx \, dy - \mathcal{H}(\varphi, \zeta) \right) dt. \tag{2.31}$$

This is known as a *canonical action principle* for the *canonical variables* φ and ζ [44]. The functional is also called the *Lagrangian*

$$\mathcal{L}(\varphi, \zeta) \quad := \quad \int \left(\iint \varphi \frac{\partial \zeta}{\partial t} \, dx \, dy - \mathcal{H}(\varphi, \zeta) \right) dt. \tag{2.32}$$

In order to obtain the minimal value of the total pressure, the first variation of the Lagrangian w.r.t. φ and ζ should vanish. The zero variation w.r.t. φ reads

$$\delta_\varphi \int \left(\iint \varphi \frac{\partial \zeta}{\partial t} \, dx \, dy - \mathcal{H}(\varphi, \zeta) \right) dt \quad = \quad 0,$$

hence

$$\frac{\partial \zeta}{\partial t} \quad = \quad \delta_\varphi H(\varphi, \zeta). \tag{2.33}$$

The zero variation w.r.t ζ reads

$$\begin{aligned} \delta_\zeta \int \left(\iint \varphi \frac{\partial \zeta}{\partial t} \, dx \, dy - \mathcal{H}(\varphi, \zeta) \right) dt \quad &= \quad 0, \\ \iint \left(- \int \frac{\partial \varphi}{\partial t} \delta\zeta \; dt + \varphi \, \delta\zeta|_{t_{end}} \right) dx \, dy \quad &= \quad \int \delta_\zeta \mathcal{H}(\varphi, \zeta) \; dt, \end{aligned}$$

hence

$$\frac{\partial \zeta}{\partial t} \quad = \quad -\delta_\zeta H(\varphi, \zeta). \tag{2.34}$$

Summarizing, the zero variations w.r.t. φ and ζ result in

$$\frac{\partial \zeta}{\partial t} \quad = \quad \delta_\varphi H(\varphi, \zeta), \tag{2.35a}$$

$$\frac{\partial \varphi}{\partial t} \quad = \quad -\delta_\zeta H(\varphi, \zeta), \tag{2.35b}$$

with the Hamiltonian density H given by $\iint H \; dx \, dy = \mathcal{H}$. These equations are the first two of the Hamiltonian description (2.8). The other one will be derived in next chapter.

3 Vertical structure of the flow

In Chapter 2 the Hamiltonian description (2.35) has been derived from the Euler equations (2.1). In Equation (2.22) the velocity potential ϕ has been written in the form of a series in vertical shape functions. These shape functions will be analyzed in more detail in this chapter. First, a general model will be considered in Section 3.1. Then, two different choices of shape functions will be discussed, namely the parabolic and cosine-hyperolic model in Section 3.2 and 3.3, respectively.
With the surface velocity potential (2.23), the series expansion of ϕ reads

$$\begin{aligned} \phi(x,y,z,t) &= \varphi(x,y,t) + \sum_{m=1}^{M} f_m(z)\psi_m(x,y,t), && \text{(3.1a)} \\ f_m &= 0 \qquad \text{for } z = \zeta(x,y,t). && \text{(3.1b)} \end{aligned}$$

The shape functions f_m have to be prescribed and represent the vertical velocity shape. Because of (3.1b), all shape functions f_m depend on ζ and therefore we have $\nabla f_m = \frac{\partial f_m}{\partial \zeta}\nabla\zeta$. Moreover, the shape functions f_m may also depend on h and shape parameters κ_m. The functions ψ_m are not known a priori and therefore become variables for the total pressure. This affects the minimization problem (2.4), consequently, the minimization problem (2.7) changes to

$$\min_{\varphi,\zeta,\psi_1,\psi_2,\ldots,\psi_M} \int \left(\iint \varphi \frac{\partial \zeta}{\partial t}\, dx\, dy - \mathcal{H}(\varphi,\zeta,\psi_1,\psi_2,\ldots,\psi_M) \right) dt, \tag{3.2}$$

for the Hamiltonian $\mathcal{H}$. The minimized functional is called the Lagrangian (see Equation (2.32)). The zero variations of the Lagrangian w.r.t. φ and ζ yield Equations (2.35). Because the first term of the Lagrangian is independent of ψ_m, the zero variation w.r.t. ψ_m yields a zero variation of the Hamiltonian w.r.t. ψ_m. Hence, we get the following set of equations

$$\begin{aligned} \frac{\partial \zeta}{\partial t} - \delta_\varphi H(\varphi,\zeta,\psi_1,\ldots,\psi_M) &= 0, && \text{(3.3a)} \\ \frac{\partial \varphi}{\partial t} + \delta_\zeta H(\varphi,\zeta,\psi_1,\ldots,\psi_M) &= 0, && \text{(3.3b)} \\ \delta_{\psi_m} H(\varphi,\zeta,\psi_1,\ldots,\psi_M) &= 0 \qquad \text{for } m = 1,2,\ldots,M. && \text{(3.3c)} \end{aligned}$$

These equations describe the variational Boussinesq model for fluid motions with a predefined vertical shape.

3.1 General series model

The Hamiltonian system (3.3) is written in a variational form. For computational purposes, the variations of the Hamiltonian density functional H should be written out in terms of the basic variables φ, ζ and ψ_m. First, this will be done for the general case. In Sections 3.2 and 3.3 the shape function will be chosen according to two different models.
Let's assume that the bottom does not change directly in time, so $\frac{\partial h}{\partial t} = 0$, and let's assume a mildly sloping bottom profile as well, i.e., $\nabla h = 0$. With these assumptions and using Hamiltonian (2.30) and the kinetic energy (2.25), the Hamiltonian density reads

$$H(\varphi,\zeta,\psi_m) = \int_{-h}^{\zeta} \frac{1}{2}\left(\nabla\left(\varphi + \sum_{m=1}^{M} f_m\psi_m \right) \right)^2 dz + \frac{1}{2} g(\zeta^2 - h^2). \tag{3.4}$$

The Hamiltonian density (3.4) can now be written as

$$\begin{aligned} H(\varphi,\zeta,\psi_m) &= \frac{1}{2}(h+\zeta)(\nabla\varphi)^2 + \frac{1}{2}\sum_{m,n=1}^{M} F_{mn}\nabla\psi_m\cdot\nabla\psi_n + \frac{1}{2}(\nabla\zeta)^2\sum_{m,n=1}^{M} G_{mn}\psi_m\psi_n \\ &\quad + \frac{1}{2}\sum_{m,n=1}^{M} K_{mn}\psi_m\psi_n + \nabla\varphi\cdot\sum_{m=1}^{M} P_m\nabla\psi_m + \nabla\varphi\cdot\nabla\zeta\sum_{m=1}^{M} Q_m\psi_m \\ &\quad + \nabla\zeta\cdot\sum_{m,n=1}^{M} R_{mn}\psi_n\nabla\psi_m + \frac{1}{2}g(\zeta^2-h^2), \end{aligned} \tag{3.5}$$

with $F_{mn}, G_{mn}, K_{mn}, P_m, Q_m$ and R_{mn} functionals given by

$$F_{mn} := \int_{-h}^{\zeta} f_m f_n \, dz, \tag{3.6a}$$

$$G_{mn} := \int_{-h}^{\zeta} \frac{\partial f_m}{\partial\zeta}\frac{\partial f_n}{\partial\zeta} \, dz, \tag{3.6b}$$

$$K_{mn} := \int_{-h}^{\zeta} \frac{\partial f_m}{\partial z}\frac{\partial f_n}{\partial z} \, dz, \tag{3.6c}$$

$$P_m := \int_{-h}^{\zeta} f_m \, dz, \tag{3.6d}$$

$$Q_m := \int_{-h}^{\zeta} \frac{\partial f_m}{\partial\zeta} \, dz, \tag{3.6e}$$

$$R_{mn} := \int_{-h}^{\zeta} f_m\frac{\partial f_n}{\partial\zeta} \, dz. \tag{3.6f}$$

For a thorough derivation, see Appendix E.1.

Note that R_{mn} is the only integral being non-symmetric in m and n and that the functionals depend on ζ, because the integrals are over the vertical fluid domain from depth h to water level ζ.

With the general Hamiltonian density (3.5), the Hamiltonian system (3.3) can now be rewritten in terms of the basic variables φ, ζ and ψ_m (see Appendix E.1). Subsequently, the continuity equation (3.3a) reads

$$\frac{\partial\zeta}{\partial t} + \nabla\cdot\left((h+\zeta)\nabla\varphi + \sum_{m=1}^{M} P_m\nabla\psi_m + \nabla\zeta\sum_{m=1}^{M} Q_m\psi_m\right) = 0. \tag{3.7}$$

The second equation (3.3b) from the Hamiltonian system becomes

$$\frac{\partial\varphi}{\partial t} + \frac{1}{2}(\nabla\varphi)^2 + g\zeta + \mathcal{R} = 0, \tag{3.8}$$

which is equivalent to the Bernoulli equation (2.21), but now with an extra *non-hydrostatic*

term $\mathcal{R}(x,y,t)$, given by

$$\begin{aligned}\mathcal{R} \;:=\; & \frac{1}{2}\sum_{m,n=1}^{M} F'_{mn}\nabla\psi_m\cdot\nabla\psi_n + \frac{1}{2}\sum_{m,n=1}^{M}\left((\nabla\zeta)^2 G'_{mn} + K'_{mn}\right)\psi_m\psi_n \\ & +\nabla\varphi\cdot\sum_{m=1}^{M}\left(P'_m\nabla\psi_m + Q'_m\psi_m\nabla\zeta\right) + \nabla\zeta\cdot\sum_{m,n=1}^{M} R'_{mn}\psi_n\nabla\psi_m \\ & -\nabla\cdot\left(\sum_{m=1}^{M} Q_m\psi_m\nabla\varphi + \sum_{m,n=1}^{M}\left(R_{mn}\psi_n\nabla\psi_m + G_{mn}\psi_m\psi_n\nabla\zeta\right)\right), \end{aligned} \tag{3.9}$$

with prime indicating a variation w.r.t. ζ, so $F'_{mn} = \frac{\delta_\zeta F_{mn}}{\delta\zeta}$.
Equation (3.3c), given by the zero first variation of the Hamiltonian w.r.t. ψ_ℓ, can be written as

$$\begin{aligned} & Q_\ell\;\nabla\varphi\cdot\nabla\zeta + \sum_{m=1}^{M}\left(K_{\ell m} + (\nabla\zeta)^2 G_{\ell m}\right)\psi_m + \nabla\zeta\cdot\sum_{m=1}^{M} R_{m\ell}\nabla\psi_m \\ & \qquad -\nabla\cdot\left(\sum_{m=1}^{M} F_{\ell m}\nabla\psi_m + P_\ell\nabla\varphi + \sum_{m=1}^{M} R_{\ell m}\psi_m\nabla\zeta\right) \;=\; 0, \end{aligned} \tag{3.10}$$

for $\ell = 1, 2, \ldots, M$.
Equations (3.7), (3.8) and (3.10) are the main equations of the general variational Boussinesq model, in terms of the vertical shape functions f_m. In sections 3.2 and 3.3 two different models of the vertical shape f_m will be considered.

3.1.1 Shallow water equations

The equations in the variational Boussinesq model, as described above, are a generalization of the more well-known shallow water equations. The main difference is in modelling the vertical structure of the fluid flow. In the Boussinesq model, the shape functions $f_m(z)$ and the variables $\psi_m(x,y,t)$ give the vertical motions, allowing variations in the vertical velocity of the fluid. The shallow water equations uses vertically averaged velocities instead [13]. Therefore, no changes in vertical motions can be derived from the shallow water equations.
To reduce the variational Boussinesq model to the shallow water equation, one has to ignore ψ, that is, omit all terms with ψ and Equation (3.10). Substitute the velocity for the potential and differentiate Equation (3.8) to both x and y. Then, for irrotational flow, equations (3.7) and (3.8) become

$$\frac{\partial\zeta}{\partial t} + \frac{\partial(h+\zeta)u}{\partial x} + \frac{\partial(h+\zeta)v}{\partial y} \;=\; 0, \tag{3.11a}$$

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + g\frac{\partial\zeta}{\partial x} \;=\; 0, \tag{3.11b}$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + g\frac{\partial\zeta}{\partial y} \;=\; 0, \tag{3.11c}$$

which are the *shallow water equations* [22]; see also Appendix G.

3.2 Parabolic model

The first model considered for the vertical shape is a parabolic one [20]. For this model, the shape function is given by

$$f^{(p)} \;:=\; \frac{1}{2}(z-\zeta)\left(1+\frac{h+z}{h+\zeta}\right). \tag{3.12}$$

Note that we have $f=0$ at $z=\zeta$ and that the series expansion consists of only one shape function, so $M=1$. When the parabolic shape (3.12) is substituted into the Hamiltonian (3.4), one gets

$$\begin{aligned} H^{(p)} \;=\; & \frac{1}{2}(h+\zeta)\Big(\nabla\varphi-\frac{2}{3}\psi\nabla\zeta-\frac{1}{3}(h+\zeta)\nabla\psi\Big)^2 \\ & +\frac{1}{90}(h+\zeta)\left(\psi\nabla\zeta-(h+\zeta)\nabla\psi\right)^2+\frac{1}{6}(h+\zeta)\psi^2+\frac{1}{2}g(\zeta^2-h^2). \end{aligned} \tag{3.13}$$

In Appendix E.2 the derivation of this Hamiltonian is shown, as well as the derivation of the Hamiltonian system, given by

$$\frac{\partial\zeta}{\partial t}+\nabla\cdot\left((h+\zeta)\Big(\nabla\varphi-\frac{2}{3}\psi\nabla\zeta-\frac{1}{3}(h+\zeta)\nabla\psi\Big)\right) \;=\; 0, \tag{3.14a}$$

$$\begin{aligned} \frac{\partial\varphi}{\partial t}+\frac{1}{2}\left(\nabla\varphi-\frac{2}{3}\psi\nabla\zeta-\frac{2}{3}(h+\zeta)\nabla\psi\right)^2 \\ +\frac{1}{6}\Big(1+\frac{1}{5}(\nabla\zeta)^2\Big)\psi^2-\frac{1}{45}\Big((h+\zeta)\nabla\psi+\psi\nabla\zeta\Big)^2 \\ +\nabla\cdot\left((h+\zeta)\Big(\frac{2}{3}\nabla\varphi-\frac{7}{15}\psi\nabla\zeta-\frac{1}{5}(h+\zeta)\nabla\psi\Big)\psi\right)+g\zeta \end{aligned} \;=\; 0, \tag{3.14b}$$

$$\begin{aligned} (h+\zeta)\psi\Big(\frac{1}{3}+\frac{7}{15}(\nabla\zeta)^2\Big)-\Big(\frac{2}{3}(h+\zeta)\nabla\varphi-\frac{1}{5}(h+\zeta)^2\nabla\psi\Big)\cdot\nabla\zeta \\ +\nabla\cdot\left(\frac{1}{3}(h+\zeta)^2\nabla\varphi-\frac{1}{5}(h+\zeta)^2\psi\nabla\zeta-\frac{2}{15}(h+\zeta)^3\nabla\psi\right) \end{aligned} \;=\; 0. \tag{3.14c}$$

As derived in (D.4), the depth averaged horizontal velocity $\mathbf{U}$ for a parabolic shape reads

$$\mathbf{U}(x,y) \;=\; \nabla\varphi-\frac{2}{3}\psi\nabla\zeta-\frac{1}{3}(h+\zeta)\nabla\psi. \tag{3.15}$$

With this, the first equation (3.14a) becomes

$$\frac{\partial\zeta}{\partial t}+\nabla\cdot\big((h+\zeta)\mathbf{U}\big) \;=\; 0. \tag{3.16}$$

This equation represents mass conservation, for a time independent bottom shape [22] (see also Equation (3.11)). The second equation (3.14b) represents the Bernoulli equation. The third equation (3.14c) is an elliptic partial differential equation, because the second-order derivatives of ψ are in the form of the Laplace operator [45].

3.3 Cosine-hyperbolic model

Another physically realistic choice for the vertical shape function is a cosine-hyperbolic one [21], given by

$$f^{(c)} \;:=\; \cosh(\kappa(h+z)) - \cosh(\kappa(h+\zeta)), \tag{3.17}$$

with $\kappa(x,y,t)$ a shape parameter. In Section 4.5 it will be explained that the parameter κ is related to the wave number.
Note that the cosine-hyperbolic model is related to the parabolic model by $\lim_{\kappa\to 0} \frac{f^{(c)}}{\kappa^2(h+\zeta)} = f^{(p)}$. At the water surface, the condition $f|_{z=\zeta} = 0$ is satisfied. At the bottom, the boundary condition (2.18c) can be written as $|\mathbf{n}_b|\frac{\partial h}{\partial t} = -\nabla\varphi\cdot\nabla h - \frac{\partial f}{\partial\zeta}\psi\nabla\zeta\cdot\nabla h - f\nabla\psi\cdot\nabla h - \frac{\partial f}{\partial z}\psi$ at $z=-h$. With the mild slope condition $\nabla h = 0$ this reduces to $\psi\frac{\partial f}{\partial z}|_{z=-h} = 0$, which is satisfied by both the parabolic and cosine-hyperbolic shape functions.
In Appendix E.3 the derivation of the Hamiltonian system with a cosine-hyperbolic shape function is given. The resulting system reads

$$\frac{\partial\zeta}{\partial t} + \nabla\cdot\big((h+\zeta)\,\mathbf{U}\big) \;=\; 0, \tag{3.18a}$$

$$\begin{aligned}
&\frac{\partial\varphi}{\partial t} + \frac{1}{2}\mathbf{U}^2 + g\zeta + \frac{1}{2}\kappa^2\mathcal{S}^2\psi^2 + \frac{1}{2}\mathcal{D}^2(\nabla\psi)^2 \\
&-\kappa(h+\zeta)\mathbf{U}\cdot\left(\Big(\mathcal{S} - \frac{\mathcal{D}}{\kappa(h+\zeta)}\Big)\nabla\psi + \kappa\,\mathcal{C}\psi\nabla\zeta\right) + \nabla\cdot\big(\kappa(h+\zeta)\mathcal{S}\mathbf{U}\psi\big) \;=\; 0,
\end{aligned} \tag{3.18b}$$

$$\begin{aligned}
&-\kappa\,(h+\zeta)\,\mathcal{S}\left(\nabla\varphi - \mathcal{D}\nabla\psi - \kappa\mathcal{S}\psi\nabla\zeta\right)\cdot\nabla\zeta \\
&+\frac{1}{2}\kappa\big(\mathcal{S}\,\mathcal{C} - \kappa(h+\zeta)\big)\psi + \nabla\cdot\left((h+\zeta)\mathcal{D}\left(\nabla\varphi - \kappa\mathcal{S}\psi\nabla\zeta\right)\right) \\
&+\nabla\cdot\left(\frac{1}{\kappa}\Big(\frac{\mathcal{S}^2}{\kappa(h+\zeta)} - \mathcal{D}^2\kappa(h+\zeta) - \frac{1}{2}\kappa(h+\zeta) - \frac{1}{2}\mathcal{S}\mathcal{C}\Big)\nabla\psi\right) \;=\; 0,
\end{aligned} \tag{3.18c}$$

with the functionals

$$\mathcal{D} \;:=\; \cosh(\kappa(h+\zeta)) - \frac{\sinh(\kappa(h+\zeta))}{\kappa(h+\zeta)}, \tag{3.19a}$$

$$\mathcal{S} \;:=\; \sinh(\kappa(h+\zeta)), \tag{3.19b}$$

$$\mathcal{C} \;:=\; \cosh(\kappa(h+\zeta)). \tag{3.19c}$$

The depth-averaged velocity used in the equations is given by

$$\mathbf{U} \;:=\; \nabla\varphi - \mathcal{D}\nabla\psi - \kappa\mathcal{S}\psi\nabla\zeta, \tag{3.20}$$

as derived in (D.6).
Equations (3.14) and (3.18) give two different models for the non-linear variational Boussinesq model. In Chapter 4 they will be linearized to get the model equations.

4 Linearized variational Boussinesq model

The variational Boussinesq model presented in the previous chapters resulted in a system of three equations (3.7),(3.8),(3.10). This non-linear Hamiltonian system is the basis for the variational Boussinesq model. In order to simplify these equations and thus reducing the computational effort, the Hamiltonian will be linearized.

4.1 Average velocity

The velocity of the fluid is given by $\mathbf{u} = \nabla\Phi$. The velocity of the fluid motion can be divided in two components, namely an average velocity consisting mainly of the current and a smaller term for the velocity due to wave motions. So

$$\nabla\Phi \quad = \quad \widetilde{\mathbf{U}} + \nabla\widetilde{\phi}, \tag{4.1}$$

with $\widetilde{\mathbf{U}}(x, y, t)$ the average horizontal velocity and $\nabla\widetilde{\phi}$ the velocity of the fluid due to wave motions. The average velocity $\widetilde{\mathbf{U}}$ is not the same as the depth averaged velocity $\mathbf{U}$, presented in Appendix D, because $\widetilde{\mathbf{U}}$ has also been averaged over time. Nevertheless, it may depend on time, but not as quick as the other variables.

As in Equation (3.1), the potential $\widetilde{\phi}$ will be written in the form of a surface velocity potential $\widetilde{\varphi}$ and a series expansion in vertical shape functions f_m $(m = 1, 2, \ldots, M)$. The two models for the vertical shape function, parabolic (3.12) and cosine-hyperbolic (3.17), both consist of only one shape function. We thus take $M = 1$ in the following, so

$$\widetilde{\phi}(x, y, z, t) \quad = \quad \widetilde{\varphi}(x, y, t) + f(z)\,\psi(x, y, t), \tag{4.2a}$$

$$f \quad = \quad 0 \qquad \text{for } z = \overline{\zeta}. \tag{4.2b}$$

In Equation (3.1b) the vertical shape function f was taken zero at the actual water level ζ. As an approximation, we assume the shape function to be zero-valued at the mean water level $\overline{\zeta}$, as seen in Equation (4.2b). The advantage of this approximation is that f does not depend on ζ, so $\frac{\partial f}{\partial \zeta} = 0$.

The horizontal gradient of f now becomes $\nabla f = \frac{\partial f}{\partial h}\nabla h + \frac{\partial f}{\partial \overline{\zeta}}\nabla\overline{\zeta}$. We have the mild slope condition for the bottom $\nabla h = 0$ and an additional assumption of $\nabla\overline{\zeta} = 0$, implying an almost constant average water level. Hence, the horizontal gradient of the vertical shape function f is zero.

From now on, the tildes will be omitted for convenience. However, one should keep in mind that there is a slight distinction, given by the current $\widetilde{\mathbf{U}}$, between the potential φ used in the remainder of this report and the one used previously.

In order to get the linearized Hamiltonian system, substitute the velocity potential (4.1) in Equation (3.3). The main term in the Hamiltonian (2.30) is $\frac{1}{2}(\nabla\Phi)^2$ from the kinetic

energy (2.25) and can be rewritten as

$$\begin{aligned}
\int_{-h}^{\zeta} \frac{1}{2}(\nabla\Phi)^2 dz &= \int_{-h}^{\zeta} \frac{1}{2}(\mathbf{U}+\nabla\phi)^2 dz \\
&= \int_{0}^{\zeta} \left(\frac{1}{2}\mathbf{U}^2 + \mathbf{U}\cdot\nabla(\varphi+f\psi) + \frac{1}{2}\big(\nabla(\varphi+f\psi)\big)^2\right) dz \\
&\quad + \int_{-h}^{0} \left(\frac{1}{2}\mathbf{U}^2 + \mathbf{U}\cdot\nabla(\varphi+f\psi) + \frac{1}{2}\big(\nabla(\varphi+f\psi)\big)^2\right) dz \\
&= \frac{1}{2}\zeta\mathbf{U}^2 + \zeta\mathbf{U}\cdot\nabla\varphi + \mathbf{U}\cdot\nabla\psi\int_0^{\zeta} f dz + \int_0^{\zeta}\left(\frac{1}{2}(\nabla\varphi+f\nabla\psi)^2 + \frac{1}{2}(f'\psi)^2\right) dz \\
&\quad + \frac{1}{2}h\mathbf{U}^2 + h\mathbf{U}\cdot\nabla\varphi + \mathbf{U}\cdot\nabla\psi\int_{-h}^{0} f dz + \int_{-h}^{0}\left(\frac{1}{2}(\nabla\varphi+f\nabla\psi)^2 + \frac{1}{2}(f'\psi)^2\right) dz
\end{aligned} \tag{4.3}$$

Some terms of this expression can already be omitted.

- The term $\frac{1}{2}h\mathbf{U}^2$ has zero variations w.r.t. φ, ζ and ψ and will therefore not appear in the Hamiltonian system.
- Because $f = 0$ at $z = \bar{\zeta}$, the term $\mathbf{U}\cdot\nabla\psi\int_0^{\zeta} f dz$ vanishes.
- The integral $\int_0^{\zeta}\left(\frac{1}{2}(\nabla\varphi+f\nabla\psi)^2 + \frac{1}{2}(f'\psi)^2\right) dz$ has a cubic contribution in the Lagrangian and therefore quadratic contribution in the model equations. By performing a linearization, this integral will vanish.

We end up with the linearized Lagrangian

$$\begin{aligned}
\mathcal{L}_0 &= \int\left(\iint \varphi\frac{\partial\zeta}{\partial t}\, dx\, dy - \iint\left(\frac{1}{2}\zeta\mathbf{U}^2 + h\mathbf{U}\cdot\nabla\varphi + \mathbf{U}\cdot\nabla\psi\int_{-h}^{0} f\, dz + \zeta\mathbf{U}\cdot\nabla\varphi\right.\right. \\
&\quad \left.\left. + \int_{-h}^{0}\left(\frac{1}{2}(\nabla\varphi+f\nabla\psi)^2 + \frac{1}{2}(f'\psi)^2\right) dz + \frac{1}{2}g(\zeta^2-h^2)\right) dx\, dy\right) dt.
\end{aligned} \tag{4.4}$$

As explained in the next section, more terms of this linearized Lagrangian will be omitted.

4.1.1 Galilean invariance

The model is made in an fixed frame of reference. One can also take a frame of reference comoving with the current, or with a ship. When the velocity of this moving frame is constant, one speaks of a *Galilean transformation* [25]. We want the model to be equivalent in both frames, which is called *Galilean invariance*. This requirement can only hold when some of the terms in the linearized Lagrangian are omitted.
A Galilean transformation is a transformation from the frame (x, y, z, t) to $(\tilde{x}, \tilde{y}, \tilde{z}, \tilde{t})$ given by (see [25])

$$\tilde{x} = x - \tilde{U}t, \tag{4.5a}$$
$$\tilde{y} = y - \tilde{V}t, \tag{4.5b}$$
$$\tilde{z} = z, \tag{4.5c}$$
$$\tilde{t} = t, \tag{4.5d}$$

with $\tilde{U}, \tilde{V}$ constant velocities of the moving frame.
The derivatives in the moving frame will become

$$\frac{\partial}{\partial x} = \frac{\partial \tilde{x}}{\partial x}\frac{\partial}{\partial \tilde{x}} = \frac{\partial}{\partial \tilde{x}} \quad \text{and} \quad \frac{\partial}{\partial y} = \frac{\partial \tilde{y}}{\partial y}\frac{\partial}{\partial \tilde{y}} = \frac{\partial}{\partial \tilde{y}}, \tag{4.6}$$

so $\nabla = \tilde{\nabla}$.
For the time derivative, we have

$$\frac{\partial}{\partial t} = \frac{\partial \tilde{t}}{\partial t}\frac{\partial}{\partial \tilde{t}} + \frac{\partial \tilde{x}}{\partial t}\frac{\partial}{\partial \tilde{x}} + \frac{\partial \tilde{y}}{\partial t}\frac{\partial}{\partial \tilde{y}} = \frac{\partial}{\partial \tilde{t}} - \tilde{U}\frac{\partial}{\partial \tilde{x}} - \tilde{V}\frac{\partial}{\partial \tilde{y}} = \frac{\partial}{\partial \tilde{t}} - \tilde{\mathbf{U}} \cdot \nabla. \tag{4.7}$$

Observe that in a variational setting, we have

$$\int \iint \varphi \frac{\partial \zeta}{\partial t}\, dx\, dy\, dt \;=\; \int \iint -\zeta \frac{\partial \varphi}{\partial t}\, dx\, dy\, dt, \tag{4.8}$$

by partial integration.
Applying the Galilean transformation (4.5) to this integral gives

$$\begin{aligned} \int \iint -\zeta \frac{\partial \varphi}{\partial t}\, dx\, dy\, dt &= \int \iint -\tilde{\zeta}\left(\frac{\partial \tilde{\varphi}}{\partial \tilde{t}} - \tilde{\mathbf{U}} \cdot \tilde{\nabla}\tilde{\varphi}\right) d\tilde{x}\, d\tilde{y}\, d\tilde{t} \\ &= \int \iint \left(\tilde{\varphi}\frac{\partial \tilde{\zeta}}{\partial \tilde{t}} + \tilde{\zeta}\tilde{\mathbf{U}} \cdot \tilde{\nabla}\tilde{\varphi}\right) d\tilde{x}\, d\tilde{y}\, d\tilde{t}. \end{aligned} \tag{4.9}$$

Using this expression, application of the Galilean transformation to the linearized Lagrangian (4.4) yields

$$\begin{aligned} \tilde{\mathcal{L}}_0 \;=\; \int \Bigg(\iint \tilde{\varphi}\frac{\partial \tilde{\zeta}}{\partial \tilde{t}}\, d\tilde{x}\, d\tilde{y} + \iint \tilde{\zeta}\tilde{\mathbf{U}} \cdot \tilde{\nabla}\tilde{\varphi}\, d\tilde{x}\, d\tilde{y} - \iint \Bigg(\frac{1}{2}\tilde{\zeta}\mathbf{U}^2 + \tilde{h}\mathbf{U} \cdot \tilde{\nabla}\tilde{\varphi} + \mathbf{U} \cdot \tilde{\nabla}\tilde{\psi} \int_{-h}^{0} f\, dz \\ + \tilde{\zeta}\mathbf{U} \cdot \tilde{\nabla}\tilde{\varphi} + \int_{-h}^{0} \left(\frac{1}{2}(\tilde{\nabla}\tilde{\varphi} + f\tilde{\nabla}\tilde{\psi})^2 + \frac{1}{2}(f'\tilde{\psi})^2\right) dz + \frac{1}{2}g(\tilde{\zeta}^2 - \tilde{h}^2)\Bigg) d\tilde{x}\, d\tilde{y}\Bigg) d\tilde{t}. \end{aligned} \tag{4.10}$$

Let's consider a frame of reference moving with the average velocity, i.e., $\tilde{\mathbf{U}} = \mathbf{U}$. Then the linearized Lagrangian becomes

$$\begin{aligned} \tilde{\mathcal{L}}_0 \;=\; \int \Bigg(\iint \tilde{\varphi}\frac{\partial \tilde{\zeta}}{\partial \tilde{t}}\, d\tilde{x}\, d\tilde{y} - \iint \Bigg(\int_{-h}^{0} \left(\frac{1}{2}(\tilde{\nabla}\tilde{\varphi} + f\tilde{\nabla}\tilde{\psi})^2 + \frac{1}{2}(f'\tilde{\psi})^2\right) dz + \frac{1}{2}g(\tilde{\zeta}^2 - \tilde{h}^2) \\ + \frac{1}{2}\tilde{\zeta}\mathbf{U}^2 + \tilde{h}\mathbf{U} \cdot \tilde{\nabla}\tilde{\varphi} + \mathbf{U} \cdot \tilde{\nabla}\tilde{\psi} \int_{-h}^{0} f\, dz\Bigg) d\tilde{x}\, d\tilde{y}\Bigg) d\tilde{t}. \end{aligned} \tag{4.11}$$

Note that the term $\zeta\mathbf{U} \cdot \varphi$ has now vanished. However, there are still some terms in the Lagrangian depending on the average velocity. To make the linearized Lagrangian Galilean invariant, these three terms have to be omitted. This yields

$$\begin{aligned} \mathcal{L}_0 \;=\; \int \Bigg(\iint \varphi \frac{\partial \zeta}{\partial t}\, dx\, dy - \iint \Bigg(\zeta\mathbf{U} \cdot \nabla\varphi \\ + \int_{-h}^{0} \left(\frac{1}{2}(\nabla\varphi + f\nabla\psi)^2 + \frac{1}{2}(f'\psi)^2\right) dz + \frac{1}{2}g(\zeta^2 - h^2)\Bigg) dx\, dy\Bigg) dt, \end{aligned} \tag{4.12}$$

the linearized Lagrangian, invariant under a Galilean transformation.
With this expression (4.12), the linearized Hamiltonian can be written as

$$\mathcal{H}_0 \;=\; \iint \left(\zeta \mathbf{U}\cdot\nabla\varphi + \int_{-h}^{0} \Big(\frac{1}{2}(\nabla\varphi + f\nabla\psi)^2 + \frac{1}{2}(f'\psi)^2 \Big) dz + \frac{1}{2} g(\zeta^2 - h^2) \right) dx\, dy. \quad (4.13)$$

Similar to (3.3), the linearized Hamiltonian system reads

$$\frac{\partial \zeta}{\partial t} - \delta_\varphi H_0 \;=\; 0, \quad (4.14a)$$

$$\frac{\partial \varphi}{\partial t} + \delta_\zeta H_0 \;=\; 0, \quad (4.14b)$$

$$\delta_\psi H_0 \;=\; 0. \quad (4.14c)$$

These three equations are the basis of the linear variational Boussinesq model. In the next sections the shape models we have discussed in Chapter 3 are applied to these equations.

4.2 Linearized general series model

The Hamiltonian system (3.3) has been changed into a linearized equivalent (4.14). This system is given in terms of the variations of the linearized Hamiltonian. First, these variations are calculated in terms of the basic variables. This will be done for the general series model (4.2). The derivations are shown in more detail in Appendix E.4.
The linearized Hamiltonian for the general series model is given by (see Equation (E.33)):

$$\begin{aligned} \mathcal{H}_0 \;=\; \iint \Big(& \zeta \mathbf{U}\cdot\nabla\varphi + \frac{1}{2} h \left(\nabla\varphi - \mathcal{D}_0 \nabla\psi\right)^2 \\ & + \frac{1}{2}\left(\mathcal{N}_0 - h\mathcal{D}_0^2\right)(\nabla\psi)^2 + \frac{1}{2}\mathcal{M}_0\psi^2 + \frac{1}{2} g(\zeta^2 - h^2) \Big) dx\, dy, \end{aligned} \quad (4.15)$$

with the functionals

$$\mathcal{D}_0 \;:=\; -\frac{1}{h}\int_{-h}^{0} f \; dz, \quad (4.16a)$$

$$\mathcal{M}_0 \;:=\; \int_{-h}^{0} f'^2 \; dz, \quad (4.16b)$$

$$\mathcal{N}_0 \;:=\; \int_{-h}^{0} f^2 \; dz. \quad (4.16c)$$

Note that these functionals are independent of the basic variables φ, ζ and ψ and are therefore constants.
The zero variations of the linearized Lagrangian give (see Equation (E.38))

$$\frac{\partial \zeta}{\partial t} + \nabla\cdot\left(\zeta\,\mathbf{U} + h\,\nabla\varphi - h\,\mathcal{D}_0\,\nabla\psi\right) \;=\; 0, \quad (4.17a)$$

$$\frac{\partial \varphi}{\partial t} + \mathbf{U}\cdot\nabla\varphi + g\zeta \;=\; 0, \quad (4.17b)$$

$$\mathcal{M}_0\,\psi + \nabla\cdot\left(h\,\mathcal{D}_0\,\nabla\varphi - \mathcal{N}_0\,\nabla\psi\right) \;=\; 0. \quad (4.17c)$$

These are the three basic equations of the linearized variational Boussinesq model for general vertical series.

4.3 Linearized parabolic model

The linearized Hamiltonian system (4.14) for the general series model will now be applied to the parabolic model (3.12). The functionals (4.16) can be evaluated with the integrals given in Appendix E.2. We have

$$\mathcal{D}_0^{(p)} = \frac{1}{3}h, \tag{4.18a}$$

$$\mathcal{M}_0^{(p)} = \frac{1}{3}h, \tag{4.18b}$$

$$\mathcal{N}_0^{(p)} = \frac{2}{15}h^3. \tag{4.18c}$$

The linearized parabolic Hamiltonian now becomes

$$\begin{aligned}\mathcal{H}_0^{(p)} = \iint \Big(&\zeta \mathbf{U}\cdot\nabla\varphi + \frac{1}{2}h\Big(\nabla\varphi - \frac{1}{3}h\nabla\psi\Big)^2 \\ &+ \frac{1}{90}h^3(\nabla\psi)^2 + \frac{1}{6}h\psi^2 + \frac{1}{2}g(\zeta^2 - h^2)\Big)\,dx\,dy.\end{aligned} \tag{4.19}$$

The linearized parabolic Hamiltonian system is given by

$$\frac{\partial\zeta}{\partial t} + \nabla\cdot\Big(\zeta\,\mathbf{U} + h\,\nabla\varphi - \frac{1}{3}h^2\,\nabla\psi\Big) = 0, \tag{4.20a}$$

$$\frac{\partial\varphi}{\partial t} + \mathbf{U}\cdot\nabla\varphi + g\zeta = 0, \tag{4.20b}$$

$$h\,\psi + \nabla\cdot\Big(h^2\nabla\varphi - \frac{2}{5}h^3\,\nabla\psi\Big) = 0. \tag{4.20c}$$

This system contains the model equations for parabolic-shaped fluid flow.

4.4 Linearized cosine-hyperbolic model

For the cosine-hyperbolic model (3.17), the linearized Hamiltonian system will be derived by substituting the parameters (4.16) in Equation (4.17). The functionals are derived with the integrals given in Appendix E.3. We have

$$\mathcal{D}_0^{(c)} = \mathcal{C}_0 - \frac{\mathcal{S}_0}{\kappa h}, \tag{4.21a}$$

$$\mathcal{M}_0^{(c)} = \frac{1}{2}\kappa\,\mathcal{S}_0\,\mathcal{C}_0 - \frac{1}{2}\kappa^2 h, \tag{4.21b}$$

$$\mathcal{N}_0^{(c)} = -\frac{3}{2}\frac{1}{\kappa}\,\mathcal{S}_0\,\mathcal{C}_0 + \frac{1}{2}h + h\,\mathcal{C}_0^2, \tag{4.21c}$$

with the functionals

$$\mathcal{S}_0 := \sinh(\kappa h), \tag{4.22a}$$

$$\mathcal{C}_0 := \cosh(\kappa h). \tag{4.22b}$$

The linearized Hamiltonian is calculated by substituting these parameters in Equation (4.15) and reads

$$\begin{aligned}\mathcal{H}_0^{(c)} &= \iint \Bigg(\zeta \mathbf{U}\cdot\nabla\varphi + \frac{1}{2}h\Big(\nabla\varphi - \Big(\mathcal{C}_0 - \frac{\mathcal{S}_0}{\kappa h}\Big)\nabla\psi\Big)^2 + \frac{1}{4}\kappa\big(\mathcal{S}_0\,\mathcal{C}_0 - \kappa h\big)\psi^2 \\ &\quad + \frac{1}{2}\Big(-\frac{3}{2}\frac{1}{\kappa}\,\mathcal{S}_0\,\mathcal{C}_0 + \frac{1}{2}h + \frac{\mathcal{S}_0}{\kappa}\Big(2\mathcal{C}_0 - \frac{\mathcal{S}_0}{\kappa h}\Big)\Big)(\nabla\psi)^2 + \frac{1}{2}g\big(\zeta^2 - h^2\big)\Bigg)dx\,dy. \quad (4.23)\end{aligned}$$

With Equation (4.17), the linearized cosine-hyperbolic Hamiltonian system becomes

$$\begin{aligned}\frac{\partial\zeta}{\partial t} + \nabla\cdot\Big(\zeta\,\mathbf{U} + h\,\nabla\varphi - h\,\big(\mathcal{C}_0 - \frac{\mathcal{S}_0}{\kappa h}\big)\,\nabla\psi\Big) &= 0, \quad (4.24a)\\ \frac{\partial\varphi}{\partial t} + \mathbf{U}\cdot\nabla\varphi + g\zeta &= 0, \quad (4.24b)\\ \frac{1}{2}\kappa\big(\mathcal{S}_0\,\mathcal{C}_0 - \kappa h\big)\,\psi + \nabla\cdot\Bigg(\Big(h\,\mathcal{C}_0 - \frac{\mathcal{S}_0}{\kappa}\Big)\,\nabla\varphi - \Big(\frac{1}{2}h - \frac{3}{2}\frac{1}{\kappa}\,\mathcal{S}_0\,\mathcal{C}_0 + h\,\mathcal{C}_0^2\Big)\,\nabla\psi\Bigg) &= 0. \quad (4.24c)\end{aligned}$$

Similar to (4.20), these are the main equations of the cosine-hyperbolic model.

4.5 Dispersion relation

In the cosine-hyperbolic model (3.17), the shape parameter κ has been introduced, which may depend on the horizontal coordinates x and y as well as time t. This parameter has to be defined, either as input, or as a function of the basic variables. In this section, we will derive that κ can be related to the characteristic wave number.
Let's consider a harmonic wave:

$$\zeta = a\sin(\omega t - kx), \quad (4.25)$$

with a denoting the amplitude, ω the frequency and k the wave number. A harmonic wave can also be written in terms of the wave length L and period T, by substituting $k = \frac{2\pi}{L}$ and $\omega = \frac{2\pi}{T}$. When considering a two-dimensional plane wave in a moving fluid, the absolute frequency ω changes in a relative (or intrinsic) frequency σ. This relation is given by $\omega = \sigma + kU_n$ with U_n the component of the current in the wave direction [13]. For a given angle θ between current and the direction of wave propagation, we have $U_n = |\mathbf{U}|\cos(\theta)$.
The wave number k and relative frequency σ are related to each other. For linear wave theory, this *dispersion relation* is given by (see [13])

$$\sigma^2 = gk\tanh(kh). \quad (4.26)$$

For the variational Boussinesq model, given by Equation (4.17), the dispersion relation reads (see [18])

$$\sigma^2 = ghk^2\frac{\mathcal{M}_0 + k^2(\mathcal{N}_0 - h\mathcal{D}_0^2)}{\mathcal{M}_0 + k^2\mathcal{N}_0}. \quad (4.27)$$

Substituting the parameters (4.21) for the cosine-hyperbolic model, we get (see [21])

$$\sigma^2 = g\kappa\frac{k^2}{\kappa^2}\frac{\kappa h\left(\frac{\mathcal{S}_0^{(c)}\mathcal{C}_0^{(c)}}{\kappa h} - 1\right)\left(1 - \frac{k^2}{\kappa^2}\right) + 2\frac{k^2}{\kappa^2}\mathcal{S}_0^{(c)}\mathcal{D}_0^{(c)}}{\left(\frac{\mathcal{S}_0^{(c)}\mathcal{C}_0^{(c)}}{\kappa h} - 1\right)\left(1 - \frac{k^2}{\kappa^2}\right) + 2\frac{k^2}{\kappa^2}\mathcal{C}_0^{(c)}\mathcal{D}_0^{(c)}}. \quad (4.28)$$

When substituting $\kappa = k$ in Equation (4.28). the exact dispersion relation (4.26) is obtained. At first sight, taking k for the shape parameter κ seems to be a good choice. However, this is not very practical because in reality there is not one wave number. Real waves are a superposition of a lot of harmonic waves with different wave numbers. A straightforward solution is to deal with κ as an input variable, which should be taken as close as possible to the expected characteristic wave number.
For small κ we have $\lim_{\kappa h \to 0} f^{(c)} = 0$, because $|z|, |\zeta| \leq h$. A zero shape function f will give an undetermined ψ. To avoid this, the parameter κ may not be taken too small. With some trial a bound of

$$\kappa \quad \geq \quad \frac{\pi}{2h} \tag{4.29}$$

can be obtained [19, 17]
Therefore, κ is defined as the maximum of $\frac{\pi}{2h}$ and an input parameter κ_c.

4.6 Ship

In the previous sections the variational Boussinesq model for water waves has been presented. A moving ship has not been included yet in this wave model. In this section, a method for incorporating a moving ship, which interacts with the wave pattern, is discussed. It will model a ship as a pressure pulse on the water surface.
In the pressure functional, given by Equation (2.13), a source term is added. The pressure of a ship on the water surface specifies this term. When the model is adapted to this different pressure term, the Bernoulli equations (4.17b) will change to

$$\frac{\partial \varphi}{\partial t} + \mathbf{U} \cdot \nabla \varphi + g\zeta \quad = \quad P_s, \tag{4.30}$$

with $P_s := -\frac{p_s}{\rho}$ and p_s the predefined pressure modelling the ship. Note that p_s is zero outside the area of the ship. For a derivation of this equation, see Appendix G.
To define p_s, one can look at the hydrostatic pressure $p_h = \rho g(\zeta - d)$ at depth d. Then, assuming a zero water level, we have $P_h(d) = gd$. Hence, one can model a ship with draft d_s by taking $P_s = gd_s\alpha(x, y)$, with $\alpha(x, y)$ a shape function with one in the middle of the ship and zero on the horizontal boundary of the ship. The function α is similar to a Heaviside function, but with a smooth transition at the edges, for example $\alpha = 1 - \sin^{12}(\frac{\pi}{2}r)$, with r the scaled distance to the center of the ship.

5 The model equations summarized

In Chapter 2 the variational Boussinesq model has been derived. Minimizing the total pressure yielded a Hamiltonian system. The vertical structure of the fluid flow has been introduced in Chapter 3 for two different shape functions: the parabolic and cosine-hyperbolic model. The linearization, explained in Chapter 4, simplified the Hamiltonian system. For convenience, the resulting equations will be summarized in this section.
The model results in the three model equations (4.17), adapted with Equation (4.30) of the ship, reads

$$\frac{\partial \zeta}{\partial t} + \nabla \cdot \left(\zeta \, \mathbf{U} + h \, \nabla \varphi - h \, \mathcal{D}_0 \, \nabla \psi \right) = 0, \tag{5.1a}$$

$$\frac{\partial \varphi}{\partial t} + \mathbf{U} \cdot \nabla \varphi + g \zeta = P_s, \tag{5.1b}$$

$$\mathcal{M}_0 \, \psi + \nabla \cdot \left(h \, \mathcal{D}_0 \, \nabla \varphi - \mathcal{N}_0 \, \nabla \psi \right) = 0, \tag{5.1c}$$

for the three basic variables water level ζ, surface velocity potential φ and vertical structure ψ. The functionals depend on the choice of the vertical shape model. For a parabolic vertical velocity, we have Equation (4.18), i.e.,

$$\mathcal{D}_0^{(p)} = \frac{1}{3} h, \tag{5.2a}$$

$$\mathcal{M}_0^{(p)} = \frac{1}{3} h, \tag{5.2b}$$

$$\mathcal{N}_0^{(p)} = \frac{2}{15} h^3. \tag{5.2c}$$

Note that the parameters are always positive for positive water depth ($h > 0$).
For the cosine-hyperbolic model, we have Equation (4.21), so

$$\mathcal{D}_0^{(c)} = \cosh(\kappa h) - \frac{\sinh(\kappa h)}{\kappa h}, \tag{5.3a}$$

$$\mathcal{M}_0^{(c)} = \frac{1}{2} \kappa \, \sinh(\kappa h) \, \cosh(\kappa h) - \frac{1}{2} \kappa^2 h, \tag{5.3b}$$

$$\mathcal{N}_0^{(c)} = -\frac{3}{2} \frac{1}{\kappa} \, \sinh(\kappa h) \, \cosh(\kappa h) + \frac{1}{2} h + h \, (\cosh(\kappa h))^2. \tag{5.3c}$$

In Appendix F, it has been shown that $\mathcal{D}_0^{(c)}, \mathcal{M}_0^{(c)}, \mathcal{N}_0^{(c)} \geq 0 \; \forall \, \kappa, h \in \mathbb{R}^+$ and $\mathcal{D}_0^{(c)}, \mathcal{M}_0^{(c)}, \mathcal{N}_0^{(c)} = 0 \Leftrightarrow \kappa h = 0$. So for the cosine-hyperbolic model, the parameters are positive for positive water depth.
The first two model equations (5.1a),(5.1b) have a first order time derivative and up to second order spatial derivatives. In Section 7.3, the numerical discretization of it will be derived with the finite volume method and the leapfrog method.
The third model equation (5.1c) is an elliptic partial differential equation in ψ. The numerical discretization of it will lead to a linear system of equations. To solve this efficiently, an iterative method will be used. Building an efficient linear solver for this elliptic equation will be the main goal for this project.

6 Boundary conditions

In Section 2.2 the boundary conditions at the surface and the bottom have been derived. In the horizontal direction, boundaries have not been specified yet. However, the computational domain will be finite and therefore horizontal boundaries should be introduced.
Boundaries can be divided mainly in two different categories: closed and open boundaries. Closed boundaries are boundaries like the shore and breakwaters. No water can flow through these boundaries. Through open boundaries water can propagate freely. These are in fact no physical boundaries, but imposed by the finite computational domain.
In the derivation of the Hamiltonian system, Green's theorem has been applied to rewrite the equations. Because the undefined horizontal domain, no integral over the horizontal boundary occurred in the derivation. For a domain Ω enclosed by the boundary Γ, application of Green's theorem will give a boundary integral.
Because energy can leave and enter the domain through the boundaries, the linearized Lagrangian (4.12) will change in

$$\mathcal{L}_0 = \int \left(\iint_\Omega \varphi \frac{\partial \zeta}{\partial t} \, dx \, dy - \mathcal{H}_0 + \int_\Gamma (\varphi F_\zeta + \zeta F_\varphi + \psi F_\psi) \, ds \right) dt, \tag{6.1}$$

with F the fluxes through the boundary [19].
The derivations explained in Chapter 4 and written out in Appendix E are briefly redone for the bounded Lagrangian (6.1). The unit outward vector on the boundary Γ is denoted by $\mathbf{n}$. Note that the first term in the Lagrangian (6.1) will not lead to a boundary integral, therefore it suffices to look at the zero variations of the linearized Hamiltonian (4.15). The zero variation w.r.t φ is given by

$$\begin{aligned} \delta_\varphi \mathcal{H}_0 &= \iint (\zeta \mathbf{U} \cdot \nabla \delta\varphi + h \left(\nabla\varphi - \mathcal{D}_0 \nabla\psi\right) \cdot \nabla \delta\varphi) \, dx \, dy \\ &= -\iint_\Omega \left(\nabla \cdot \zeta \mathbf{U} \; \delta\varphi + \nabla \cdot h \left(\nabla\varphi - \mathcal{D}_0 \nabla\psi\right) \; \delta\varphi\right) dx \, dy \\ &\quad + \int_\Gamma \left(\mathbf{n} \cdot \zeta \mathbf{U} \; \delta\varphi + \mathbf{n} \cdot h \left(\nabla\varphi - \mathcal{D}_0 \nabla\psi\right) \; \delta\varphi\right) ds. \end{aligned} \tag{6.2}$$

The integral over Ω leads to the continuity equation (4.17a), the integral over Γ will be dealt with in this chapter.
The zero variation of the Hamiltonian w.r.t ζ (see Equation (E.36)) does not result in a boundary integral.
Taking the zero variation of the Hamiltonian w.r.t ψ gives

$$\begin{aligned} \delta_\psi \mathcal{H}_0 &= \iint \left(h \left(\nabla\varphi - \mathcal{D}_0 \nabla\psi\right) \cdot \left(-\mathcal{D}_0 \nabla \delta\psi\right) + \left(\mathcal{N}_0 - h\mathcal{D}_0^2\right) \nabla\psi \cdot \nabla\delta\psi + \mathcal{M}_0 \psi \, \delta\psi\right) dx \, dy \\ &= \iint_\Omega \left(\nabla \cdot \left(h\mathcal{D}_0 \left(\nabla\varphi - \mathcal{D}_0 \nabla\psi\right)\right) \delta\psi - \nabla \cdot \left(\left(\mathcal{N}_0 - h\mathcal{D}_0^2\right) \nabla\psi\right) \delta\psi + \mathcal{M}_0 \psi \, \delta\psi\right) dx \, dy \\ &\quad + \int_\Gamma \left(\mathbf{n} \cdot \left(h\mathcal{D}_0 \left(\nabla\varphi - \mathcal{D}_0 \nabla\psi\right)\right) \delta\psi - \mathbf{n} \cdot \left(\left(\mathcal{N}_0 - h\mathcal{D}_0^2\right) \nabla\psi\right) \delta\psi\right) ds. \end{aligned} \tag{6.3}$$

The integral over Ω yields the elliptic differential equation (4.17c).

The variations should be zero for minimal pressure, therefore the integrand of the boundary integrals in Equations (6.2) and (6.3) should equal the fluxes on the boundary Γ, i.e.,

$$\Big(\zeta\,\mathbf{U} + h\,\nabla\varphi - h\,\mathcal{D}_0\,\nabla\psi\Big)\cdot\mathbf{n} \;=\; F_\zeta, \tag{6.4a}$$

$$\Big(h\,\mathcal{D}_0\,\nabla\varphi - \mathcal{N}_0\,\nabla\psi\Big)\cdot\mathbf{n} \;=\; F_\psi. \tag{6.4b}$$

In the next sections the fluxes F_ζ and F_ψ are chosen according to the type of boundary.
Note that there are only two boundary conditions, whereas there are three equations. This is because with $\psi = 0$, the first two model equations (5.1a) and (5.1b) become a hyperbolic set of equations, which requires one boundary condition [19]. The third model equation (5.1c) is an elliptic one and therefore requires an additional boundary condition.
The boundary conditions (6.4) can be rewritten as

$$h\left(\mathcal{N}_0 - h\mathcal{D}_0^2\right)\nabla\varphi\cdot\mathbf{n} \;=\; \mathcal{N}_0\,F_\zeta + h\,\mathcal{D}_0\,F_\psi - \mathcal{N}_0\,\zeta\,\mathbf{U}\cdot\mathbf{n}, \tag{6.5a}$$

$$\left(\mathcal{N}_0 - h\mathcal{D}_0^2\right)\nabla\psi\cdot\mathbf{n} \;=\; \mathcal{D}_0\,F_\zeta + F_\psi - \mathcal{D}_0\,\zeta\,\mathbf{U}\cdot\mathbf{n}. \tag{6.5b}$$

This shows that two boundary conditions are sufficient, because condition (6.4a) can be used for the continuity equation (5.1a), condition (6.5a) for the Bernoulli equation (5.1b) and condition (6.4b) for the elliptic equation (5.1c).

6.1 Closed boundaries

In Chapter 2 the impermeability of the water bottom (2.20c) was derived as a condition of zero normal flow at the bottom surface. Closed boundaries in the horizontal plane behave similarly. At these boundaries the fluid cannot flow through the boundary, yielding a zero normal velocity. Examples of these boundaries are shores, cliffs, walls and piers. For closed boundaries, the following condition has to be satisfied:

$$\mathbf{u}\cdot\mathbf{n}_c \;=\; 0 \quad \text{or} \quad \left.\frac{\partial\phi}{\partial n}\right|_{\Gamma_c} \;=\; 0 \tag{6.6}$$

with Γ_c the closed part of the boundary and $\mathbf{n}_c$ the unit outward normal on it. These boundary conditions are also called reflective boundary conditions, because incoming waves will reflect back into the domain.
With the expression of the velocity potential ϕ in Equation (4.2), the boundary condition (6.6) can be written as $(\mathbf{U} + \nabla\varphi + f\nabla\psi)\cdot\mathbf{n}_c = 0$. The current $\mathbf{U}$ is an input-variable, and it is assumed that $\mathbf{U}\cdot\mathbf{n}_c = 0$ holds. The boundary condition is than satisfied when

$$\nabla\varphi\cdot\mathbf{n}_c = 0 \quad \text{and} \quad \nabla\psi\cdot\mathbf{n}_c = 0. \tag{6.7}$$

Substituting boundary conditions (6.7) in Equation (6.5) gives

$$F_\zeta = 0 \quad \text{and} \quad F_\psi = 0. \tag{6.8}$$

So another way of characterizing closed boundaries is by imposing zero fluxes at the boundary.

6.2 Open boundaries

When modelling an open sea, there are no physical boundaries. Because the computational domain has to be finite, artificial boundaries are included, which are called *open boundaries*. Waves and water particles can propagate freely through these boundaries. The fluxes at the boundary will therefore be nonzero in general. Because the outgoing flow should vanish, these boundaries are also called *absorbing boundaries*.
Because they are not natureal, absorbing boundaries are quite difficult to build. Especially for non-uniform flows there will always be some reflection, therefore these boundaries are often called *weakly-reflective boundaries*. The type of weakly-reflective boundary used in this model is the Sommerfeld boundary condition [19].
Let's consider a quantity $q(x, y, t)$, representing a simple wave-like pattern. Its wave velocity is given by $\mathbf{c} = c_0 \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \end{bmatrix}$ with $c_0 > 0$ the wave speed and θ the angle of the propagation direction. Wave phenomena satisfy the *wave equation*

$$\frac{\partial^2 q}{\partial t^2} = c_0^2 \left(\frac{\partial^2 q}{\partial x^2} + \frac{\partial^2 q}{\partial y^2} \right). \tag{6.9}$$

Solutions of this differential equations are of the form

$$q(x, y, t) = \tilde{q} \left(\frac{\cos(\theta)}{c_0} x + \frac{\sin(\theta)}{c_0} y - t \right). \tag{6.10}$$

The argument of $\tilde{q}$ is called the characteristic.
The *Sommerfeld condition* reads

$$\cos(\theta) \frac{\partial q}{\partial t} + c_0 \frac{\partial q}{\partial n} = 0, \tag{6.11}$$

with $\frac{\partial q}{\partial n} = \nabla q \cdot \mathbf{n}$ and $\mathbf{n}$ the unit outward normal at the boundary.
At the boundaries, the wave field q can be decoupled in an ingoing and an outgoing wave field, denoted by q_{in} and q_{out}. The outgoing wave field, which has to be absorbed, is now given by $q_{\text{out}} = q - q_{\text{in}}$. Applying the Sommerfeld condition to the outgoing wave field yields

$$\cos(\theta) \frac{\partial q}{\partial t} + c_0 \frac{\partial q}{\partial n} = \cos(\theta) \frac{\partial q_{\text{in}}}{\partial t} + c_0 \frac{\partial q_{\text{in}}}{\partial n}. \tag{6.12}$$

Taking $q = F_\zeta$ and $q = F_\psi$ gives the Sommerfeld boundary conditions for the open boundaries. Incoming waves should therefore be prescribed in the form of $F_{\zeta,\text{in}}$ and $F_{\psi,\text{in}}$. The parameters θ and c_0 have to be prescribed by characteristic values, although ideally they are calculated at each stage.

7 Numerical discretization

In the previous chapters the model has been presented, leading to the model equations (5.1). In this chapter, the numerical discretization of these equations will be presented (see [19]). To start with, a computational domain is introduced. As spatial discretization, the finite volume method will be used. The leapfrog method will be used for the time integration.

7.1 Computational domain

The computational domain will be taken rectangular, with size $L_x \times L_y$. On this domain an equidistant grid is used with nodes on the boundary. In the x-direction there are N_x grid points and in the y-direction there are N_y grid points. Because the mesh is equidistant, the mesh sizes are $\Delta x = \frac{L_x}{N_x - 1}$ and $\Delta y = \frac{L_y}{N_y - 1}$ in the x- and y-direction, respectively.
The nodes are numbered by (m, n) with $m = 1, 2, \ldots, N_x$ and $n = 1, 2, \ldots, N_y$. Node $(1, 1)$ is positioned at the physical coordinates $(x, y) = (0, 0)$ and node (N_x, N_y) corresponds to $(x, y) = (L_x, L_y)$.

7.2 Spatial discretization

The model equations (5.1) are discretized with the finite volume method. The rectangular volumes are centered around the grid points and have size $\Delta x \times \Delta y$. The derivatives will be approximated with centered differences yielding a five-point stencil.
Because the grid is rectangularly structured, one can use the following notation of the nodes:

$$\text{center:} \quad \zeta_C := \zeta_{(m,n)}, \tag{7.1a}$$
$$\text{east:} \quad \zeta_E := \zeta_{(m+1,n)}, \tag{7.1b}$$
$$\text{north:} \quad \zeta_N := \zeta_{(m,n+1)}, \tag{7.1c}$$
$$\text{west:} \quad \zeta_W := \zeta_{(m-1,n)}, \tag{7.1d}$$
$$\text{south:} \quad \zeta_S := \zeta_{(m,n-1)}. \tag{7.1e}$$

Note that the names of the wind directions are according to the computational domain. This can be different with the physical wind directions.
An overbar notation is used to indicate the mean of two nodes:

$$\overline{\zeta_E} := \frac{\zeta_E + \zeta_C}{2}, \quad \overline{\zeta_N} := \frac{\zeta_N + \zeta_C}{2}, \quad \overline{\zeta_W} := \frac{\zeta_W + \zeta_C}{2}, \quad \text{and} \quad \overline{\zeta_S} := \frac{\zeta_S + \zeta_C}{2}. \tag{7.2}$$

For the other variables, the overbar notation is used similarly. Note that because of the equidistant grid, these are central averages on the edges of the finite volumes.
To illustrate the finite volume method used, it will be written out for some terms of the model equations. Let's consider an arbitrary volume Ω_{mn} with boundary Γ_{mn}, centered around an internal grid point. The discretization of the boundaries will be done in Section 7.2.1.
Term without spatial derivatives will be evaluated as

$$\iint_{\Omega_{mn}} \frac{\partial \zeta}{\partial t} \, dx \, dy \approx \frac{\partial \zeta_C}{\partial t} \Delta x \Delta y. \tag{7.3}$$

So these terms are approximated by the value in the center multiplied with the volume size.

Terms with a first order spatial derivative are rewritten with Gauss's divergence theorem[6] and approximated with the central averages (7.2), for example

$$\begin{aligned}\iint_{\Omega_{mn}} \nabla\cdot(\zeta\mathbf{U})\;dx\,dy &= \int_{\Gamma_{mn}} \zeta\mathbf{U}\cdot\mathbf{n}\;ds \\ &\approx \overline{\zeta_E}\,\overline{U_E}\,\Delta y + \overline{\zeta_N}\,\overline{V_N}\,\Delta x - \overline{\zeta_W}\,\overline{U_W}\,\Delta y - \overline{\zeta_S}\,\overline{V_S}\,\Delta x. \end{aligned} \tag{7.4}$$

The second order spatial derivatives are also rewritten in terms of a boundary integral. The normal derivatives on the boundary are approximated by central differences. For example

$$\begin{aligned}\iint_{\Omega_{mn}} \nabla\cdot(h\,\nabla\varphi)\;dx\,dy &= \int_{\Gamma_{mn}} h\frac{\partial\varphi}{\partial n}\;ds \\ &\approx \left(\overline{h_E}\,\frac{\varphi_E-\varphi_C}{\Delta x}\right)\Delta y + \left(\overline{h_N}\,\frac{\varphi_N-\varphi_C}{\Delta y}\right)\Delta x \\ &\quad + \left(\overline{h_W}\,\frac{\varphi_W-\varphi_C}{\Delta x}\right)\Delta y + \left(\overline{h_S}\,\frac{\varphi_S-\varphi_C}{\Delta y}\right)\Delta x. \end{aligned} \tag{7.5}$$

In the second model equation (5.1b), the term $\mathbf{U}\cdot\nabla\varphi$ occurs. In its discretized form this term reads

$$\begin{aligned}\iint_{\Omega_{mn}} \mathbf{U}\cdot\nabla\varphi\;dx\,dy &\approx \left(\frac{1}{2}\left(\left(U\frac{\partial\varphi}{\partial x}\right)_E + \left(U\frac{\partial\varphi}{\partial x}\right)_W\right) + \frac{1}{2}\left(\left(V\frac{\partial\varphi}{\partial y}\right)_N + \left(V\frac{\partial\varphi}{\partial y}\right)_S\right)\right)\Delta x\Delta y \\ &\approx \frac{1}{2}\left(\overline{U_E}\frac{\varphi_E-\varphi_C}{\Delta x} - \overline{U_W}\frac{\varphi_W-\varphi_C}{\Delta x} + \overline{V_N}\frac{\varphi_N-\varphi_C}{\Delta y} - \overline{V_S}\frac{\varphi_S-\varphi_C}{\Delta y}\right)\Delta x\Delta y. \end{aligned} \tag{7.6}$$

The first approximation is done by first evaluating the integrand in the center, equivalent to the terms without spatial differencing. Then the averages $(U\frac{\partial\varphi}{\partial x})_C \approx \frac{1}{2}((U\frac{\partial\varphi}{\partial x})_E + (U\frac{\partial\varphi}{\partial x})_W)$ and $(V\frac{\partial\varphi}{\partial y})_C \approx \frac{1}{2}((V\frac{\partial\varphi}{\partial y})_N + (V\frac{\partial\varphi}{\partial y})_S)$ are applied. For the second approximation in Equation (7.6), use has been made of central differencing.

[6] $\int_\Omega \nabla\cdot\mathbf{u}\;d\Omega = \int_\Gamma \mathbf{u}\cdot\mathbf{n}\;d\Gamma$, for a domain Ω with boundary Γ and the unit outward normal $\mathbf{n}$.

Now the Hamiltonian system (5.1) can be approximated for the internal elements by

$$
\begin{aligned}
\Delta x\Delta y\frac{\partial \zeta_C}{\partial t} + \frac{1}{2}\,\overline{U_E}\,\Delta y\,\zeta_E + \frac{1}{2}\,\overline{V_N}\,\Delta x\,\zeta_N - \frac{1}{2}\,\overline{U_W}\,\Delta y\,\zeta_W - \frac{1}{2}\,\overline{V_S}\,\Delta x\,\zeta_S \\
+\left(\frac{1}{2}\,\overline{U_E}\,\Delta y + \frac{1}{2}\,\overline{V_N}\,\Delta x - \frac{1}{2}\,\overline{U_W}\,\Delta y - \frac{1}{2}\,\overline{V_S}\,\Delta x\right)\zeta_C \\
+\frac{\Delta y}{\Delta x}\,\overline{h_E}\,\varphi_E + \frac{\Delta x}{\Delta y}\,\overline{h_N}\,\varphi_N + \frac{\Delta y}{\Delta x}\,\overline{h_W}\,\varphi_W + \frac{\Delta x}{\Delta y}\,\overline{h_S}\,\varphi_S \\
-\left(\frac{\Delta y}{\Delta x}\,\overline{h_E} + \frac{\Delta x}{\Delta y}\,\overline{h_N} + \frac{\Delta y}{\Delta x}\,\overline{h_W} + \frac{\Delta x}{\Delta y}\,\overline{h_S}\right)\varphi_C \\
-\frac{\Delta y}{\Delta x}\,\overline{h_E}\,\overline{\mathcal{D}_{0E}}\,\psi_E - \frac{\Delta x}{\Delta y}\,\overline{h_N}\,\overline{\mathcal{D}_{0N}}\,\psi_N - \frac{\Delta y}{\Delta x}\,\overline{h_W}\,\overline{\mathcal{D}_{0W}}\,\psi_W - \frac{\Delta x}{\Delta y}\,\overline{h_S}\,\overline{\mathcal{D}_{0S}}\,\psi_S \\
+\left(\frac{\Delta y}{\Delta x}\,\overline{h_E}\,\overline{\mathcal{D}_{0E}} + \frac{\Delta x}{\Delta y}\,\overline{h_N}\,\overline{\mathcal{D}_{0N}} + \frac{\Delta y}{\Delta x}\,\overline{h_W}\,\overline{\mathcal{D}_{0W}} + \frac{\Delta x}{\Delta y}\,\overline{h_S}\,\overline{\mathcal{D}_{0S}}\right)\psi_C &= 0, \quad (7.7a)\\
\Delta x\Delta y\frac{\partial \varphi_C}{\partial t} + \frac{1}{2}\overline{U_E}\Delta y\,\varphi_E + \frac{1}{2}\overline{V_N}\Delta x\,\varphi_N - \frac{1}{2}\overline{U_W}\Delta y\,\varphi_W - \frac{1}{2}\overline{V_S}\Delta x\,\varphi_S \\
-\left(\frac{1}{2}\overline{U_E}\Delta y + \frac{1}{2}\overline{V_N}\Delta x - \frac{1}{2}\overline{U_W}\Delta y - \frac{1}{2}\overline{V_S}\Delta x\right)\varphi_C + \Delta x\Delta y\, g\,\zeta_C &= 0, \quad (7.7b)\\
\frac{\Delta y}{\Delta x}\,\overline{h_E}\,\overline{\mathcal{D}_{0E}}\,\varphi_E + \frac{\Delta x}{\Delta y}\,\overline{h_N}\,\overline{\mathcal{D}_{0N}}\,\varphi_N + \frac{\Delta y}{\Delta x}\,\overline{h_W}\,\overline{\mathcal{D}_{0W}}\,\varphi_W + \frac{\Delta x}{\Delta y}\,\overline{h_S}\,\overline{\mathcal{D}_{0S}}\,\varphi_S \\
-\left(\frac{\Delta y}{\Delta x}\,\overline{h_E}\,\overline{\mathcal{D}_{0E}} + \frac{\Delta x}{\Delta y}\,\overline{h_N}\,\overline{\mathcal{D}_{0N}} + \frac{\Delta y}{\Delta x}\,\overline{h_W}\,\overline{\mathcal{D}_{0W}} + \frac{\Delta x}{\Delta y}\,\overline{h_S}\,\overline{\mathcal{D}_{0S}}\right)\varphi_C \\
-\frac{\Delta y}{\Delta x}\,\overline{\mathcal{N}_{0E}}\,\psi_E - \frac{\Delta x}{\Delta y}\,\overline{\mathcal{N}_{0N}}\,\psi_N - \frac{\Delta y}{\Delta x}\,\overline{\mathcal{N}_{0W}}\,\psi_W - \frac{\Delta x}{\Delta y}\,\overline{\mathcal{N}_{0S}}\,\psi_S \\
+\left(\frac{\Delta y}{\Delta x}\,\overline{\mathcal{N}_{0E}} + \frac{\Delta x}{\Delta y}\,\overline{\mathcal{N}_{0N}} + \frac{\Delta y}{\Delta x}\,\overline{\mathcal{N}_{0W}} + \frac{\Delta x}{\Delta y}\,\overline{\mathcal{N}_{0S}}\right)\psi_C + \Delta x\Delta y\mathcal{M}_{0C}\,\psi_C &= 0, \quad (7.7c)
\end{aligned}
$$

the stencil for the discretized Hamiltonian system.

7.2.1 Discretization of the boundaries

In Chapter 6, the boundary condition have been derived. For both the closed and open boundaries, the boundary condition (6.4) is given as a flux on the boundary. The discretization of the model equations has been done with the finite volume method, in which the volume integral was rewritten to a boundary integral. The specified fluxes on the boundary can straightforwardly be applied in this way.
Note that the closed boundaries follow from the bottom profile. When $h < 0$ at a grid point, it is on dry land. So the contour line $h = 0$ gives the closed boundary.

7.3 Time integration

In Section 7.2, the spatial discretization of the model equations (5.1) has been performed with the finite volume method. On node (i, j), the discretized versions of the variables ζ, φ and ψ are given by ζ_{ij}, φ_{ij} and ψ_{ij}. A one dimensional ordering gives vectors $\vec{\zeta}$, $\vec{\varphi}$ and $\vec{\psi}$ containing the values of the variables on all nodes. For the following derivations, it is not necessary to

specify the ordering of nodes. The spatial discretization can be written as

$$\frac{d}{dt}\begin{bmatrix} \vec{\zeta} \\ \vec{\varphi} \\ \mathbf{0} \end{bmatrix} + \begin{bmatrix} S_{\zeta\zeta} & S_{\zeta\varphi} & S_{\zeta\psi} \\ S_{\varphi\zeta} & S_{\varphi\varphi} & S_{\varphi\psi} \\ S_{\psi\zeta} & S_{\psi\varphi} & S_{\psi\psi} \end{bmatrix} \begin{bmatrix} \vec{\zeta} \\ \vec{\varphi} \\ \vec{\psi} \end{bmatrix} = \mathbf{0} \tag{7.8}$$

This system can be rewritten as

$$\dot{\mathbf{q}} = L\,\mathbf{q} + \mathbf{f}, \tag{7.9a}$$
$$S\vec{\psi} = \mathbf{b} \tag{7.9b}$$

with $\mathbf{q} = \begin{bmatrix} \vec{\zeta} \\ \vec{\varphi} \end{bmatrix}$ and $\dot{\mathbf{q}}$ its time derivative. The matrix $L = -\begin{bmatrix} S_{\zeta\zeta} & S_{\zeta\varphi} \\ S_{\varphi\zeta} & S_{\varphi\varphi} \end{bmatrix}$ is the spatial discretization matrix and $\mathbf{f} = -\begin{bmatrix} S_{\zeta\psi}\vec{\psi} \\ S_{\varphi\psi}\vec{\psi} \end{bmatrix}$. In Equation (7.9b), we have $S = S_{\psi\psi}$ and $\mathbf{b} = -S_{\psi\varphi}\vec{\varphi}$, because $S_{\psi\zeta}$ equals zero.

The state at time t_n is denoted by $\mathbf{q}^n := \mathbf{q}(t_n)$. For given states $\mathbf{q}^n, \vec{\psi}^n$ and $\mathbf{q}^{n-1}, \vec{\psi}^{n-1}$, the state $\mathbf{q}^{n+1}$ is calculated by applying the leapfrog method to Equation (7.9a). Subsequently, the linear system $S\vec{\psi}^{n+1} = \mathbf{b}^{n+1}$ is solved. The leapfrog method (see [16]) will be derived in this section. Solving the linear system (7.9b) is the main topic of the research for this thesis. Methods to solve this linear system will be discussed in Chapter 8.

The variable time step Δt is given by $\Delta t_+ := t_{n+1} - t_n$ and $\Delta t_- := t_n - t_{n-1}$. A Taylor-series expansion gives

$$\mathbf{q}^{n+1} = \mathbf{q}^n + \Delta t_+ \dot{\mathbf{q}}^n + \frac{1}{2}\Delta t_+^2 \ddot{\mathbf{q}}^n + \frac{1}{6}\Delta t_+^3 \dddot{\mathbf{q}}^n + \mathcal{O}(\Delta t_+^4), \tag{7.10a}$$
$$\mathbf{q}^{n-1} = \mathbf{q}^n - \Delta t_- \dot{\mathbf{q}}^n + \frac{1}{2}\Delta t_-^2 \ddot{\mathbf{q}}^n - \frac{1}{6}\Delta t_-^3 \dddot{\mathbf{q}}^n + \mathcal{O}(\Delta t_-^4). \tag{7.10b}$$

To get rid of the second order derivative, we multiply the second equation by $\left(-\frac{\Delta t_+^2}{\Delta t_-^2}\right)$ and add the equations:

$$\begin{aligned} \mathbf{q}^{n+1} - \frac{\Delta t_+^2}{\Delta t_-^2}\mathbf{q}^{n-1} = {} & \mathbf{q}^n + \Delta t_+ \dot{\mathbf{q}}^n + \frac{1}{2}\Delta t_+^2 \ddot{\mathbf{q}}^n + \frac{1}{6}\Delta t_+^3 \dddot{\mathbf{q}}^n + \mathcal{O}(\Delta t_+^4) \\ & - \frac{\Delta t_+^2}{\Delta t_-^2}\mathbf{q}^n + \frac{\Delta t_+^2}{\Delta t_-}\dot{\mathbf{q}}^n - \frac{1}{2}\Delta t_+^2 \ddot{\mathbf{q}}^n + \frac{1}{6}\Delta t_+^2 \Delta t_- \dddot{\mathbf{q}}^n + \mathcal{O}(\Delta t_+^2 \Delta t_-^2). \end{aligned} \tag{7.11}$$

We introduce a new variable $\beta := \frac{\Delta t_+}{\Delta t_-}$, denoting the change in time step. Let's assume that the time steps have the same order of magnitude, so $\mathcal{O}(\Delta t_+) = \mathcal{O}(\Delta t_-)$ and $\beta = \mathcal{O}(1)$. Then, the above equation reads

$$\mathbf{q}^{n+1} = \beta^2 \mathbf{q}^{n-1} + (1-\beta^2)\mathbf{q}^n + \Delta t_+(1+\beta)\dot{\mathbf{q}}^n + \mathcal{O}(\Delta t_\pm^3), \tag{7.12}$$

giving an expression of $\mathbf{q}^{n+1}$ in terms of $\mathbf{q}^{n-1}$, $\mathbf{q}^n$ and $\dot{\mathbf{q}}^n$. Equation (7.12) is used as the numerical time scheme for solving Equation (7.9a) [16]. Note that it is an explicit scheme and depends on the two previous time steps (a so-called multistep method).

For equidistant time steps, i.e., $\beta = 1$, Equation (7.12) reduces to

$$\dot{\mathbf{q}}^n = \frac{\mathbf{q}^{n+1} - \mathbf{q}^{n-1}}{2\Delta t} + \mathcal{O}(\Delta t^2), \tag{7.13}$$

the so-called *leapfrog method*. For the test equation $\dot{y} = \lambda y$, the leapfrog method is stable only for $\lambda\Delta t \in [-i, i]$ (see [35, 9]). The stability on the imaginary axis is important, because many wave models have purely imaginary eigenvalues [45].

7.3.1 Initial conditions

The variational Boussinesq model is written in the time domain. Initial conditions for the basic variables have to be specified. The discretized system of the model equations (7.9) shows that the variable ψ is decoupled from the system with the time derivative. Therefore, only initial conditions for ζ and φ are needed. The initial state of ψ can then be calculated with Equation (7.9b).
Equation (7.9a), which contains a time derivative of ζ and ϕ, is discretized with the leapfrog scheme (7.12), which is a two-step method. This requires an extra initial condition at time t_{-1}. To prevent this extra condition, one can also choose to use Euler Forward or another one-step method for calculating the variables at t_1.
An easy choice for the initial conditions is a zero initial state, that is,

$$\zeta^0_{(m,n)} = 0, \qquad \zeta^{-1}_{(m,n)} = 0, \tag{7.14a}$$

$$\varphi^0_{(m,n)} = 0, \qquad \varphi^{-1}_{(m,n)} = 0, \tag{7.14b}$$

$$\psi^0_{(m,n)} = 0, \qquad \psi^{-1}_{(m,n)} = 0, \tag{7.14c}$$

for all nodes (m, n). These initial conditions will be used in wave model.

8 Numerical linear algebra

Numerical discretization of the model equations results in a system of linear equations for the discrete model variables. One of the discrete mode equations is of the form $S\vec{\psi} = \mathbf{b}$, with S the discretization matrix. The vector $\vec{\psi}$ has to be solved, it contains the value of $\psi(x, y)$ in every grid point and has therefore a length of $N_x \cdot N_y$, with N_x and N_y denoting the number of nodes in the x- and y-direction. Because the dimensions can become large, solving this linear system will be computational expensive. To reduce the computational effort, numerous numerical methods for solving linear systems efficiently are available in the literature, see [32] and [42].
The method used in the model for solving $S\vec{\psi} = \mathbf{b}$ is the *conjugate gradients method* [12]. This is an often used method for symmetric positive definite matrices, which can be targeted to specific systems by applying a preconditioner. The conjugate gradient method will be derived in Section 8.3. In Section 8.4 the concept of preconditioning will be explained.

8.1 Properties of the matrix

The linear system which has to be solved is

$$S\vec{\psi} = \mathbf{b} \tag{8.1}$$

as defined in Equation (7.9b), with $\vec{\psi}$ a vector containing the values of ψ on all grid points. The matrix S and vector $\mathbf{b}$ are filled according to the numerical discretization as presented in Chapter 7. More precisely, the linear system is equivalent to Equation (7.7c). In this section some properties of the matrix S are presented. These properties will determine the method for solving this linear system.
The size of the matrix S is determined by the computational domain (see Section 7.1). The rectangular grid consists of N_x and N_y nodes in the x- and y-direction, respectively. Therefore the matrix S has dimension $N_xN_y \times N_xN_y$. Due to the requirement of real-time calculation and the limitations of current technology, typical values for a grid of 5×5 m and a timestep of $\Delta t = 0.05$ s are $N_x = 200$ and $N_y = 100$, so the matrix is of the order $20\,000 \times 20\,000$. In the future, the dimensions of the domain have to be much larger.
To assess the properties of the matrix, we consider Equation (7.7c), which determines

$$\begin{bmatrix} 0 & -\frac{\Delta x}{\Delta y}\overline{\mathcal{N}_{0N}} & 0 \\ -\frac{\Delta y}{\Delta x}\overline{\mathcal{N}_{0W}} & \frac{\Delta y}{\Delta x}\overline{\mathcal{N}_{0E}} + \frac{\Delta x}{\Delta y}\overline{\mathcal{N}_{0N}} + \frac{\Delta y}{\Delta x}\overline{\mathcal{N}_{0W}} + \frac{\Delta x}{\Delta y}\overline{\mathcal{N}_{0S}} + \Delta x\Delta y\mathcal{M}_{0C} & -\frac{\Delta y}{\Delta x}\overline{\mathcal{N}_{0E}} \\ 0 & -\frac{\Delta x}{\Delta y}\overline{\mathcal{N}_{0S}} & 0 \end{bmatrix}, \tag{8.2}$$

a five-point stencil. When using a lexicographical ordering[7] of the grid points, this leads to a *pentadiagonal*[8] matrix. For a horizontal numbering, half the *bandwidth*[9] will be N_x, for vertical numbering N_y.

[7] A lexicographical ordering is a horizontal or a vertical numbering

[8] A pentadiagonal matrix is a matrix with non-zero elements on five diagonals and zero elements on all other diagonals.

[9] For b half of the bandwith of a matrix A, we have that all elements outside the b^{th} diagonal are zero, i.e., $(a_{ij})_{|i-j|>b} = 0$.

With the overbar as defined in Equation (7.2) and the fact that the parameters are positive (see Chapter 5), one can see that the center term in the stencil (8.2) is positive and the outer elements are negative. Observe that the sum of all elements is positive. Therefore, the matrix is *diagonally dominant*, i.e., the center element is at least the sum of the absolute values of the outer elements. Because we have $h > 0$, it holds that $\mathcal{M}_{0C} > 0$ and therefore the matrix S is strictly diagonally dominant.
With stencil (8.2), we see that the center element has a value of $\mathcal{O}(1 + h^2)$, while the sum of the absolute values of the outer elements is $\mathcal{O}(1)$; it is assumed that $\mathcal{O}(\Delta x) = \mathcal{O}(\Delta y) = \mathcal{O}(h)$. So for small mesh sizes h, the diagonally dominance of the matrix is not very strongly.
Because the matrix is strictly diagonally dominant with positive elements on the main diagonal, Gershgorin's circle theorem (see [46]) states that the real part of all eigenvalues are strictly positive. So S has no eigenvalue zero and is therefore nonsingular, i.e., the inverse S^{-1} exists.
The diagonal elements are strictly positive ($S_{ii} > 0$), while the outer elements are negative ($S_{ij} \leq 0$). Because of the strictly diagonally dominance, the matrix is an *M-matrix*[10].
To verify the symmetry of the matrix, one has to compare the outer diagonals with each other. For example, the contribution of the west neighbour on node (i, j) has to be the same as the contribution of the east neigbour on node $(i-1, j)$. So for symmetry, we need $\left(-\frac{\Delta y}{\Delta x}\overline{\mathcal{N}_{0W}}\right)_{i,j} = \left(-\frac{\Delta y}{\Delta x}\overline{\mathcal{N}_{0E}}\right)_{i-1,j}$. Because of the central differencing of the fluxes in the finite volume method, the mesh sizes are the same. With the overbar notation (7.2), the requirement is rewritten as $(\mathcal{N}_{0W} + \mathcal{N}_{0C})_{i,j} = (\mathcal{N}_{0C} + \mathcal{N}_{0E})_{i-1,j} \Leftrightarrow (\mathcal{N}_{0,i-1,j} + \mathcal{N}_{0,i,j}) = (\mathcal{N}_{0,i-1,j} + \mathcal{N}_{0,i,j})$, which is clearly satisfied. Applying the same reasoning to the other neighbours shows that the matrix is symmetric, i.e., $S^T = S$. Because the boundary conditions (6.4) are given in the form of fluxes, the discretization (see Section 7.2.1) of this will not destroy the symmetry of the matrix.
Note that the matrix is real-valued and the Euclidean inner product can therefore be used. Because of symmetry, we have $\langle S\mathbf{x}, \mathbf{x}\rangle_2 = \mathbf{x}^T S^T \mathbf{x} = \mathbf{x}^T S\mathbf{x} = \langle \mathbf{x}, S\mathbf{x}\rangle_2$. This shows that the matrix S is *self adjoint* in the Euclidean inner product space. The self adjointness of the matrix implies that all eigenvalues are real valued [23].
From Gershgorin's circle theorem we already know that the real part of the eigenvalues are all strictly positive. From symmetry, the eigenvalues are real. So all eigenvalues λ_i of S satisfy $\lambda_i > 0$. This yields that S is *positive definite*, i.e., $\mathbf{x}^T S\mathbf{x} > 0 \ \forall \mathbf{x} \neq \mathbf{0}$.
Summarizing, the matrix S is real valued, pentadiagonal, strictly diagonally dominant, symmetric and positive definite.

Dry nodes On dry land (for example coasts, islands and piers), the model equations do not have to be solved. However, there are grid points on this part of the domain. These nodes occur in the linear system $S\psi = \mathbf{b}$. The variable ψ is taken zero on dry land. In the matrix S, on the row corresponding to a dry node, the elements on the outer diagonals are taken zero and on the main diagonal the elements are one. A zero right hand side will give

[10] A matrix A is an M-matrix if $a_{ii} > 0$ and $a_{ij} \leq 0$ for $i \neq j$ and $A^{-1} \geq 0$ [32]. The last requirement is equivalent with $\rho(I - D^{-1}A) < 1$, with $\rho(\cdot)$ the spectral radius (maximum absolute eigenvalue) and D the diagonal of A. When A is strictly diagonally dominant, the diagonal of $I - D^{-1}A$ is zero and the sum of the absolute values of the outer elements is smaller than one. With Gershgorin's circle theorem, the spectral radius is smaller than one.

$\psi_{mn} = 0$ for dry nodes (m, n). The properties of the matrix as presented before still apply for the matrix adapted to dry nodes.

8.2 Krylov subspace methods

A class of linear solvers are the Krylov subspace methods. These are iterative methods, i.e., for a starting vector ψ^0, the method generates a series of vectors $\psi^1, \psi^2, \ldots$ converging to $\psi = S^{-1}\mathbf{b}$. Let's consider *Richardson's iteration* [31], given by

$$\begin{aligned} \psi^{i+1} &= \psi^i + \tau(\mathbf{b} - S\psi^i) \\ &= (I - \tau S)\psi^i + \tau\mathbf{b}, \end{aligned} \tag{8.3}$$

for $i = 0, 1, 2, \ldots$ and a parameter $\tau \neq 0$. For an invertible matrix M, the linear system $S\psi = \mathbf{b}$ is equivalent with

$$M^{-1}S\psi = M^{-1}\mathbf{b}, \tag{8.4}$$

with M^{-1} called the *left preconditioner* of S. Matrix M has to be chosen such that system (8.4) is easier to solve than the original one. Now apply Richardson's iteration for this preconditioned system with $\tau = 1$, so

$$\begin{aligned} \psi^{i+1} &= \psi^i + M^{-1}(\mathbf{b} - S\psi^i) \\ &= (I - M^{-1}S)\psi^i + M^{-1}\mathbf{b}. \end{aligned} \tag{8.5}$$

This recursion is the basis for a class of methods called the *basic iterative methods*. For a converging ψ^i, the term $\mathbf{r}^i := \mathbf{b} - S\psi^i$ will go to zero and is therefore called the *residual*. For the initial vector ψ^0, the first iterates are

$$\begin{aligned} \psi^1 &= \psi^0 + M^{-1}\mathbf{r}^0, \\ \psi^2 &= \psi^1 + M^{-1}\mathbf{r}^1 = \psi^0 + M^{-1}\mathbf{r}^0 + M^{-1}(\mathbf{b} - S(\psi^0 + M^{-1}\mathbf{r}^0)), \\ &= \psi^0 + 2M^{-1}\mathbf{r}^0 - (M^{-1}S)M^{-1}\mathbf{r}^0 \\ \psi^3 &= \psi^2 + M^{-1}(\mathbf{b} - S\psi^2) \\ &= \psi^0 + 2M^{-1}\mathbf{r}^0 - (M^{-1}S)M^{-1}\mathbf{r}^0 + M^{-1}(\mathbf{b} - S(\psi^0 + 2M^{-1}\mathbf{r}^0 - (M^{-1}S)M^{-1}\mathbf{r}^0)) \\ &= \psi^0 + 3M^{-1}\mathbf{r}^0 - 3(M^{-1}S)M^{-1}\mathbf{r}^0 + (M^{-1}S)^2M^{-1}\mathbf{r}^0, \\ &\vdots \end{aligned}$$

From this, we can see that

$$\psi^i \in \psi^0 + \text{span}\left\{M^{-1}\mathbf{r}^0, (M^{-1}S)(M^{-1}\mathbf{r}^0), \ldots, (M^{-1}S)^{i-1}(M^{-1}\mathbf{r}^0)\right\}. \tag{8.6}$$

The subspace

$$K^i(S, \mathbf{r}^0) := \text{span}\left\{\mathbf{r}^0, S\mathbf{r}^0, \ldots, S^{i-1}\mathbf{r}^0\right\} \tag{8.7}$$

is called the *Krylov space* of dimension i for matrix S and residual $\mathbf{r}^0$. For basic iterative methods we have $\psi^i \in \psi^0 + K^i(M^{-1}S, M^{-1}\mathbf{r}^0)$ for the preconditioned system (8.4). Linear solvers using this property are called *Krylov subspace methods* [32].
With a similar derivation, we have $\psi^i = (I - M^{-1}S)^i\psi^0 + \sum_{j=0}^{i-1}(I - M^{-1}S)^j M^{-1}\mathbf{b}$. One can recognize the geometric series and for $i \to \infty$ the Neumann series. When the series converges, we have $\psi^\infty = (M^{-1}S)^{-1}M^{-1}\mathbf{b} = S^{-1}\mathbf{b}$, which is the solution of the linear system.

8.3 Conjugate gradients method

For the linear system $S\psi = b$, we search for estimates $\psi^k \in K^k\{S, b\}$ such that $||\psi - \psi^k||$ is minimal. To get only one minimization problem at each iteration step, we search in the direction of $p^k \in K^k\{S, b\}$. The next estimate will then be $\psi^{k+1} = \psi^k + \alpha_k p^k$. The constant α_k has to be calculated such that the error $||\psi - \psi^{k+1}||$ is minimal in some norm. We have $||\psi - \psi^k - \alpha_k p^k|| = ||\psi - \psi^k|| - 2\alpha_k\langle\psi - \psi^k, p^k\rangle + \alpha_k^2||p^k||$. Equating the derivative w.r.t. α_k to zero gives

$$\alpha_k = \frac{\langle p^k, \psi - \psi^k\rangle}{\langle p^k, p^k\rangle}. \tag{8.8}$$

Let's consider the Euclidean inner product space, i.e., $\langle x, y\rangle_2 = x^T y$. The parameter α_k contains the term $(p^k)^T\psi$, which cannot be calculated because ψ is unknown. Therefore, the Euclidean norm cannot be used. Note that $S\psi$ is known: $S\psi = b$. Hence, consider

$$\langle x, y\rangle_S = x^T S y, \tag{8.9}$$

the *matrix inner product*. Observe that this is equivalent with $\langle x, Sy\rangle_2$. If S is spd[11], this inner product is well defined. Because $S(\psi - \psi^k) = S\psi - S\psi^k = b - S\psi^k = r^k$, we have

$$\alpha_k = \frac{\langle p^k, r^k\rangle_2}{\langle p^k, Sp^k\rangle_2}, \tag{8.10}$$

which can be calculated readily, because the residual r^k depends on the previous estimate ψ^k. The method is now defined, with exception of the choice of the search vector p^k. In fact, there are many options, all leading to different schemes. The choice of CG is

$$p^{k+1} = r^{k+1} + \frac{\langle r^{k+1}, r^{k+1}\rangle_2}{\langle r^k, r^k\rangle_2} p^k. \tag{8.11}$$

This choice of p^k satisfies the property

$$\langle p^k, p^j\rangle_S = 0 \quad \text{for } j = 0, 1, 2, \ldots, k-1, \tag{8.12}$$

which says that the search directions are orthogonal to each other in the matrix inner product. Summarizing, the CG-method is given by the following iteration scheme:

$$r^0 = b - S\psi^0, \tag{8.13a}$$
$$p^0 = r^0, \tag{8.13b}$$
$$\alpha_i = \frac{\langle r^i, r^i\rangle}{\langle p^i, Sp^i\rangle}, \tag{8.13c}$$
$$\psi^{i+1} = \psi^i + \alpha_i p^i, \tag{8.13d}$$
$$r^{i+1} = r^i - \alpha_i S p^i, \tag{8.13e}$$
$$\beta_i = \frac{\langle r^{i+1}, r^{i+1}\rangle}{\langle r^i, r^i\rangle}, \tag{8.13f}$$
$$p^{i+1} = r^{i+1} + \beta_i p^i, \tag{8.13g}$$

with $\langle\cdot,\cdot\rangle$ the Euclidean inner product.

[11] A spd matrix is an abbreviation for a symmetric positive definite matrix.

Convergence The convergence of the CG-method depends on the *spectrum*[12]. An upperbound of the error is given by (see [42])

$$||\psi^i - \psi||_S \quad \leq \quad 2\left(\frac{\sqrt{\frac{\lambda_{\max}}{\lambda_{\min}}}-1}{\sqrt{\frac{\lambda_{\max}}{\lambda_{\min}}}+1}\right)^i ||\psi^0 - \psi||_S, \tag{8.14}$$

with $\lambda_{\max}$ and $\lambda_{\min}$ the maximal and minimal eigenvalue of S, respectively. So the convergence mainly depends on $\frac{\lambda_{\max}}{\lambda_{\min}}$, the *spectral condition number*. From the upperbound (8.14), one can see that the smaller the condition number, the smaller the upperbound and therefore probably a faster convergence. A small condition number is equivalent to a narrow spectrum, i.e., the smallest and largest eigenvalue are close to each other.
With the upperbound (8.14) and the stencil (8.2), we can make an estimate of the convergence of CG for the system $S\psi = \mathbf{b}$. Because S is spd, Gershgorin's circle theorem gives $0 < \Delta x\Delta y\mathcal{M}_{0C} \leq \lambda_{\min} \leq \lambda_{\max} \leq \Delta x\Delta y\mathcal{M}_{0C} + 2\frac{\Delta y}{\Delta x}(\overline{\mathcal{N}_{0E}} + \overline{\mathcal{N}_{0W}}) + 2\frac{\Delta x}{\Delta y}(\overline{\mathcal{N}_{0N}} + \overline{\mathcal{N}_{0S}})$. From this, one has $\frac{\lambda_{\max}}{\lambda_{\min}} \leq 1 + \frac{2\frac{\Delta y}{\Delta x}(\overline{\mathcal{N}_{0E}}+\overline{\mathcal{N}_{0W}})+2\frac{\Delta x}{\Delta y}(\overline{\mathcal{N}_{0N}}+\overline{\mathcal{N}_{0S}})}{\Delta x\Delta y\mathcal{M}_{0C}}$. When assuming $\mathcal{O}(\Delta x) = \mathcal{O}(\Delta y) = \mathcal{O}(h)$, with h a characteristic mesh size (not the water depth as has been used earlier), we have

$$\frac{\lambda_{\max}}{\lambda_{\min}} \quad = \quad \mathcal{O}\left(h^{-2}\right). \tag{8.15}$$

From this expression for the condition number, the convergence of the CG-method can be derived[13]. When writing the upper bound of the error (8.14) as $||\psi_i - \psi||_S \leq 2C^i||\psi_0 - \psi||_S$, we get $C = 1 - \mathcal{O}(h)$. We can thus conclude that for a constant error, the number of iterations is inversely proportional to the mesh size[14].

8.4 Preconditioners

Equation (8.14) shows that the convergence behavior of the CG-method depends on the spectrum of S. Applying the CG-method to the preconditioned system (8.4), the convergence depends on the eigenvalues of $M^{-1}S$. Because the matrix M can be chosen arbitrarly, the convergence can be adjusted by choosing M such that the spectrum of $M^{-1}S$ is more favorable than the spectrum of S. From Equation (8.14) one can see that the spectral condition number should be close to one, so the smallest and largest eigenvalue should be close to each other. Loosely speaking, the preconditioner M has to be chosen such that it looks like S and the system $M\mathbf{x} = \mathbf{b}$ is relatively easy to solve [26].

Given a preconditioner M, the left preconditioned system is given by

$$M^{-1}S\psi = M^{-1}b. \tag{8.16}$$

[12]The spectrum of a matrix is the set of all eigenvalues.

[13]We have $\left(\sqrt{\frac{\lambda_{\max}}{\lambda_{\min}}}-1\right)\left(\sqrt{\frac{\lambda_{\max}}{\lambda_{\min}}}+1\right)^{-1} = \left(1-(\frac{\lambda_{\max}}{\lambda_{\min}})^{-\frac{1}{2}}\right)\left(1+(\frac{\lambda_{\max}}{\lambda_{\min}})^{-\frac{1}{2}}\right)^{-1} = \left(1-(\frac{\lambda_{\max}}{\lambda_{\min}})^{-\frac{1}{2}}\right)\cdot\left(1-(\frac{\lambda_{\max}}{\lambda_{\min}})^{-\frac{1}{2}}+(\frac{\lambda_{\max}}{\lambda_{\min}})^{-1}+\mathcal{O}\big((\frac{\lambda_{\max}}{\lambda_{\min}})^{-\frac{3}{2}}\big)\right) = 1 - 2(\frac{\lambda_{\max}}{\lambda_{\min}})^{-\frac{1}{2}} + \mathcal{O}\big((\frac{\lambda_{\max}}{\lambda_{\min}})^{-1}\big) = 1 - \mathcal{O}\big((\frac{\lambda_{\max}}{\lambda_{\min}})^{-\frac{1}{2}}\big)$, where the geometric series is used.

[14]Let's look at the error after k iterations: $||\psi_k - \psi||_S < \epsilon$, then $2C^k < \delta$, with $0 < \delta < 1$. Solving $2C^k = \delta$ for a given δ yields $k = \ln(\frac{1}{2}\delta)/\ln(C) = \ln(\frac{1}{2}\delta)/(-2h-2h^2+\mathcal{O}(h^3)) = -\ln(\frac{1}{2}\delta)\frac{1}{h}\frac{1}{1+h+\mathcal{O}(h^2)} = \mathcal{O}(\frac{1}{h})$, for $C = 1-2h$. So for a constant error, we need $\mathcal{O}(\frac{1}{h})$ iterations.

In order to preserve the symmetry, one can also consider

$$P^{-1}SP^{-T}(P^T\psi) \quad = \quad P^{-1}\mathbf{b}, \tag{8.17}$$

called the centrally preconditioned system. Note that for an spd S, the preconditioned matrix $P^{-1}SP^{-T}$ also spd[15].
In the first case, the CG-method is applied to the matrix $M^{-1}S$, while in the second case it is $P^{-1}SP^{-T}$. For a preconditioner

$$M = PP^T, \tag{8.18}$$

these two matrices are in general not the same. However, they have the same spectrum[16] and the preconditioned CG-algorithm is equivalent and therefore these two preconditioned systems will be used interchangeable.
In the wave model, several choices of preconditioner are implemented. These preconditioners are explained in subsequent chapters: diagonal scaling in Chapter 9, relaxed incomplete Cholesky in Chapter 10, and repeated red-black in Chapter 11.

8.5 Preconditioned conjugate gradient method

The conjugate gradient method is used for solving the system $S\psi = b$ and is given by the recursive algorithm (8.13). Applying the CG-method to a preconditioned system (8.17) will give a slightly different algorithm, called the *preconditioned conjugate gradient method*. Writing out the PCG-algorithm will give the pseudo code as given in Table 1. By rearranging some terms, two different implementation can be obtained. The methods are equivalent, but the differences may lead to easier implementations of the preconditioners.
The two PCG-implementations shown in Table 1 have more variables than in the original recursion (8.13), such as σ and ρ. These are dummy variables to simplify the implementation. The main difference between the two implementations of PCG are the expressions with the preconditioner P. The first choice considers the statement $q = P^{-1}SP^{-T}p$ and the second the statements $z = (PP^T)^{-1}r$ and $q = Sp$. It depends on the type of preconditioner which choice is best. For some preconditioners it is advantageous to calculate the preconditioned matrix $P^{-1}SP^{-T}$ once and use it every iteration. For other preconditioners it is better to decouple the preconditioner and the original matrix and use the statements $z = (PP^T)^{-1}r$ and $q = Sp$ separately. In the second version both the residual $\langle r_i, r_i\rangle = ||r_i||_2^2$ as well as the preconditioned residual $\langle r_i, z_i\rangle = ||r_i||_{M^{-1}}^2$ is available, whereas in the first version, only $\langle P^{-1}r_i, P^{-1}r_i\rangle == ||r_i||_{M^{-1}}^2$ is available. The availability of both residual norms gives more freedom in the choice of termination criterium, as will be seen in Section 14.9.2.

[15]Because S symmetric, $\tilde{S}^T = (P^{-1}SP^{-T})^T = P^{-TT}S^TP^{-T} = P^{-1}SP^{-T} = \tilde{S}$, so $\tilde{S}$ also symmetric. The matrix S is positive definite, i.e., $\langle y, Sy\rangle > 0$ for all y. Then $\langle x, \tilde{S}x\rangle = \langle x, P^{-1}SP^{-T}x\rangle = \langle P^{-T}x, SP^{-T}x\rangle = \langle y, Sy\rangle > 0$, so $\tilde{S}$ also positive definite.

[16]For nonsingular matrices A, B the following identity holds: $\det(AB) = \det(A)\det(B)$. Then we have $\det(\lambda I - P^{-1}SP^{-T}) = 0 \Leftrightarrow \det(\lambda PP^T - S) = 0 \Leftrightarrow \det(\lambda I - M^{-1}S) = 0$, so the same eigenvalues λ for the centrally and left preconditioned matrix.

CHOICE 1	
Input	
x	Start vector
r	Right hand side
S	Matrix
P	Preconditioner
ϵ	Tolerance
PCG-algorithm	
$r = r - Sx$	Residual
solve $r = P^{-1}r$	Map residual
$\rho_{\text{new}} = r \cdot r$	
$x = P^T x$	Map solution
$i = 0$	Iteration number
while $\rho_{\text{new}} > \epsilon^2$	Termination criterium
$i = i + 1$	
if $i = 1$	
$p = r$	Search vector
else	
$\beta = \frac{\rho_{\text{new}}}{\rho_{\text{old}}}$	
$p = r + \beta p$	Search vector
end	
solve $q = P^{-1}SP^{-T}p$	Mapped search vector
$\sigma = p \cdot q$	
$\alpha = \frac{\rho_{\text{new}}}{\sigma}$	
$x = x + \alpha p$	Update iterate
$r = r - \alpha q$	Update residual
$\rho_{\text{old}} = \rho_{\text{new}}$	
$\rho_{\text{new}} = r \cdot r$	
end	
solve $x = P^{-T}x$	Map solution
Output	
x	

CHOICE 2	
Input	
x	Start vector
r	Right hand side
S	Matrix
P	Preconditioner
ϵ	Tolerance
PCG-algorithm	
$r = r - Sx$	Residual
solve $z = (PP^T)^{-1}r$	Map residual
$\rho_{\text{old}} = \rho_{\text{new}}$	
$\rho_{\text{new}} = r \cdot z$	
$i = 0$	Iteration number
while $\rho_{\text{new}} > \epsilon^2$	Termination criterium
$i = i + 1$	
if $i = 1$	
$p = z$	Search vector
else	
$\beta = \frac{\rho_{\text{new}}}{\rho_{\text{old}}}$	
$p = z + \beta p$	Search vector
end	
$q = Sp$	Mapped search vector
$\sigma = p \cdot q$	
$\alpha = \frac{\rho_{\text{new}}}{\sigma}$	
$x = x + \alpha p$	Update iterate
$r = r - \alpha q$	Update residual
solve $z = (PP^T)^{-1}r$	Map residual
$\rho_{\text{old}} = \rho_{\text{new}}$	
$\rho_{\text{new}} = r \cdot z$	
end	
Output	
x	

Table 1: Two different implementations of the preconditioned conjugate gradient method.

9 Diagonally scaled conjugate gradient method

In Section 8.3 it has been explained that the conjugate gradient method will be used for solving the system of linear equations (8.1). To improve the convergence of the CG-method, it can be applied to the preconditioned system (8.17). The PCG-method will only have a better performance than the CG-method if the preconditioner is chosen correctly. In this section, one choice of preconditioner is explained, namely *diagonal scaling*, which is also known as the *Jacobi preconditioner*.

9.1 Structure of the preconditioner

As explained in Section 8.4, the preconditioner has to be chosen such it looks like the original matrix and the preconditioning step is relatively easy to solve. A straightforward choice is taking as preconditioner M the diagonal elements of the original matrix S. The preconditioner M is therefore a diagonal matrix and its inverse is easily calculated: $m_{ii}^{-1} = 1/m_{ii}$. This satisfies the requirement of an easy to solve preconditioned residual. The other requirement, stating that the preconditioner should look like the original matrix, is also satisfied, especially for when S is a sparse and diagonally dominant matrix.
The preconditioned matrix $\widetilde{S} = P^{-1}SP^{-T}$ is easilly calculated, because P a diagonal matrix. Moreover, the diagonal of $\widetilde{S}$ will equal one. The matrix-vector multiplication $y = \widetilde{S}x$ is therefore quite easily done. This makes choice 1 of the two PCG-implementations in Table 1 the best choice.

9.2 Spectrum of the diagonally scaled system

The main purpose of applying a preconditioner is to improve the convergence of the CG-method. As can be seen in Equation (8.14), the convergence depends on the spectrum of the matrix, i.e., the smaller the spectral condition number $\lambda_{\max}/\lambda_{\min}$, the faster the convergence will be. This is achieved by choosing the preconditioner $M = PP^T$ such that $P^{-1}SP^{-T}$ has a more favorable spectrum than S. Whether this is the case for diagonal scaling is investigated in this section for some test problems.
Looking at the test problem of an open sea with a constant depth of $30m$ (see Section 13.1), the spectra of the matrix S and the diagonally scaled matrix $P^{-1}SP^{-T}$ are calculated explicitly with `Matlab` and are shown in Figure 1.
Because the preconditioner equals the diagonal of the matrix S, the diagonal of the preconditioned matrix is equal to one. Then, Gershgorin's circle theorem states that the eigenvalues are centered around one, which can be seen in figure 1. The spectral condition number is for both matrices 1.2537 and therefore, the preconditioned CG-method is not likely to converge faster than the unpreconditioned CG-method.
The reason that the condition number does not change is that a flat bottom is considered. When the water depth and the shape parameter κ are equal for all nodes, the center element of the stencil (8.2) is also equal. This implies a constant diagonal of S and therefore $M = cI$, for some constant c, yielding $\frac{\lambda^S_{\max}}{\lambda^S_{\min}} = \frac{c\lambda^{M^{-1}S}_{\max}}{c\lambda^{M^{-1}S}_{\min}} = \frac{\lambda^{M^{-1}S}_{\max}}{\lambda^{M^{-1}S}_{\min}}$, which shows that the spectral condition numbers of S and $M^{-1}S$ are equal.

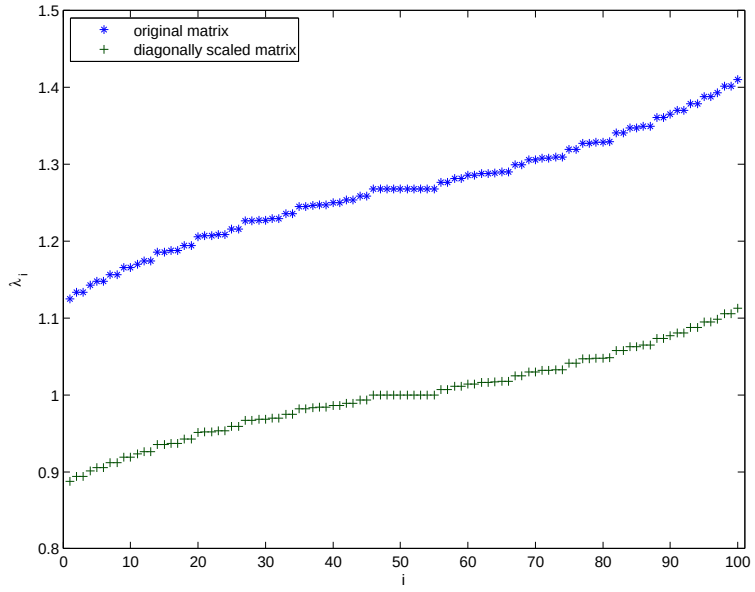

Figure 1: Spectrum of the original matrix S and the diagonally scaled matrix $P^{-1}SP^{-T}$ at an open sea with constant depth of $30m$ and a grid of 10×10 nodes.

A flat bathymetry does not show a difference in condition number. Therefore, we will consider an open sea with a deep trench in the bottom, as shown in Figure 2.

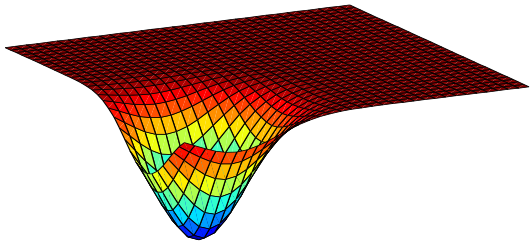

Figure 2: Bathymetry of an open sea with depth of $50\,m$ outside a trench of $100\,m$ depth and a grid of 32×32 nodes.

Because the depth varies per node, the diagonal elements of S are not constant anymore. The spectrum of the preconditioned matrix is therefore not only shifted to one, but it has really a different shape than the unpreconditioned spectrum, as seen in Figure 3. The main difference is in the large eigenvalues. Observe that the spectral condition number drops from $6.35 \cdot 10^5$ to 394.3, which is a large reduction.

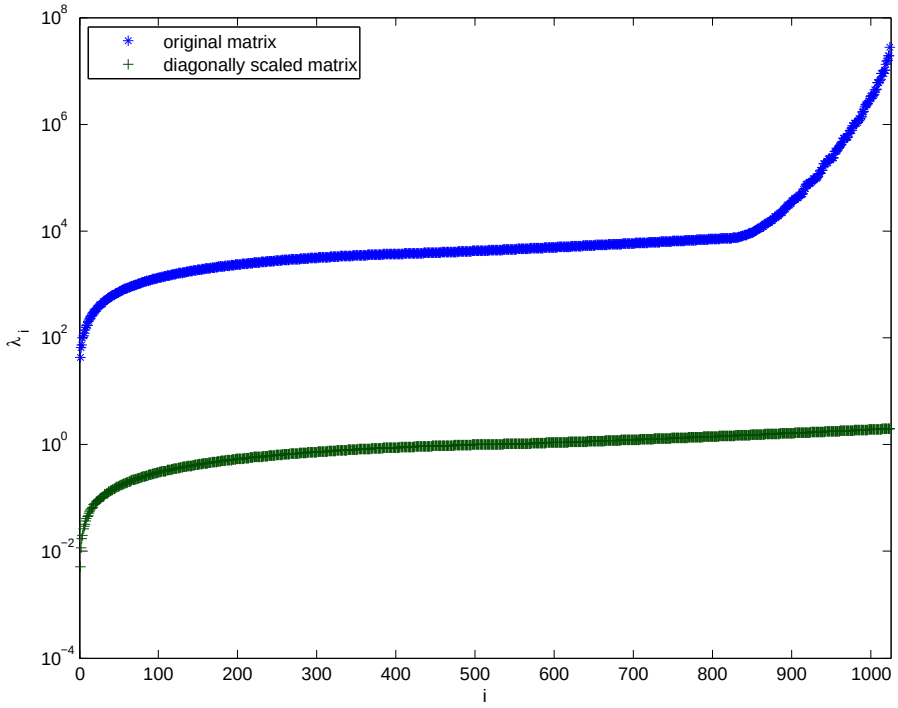

Figure 3: Spectrum of the original matrix S and the diagonally scaled matrix $P^{-1}SP^{-T}$ at the bathymetry of Figure 2.

In this section we have seen that choosing the preconditioner as the diagonal of the original matrix may lead to better convergence of the CG-method. For bathymetries with a constant depth, the condition number is not reduced. However, for varying depth profiles, diagonal scaling can reduce the condition number considerably.

10 Relaxed incomplete Cholesky decomposition

A well-known direct method for solving a linear system of equations is the LU-decomposition. Most square matrices A can be factorized in the form $A = LU$, with L a lower triangular and U an upper triangular matrix[17]. Solving a linear system $Ax = b$ can be done relatively easy by first solving $Ly = b$ and then $Ux = y$. If the matrix A is symmetric, one can find an upper triangular matrix U such that $U = L^T$. The decomposition then becomes $A = LL^T$, called the *complete Cholesky decomposition*. Because a symmetric matrix is used in the wave model, we will look at Cholesky decompositions in this section, but most results also apply for the nonsymmetric LU-decompositions.
As preconditioner, these decompositions are not usefull, because $(LL^T)^{-1}S = I$. However, one can find lower triangular matrices $K \approx L$ which are easier to solve, but still $KK^T \approx A$, and are therefore usefull as preconditioners. Calculating these matrices K can be done with an *incomplete Cholesky decomposition*, as explained in this section.

10.1 Sparsity pattern

The system to solve in the wave model is Equation (8.1): $S\psi = b$. With a lexicographical numbering of nodes, the matrix S is pentadiagonal with values on the main, first and p-th outer diagonal. This sparsity pattern[18] of S is shown in the left picture of Figure 4. The results of Figure 4 are made with `Matlab` on a grid with 10 nodes in both directions.

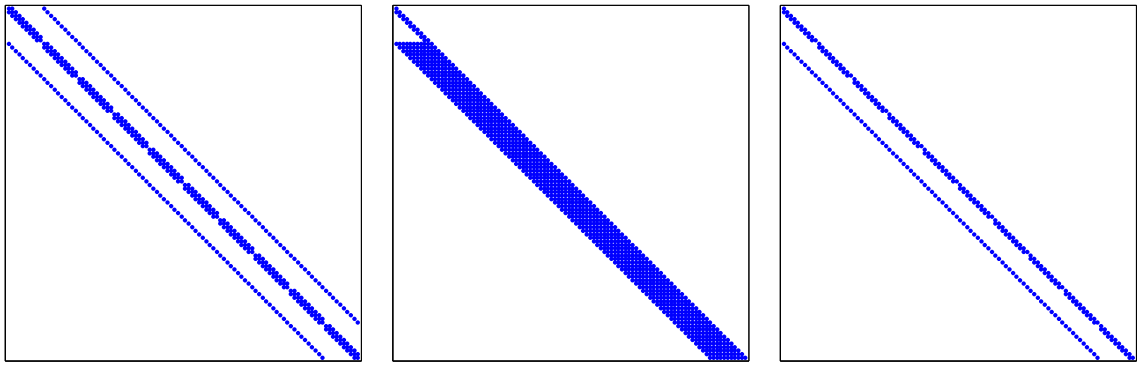

Figure 4: Sparsity pattern of S (left), L (center) and K (right).

The Cholesky factor L of S is the lower triangular matrix as shown in the center picture of Figure 4. The lower bandwith of L is the same as for S, but where S has zeros inside the band, L has nonzero elements inside the band; this is called *fill-in*.
Solving a system $Lx = b$ can be done relatively easy with forward substitution [10]. The more nonzero elements L has, the more computational effort is needed. To reduce the computation time, a predefined sparsity pattern is imposed on the Cholesky matrix. A common choice is to take the sparsity pattern of S itself. So, the decomposition of S has to result in a lower triangular matrices K which approximates S, i.e., $KK^T \approx S$ and $K + K^T$ has to have the same sparsity pattern as S. This is called *incomplete Cholesky decomposition* and the resulting lower triangular matrix K is shown in the right picture of Figure 4.

[17] A matrix L is lower triangular when $L_{ij} = 0 \; \forall \, i < j$, and U upper triangular when $U_{ij} = 0 \; \forall \, i > j$.
[18] The sparsity pattern of a matrix A is the pattern of all nonzero elements of A.

A similar choice is to take the sparsity pattern of S with m extra fill-in diagonals, called the IC(m) decomposition [26]. This results in an incomplete decomposition which approximates the matrix more accurate, but needs more computation time to solve. This method will not be used in the following, so $m = 0$ is taken.
Because $LL^T = S$, the sparsity patterns are the same. When using the incomplete Cholesky decomposition, we have $KK^T \approx S$, which has a different sparsity pattern. In the right picture of Figure 5, we see that there are nonzero elements on the $(p-1)$–st left and right diagonal, called the *fill-in elements*.

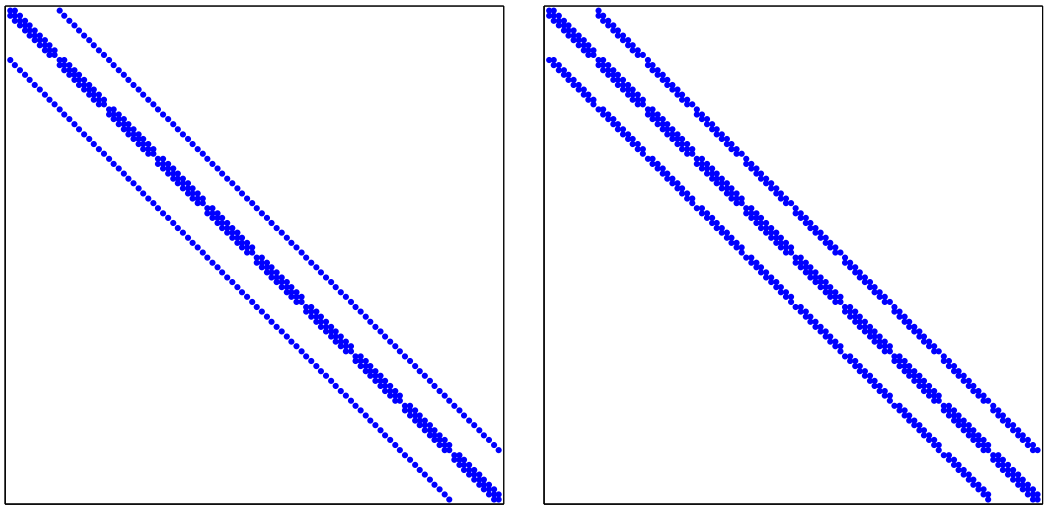

Figure 5: Sparsity pattern of LL^T (left) and KK^T (right).

Because of the sparse structure of K and $KK^T \approx S$, it is a good choice for use as a preconditioner in the CG-method.

10.2 Calculating the incomplete Cholesky decomposition

In the previous section it has been stated that the Cholesky decomposition is of the form LL^T, but one may also consider the decomposition

$$M = (D + L)D^{-1}(D + L^T), \tag{10.1}$$

with D a diagonal matrix and L a strictly lower triangular matrix, so $D_{ij} = 0 \ \forall \ i \neq j$ and $L_{ij} = 0 \ \forall \ i \leq j$. For the incomplete Cholesky decomposition, we have an additional requirement of the sparsity pattern, namely $L_{ij} = 0$ for $i \neq j + 1$ and $i \neq j + p$.
Similar to KK^T in Figure 5, the matrix M has a sparsity pattern with seven nonzero diagonals: the five nonzero diagonals of S and the fill-in elements next to the outer band. To get an algorithm for calculating the incomplete Cholesky decomposition, the matrix M will be written out for these seven diagonals. Detailed derivations are shown in Appendix H.
For the outer diagonals of M, we have $M_{i,i-p} = L_{i,i-p}$, $M_{i,i-1} = L_{i,i-1}$, $M_{i,i+1} = L^T_{i,i+1}$ and $M_{i,i+p} = L^T_{i,i+p}$. To apply M as preconditioner for S, we want $M \approx S$. Therefore, the matrix L of the incomplete Cholesky decomposition (10.1) is chosen equal to the strictly lower triangular part of S. This implies, amongst others, that the matrix L does not have to be calculated, because it is directly given by the known matrix S.

The main diagonal is now given by

$$M_{ii} = S^2_{i,i-p} D^{-1}_{i-p,i-p} + S^2_{i,i-1} D^{-1}_{i-1,i-1} + D_{ii}, \tag{10.2}$$

and the fill-in elements are given by

$$M_{i,i-p+1} = S_{i,i-p} D^{-1}_{i-p,i-p} S_{i-p,i-p+1}, \tag{10.3a}$$
$$M_{i,i+p-1} = S_{i,i-1} D^{-1}_{i-1,i-1} S_{i-1,i+p-1}. \tag{10.3b}$$

The diagonal matrix D of the decomposition has to be specified by a condition on these three diagonals. One choice is to ignore the fill-in elements and use $M_{ii} = S_{ii}$, which yields the recursive formula[19]

$$D_{ii} = S_{ii} - \frac{S^2_{i,i-p}}{D_{i-p,i-p}} - \frac{S^2_{i,i-1}}{D_{i-1,i-1}}. \tag{10.4}$$

This model for choosing L and D in Equation (10.1) is called the *incomplete Cholesky decomposition* (IC)[20] [26].
Another choice is to require equal row sums for S and M: $\sum_{j=1}^{n} S_{ij} = \sum_{j=1}^{n} M_{ij}$. Because of the choice of L to be equal to the lower part of S, we get $S_{ii} = M_{ii} + M_{i,i-p+1} + M_{i,i+p-1}$, yielding the recursive formula

$$D_{ii} = S_{ii} - \frac{S_{i,i-p}(S_{i,i-p} + S_{i-p,i-p+1})}{D_{i-p,i-p}} - \frac{S_{i,i-1}(S_{i,i-1} + S_{i-1,i+p-1})}{D_{i-1,i-1}}, \tag{10.5}$$

called the *modified incomplete Cholesky decomposition* (MIC) [11].
These two models for D can be combined by requiring

$$S_{ii} = M_{ii} + \omega \left(M_{i,i-p+1} + M_{i,i+p-1} \right), \tag{10.6}$$

with $0 \leq \omega \leq 1$ a constant, called the *relaxation parameter*. This gives

$$D_{ii} = S_{ii} - \frac{S_{i,i-p}(S_{i,i-p} + \omega S_{i-p,i-p+1})}{D_{i-p,i-p}} - \frac{S_{i,i-1}(S_{i,i-1} + \omega S_{i-1,i+p-1})}{D_{i-1,i-1}}, \tag{10.7}$$

the *relaxed incomplete Cholesky decomposition* (RIC) [4].
Note that for $\omega = 0$, the RIC-decomposition (10.7) reduces to the IC-decomposition (10.4) and for $\omega = 1$, it reduces to the MIC-decomposition (10.5).

10.3 Eisenstat's implementation

The incomplete Cholesky decomposition (10.1) results in a matrix M for which $M \approx S$ holds and will therefore be used as a preconditioner for the conjugate gradient method. During the CG-algorithm, one has to solve th system $y = M^{-1}Sx$, or a similar one. In this section, it will be explained that for incomplete Cholesky decompositions this equation can be solved efficiently with *Eisenstat's implementation* [8].

[19] For $i = 1$, use $D_{ii} = S_{ii}$, and for $1 < i < p$ use $D_{ii} = S_{ii} - \frac{S^2_{i,i-1}}{D_{i-1,i-1}}$.

[20] Confusingly, this is the same term as the general incomplete Cholesky decomposition as used before.

In order to enable Eisenstat's implementation, the matrix M will be scaled:

$$\begin{aligned}\widetilde{M} &= D^{-\frac{1}{2}} M D^{-\frac{1}{2}} \\ &= D^{-\frac{1}{2}}(D+L)D^{-1}(D+L^T)D^{-\frac{1}{2}} \\ &= (I + D^{-\frac{1}{2}} L D^{-\frac{1}{2}})(I + D^{-\frac{1}{2}} L^T D^{-\frac{1}{2}}) \\ &= (I+\widetilde{L})(I+\widetilde{L}^T), \end{aligned} \tag{10.8}$$

with the matrices D and L according to Equation (10.1) and the scaled matrix $\widetilde{L} := D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$. By the definition of the incomplete Cholesky decomposition in Section 10.2, the matrix L equals the stritly lower triangular part of S. Then, with D_S the diagonal of S, we have $S = L + D_S + L^T$ and also $\widetilde{S} = \widetilde{L} + D_{\widetilde{S}} + \widetilde{L}^T$ with $D_{\widetilde{S}} = D^{-\frac{1}{2}} D_S D^{-\frac{1}{2}}$ the diagonal of $\widetilde{S}$.
Let's observe the following derivation:

$$\begin{aligned}(I+\widetilde{L})^{-1}\widetilde{S}(I+\widetilde{L}^T)^{-1} &= (I+\widetilde{L})^{-1}(\widetilde{L} + D_{\widetilde{S}} + \widetilde{L}^T)(I+\widetilde{L}^T)^{-1} \\ &= (I+\widetilde{L})^{-1}\left((I+\widetilde{L}) + (D_{\widetilde{S}} - 2I) + (I+\widetilde{L}^T)\right)(I+\widetilde{L}^T)^{-1} \\ &= (I+\widetilde{L}^T)^{-1} + (I+\widetilde{L})^{-1}(D_{\widetilde{S}} - 2I)(I+\widetilde{L}^T)^{-1} + (I+\widetilde{L})^{-1}. \end{aligned}$$

The implementation of the matrix-vector product $\mathbf{y} = (I+\widetilde{L})^{-1}\widetilde{S}(I+\widetilde{L}^T)^{-1}\mathbf{x}$ for a given $\mathbf{x}$ can therefore be written as

$$\mathbf{z} := (I+\widetilde{L}^T)^{-1}\mathbf{x}, \tag{10.9a}$$

$$\mathbf{y} = (I+\widetilde{L})^{-1}\left(\mathbf{x} + (D_{\widetilde{S}} - 2I)\mathbf{z}\right) + \mathbf{z}, \tag{10.9b}$$

called Eisenstat's implementation.
Compared to the straightforward way of solving $\mathbf{y} = (I+\widetilde{L})^{-1}\widetilde{S}(I+\widetilde{L}^T)^{-1}\mathbf{x}$, i.e.,

$$\begin{aligned}\mathbf{v} &:= (I+\widetilde{L}^T)^{-1}\mathbf{x}, \\ \mathbf{w} &:= \widetilde{S}\mathbf{x}, \\ \mathbf{y} &= (I+\widetilde{L})^{-1}\mathbf{w}, \end{aligned}$$

this saves one matrix-vector multiplication $y = \widetilde{S}x$ at the expense of three vector updates. These three vector updates are the calculation of $\mathbf{a} := (\widetilde{D}_{\widetilde{S}} - 2I)\mathbf{z}$, $\mathbf{b} := \mathbf{x} + \mathbf{a}$ and $\mathbf{y} = \mathbf{c} + \mathbf{z}$, where it is assumed that the matrix $D_{\widetilde{S}} - 2I$ is already calculated. In general, matrix-vector multiplications are expensive and therefore Eisenstat's implementation saves some computation time.

10.4 The RICCG-method

In previous sections, we have seen that for symmetric matrices S, an incomplete Cholesky decomposition is given by Equation (10.1): $M = (D + L)D^{-1}(D + L^T)$, with L the strictly lower triangular part of the matrix S and the diagonal matrix D is calculated according to Equation (10.7).
In the derivation of the preconditioned CG-method, it is assumed that the preconditioner has the symmetric form of $M = PP^T$. In order to get this form, the matrix D is scaled to I. This is done by scaling S with D:

$$\widetilde{S} := D^{-\frac{1}{2}} S D^{-\frac{1}{2}}. \tag{10.10}$$

The incomplete Cholesky decomposition of $\widetilde{S}$ is then given by $\widetilde{M} = (I+\widetilde{L})(I+\widetilde{L}^T)$, as derived in Equation (10.8). With

$$P := (I + \widetilde{L}) \tag{10.11}$$

the preconditioner $\widetilde{M} = PP^T$ has the symmetric form as wanted for the PCG-algorithm.
The statement $q = P^{-1}\widetilde{S}P^{-T}p$ from choice 1 in Table 1 can now be solved efficiently with Eisenstat's implementation (10.9).

Summarizing, the whole linear problem is first scaled with D, as in Equation (10.10). Then, the PCG-method is applied with the relaxed incomplete Cholesky decomposition (10.11) as preconditioner.

10.5 Spectral condition number of the RIC-decomposition

In Section 8.3 it has been shown that the convergence of the CG-method depends mainly on the spectral condition number of S, which is the ratio of largest and smalles eigenvalue. For the PCG-method, the convergence depends on the spectral condition number of the preconditioned matrix $P^{-1}SP^{-T}$. As preconditioner, we will consider the relaxed incomplete Cholesky decomposition (10.7).
For a characteristic mesh size h, it can be shown that the condition number of the IC-preconditioner is $\mathcal{O}(h^{-1})$ and for the MIC-preconditioner $\mathcal{O}(h^{-2})$ [41]. Therefore, it is expected that MICCG will converge faster, especially for larger problems.
In Figure 6, we see that the spectra of IC and MIC differ considerably. For small values of the relaxation parameter ω, the large eigenvalues are clustered at one, but the small eigenvalues are widely spread out. For large ω, the small eigenvalues are approximately one, but the large eigenvalues are not clustered. This is similar to what one might expect from the theoretical results of $\lambda_{\min}^{\mathrm{IC}} = \mathcal{O}(h^2)$ and $\lambda_{\max}^{\mathrm{IC}} = \mathcal{O}(1)$ for the IC-preconditioner; and $\lambda_{\min}^{\mathrm{MIC}} = \mathcal{O}(1)$ and $\lambda_{\max}^{\mathrm{MIC}} = \mathcal{O}(h^{-1})$ for the MIC-preconditioner [41].

The spectral condition number of the original matrix S is 160.7, which is reduced a lot with the RIC-preconditioner, as shown in Figure 7. We also see that the larger ω, the smaller the spectral condition number

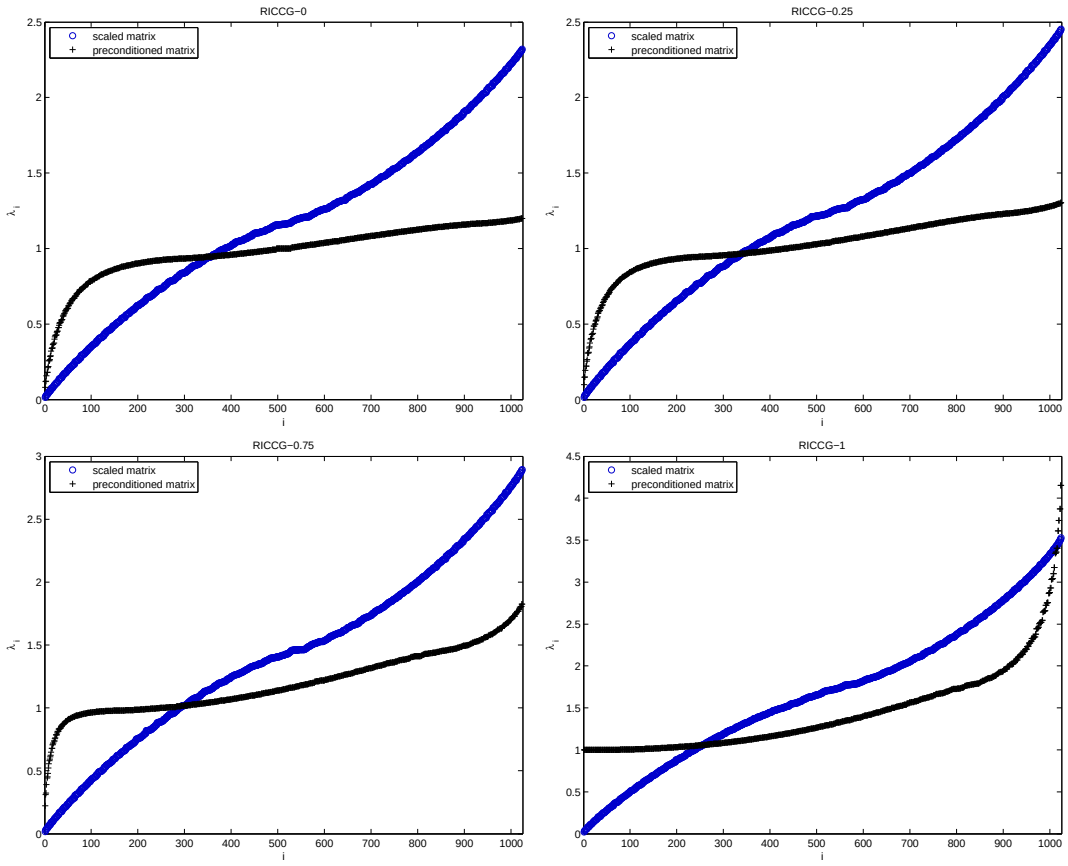

Figure 6: Spectrum of the scaled matrix $\widetilde{S}$ and preconditioned matrix $P^{-1}\widetilde{S}P^{-T}$ of RIC-preconditioner with $\omega = 0$ (upper left), $\omega = 0.25$ (upper right), $\omega = 0.75$ (lower left) and $\omega = 1$ (lower right) at a grid of 32×32 nodes.

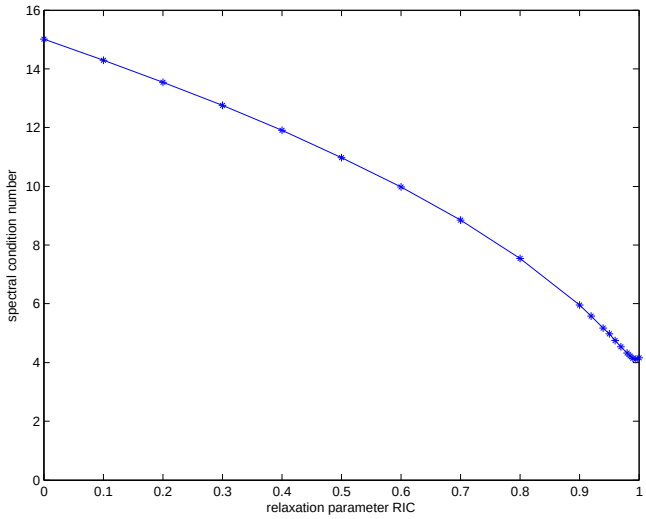

Figure 7: Spectral condition number of RIC at an open sea with 32×32 nodes.

In Section 8.3 it has been derived that a spectral condition number of $\mathcal{O}(h^{-2})$ yields $\mathcal{O}(h^{-1})$ CG-iterations. Similar, the condition number of $\mathcal{O}(h^{-1})$ for the MICCG-method will give $\mathcal{O}(h^{-\frac{1}{2}})$ CG-iterations. In order to verify these theoretical results, the methods are applied to the test problem of an open sea (see Section 13.1) with a constant depth of $50\,m$. The number of CG-iterations is averaged over 1000 timesteps, as given in Table 2 (see Section 14 for more results).

Grid	$\omega = 0$	$\omega = 0.5$	$\omega = 1$
100×100	7.822	7.637	7.399
200×200	16.357	14.387	11.450
400×400	32.895	28.184	17.144
800×800	67.233	57.611	25.758

Table 2: Mean number of RICCG-iterations at open sea.

The order of the number of iterations can be estimated with Richardson's extrapolation[21]. Because the method used three different grids and four grids are given in Table 2, we can apply Richardson's extrapolation two times. For $\omega = 0$, we get the values -0.954 and -1.054; for $\omega = 0.5$ we get -1.031 and -1.093; and for $\omega = 1$ we get -0.491 and -0.597. These values are estimates of the order, and are in accordance with the theoretical orders, i.e., $\mathcal{O}(h^{-1})$ for ICCG and $\mathcal{O}(h^{-\frac{1}{2}})$ for MICCG. In between those extreme choices, we have an estimate of $\mathcal{O}(h^{-1})$ for RIC-0.5, which is the same as for IC.
As shown in Section 8.3, these orders of the number of CG-iterations are linked with the order of the spectral condition number. To be precise, with the results of Richardson's extrapolation, the spectral condition number of ICCG is estimated by $\mathcal{O}(h^{-2})$ and for MICCG $\mathcal{O}(h^{-1})$, which equal the theoretical orders.

[21] The number of CG-iterations is given by $N_h^{\text{CG}} = \mathcal{O}(h^\alpha) = \beta h^\alpha$, with h the characteristic mesh size and β some constant. With results from three different grids, one can estimate the exponent α. For a given number of iterations at grids with mesh size h, $2h$ and $4h$, one has $\frac{\beta(4h)^\alpha - \beta(2h)^\alpha}{\beta(2h)^\alpha - \beta h^\alpha} = \frac{(4^\alpha - 2^\alpha)h^\alpha}{(2^\alpha - 1^\alpha)h^\alpha} = \frac{2^\alpha(2^\alpha - 1)}{2^\alpha - 1} = 2^\alpha$. Then α is estimated with the 2-log of the given fraction.

11 Repeated red-black preconditioner

In previous sections two preconditioners for the CG-method have been described. Diagonal scaling in Section 9 and relaxed incomplete Cholesky in Section 10. In this section another preconditioner will be derived, it uses a renumbering of the grid points according to a red-black ordering. With an elimination process, an equivalent stencil is obtained on a coarser grid, which is simplified by lumping some outer elements. This procedure is repeated several times, until the grid is coarse enough to use a direct solver on it. The resulting preconditioning method is called the *repeated red-black preconditioner*, or RRB [6, 3, 28, 39].

11.1 Repeated elimination on red-black grids

As explained in Section 7.1, the variational Boussinesq model is discretized on a rectangular grid. In previous preconditioners, a lexicographical numbering of the computational grid points is used. For the RRB-method, a red-black ordering will be taken, i.e., the red nodes are given by the points (x_{ij}, y_{ij}) with $i + j = 0 \pmod 2$, and the black nodes by $\mathbf{x}_{ij}$ with $i + j = 1 \pmod 2$. First the black points are numbered in lexicographical ordering and then the red points. With this checkerboard-like ordering and the 5-point stencil (8.2), black points only depend on red points and vice-versa.
The red nodes give a shifted grid, on which again a subdivision is performed. A second level grid is obtained by taking only the red nodes is odd rows and columns. It results in a rectangular grid with a mesh width two times larger than the first level grid. On the second level grid, a similar red-black subdivision can be made.

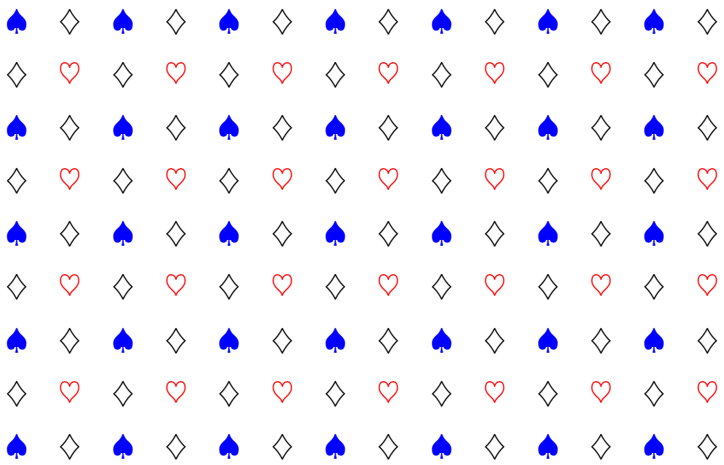

Figure 8: Domain with first level red-black (hearts and diamonds) and second level of blue nodes (spades).

With the ordering of nodes as in Figure 8, the linear system $S\psi = b$ (8.1) can be written as

$$\begin{bmatrix} D_\diamondsuit & S_{\diamondsuit\heartsuit} & S_{\diamondsuit\spadesuit} \\ S_{\heartsuit\diamondsuit} & D_\heartsuit & 0 \\ S_{\spadesuit\diamondsuit} & 0 & D_\spadesuit \end{bmatrix} \begin{bmatrix} \psi_\diamondsuit \\ \psi_\heartsuit \\ \psi_\spadesuit \end{bmatrix} = \begin{bmatrix} b_\diamondsuit \\ b_\heartsuit \\ b_\spadesuit \end{bmatrix}, \tag{11.1}$$

with D diagonal matrices. Because of the symmetric stencil (8.2), we have $S_{\heartsuit\diamondsuit} = S^T_{\diamondsuit\heartsuit}$ and $S_{\spadesuit\diamondsuit} = S^T_{\diamondsuit\spadesuit}$. Note that the size of $\psi_\diamondsuit$ is two times larger than $\psi_\heartsuit$ and $\psi_\spadesuit$.

The diagonal matrix $D_\diamondsuit^{-1}$ is easily computed. Therefore, the $\diamondsuit$-nodes can be decoupled with Gaussian elimination. This can be represented by the map $S = L_1 S_1 L_1^T$, i.e.,

$$\begin{bmatrix} I_\diamondsuit & 0 & 0 \\ S_{\heartsuit\diamondsuit} D_\diamondsuit^{-1} & I_\heartsuit & 0 \\ S_{\spadesuit\diamondsuit} D_\diamondsuit^{-1} & 0 & I_\spadesuit \end{bmatrix} \begin{bmatrix} D_\diamondsuit & 0 & 0 \\ 0 & D_\heartsuit - S_{\heartsuit\diamondsuit} D_\diamondsuit^{-1} S_{\diamondsuit\heartsuit} & -S_{\heartsuit\diamondsuit} D_\diamondsuit^{-1} S_{\diamondsuit\spadesuit} \\ 0 & -S_{\spadesuit\diamondsuit} D_\diamondsuit^{-1} S_{\diamondsuit\heartsuit} & D_\spadesuit - S_{\spadesuit\diamondsuit} D_\diamondsuit^{-1} S_{\diamondsuit\spadesuit} \end{bmatrix} \begin{bmatrix} I_\diamondsuit & D_\diamondsuit^{-1} S_{\diamondsuit\heartsuit} & D_\diamondsuit^{-1} S_{\diamondsuit\spadesuit} \\ 0 & I_\heartsuit & 0 \\ 0 & 0 & I_\spadesuit \end{bmatrix}.$$

Solving a linear system $S\psi = b$ can now be done with $L_1 S_1 L_1^T \psi = b$. Because L_1 is a lower triangular matrix, forward substitution is used. The matrix S_1 is decoupled and because $D_\diamondsuit$ diagonal, the $\diamondsuit$-subsystem of S_1 is easily solved. The subsystem of S_1 for the first level red nodes $\heartsuit$ and $\spadesuit$ is more difficult to solve. This so-called *Schur-complement* (see [7]) corresponds with a 9-point stencil

$$\begin{bmatrix} 0 & 0 & nn_1 & 0 & 0 \\ 0 & nw_1 & 0 & ne_1 & 0 \\ ww_1 & 0 & cc_1 & 0 & ee_1 \\ 0 & sw_1 & 0 & se_1 & 0 \\ 0 & 0 & ss_1 & 0 & 0 \end{bmatrix} \tag{11.2}$$

on the first level red nodes.
In order to simplify this stencil on the $\heartsuit$-nodes, the four most outer elements are lumped towards the main diagonal: $\widetilde{cc}_1 = cc_1 + ee_1 + nn_1 + ww_1 + ss_1$. This is equivalent with stating

$$(\widetilde{D}_\heartsuit)_{ii} = \sum_j \left(D_\heartsuit - S_{\heartsuit\diamondsuit} D_\diamondsuit^{-1} S_{\diamondsuit\heartsuit} \right)_{ij} \tag{11.3}$$

a diagonal matrix.
For the second level $\spadesuit$-nodes, the 9-point stencil (11.2) will stay the same. One can also choose to perform the same lumping procedure on the $\spadesuit$-nodes too, i.e., use $\widetilde{D}_\spadesuit$ similar to Equation (11.3). Both methods will result in a 9-point stencil (11.5) after the next elimination and therefore give the same structure. However, in [14] it is shown that for the Poisson equation, the first method, so lumping only the $\heartsuit$-nodes, has a smaller spectral condition number. Therefore, we will look at the first method only.
With the lumping procedure (11.3), the values $\psi_\heartsuit$ can explicitly be given by $\psi_\spadesuit$. Gaussian elimination of the $\heartsuit$-nodes then gives $\widetilde{S}_1 = L_2 S_2 L_2^T$ as

$$\begin{bmatrix} I_\diamondsuit & 0 & 0 \\ 0 & I_\heartsuit & 0 \\ 0 & -S_{\spadesuit\diamondsuit} D_\diamondsuit^{-1} S_{\diamondsuit\heartsuit} \widetilde{D}_\heartsuit^{-1} & I_\spadesuit \end{bmatrix} \begin{bmatrix} D_\diamondsuit & 0 & 0 \\ 0 & \widetilde{D}_\heartsuit & 0 \\ 0 & 0 & \widehat{S}_\spadesuit \end{bmatrix} \begin{bmatrix} I_\diamondsuit & 0 & 0 \\ 0 & I_\heartsuit & -\widetilde{D}_\heartsuit^{-1} S_{\heartsuit\diamondsuit} D_\diamondsuit^{-1} S_{\diamondsuit\spadesuit} \\ 0 & 0 & I_\spadesuit \end{bmatrix},$$

with

$$\widehat{S}_\spadesuit \;:=\; D_\spadesuit - S_{\spadesuit\diamondsuit} D_\diamondsuit^{-1} S_{\diamondsuit\spadesuit} - S_{\spadesuit\diamondsuit} D_\diamondsuit^{-1} S_{\diamondsuit\heartsuit} \widetilde{D}_\heartsuit^{-1} S_{\heartsuit\diamondsuit} D_\diamondsuit^{-1} S_{\diamondsuit\spadesuit}. \tag{11.4}$$

This matrix is given by a 9-point stencil

$$\begin{bmatrix} nw_2 & 0 & nn_2 & 0 & ne_2 \\ 0 & 0 & 0 & 0 & 0 \\ ww_2 & 0 & cc_2 & 0 & ee_2 \\ 0 & 0 & 0 & 0 & 0 \\ sw_2 & 0 & ss_2 & 0 & se_2 \end{bmatrix} \tag{11.5}$$

on the second level ♠-nodes.
As before, the 9-point stencil is simplified to a 5-point stencil by lumping the most outer elements towards the main diagonal, i.e., $\widetilde{cc}_2 = cc_1 + ne_2 + nw_2 + sw_2 + se_2$. The resulting matrix $\widetilde{S}_\spadesuit$ on the coarse grid is pentadiagonal and has the same properties as the original matrix S, i.e., $\widetilde{S}_\spadesuit$ is a symmetric positive definite M-matrix [6].

The procedure explained in this section is only one RRB iteration. It consists of an elimination $S = L_1 S_1 L_1^T$, lumping $S_1 = \widetilde{S}_1 + R_1$ and again an elimination $\widetilde{S}_1 = L_2 S_2 L_2^T$ and lumping $S_2 = \widetilde{S}_2 + R_2$. Combined, we have

$$\begin{aligned} S &= L_1 L_2 \widetilde{S}_2 L_2^T L_1^T + L_1 L_2 R_2 L_2^T L_1^T + L_1 R_1 L_1^T \\ &= LDL^T + R, \end{aligned} \tag{11.6}$$

with the lower triangular matrix

$$L := \begin{bmatrix} I_\diamondsuit & 0 & 0 \\ S_{\heartsuit\diamondsuit} D_\diamondsuit^{-1} & I_\heartsuit & 0 \\ S_{\spadesuit\diamondsuit} D_\diamondsuit^{-1} & -S_{\spadesuit\diamondsuit} D_\diamondsuit^{-1} S_{\diamondsuit\heartsuit} \widetilde{D}_\heartsuit^{-1} & I_\spadesuit \end{bmatrix} \tag{11.7}$$

and the block diagonal matrix

$$D := \begin{bmatrix} D_\diamondsuit & 0 & 0 \\ 0 & \widetilde{D}_\heartsuit & 0 \\ 0 & 0 & \widetilde{S}_\spadesuit \end{bmatrix}. \tag{11.8}$$

The matrix $R := L_1 L_2 R_2 L_2^T L_1^T + L_1 R_1 L_1^T$ contains the lumped elements. The first lumping procedure (11.3) yields the matrix $R_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & R_\heartsuit & 0 \\ 0 & 0 & 0 \end{bmatrix}$, with $R_\heartsuit = D_\heartsuit - S_{\heartsuit\diamondsuit} D_\diamondsuit^{-1} S_{\diamondsuit\heartsuit} - \widetilde{D}_\heartsuit$.
Because of the ones on the diagonal of L_1, we have $L_1 R_1 L_1^T = R_1$. Similar, we have $R_2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & R_\spadesuit \end{bmatrix}$, with $R_\spadesuit = \widehat{S}_\spadesuit - \widetilde{S}_\spadesuit$ and $L R_2 L^T = R_2$. The matrix $R = R_1 + R_2$ is then given by

$$R = \begin{bmatrix} 0 & 0 & 0 \\ 0 & R_\heartsuit & 0 \\ 0 & 0 & R_\spadesuit \end{bmatrix}. \tag{11.9}$$

The lower triangularity of L is due to the RRB ordering of the grid points. When using a lexicographical numbering, it will not have a triangular structure anymore. The same holds for the structures of D and R.
Because the coarse grid matrix $\widetilde{S}_\spadesuit$ has the same properties as S, but with a four times smaller size, one can choose to apply the same procedure again. After $k-1$ RRB iterations, we have $S = LDL^T + R$, with $D = \begin{bmatrix} D_k & 0 \\ 0 & S_k \end{bmatrix}$, a diagonal block matrix, with D_k a diagonal matrix and S_k a pentadiagonal matrix on level k. For a square domain with n^2 nodes, S_k has dimension $(n/2^{k-1})^2$, which can be much smaller than n^2. This decomposition is called the RRB-k method.
In [6] it has been proven that for a weakly diagonally dominant M-matrix, like S, the RRB-k method is well defined and the matrix LDL^T is also a weakly diagonally dominant M-matrix. Moreover, LDL^T is symmetric positive definite, for an spd S.

11.2 The RRB method as a preconditioner

The RRB-k method makes a decomposition $S = LDL^T + R$, with L a lower triangular matrix and D a block diagonal matrix. The matrix R contains the adjustments made during the lumping procedure. Similar to the scalar Cholesky decompositions in Section 10, the matrix LDL^T is a block incomplete Cholesky decomposition of S.
Because $LDL^T \approx S$ and $LDL^T\psi = b$ is relatively easy to solve, the matrix

$$M = LDL^T \tag{11.10}$$

from the RRB-k procedure can be used as a preconditioner.
During the PCG-method, $z = M^{-1}r$ has to be solved. This is done in three steps: $a = L^{-1}r$, $b = D^{-1}a$ and $z = L^{-T}b$. Because L lower triangular, calculating a is done with forward substitution and z with backward substitution. The block diagonal matrix D is scalar diagonal on all nodes except the nodes at the maximum level. The vector b is therefore straightforwardly calculated on most nodes. At the maximum level, a system $S_k b_k = a_k$ has to be solved. When the maximum level k is chosen such that it consists of only one node, it is easy to solve the coarse grid system, because it has dimension one. For smaller numbers of k, the system is given by a pentadiagonal symmetric matrix (similar to matrix (11.4)). A complete Cholesky decomposition is then used to solve the coarse grid system.

Convergence The purpose of applying a preconditioner is to improve the convergence of the CG-method. From Equation (8.14), we know that the convergence depends on the spectral condition number. For unpreconditioned CG, the spectral condition number depends quadratically on the characteristic mesh size h (see Equation (8.15)). In [6] the spectral condition number of the RRB-k preconditioner is analyzed for a Poisson equation with Dirichlet boundary conditions. Depending on the choice of k, the spectral condition number is between $\mathcal{O}(h^{-1})$ and $\mathcal{O}(1)$. In Section 11.5 we will elaborate more on the convergence properties of RRB-k.

Choosing the parameter k During the RRB procedure, the linear system reduces to a similar one on a two times coarser grid. Solving this coarse grid equation can be done by applying another RRB iteration, or by using a complete Cholesky decomposition. The level on which a complete Cholesky decomposition is made is denoted by level k. In general, the Cholesky decomposition is accurate, but slow and the RRB iteration less accurate but easy to compute. In order to choose the optimal maximum level k, the number of flops is counted in Appendix I.
For a square grid with n^2 nodes, level k consists of n_k^2 nodes, with $n_k = n(\frac{1}{2})^{k-1}$. The RRB part requires $\frac{17}{2}n^2 - 17n_k^2$ flops and solving the Cholesky decomposition $4n_k^3 + 2n_k^2$ flops. The statement of an expensive Cholesky decomposition is supported by the cubic number of flops, while RRB uses a quadratic number of flops. However, the constants are different, and setting the derivative of the total number of flops with respect to n_k to zero yields $12n_k^2 - 30n_k = 0$, with solutions $n_k = 0$ and $n_k = \frac{5}{2}$. The zero solution is a local maximum and $n_k = \frac{5}{2}$ a global minimum for positive n_k. Because n_k should be an integer, $n_k = 2$ is the optimal choice when minimizing the total number of flops. Taking k such that $n_k = 2$ will give a complete Cholesky decomposition with a half bandwith of two.
Summarizing, considering the number of flops, choosing k such that $n_k = 2$ is optimal when the number of CG-iterations is constant.

11.3 Sparsity pattern of the RRB preconditioner

The RRB-k preconditioner, given by Equation (11.10), is a block incomplete Cholesky decomposition using a red-black numbering of nodes, repeated on several levels. For a lexicographical ordering, the matrices have a banded structure, like in Figure 4. But due to the RRB-k numbering, the sparsity pattern will be different. The sparsity pattern of the matrix $L + D + L^T$ on a grid with 10×10 nodes is shown in Figure 9.

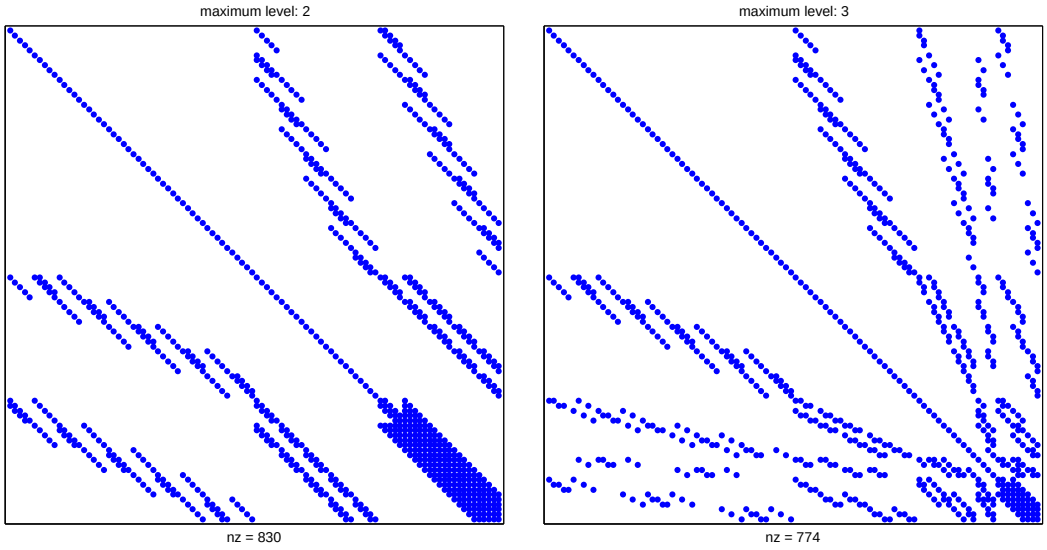

Figure 9: Sparsity pattern of $L + D + L^T$ with a maximum level of 2 (left) and 3 (right).

The maximum level is numbered at the end and therefore occurs in the lower right block of D. The whole band of this submatrix is filled in because of the complete Cholesky decomposition. When the maximum level is taken smaller, the Cholesky decomposition is also smaller and less fill-in elements occur. This is seen by the number of nonzero elements, which is 830 for $k = 2$ and 774 for $k = 3$.
The original matrix S has only coupling between the first level black and red nodes. In the left picture of Figure 10, one can see this by the diagonal lower right block. The sparsity pattern of S is similar to the structure in Equation (11.1).
When looking at the sparsity patter of the preconditioner $M = LDL^T$ in Figure 10, we see a similar structure as for $L+D+L^T$. But now extra fill-in elements occur in the block diagonal matrix D.

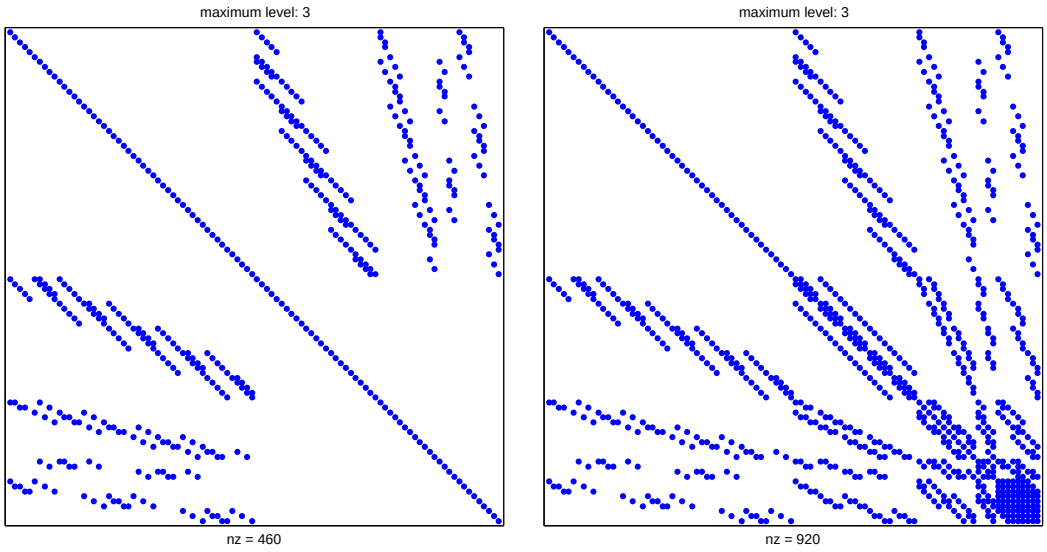

Figure 10: Sparsity pattern of original matrix S (left) and preconditioner LDL^T (right).

11.4 Lumping procedure during RRB

In the RRB preconditioner, at several levels 9-point stencils are reduced to 5-point stencils with a lumping procedure. As given in Equation (11.3), this is done by adding the four most outer elements to the center element. This is similar to the modified incomplete Cholesky decomposition from Equation (10.5). One can also use the idea of a relaxation parameter ω, resulting in the lumped 5-point stencil

$$\begin{bmatrix} 0 & nn & 0 \\ ww & cc + \omega(ne + nw + sw + se) & ee \\ 0 & ss & 0 \end{bmatrix}, \tag{11.11}$$

and similar for other stencils. For $\omega = 1$ it reduces to the stencil as considered before. Taking $\omega = 0$ is similar to an incomplete Cholesky decomposition.
Another option is to lump the most outer elements to the other outer elements, i.e.,

$$\begin{bmatrix} 0 & nn + \frac{1}{2}(ne + nw) & 0 \\ ww + \frac{1}{2}(nw + sw) & cc & ee + \frac{1}{2}(se + ne) \\ 0 & ss + \frac{1}{2}(sw + se) & 0 \end{bmatrix}. \tag{11.12}$$

In [30] another lumping strategy is proposed, for use of a slightly different RRB method applied to a convection-diffusion equation. The lumped stencil reads

$$\begin{bmatrix} 0 & nn + ne + nw & 0 \\ ww + nw + sw & cc - (ne + nw + sw + se) & ee + se + ne \\ 0 & ss + sw + se & 0 \end{bmatrix}. \tag{11.13}$$

These three lumping procedures are applied to the RRB preconditioner in the wave model. In Table 3 some results are shown for two test problems (see Chapter 13). Because the structure of the preconditioner does not change with the lumping procedure, only its values change, the computation time for solving a linear system is proportional to the number of CG-iterations.

	$\omega = 1$	$\omega = 0.5$	$\omega = 0$	Eq. (11.12)	Eq. (11.13)
Open sea of 200×200 nodes	7.755	11.584	13.511	10.731	22.031
Port of 120×240 nodes	6.975	10.824	12.754	11.550	25.408

Table 3: Average number of CG-iterations over 1000 time iterations, with RRB-$k_{\max}$ preconditioner and several lumping procedures.

With the results in Table 3, observe that taking $\omega = 1$ in stencil (11.11) gives the lowest number of CG-iterations of the different lumping procedures. Therefore the lumping strategy according to the modified incomplete Cholesky decomposition will be used.

11.5 Spectral condition number of the RRB-k preconditioner

Because preconditioners M are approximations of the original matrix S, the spectral condition number of the preconditioned matrix $M^{-1}S$ is smaller than the one of S. This gives a sharper upperbound (8.14) of the error during the CG-method and therefore a better convergence.
In [6] and [14], the RRB-k preconditioner is investigated for the two-dimensional Poisson equation, i.e., $\Delta u = f$, which is similar to the elliptic model equation (5.1c). The Poisson equation is discretized on a square domain with n nodes in both x- and y-direction and h denoting the characteristic mesh width. In both references, the levels are numbered different than is done in this section: $\tilde{k} = 2(k-1)$.
In [6], Brand has shown that the spectral condition number satisfy.

$$\left(\frac{\lambda_{\max}}{\lambda_{\min}}\right)_{\text{RRB-}\tilde{k}} \leq 2^{\tilde{k}/2}. \tag{11.14}$$

Let's consider the extreme choices $\tilde{k}_{\min} = 0$ and $\tilde{k}_{\max} = 2\log_2(n)$. The minimal choice $\tilde{k}_{\min}$ yields a spectral condition number of one. This is expected, because the complete Cholesky decomposition is performed on the whole grid and thus $M = S$, an exact preconditioner. For $\tilde{k}_{\max}$, the RRB iteration is performed until a maximum level of one node. The upperbound (11.14) gives $2^{\tilde{k}_{\max}/2} = n$, so a spectral condition number of $\mathcal{O}(h^{-1})$. For intermediate choices of $\tilde{k}$, the condition number will be in between $\mathcal{O}(1)$ and $\mathcal{O}(h^{-1})$.
We will look at two special choices of $\tilde{k}$. In [6], Brand takes $\tilde{k}_B = \log_2(n)$ as maximum level, which will then have n nodes. This choice is substituted in the upper bound (11.14) and gives a spectral condition number of $\mathcal{O}(h^{-\frac{1}{2}})$. With a different reasoning, in [14], Ciarlet determines the maximum level with $\tilde{k}_C = \frac{2}{3}\log_2(n) + \frac{4}{3}$, and proves that the spectral condition number is $\mathcal{O}(h^{-\frac{1}{3}})$.
The difference between these results is in the choice of maximum level. For $n > 16$, we have $\tilde{k}_C < \tilde{k}_B$. So, for most grids, the maximum level with $\tilde{k}_C$ has more nodes than with $\tilde{k}_B$. This gives a larger effect of the complete Cholesky decomposition and thus a more accurate preconditioner.
Observe that the choice of k according to minimal number of flops from Section 11.2 is given by $\tilde{k}_{\text{flops}} = \tilde{k}_{\max} - 4$, which gives a spectral condition number of $\mathcal{O}(h^{-1})$.

In order to verify these results of the spectral condition number, one can look at the number of CG-iterations. According to the theory in Section 8.3, the spectral condition number of $\mathcal{O}(h^{-1})$ for RRB-$k_{\max}$ yields $\mathcal{O}(h^{-\frac{1}{2}})$ CG-iterations. The number of CG-iterations, averaged over 1000 time steps, at an open sea (see Section 13.1) is presented in Table 4.

Grid	$k_{\max}$	k_{flops}	k_B	k_C
100×100	6.732	6.732	6.732	6.732
200×200	7.755	7.755	7.755	7.755
400×400	8.994	8.994	8.980	8.980
800×800	11.503	11.503	11.503	11.494

Table 4: Mean number of CG-iterations with RRB-k preconditioner at open sea.

As is done in Section 10.5, Richardson's extrapolation can give an estimate of the order of the number of CG-iterations needed. For $k_{\max}$ and k_{flops}, we expect $\mathcal{O}(h^{-\frac{1}{2}})$ CG-iterations. However, Richardson's extrapolation applied to the three coarsest grids give an order of -0.28, and to the finest grids -1.02. For the other choices k_B and k_C similar results are obtained. Because these results differ considerably between the grids, we can conclude that the mesh width is too large for this analysis. The Richardson extrapolation is not converged yet and will therefore give an inaccurate estimate.
The result of $\mathcal{O}(h^{-\frac{1}{2}})$ CG-iterations for $k_{\max}$ implies that for small h, the number of CG-iterations has to increase with a factor $\sqrt{2}$ in case of a two times coarser grid. The results in Table 4 show that the number of CG-iterations increase with less than a factor $\sqrt{2}$, which gives even a better convergence.
Note that the CG-method is performed on the first Schur-complement instead of the matrix $M^{-1}S$ for the whole domain. This is because the first level black points are eliminated exactly.

12 Deflation

Three different preconditioners for the CG-method have been discussed in previous sections. The *deflation method* can be used on top of these preconditioners. Its basic idea is to map some predefined vectors into the null-space. When the deflation vectors are chosen correctly, the spectral condition number will decrease, which can improve the performance of these PCG-methods. In this section the deflation method and its properties will be explained; for a more detailed description, the reader is referred to [27, 36, 37, 38].

12.1 Derivation of the deflation method

The error during the PCG-iteration is bounded by Equation (8.14). Improving this upper bound is mainly achieved by decreasing the spectral condition number $\lambda_{\max}/\lambda_{\min}$, the ratio of the maximal and minimal eigenvalue. When applying the CG-method to a singular and positive semi-definite system, there are zero eigenvalues, so $\lambda_{\min} = 0$ and the condition number is undetermined. As will be explained in Section 12.3, the spectral condition number will then be given by $\lambda_{\max}/\tilde{\lambda}_{\min}$, with $\tilde{\lambda}_{\min}$ the smallest nonzero eigenvalue.
The deflation method exploits this property by defining a matrix Q, such that QS has some zero eigenvalues and the other eigenvalues approximately equal to the eigenvalues of S. The spectral condition number of QS is than given by $\lambda^{QS}_{\max}/\tilde{\lambda}^{QS}_{\min}$, which is likely to be smaller than $\lambda^{S}_{\max}/\lambda^{S}_{\min}$, thus giving a better convergence of the CG-method.
This projection matrix Q is defined by

$$Q \;:=\; I - SZ\left(Z^T SZ\right)^{-1} Z^T \tag{12.1}$$

and is called the *deflation matrix* [36]. The *deflation subspace matrix* Z has to be specified by the user. The columns of Z are the *deflation vectors* and span the null-space of QS. For n grid points, $Q \in \mathbb{R}^{n\times n}$ and $Z \in \mathbb{R}^{n\times k}$, with k the number of deflation vectors, usually much smaller than n.

12.2 Properties of the deflation matrix

The deflation matrix Q is defined in Equation (12.1) and the discretization matrix S in Equation (8.1). In this section, the following properties will be explained (for similar theorems and proofs, see [36]):

(a) Q is a projection: $Q^2 = Q$;

(b) QS is symmetric: $(QS)^T = QS$;

(c) QS is positive semi-definite: $\langle x, QSx\rangle \geq 0 \quad \forall\, x \in \mathbb{R}^n$;

(d) Z is the null-space of QS: $QSx = 0$ iff $x \in \text{span}\{z_1, z_2, \ldots, z_k\}$.

Proof (a) The matrix Q is called a projection if it satisfies $Q^2 = Q$. We have

$$\begin{aligned} Q^2 &= \left(I - SZ\left(Z^TSZ\right)^{-1}Z^T\right)^2 \\ &= I - 2SZ\left(Z^TSZ\right)^{-1}Z^T + SZ\left(Z^TSZ\right)^{-1}Z^TSZ\left(Z^TSZ\right)^{-1}Z^T \\ &= I - 2SZ\left(Z^TSZ\right)^{-1}Z^T + SZ\left(Z^TSZ\right)^{-1}Z^T \\ &= I - SZ\left(Z^TSZ\right)^{-1}Z^T \\ &= Q, \end{aligned}$$

so Q a projection. In general, it will not be an orthogonal projection, because Q is nonsymmetric[22].

Proof (b) The symmetry of QS follows from

$$\begin{aligned} (QS)^T &= SQ^T \\ &= S\left(I - Z\left(Z^TSZ\right)^{-1}Z^TS\right) \\ &= S - SZ\left(Z^TSZ\right)^{-1}Z^TS \\ &= \left(I - SZ\left(Z^TSZ\right)^{-1}Z^T\right)S \\ &= QS, \end{aligned}$$

where the symmetry of S is used.

Proof (c) First observe that with the properties $Q = Q^2$ and $QS = SQ^T$, we can write

$$QS = Q^2S = QSQ^T.$$

Then we have, for the Euclidean inner product,

$$\langle x, QSx\rangle = \langle x, QSQ^Tx\rangle = \langle Q^Tx, SQ^Tx\rangle = \langle y, Sy\rangle,$$

with $y = Q^Tx$. For all vectors $y \neq 0$, the positive definiteness of S yields $\langle y, Sy\rangle > 0$, and for $y = 0$, we have $\langle y, Sy\rangle = 0$. Because of the identity[23] $Q^TZ = 0$, the null-space of Q^T contains the columns z_i of Z. For $x = z_i$, we get $y = 0$ and therefore $\langle x, QSx\rangle = 0$. Because there is at least one nontrivial z_i, the null-space of Q^T is also nontrivial and we get

$$\langle x, QSx\rangle \geq 0$$

for all $x \in \mathbb{R}^{n\times n}$, and $\langle x, QSx\rangle = 0$ for at least one nontrivial vector. Concluding, the matrix QS is positive semi-definite.

Proof (d) The null space of QS contains $\text{span}\{z_1, \ldots, z_k\}$, because

$$QSZ = SZ - SZ\left(Z^TSZ\right)^{-1}Z^TSZ = SZ - SZ = 0.$$

[22] A projection matrix Q is an orthogonal projection if Q is self-adjoint. For the Euclidean inner product space, this reduces to symmetry: $Q^T = Q$. With $B := Z\left(Z^TSZ\right)^{-1}Z^T$, we have $Q = I - SB$ and $Q^T = I - BS$, which are in general not the same.

[23] It holds that $Q^TZ = (I - Z(Z^TSZ)^{-1}Z^TS)Z = Z - Z(Z^TSZ)^{-1}Z^TSZ = Z - Z = 0$.

Now we need to show that this is the whole null-space of QS. In order to do this correctly, assume that the deflation vectors z_i are linearly independent, so $\text{rank}(Z) = k$. Since $QSZ = 0$ and S nonsingular, we have $\dim(\mathcal{N}(Q)) \geq k$, with $\mathcal{N}$ denoting the null-space:

$$\mathcal{N}(Q) = \{y \in \mathbb{R}^n \,|\, Qy = 0\}.$$

In order to investigate the range of Q, denoted by

$$\mathcal{R}(Q) = \{y \in \mathbb{R}^n \,|\, y = Qx \text{ for a } x \in \mathbb{R}^n\},$$

take an arbitrary vector y perpendicular to Z, i.e, $\langle z_i, y\rangle_2 = 0$ for $1 \leq i \leq k$. Then $Z^T y = \begin{bmatrix} z_1^T y \\ \vdots \\ z_k^T y \end{bmatrix} = \begin{bmatrix} \langle z_1, y\rangle_2 \\ \vdots \\ \langle z_k, y\rangle_2 \end{bmatrix} = 0$, yielding $Qy = y - SZ(Z^T SZ)^{-1} Z^T y = y - SZ(Z^T SZ)^{-1} 0 = y$. Since $\dim\{z_i, \ldots, z_k\} = k$, the orthocomplement satisfies $\dim\{y \in \mathbb{R}^n \,|\, \langle z_i, y\rangle_2 = 0 \text{ for } i = 1, \ldots, k\} = n - k$. Together with $Qy = y$ we have $\dim(\mathcal{R}(Q)) \geq n - k$. Because the dimension of the null-space and range have to sum up to n, we get $\dim(\mathcal{N}(Q)) = k$, and $\mathcal{N}(Q) = \text{span}\{Sz_1, \ldots, Sz_k\}$. The zero null-space of S finally yields

$$\mathcal{N}(QS) = \text{span}\{z_1, z_2, \ldots, z_k\},$$

the null-space of QS.

12.3 The conjugate gradient method applied to singular systems

During the derivation of the CG-method in Section 8.3, it was stated that the matrix which the CG-method is applied for should be symmetric positive definite and thus nonsingular. As will be shown in Section 12.4, in the deflation method the CG-method will be applied to a singular system. In this section we will explain that this is allowed (see also [40, 15]).
A linear system $Ax = b$ is considered for which $A \in \mathbb{R}^{n\times n}$ is singular and symmetric positive semi-definite, so A has only nonnegative eigenvalues. Since A singular, the null-space of A is nontrivial, i.e., the null-space contains at least one nonzero vector. Also observe that there is a solution of $Ax = b$ iff $b \in \mathcal{R}(A)$, the range of A, which will be assumed in the following analysis.
Because A symmetric and real-valued, thus self-adjoint in the Euclidean space, it has only real eigenvalues and no generalized eigenvectors; moreover, it has a complete set of orthogonal eigenvectors [23]. Any vector can therefore be written as a linear combination of eigenvectors u_i with λ_i its corresponding eigenvalue. For the initial residual $r_0 = b - Ax_0$, we write $r_0 = \sum_{i=1}^n \alpha_i u_i$, with α_i some constants. The assumption $b \in \mathcal{R}(A)$ implies $r_0 \in \mathcal{R}(A)$. Since $\mathcal{R}(A) \cap \mathcal{N}(A) = 0$, we have $r_0 \notin \mathcal{N}(A)$ and thus $\alpha_i = 0$ for all i with $\lambda_i = 0$.
The eigenvalues λ_i for which $\alpha_i \neq 0$ are called the *active* ones; in view of the Krylov subspace (8.7) the other eigenvalues and eigenvectors do not participate in the CG process [40]. In particular, the zero eigenvalues of A are not active and therefore do not influence the CG-method.

From Equation (8.14), it can be concluded that the convergence of CG depends mainly on the spectral condition number $\lambda_{\max}/\lambda_{\min}$. The results in this section imply that this can be changed to the ratio of the maximal and minimal active eigenvalue. In general, this will be the ratio between the maximal and minimal nonzero eigenvalue.

12.4 Deflated conjugate gradient method

The deflation method uses a projection Q which maps some predefined vectors z_i into the null-space of QS. The zero eigenvalues of the singular QS will probably yield a more favorable spectrum of QS than S. Therefore, CG-method is not applied to the original system $S\psi = b$ (8.1), but to the deflated system

$$QS\tilde{\psi} = Qb. \tag{12.2}$$

Because of the singularity, the solution $\tilde{\psi}$ is not unique and is therefore not necessarily the solution of the original system. However, one can show that the vector $Q^T\tilde{\psi}$ is uniquely and well defined, and satisfies $Q^T\tilde{\psi} = Q^T\psi$ [36]. Therefore, let's split ψ as

$$\psi = (I - Q^T)\psi + Q^T\tilde{\psi}. \tag{12.3}$$

The first part on the right hand side of this equation can be rewritten as

$$\begin{aligned} \left(I - Q^T\right)\psi &= \left(I - \left(I - Z\left(Z^TSZ\right)^{-1}Z^TS\right)\right)\psi \\ &= Z\left(Z^TSZ\right)^{-1}Z^TS\psi \\ &= Z\left(Z^TSZ\right)^{-1}Z^Tb. \end{aligned} \tag{12.4}$$

This expression depends only on the known right hand side b, not anymore on ψ and can therefore be computed without the use of a linear solver.
Summarizing, the deflated system (12.2) is solved for $\tilde{\psi}$ with the CG-method. The solution ψ of the original system is then calculated with

$$\begin{aligned} \psi &= Z\left(Z^TSZ\right)^{-1}Z^Tb + Q^T\tilde{\psi} \\ &= \tilde{\psi} + Z\left(Z^TSZ\right)^{-1}Z^T\left(b - S\tilde{\psi}\right). \end{aligned} \tag{12.5}$$

Calculating the solution $\tilde{\psi}$ of Equation (12.2) with CG will consume most computation time. The deflation subspace matrix Z, which determines Q, should therefore be chosen such that the deflated CG-method converges faster than the undeflated version, i.e., the spectrum of QS has to be more favorable than the spectrum of S.

12.4.1 Deflated preconditioned conjugate gradient method

The deflated system (12.2) is solved with the conjugate gradient method. Applying a preconditioner can improve the convergence of CG. So, the linear system of equations

$$M^{-1}QS\tilde{\psi} = M^{-1}Qb \tag{12.6}$$

is solved with CG for a preconditioner $M = PP^T$, which is still based on S. Another possibility is to apply the CG-method to the system

$$\widetilde{Q}\widetilde{S}\widetilde{\tilde{\psi}} = \widetilde{Q}\widetilde{b}, \tag{12.7}$$

with the preconditioned variables

$$\widetilde{Q} = P^{-1}QP, \quad \widetilde{S} = P^{-1}SP^{-T}, \quad \widetilde{\tilde{\psi}} = P^T\tilde{\psi}, \quad \text{and} \quad \widetilde{b} = P^{-1}b.$$

$x_0 = \psi_0$	Initial solution
$r_0 = b - Sx_0$	Initial residual
$r_0 = Qr_0$	Deflated residual
$i = 0$	Number of iterations
while $\|\|r_i\|\| > \epsilon$	
$s_i = M^{-1}r_i$	Preconditioned residual
$i = i + 1$	
if $i = 1$	
$p_i = s_{i-1}$	Search vector
else	
$\beta_i = \dfrac{\langle r_{i-1}, s_{i-1}\rangle}{\langle r_{i-2}, s_{i-2}\rangle}$	
$p_i = s_{i-1} + \beta_i p_{i-1}$	Update search vector
end	
$q_i = QSp_i$	Map search vector
$\alpha_i = \dfrac{\langle r_{i-1}, s_{i-1}\rangle}{\langle p_i, q_i\rangle}$	
$x_i = x_{i-1} + \alpha_i p_{i-1}$	Update iterate
$r_i = r_{i-1} - \alpha_i q_{i-1}$	Update residual
end	
$x = x_i$	Solution of DPCG
$\psi = x + Z\left(Z^T SZ\right)^{-1} Z^T (b - Sx)$	Solution

Table 5: Algorithm of the deflated preconditioned conjugate gradient method.

Both versions of the deflated PCG-method are equivalent [36]. The algorithm of the deflated preconditioned conjugate gradient method is given in Table 5.
Note that the preconditioner M occurs in a different equation than Q and S. It can therefore be implemented separately, thus making it easy to combine different kinds of preconditioners with deflation.

12.5 Choice of deflation vectors

The goal of the deflation method is to increase the convergence of CG. Similar to preconditioning, deflation changes the spectrum of the matrix applied to the CG-method. More precisely, the spectrum will contain some zero eigenvalues. In Section 12.3 it has been explained that the zero eigenvalues are not active and will not influence the convergence of CG. The spectrum of QS will therefore have less active eigenvalues and the spectral condition number is likely to be smaller than the one of S. In [36] it has been shown that the deflated spectral condition number is always smaller than the undeflated one, also in case of preconditioned matrices.
The deflation matrix Q depends on the matrix Z, which columns are the deflation vectors z_i. These deflation vectors have to be chosen such that the spectral condition number will decrease as much as possible.
Because the spectral condition number explicitly depends on the smallest nonzero eigenvalue, it is a good idea to deflate the smallest eigenvalues. This is achieved by defining the k

deflation vectors z_i as the eigenvectors corresponding to the smallest eigenvalues, and is called *eigenvector deflation*. Although this is theoretically an excellent choice, in practice the eigenvectors are not known in advance and are hard to calculate.
An method to choose the deflation vectors easier is *subdomain deflation*. The domain Ω is subdivided in k subdomains Ω_i, which are disjoint and cover the whole domain, i.e., $\Omega_i \cap \Omega_j = \emptyset$ for all $i \neq j$ and $\cup_{i=1}^{k} \Omega_i = \Omega$. The subdomain deflation vectors $\mathbf{z}_i$ are defined as

$$(z_i)_j \quad := \quad \begin{cases} 1, & x_j \in \Omega_i \,; \\ 0, & x_j \in \Omega \setminus \Omega_i \,, \end{cases} \tag{12.8}$$

with x_j grid points [38]. The subdomains Ω_i are chosen rectangular, as in figure 11.

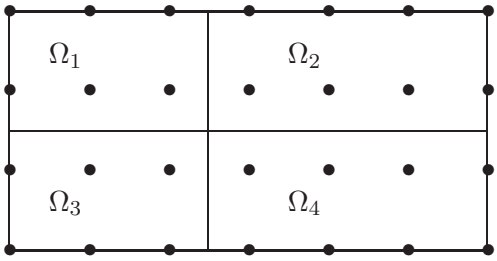

Figure 11: Domain divided in four rectangular subdomains.

12.6 Spectrum of the deflated matrices

In order to verify some properties of the deflation matrix, we will look at the matrix S from the test problem of an open sea (see Section 13.1). The deflation matrix Q is given by subdomain deflation, with k rectangular subdomains. Two different bathymetries are used, one with a constant depth of $30\,m$ and one with a trench in it. In Figure 12 the spectra, calculated with `Matlab`, are shown.

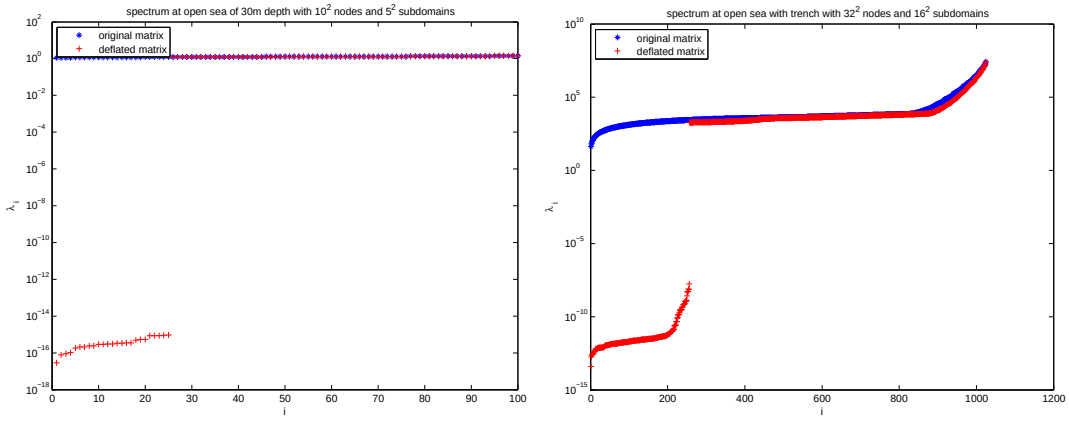

Figure 12: Spectrum of the original matrix S and deflated matrix QS.

Both spectra clearly show that QS has k zero eigenvalues, with k the number of deflation vectors. Because of rounding errors, they are not exactly equal to zero, as can be see in Figure 12.

For exact eigenvalue deflation, the nonzero eigenvalues of QS will be the same as the largest eigenvalues of S. As can be seen in Figure 12, this is only approximately true for subdomain deflation. So, the largest deflated eigenvalues are almost the same as the undeflated ones, but the the smallest nonzero eigenvalues of the deflated matrix differ from the smallest eigenvalues of the undeflated matrix. By applying deflation, the spectral condition number is decreased. For the first test problem considered, the spectral condition number reduces from 1.2537 to 1.1742 and for the second from 635 153 to 13 639. Because of the smaller spectral condition number, the deflation method is likely to converge faster. More results of the deflation method will be presented in Section 14.5.

12.7 Implementation aspects of the deflation method

In every iteration of the deflated CG-method, as given in Table 5, the matrix-vector multiplication $y = Qx$ has to be calculated, with Q the deflation matrix defined in Equation (12.1). This multiplication is divided in several steps:

- $\mathbf{a} = Z^T\mathbf{x}$;
- $\mathbf{b} = (Z^TSZ)^{-1}\mathbf{a}$;
- $\mathbf{c} = Z\mathbf{b}$;
- $\mathbf{d} = S\mathbf{c}$;
- $\mathbf{y} = \mathbf{x} - \mathbf{d}$.

To calculate $\mathbf{a}$, one should observe that $Z^T\mathbf{x}$ can be written in terms of inner products: $a_i = \langle \mathbf{z}_i, \mathbf{x}\rangle_2$ for $1 \leq i \leq k$. With the choice of subdomain deflation as in Equation (12.8), a_i is given by the sum of the elements of $\mathbf{x}$ inside subdomain i, i.e., $a_i = \sum_{j\in\Omega_i} x_j$.
In a similar way $\mathbf{c}$ can be calculated. The vector $\mathbf{b} \in \mathbb{R}^k$ contains values b_i for each subdomain i. This value is distributed to all grid points inside the corresponding subdomain, i.e., $x_j = b_i \ \forall j \in \Omega_i$.
On the coarse grid, one has to solve $Z^TSZ\mathbf{b} = \mathbf{a}$, which is a system of k linear equations. Note that Z^TSZ is symmetric and has a banded structure, with $\sqrt{k}$ half the bandwith. Since in general $k \ll n$, a complete Cholesky decomposition of Z^TSZ is calculated for solving $\mathbf{b}$ efficiently and accurately.
The matrix-vector multiplication $S\mathbf{c}$ is already implemented for use in the CG-iteration and can therefore readily be calculated.
Often, the matrix $SZ \in \mathbb{R}^{n\times k}$ is calculated explicitly in order to calculate $\mathbf{d} = SZ\mathbf{b}$ in one step. However, for the considered model, this does not lead to a larger efficiency.
For node j in the interior of subdomain i, the corresponding row of SZ is given by zeros, except element $(SZ)_{ji}$, which is given by the rowsum of S. For some matrices (for example the discretized Poisson equation), this is again zero, thus further simplifying SZ [36]. This simplification is not the case for the matrix S in Equation (8.1), because it is strictly diagonally dominant. For nodes j on the boundary of subdomain i, row i of SZ will have several nonzero elements. So calculating $\mathbf{d} = SZ\mathbf{b}$ in one step is not that easy.
On the other hand, calculating $Z\mathbf{b}$ is easily done for linear subdomain deflation. Because S is pentadiagonal, the matrix-vector multiplication $S\mathbf{c}$ is calculated with only $9n$ flops. Concluding, for this case of a sparse matrix S and subdomain deflation matrix Z with only zeros

and ones, calculating $\mathbf{d} = SZ\mathbf{b}$ in one step and calculating $\mathbf{c} = Z\mathbf{b}$ and $\mathbf{d} = S\mathbf{c}$ separately are done with a comparable amount of computation time.
In order to check this, both choices are implemented. Implementing $SZ\mathbf{b}$ is done in two different ways: one which uses stencils and one which labels the nodes. Two test problems are considered: an open sea with 200×200 nodes and a port of 120×240 nodes with incoming waves. At the first problem the deflated diagonally scaled CG-method is used with 40×40 subdomains. The deflated MICCG-method with 10×10 subdomains is applied to the second test problem.

	S and Z separately	SZ in one step	
Port with DDgsCG 40×40	98.7679	115.9915	109.8509
Open Sea with DMICCG 10×10	78.1952	89.8005	83.013

Table 6: CPU-time of linear solves of 1000 time iterations.

The CPU-time used for the wave model with the different implementations of $SZ\mathbf{b}$ is given in Table 6. It shows that the computational time does not differ a lot. In both cases, the best choice is implementing $S\mathbf{c}$ and $Z\mathbf{b}$ separately.

13 Test problems

Various models exist to test the implementation of the wave model. The choices of for example the bottom topography, incoming waves on the open boundaries, moving ships and mesh size determine the test problem.
These test problems can be used for investigating the performance of the linear solver. We can easily change the complexity of the test problems by increasing the number of grid points for each domain.

13.1 Open sea

An easy domain for a test problem is a rectangular domain with constant depth. This is a model for a part of a sea. The boundaries are all open, on which incoming waves can be defined.

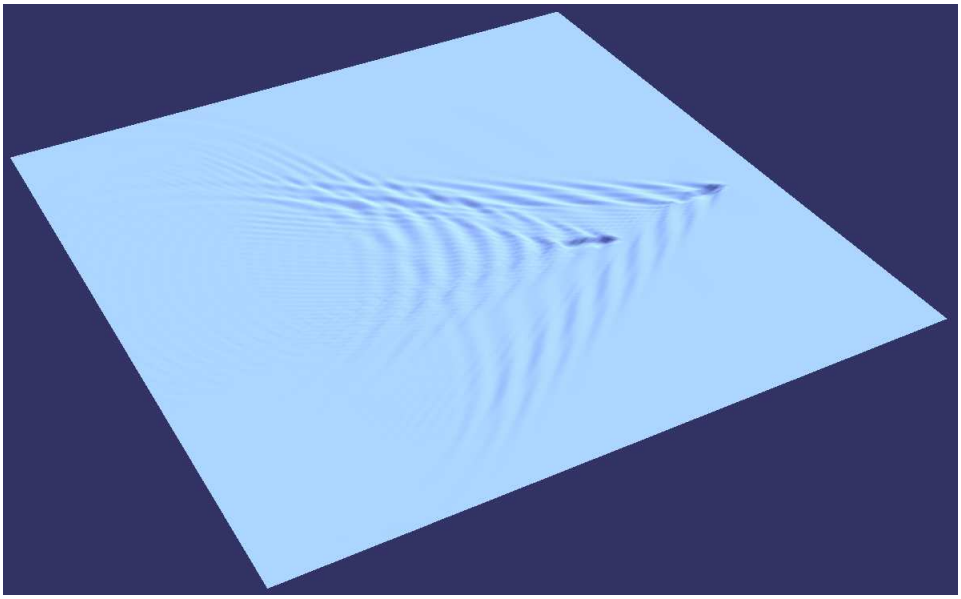

Figure 13: Two ships with intersection courses.

In Figure 13 an example is shown. Two ships with different headings have been modelled by a predefined pressure term. No incoming waves and currents are specified. The size of the domain shown in the figure is a square kilometer, with mesh size of $5 \times 5\,m$ and a depth of $30\,m$, so a computational domain of 200×200 nodes.

13.2 Port

A lot of wave phenomena, like shoaling, occur due to changes in water depth. In order to check the performance of the Boussinesq model with varying water depths, an artificial port has been developed. Present are a beach, a harbour and a shallow part. The depth in the largest part of the domain is $30\,m$. The beach has a length of $200\,m$. The harbor has constant depth and is divided from the sea by a pier of $10\,m$ width. The shallow part raises the bottom to $2\,m$ depth, with a radius of $125\,m$. The domain has a size of $600 \times 1200\,m$ with finite volumes of $5 \times 5\,m$.

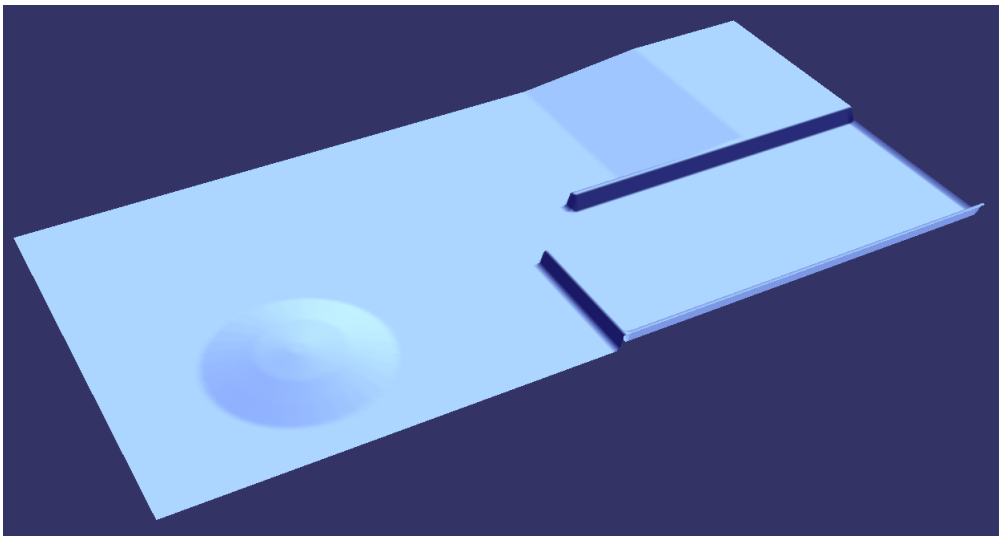

Figure 14: Bathymetry of the port.

13.2.1 Wave patterns due to varying water depth

When applying the variational Boussinesq model to a varying bathymetry, some wave phenomena can be observed. In the next results, four phenoma can be seen: shoaling, refraction, reflection and diffraction. See [13] for more information about wave phenomena.
In Figure 15 a harmonic wave is entering the domain at the north and west boundaries. At the shallow part of the sea, the water depth decreases. Then, the wave height will increase, as is seen in the figure. This change in wave height is called *shoaling*. Near the beach, the waves are moving with an angle to the shore. Due to the difference in water depth over a wave crest, the waves are changing direction towards the shore, called *refraction*.

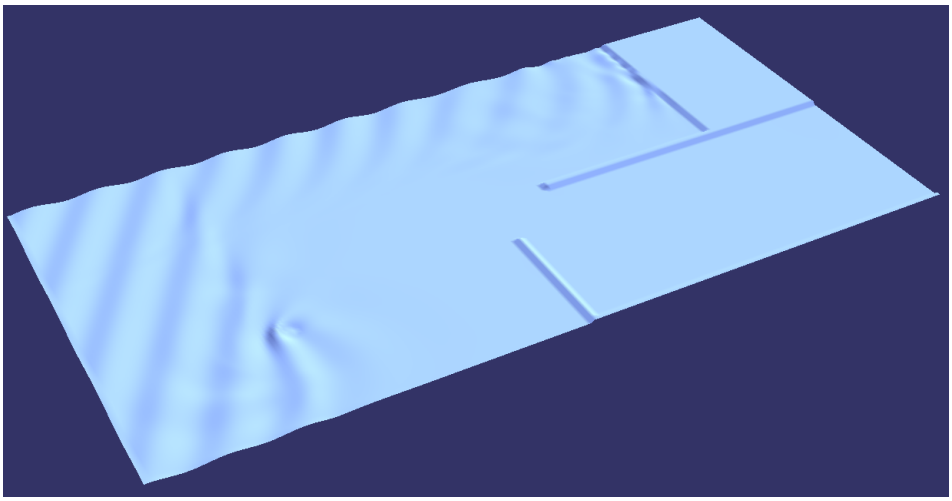

Figure 15: Shoaling and refraction.

When waves will move onto walls, it will reflect back into the sea. The *reflection* of the waves at the pier can be seen in Figure 16. The waves from the southwest corner move past the pier. Right after the pier, there is a sharp difference in wave amplitude. This causes *diffraction* of the waves towards the harbour.

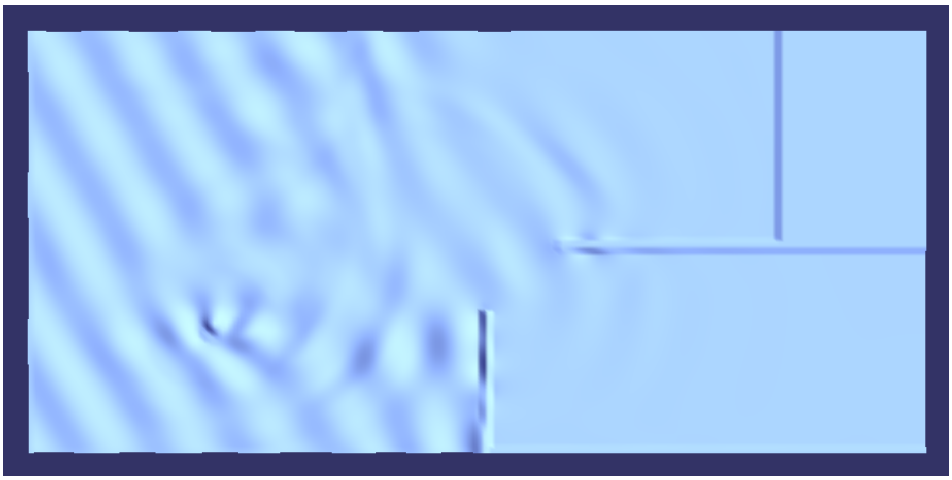

Figure 16: Reflection and refraction.

13.3 IJssel

A more realistic problem is given by a model based on the river IJssel. The river has a depth of around $4\,m$. The width of the river is around $60\,m$ and we consider a length of $800\,m$. On this $200 \times 800\,m$ sized domain, a uniform $2 \times 2\,m$ grid has been specified.

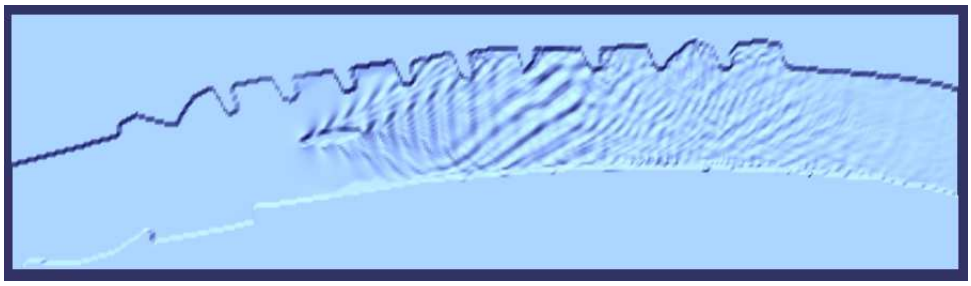

Figure 17: A ship sailing through the river IJssel.

In Figure 17, the considered part of the IJssel is shown. It consist of a bending river with a smooth curving boundary at the left bank. The right bank is more irregular, with breakwaters. In the figure, it can be seen that the boundaries reflect the waves produced by the ship.

14 Results

In the first part of this thesis, a study about the variational Boussinesq model for waves has been presented. One of the discretized model equations results in a linear system of equations, solved with an iterative method. A main part of the research is the development of a linear solver with improved performance. To make it more clear what is meant with improved performance, some assessment criteria will be discussed in Section 14.1.
As explained in previous chapters, the conjugate gradient method is used for solving the linear system. Several preconditioners are applied to improve the convergence, namely diagonal scaling, relaxed incomplete Cholesky and repeated red-black. The deflation method is combined with the preconditioners.
Several test problems have been used to obtain some results, from which some properties of the method can be derived. In the results as given in this section, a characteristic test problem is used; other test problems give similar results.

14.1 Criteria for the performance assessment

In order to compare the linear solvers with each other and to assess their performance, some assessment criteria will be explained in this section.

Efficiency One of the most important properties of a solver is its efficiency. The efficiency is twofold: computing time and storage. The computing time is measured in terms of CPU-time, so the time it takes to solve the linear system on a standard computer.
Several variables have to be stored for the methods. A lot of them scale with the number of grid points, which can become quite large. To save memory, as less variables as possible have to be stored.

Real-time The goal of the variational Boussinesq model is the use in a real-time ship simulator. This requires a large efficiency. Moreover, the time to solve the system is fixed to a few hundreds of a second and will in practice never be constant. During the calculation of the wave model, other processes run simultaneously, which may cause delays in the calculation process.

Convergence with mesh-refinement It is likely that in the future the computational domains will be larger. More unknowns are used in the model, resulting in a larger system of equations. Because the efficiency requirements still have to be satisfied, the solver should not scale disproportional with the number of nodes. For the same tolerance, the CG-method needs $\mathcal{O}(h^{-1})$ iterations. Preconditioners and other methods can reduce this, to $\mathcal{O}(\infty)$ in the ideal case, which is called grid-independent convergence. Increasing the number of nodes then will not lead to slower convergence.

Parallelization To reduce the wall-clock time of the methods, one can choose to parallelize the solver over several computing units. The parallelization of linear solvers is not always straightforward and the possibilities for parallelization depends on the method used. The actual parallelization over a shared memory computer is beyond the scope of this project. But we will look at some results of parallel methods on one CPU.

Robustness The wave model have to be applicable to a wide range of bathymetries, including beaches, harbours, rivers and trenches. The linear solver have to be robust enough to solve the different linear systems for these bathymetries.

14.2 Overall behaviour of the CG-method

The conjugate gradient method is an iterative solver for a linear system of equations. So, an initial estimate of the solution is updated in several steps until the estimate is well enough. This is achieved by iterating the method until a norm of the residual is smaller than a predefined value, called the termination criterium.
In the CG-method, every iteration the value $\rho_i = \langle r_i, z_i \rangle_2$ is calculated, with r_i denoting the residual at iteration i and $z_i = M^{-1} r_i$. Observe that $\langle r_i, z_i \rangle_2 = \langle r_i, r_i \rangle_{M^{-1}} = ||r_i||^2_{M^{-1}} = ||\tilde{r}_i||^2_2$, with $\tilde{r}_i = P^{-1} r_i$ denoting the preconditioned residual and $||\cdot||_2$ denoting the Euclidean norm. The value ρ_i is an estimate of the error at iteration i, and is readily available. The termination criterium is therefore based on $\rho_i = ||r_i||_{M^{-1}}$. As will be discussed in Section 14.9, an absolute termination criterium is used:

$$||r_i||_{M^{-1}} \leq \epsilon,$$

with $\epsilon = 2 \cdot 10^{-6}$ taken in all results which will be presented.

To analyze the behaviour of ρ_i for different CG-iterations i, it is shown in Figure 18. The test problem used is an open sea with 200×200 nodes and the MICCG-method.

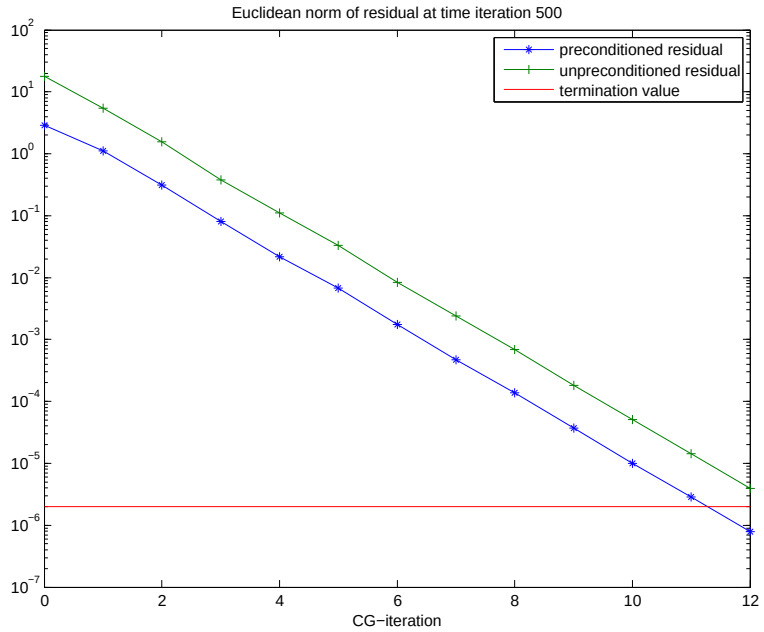

Figure 18: Euclidean norm of preconditioned and unpreconditioned residual in the MICCG-method, at open sea of 200×200 nodes.

The residual is decreasing monotonically with each CG-iteration, until the termination value is reached, as is seen in Figure 18. The unpreconditioned residual is shifted one order of magnitude, but has almost the same structure as the preconditioned residual.

The linear solver is a part of the wave model and will be used in every timestep. To analyze how the performance of the linear solver changes with simulation time, a simulation of the variational Boussinesq model with 1000 timesteps of $0.05\,s$ is performed on the same test problem as before. The number of CG-iterations used for each timestep as well as the CPU-time used for solving Equation (8.1) with MICCG is shown in Figure 19.

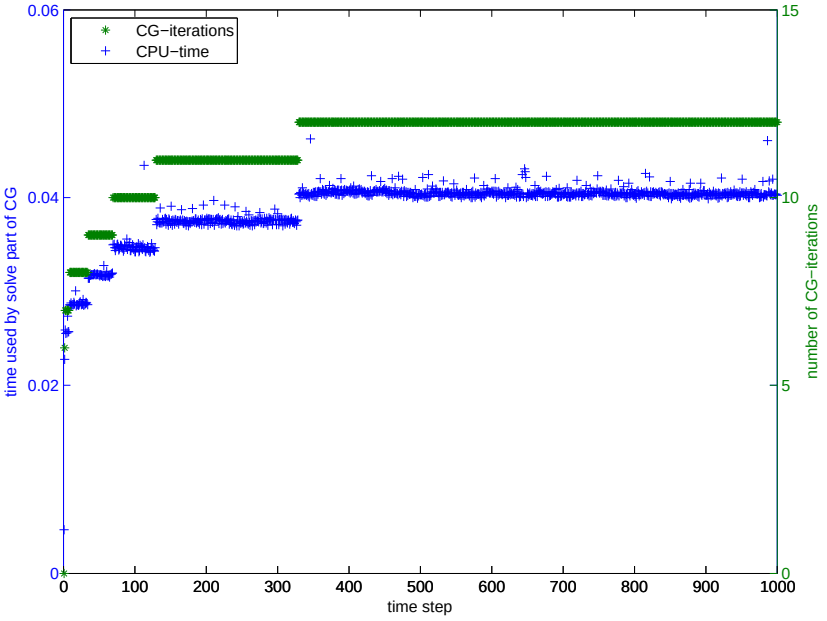

Figure 19: Results at open sea of 200×200 nodes and MIC preconditioner.

The number of CG-iterations needed for a separate time step is increasing in time. At the start of the simulation only 6 CG-iterations are needed, while at the end it is 12. This is because a zero initial state is taken, hence an almost undisturbed water surfuce. At the first few time steps, the two ships sailing into the domain cause a wave pattern which is limited to a small part of the domain. Later on, the waves are propagated into the whole domain. The water height becomes more irregular and the linear system will be harder to solve, resulting in more CG-iterations. After 400 timesteps the wave pattern is developed and the number of CG-iterations per timestep remains constant.
Two timers have been implemented in the code of the wave model. One gives the CPU-time used for calculating the preconditioner, the other gives the time for solving the linear system with the CG-method. The CPU-time needed for solving the linear system clearly depends on the number of CG iterations and is, on average, $0.0388\,s$ per time step. Note that in order to calculate real-time, the CPU-time should be less than the timestep of $0.05\,s$, which is the case for this test problem. Calculating the preconditioner is relatively cheap, it takes only $4.4\,ms$.

14.3 Varying the maximum level in RRB

The repeated red-black preconditioner uses recursively a red-black ordering of nodes until a maximum level has been reached. On this coarse grid a complete Cholesky decomposition is performed. The maximum level is denoted by k and should be at least 2, because $k = 1$ results in a complete Cholesky decomposition on the whole domain. The maximal value $k_{\max}$

is such that the maximum level contains only one node. All integer values between those two are allowed. In this section we will look at the influence of this parameter on the performance of RRB-k.
In Section 11.5 some theoretical results about the influence of the parameter k in the RRB-preconditioner have been derived. For $k_{\max}$ the spectral condition number is $\mathcal{O}(h^{-1})$, while smaller values of k yield a lower order of the spectral condition number. Therefore, we expect that decreasing k will also give a decrease in the number of CG-iterations. In order to examine this, the RRB-k method is applied to the IJssel problem, given in Section 13.3, for different values of k. In Table 7 the results are shown.

Solver	Size of max level	Number of CG-iter.	CPU-time 1000 solves
RRB-10	1×1	6.205	53.46
RRB-9	1×2	6.205	45.41
RRB-8	1×4	6.205	42.66
RRB-7	2×7	6.205	40.59
RRB-6	4×13	6.205	40.33
RRB-5	7×26	6.205	39.64
RRB-4	13×51	6.205	40.96
RRB-3	26×101	6.186	48.64
RRB-2	52×202	6.174	103.03

Table 7: Results for RRB-k at the IJssel of 104×404 nodes with varying k.

The number of CG-iterations is exactly the same for $4 \leq k \leq 10$, so the choice of k has less effect on the convergence of CG than expected. Suprisingly, the CPU-time decreases for a constant number of CG-iterations. A reason is that the number of flops is minimal for $k = 7$, as explained in Section 11.2, however, the CPU-time is still not minimal.
The drop in CPU-time with constant number of CG-iterations can also be explained with caching effects and implementational issues. The RRB-preconditioner uses several grids with different mesh sizes. The implemented loops will therefore jump with different distances through an array, which can be slow. For the complete Cholesky decomposition, the elements on the coarsest grid are reordered, thus simplifying the loops through the arrays.
The influence of implementation aspects will not be investigated thoroughly, however, to show that it is notable, the following experiment is done. To solve a system $LL^T x = b$ from the Cholesky decomposition, forward and backward substitution is used. The algorithms are equivalent, except of the directions of the loops. So the same number of flops, but a slightly different implementation. A timer is implemented for these two similar methods, for which the same CPU-time is expected.
The results in Table 8 show a considerable difference in CPU-time between forward and backward substitution. While forward substitution performs better with large systems, the backward substitution is faster on small systems.

number of elements	100^2	50^2	25^2	13^2	7^2	4^2	2^2	1^2
forward substitution	52.06	7.42	1.02	0.116	0.0242	0.0091	0.0048	0.0039
backward substitution	93.43	9.45	1.26	0.168	0.0296	0.0078	0.0029	0.0018

Table 8: The CPU-time for the forward and backward subsitution of the Cholesky decomposition, on an open sea of 200×200 nodes.

Although this does not clarify the change in CPU-time for constant number of CG-iterations fully, it shows that the number of flops is not always a good estimate of the computation time and that implementation and caching may have influence on the performance.

We will now return to the results in Table 7. The choice of $k = 5$ gives the lowest amount of CPU-time although the number of CG-iterations is the same. Compared to $k_{\max}$, the CPU-time dropped 25%.
For a minimal number of flops, the maximum level is given by $k = 7$ (see Section 11.2), which is too large to be optimal. On the other hand, the choices k_B and k_C from [6] and [14] (see Section 11.5) give $k = 4$, which is somewhat too low. When considering the results of other test problems, the same is observed: k_{flops} too large and k_B and k_C too small. So an average of these choices would be a good estimate of the optimal k.

Summarizing, the results of RRB-k show that the number of CG-iterations is almost constant for varying k. Only for very small k, there is a small decrease in CG-iterations, but at the expense of a sharp increase in CPU-time. Although the number of CG-iterations changes hardly, the CPU-time changes due to a decrease in flops and probably implementational effects. The difference in CPU-time between the maximal and optimal value of k is between 5 and 25% depending on the test problem.

14.3.1 Use of Lapack routines

The complete Cholesky decomposition on the maximum level is implemented both with an own implemented `C++` routine and a `Lapack` routine[24]. The advantage of the `C++` routine is the usage of the same data structures as in the wave model. The `Lapack` routine is optimized, but has some overhead since the data has to be converted to the structure of `Lapack`.
When comparing the CPU-time results, we see that for small decompositions, $k > 3$, the `C++` routine is slightly faster. For small k, so a complete Cholesky decomposition with relatively many elements, the `Lapack` routine is much faster. However, the total CPU-time for these cases is already large.

14.3.2 Cholesky decomposition

The Cholesky decomposition used in the RRB-k method can also be used for solving the whole linear system. The CPU-time needed for calculating the preconditioner and solving the linear system is given in Table 9.
Calculating the Cholesky decomposition is much more expensive than the RRB-preconditioner. The CPU-time for solving a linear system is somewhat larger for the complete Cholesky decomposition than for RRB-$k_{\max}$, but smaller than for RRB-2.

[24] `Lapack` is a standard package with several routines of direct solvers, among others the Cholesky decomposition. The routines are optimized by several authors [2].

	RRB-$k_{\max}$	RRB-2	Cholesky	`Lapack` Cholesky
preconditioner	0.0036	0.014	0.083	0.11
solve	0.0064	0.019	0.014	0.0085

Table 9: The average CPU-time of one timestep, at open sea of 100×100 nodes.

With mesh-refinement, the differences between the RRB-method and the complete Cholesky decomposition becomes larger. Because on a coarse grid of 100×100 the RRB-method is already faster, the complete Cholesky decomposition is not useful for the considered problems.

14.4 Influence of the relaxation parameter on RICCG

The relaxed incomplete Cholesky preconditioned CG-method contains a relaxation parameter ω, as explained in Chapter 10. For $\omega = 0$, the RIC decomposition reduces to the incomplete Cholesky decomposition and for $\omega = 1$ it reduces to the modified incomplete Cholesky decomposition. The relaxation parameter determines the lumping procedure during the incomplete decomposition and changes only the value of the preconditioner, not the structure. Therefore, each iteration of RICCG requires the same computation effort for every value of the parameter. The CPU-time is thus proportial to the number of CG-iterations.
In Figure 6 of Section 10.5, the spectral condition number has been given for several relaxation parameters. It shows that the spectral number decreases for increasing relaxation parameter. However, near $\omega = 1$, the spectral condition number is not monotone.

ω	100×100	200×200	400×400	800×800
0	7.822	16.357	32.895	67.233
0.95	6.826	10.910	18.522	35.600
1	7.399	11.450	17.144	25.758

Table 10: The average number of CG-iterations at an open sea.

The number of CG-iterations, listed in Table 10, show that $\omega = 1$ gives the smallest number at the fine grids. For the coarse grids, $\omega = 0.95$ is a better choice. Because the small improvements of $\omega = 0.95$ over $\omega = 1$ in the coarse test problems, and because we are mainly interested in fine grids, $\omega = 1$ is the default choice.

As will be explained in Section 14.6, combining the RICCG-method with deflation changes the influence of the relaxation parameter.

14.5 Varying the number of deflation vectors

The deflation method projects some vectors into the null-space, resulting in a smaller spectral condition number, as explained in Chapter 12. Subdomain deflation is implemented with rectangular subdomains. The number of subdomains can be chosen by the user. In this section, results are discussed for varying number of deflation vectors. As preconditioner, diagonal scaling is used. Results for the combination of deflation and the RIC-preconditioner will be given in Section 14.6.
For increasing number of deflation vectors, the spectral condition number decreases, often yielding a reduction in CG-iteration. As seen in Table 11, this is true. However, each iteration

Number of subdomains	Mean number CG-iterations	CPU-time 1000 solves
0×0	51.204	113.5
1×1	51.205	209.8
5×5	51.075	208.3
10×10	48.817	200.6
20×20	40.224	171.2
40×40	26.820	140.6
80×80	17.014	210.8

Table 11: Results at open sea of 200×200 nodes, with deflated diagonally scaled CG.

is more expensive for larger numbers of deflation vectors. The result of these two effects is a larger amount of CPU-time for the considered test problem.
Between zero and one deflation vector, there is a jump in CPU-time. With zero deflation vectors, the undeflated CG-method is obtained. For one deflation vector, the deflated CG-method has to calculate $y = Qx$ every iteration, with Q the deflation matrix given by Equation (12.1). Since the coarse grid matrix $Z^T SZ$ is only one element large, no Cholesky decomposition is needed, but one still has to calculate this coarse grid matrix. Also an extra matrix-vector multiplication $y = Sx$ is required inside the deflation matrix. As explained in Section 12.7, calculating $y = SZx$ in one step takes a similar amount of computation time. So, although only one deflation vector is used, the extra work which has to be done for deflation is considerably large.

Summarizing, increasing the number of deflation vectors on one hand reduces the number of CG-iterations, but on the other hand, results in CG-iterations that are computationally more expensive.
For the considered test problem, the deflation method does not give a reduction in CPU-time. At finer grids, the deflation method can be faster than the undeflated version. For example, at an open sea of 400×400 nodes, the undeflated diagonally scaled CG-method uses $1703\,s$ CPU-time, while for 100^2 subdomains, this is $1018\,s$.

14.6 The deflated RICCG-method

In previous sections, results are discussed about the relaxation parameter in the RICCG-method and the number of deflation vectors in the deflation method with diagonally scaling. It was concluded that $\omega = 1$ is the optimal choice as relaxation parameter. However, when RICCG is combined with deflation, this will not be the case anymore.
The spectral condition number of RIC-preconditioned matrices depends on the choice of relaxation parameter ω (see Section 10.5). For $\omega = 0$, the condition number is $\mathcal{O}(h^{-2})$, while for $\omega = 1$, we have $\mathcal{O}(h^{-1})$. This partially explaines the good performance of RICCG at $\omega = 1$. Figure 6 show that also the structure of the spectra of these extreme choices are quite different. For $\omega = 0$ there are some isolated low eigenvalues and larger eigenvalues

clustered around one. For $\omega = 1$ it is the other way around: small eigenvalues near one and large eigenvalues spreaded out.
The deflation method maps some vectors in the null-space, which leads to some zero eigenvalues of the deflated matrix. The remaining nonzero eigenvalues are approximately the same as the largest eigenvalues of the undeflated method, as is shown in Figure 12. Loosely speaking, the smallest eigenvalues are deflated to zero.
Applying the deflation method to MIC, so $\omega = 1$, will hardly reduce the spectral condition number. Since the smaller eigenvalues are clustered around one, the smallest active eigenvalue of the deflated matrix is also approximately one. The largest eigenvalue hardly changes with subdomain deflation and therefore the deflation method does not reduce the condition number of the MIC-preconditioner.
In the case of $\omega = 0$, the deflation method is effective. The smallest few eigenvalues with values of $\mathcal{O}(h^2)$ are deflated to zero, while the largest eigenvalue remains $\mathcal{O}(1)$. This reduction of the spectral condition number of the IC-preconditioner will probably give better convergence.

Number of subdomains	Mean number CG-iterations	CPU-time 1000 solves	Mean number CG-iterations	CPU-time 1000 solves
	$\omega = 0$		$\omega = 1$	
0×0	16.357	71.02	11.450	50.35
1×1	16.356	104.18	11.450	74.48
5×5	16.332	103.89	11.450	74.85
10×10	15.500	99.98	11.449	75.47
20×20	13.128	87.27	11.443	76.51
40×40	9.662	76.11	11.398	87.98
80×80	7.540	120.83	9.975	153.10

Table 12: Results at open sea of 200×200 nodes, with DRICCG and varying number of deflation vectors.

In Table 12 results of the deflated IC and MIC preconditioned CG-method is given. There is a clear difference between the two preconditioners. As expected, the number of CG-iterations decreases sharply for $\omega = 0$, while it is almost constant for $\omega = 1$.

The advantage of $\omega = 1$ is the small number of CG-iterations in the undeflated method. While $\omega = 0$ has a large reduction in CG-iterations in the deflated version. By choosing the relaxation parameter in between those extreme choices, the advantages of these methods can be combined. However, also the disadvantages are combined and therefore the optimal choice is not clear in advance.
In Table 13 some results are given for several choices of the relaxation parameter. Without deflation, the number of CG-iterations and also CPU-time is monotonically decreasing for increasing relaxation parameter. When deflation is applied, the number of CG-iterations is decreasing faster for small values of ω. In the case of 70^2 deflation vectors, $\omega = 0.75$ is the optimal choice. Unfortunately, there is no clear pattern in the combined influence of relaxation parameter and number of deflation vectors on the performance of DRICCG.

	# CG-it.	CPU-time	# CG-it.	CPU-time
# subd.	0×0		70×70	
$\omega = 0$	32.9	981.0	13.8	657.3
$\omega = 0.25$	30.9	924.7	13.5	598.8
$\omega = 0.5$	28.2	845.0	13.2	589.5
$\omega = 0.75$	24.3	738.1	12.6	566.8
$\omega = 1$	17.1	525.3	16.6	744.1

Table 13: Results at open sea of 400×400 nodes, with DRICCG and varying relaxation parameter ω.

Considering only the number of CG-iterations, the main observation is that for small numbers of deflation vectors, $\omega = 1$ gives the lowest number of CG-iterations. However, the convergence is improving very slow for larger numbers of deflation vectors. On the contrary, $\omega = 0$ converges slow for undeflated CG, but the number of CG-iterations decreases sharp when deflation is applied. For small subdomains, so many deflation vectors, the convergence of RICCG at $\omega = 0$ becomes better than for $\omega = 1$. And values of ω in between zero and one are in many cases better than both extreme choices.
For the CPU-time, we can conclude that given a value of ω, deflation may reduce the CPU-time needed, and given the number of deflation vectors, relaxation may improve the CG-method. However, the undeflated MICCG-method uses in most cases the least CPU-time.

14.6.1 Using Lapack inside the deflation method

In the deflation method, a linear system has to be solved on the coarse grid. This is done with a complete Cholesky decomposition. Both an own implemented `C++` routine and a `Lapack` routine have been used.
For small numbers of deflation vectors the Cholesky decomposition is also small and will use only a small part of the total computation time. However, the `C++` implementation is slightly faster. When using small subdomains, the Cholesky decomposition will be more important and will consume the largest part of the CPU-time. For these large decompositions, the `Lapack` routines can reduce the total CPU-time a lot. At intermediate choices, the differences are small.

14.6.2 Estimating the spectral condition number

In Section 10.5 it has been shown that Richardson's extrapolation estimates the order of the number of CG-iterations quite well. Thus also a good estimate of the order of the spectral condition number. For the undeflated RIC preconditioner, the condition number of $\mathcal{O}(h^{-2})$ for $\omega = 0$ and $\mathcal{O}(h^{-1})$ for $\omega = 1$ have been estimated. With the results from the deflation method, the same can be done for the deflated RICCG-method.
As test problem an open sea is considered for different mesh sizes. To compare the results, the same number of subdomains are used. Results for the deflated ICCG method are given in Table 14.

	0 subd.	20^2 subd.	50^2 subd.
100×100	7.822	6.773	4.879
200×200	16.357	13.128	8.732
400×400	32.895	25.867	16.087
800×800	67.233	52.541	30.211

Table 14: Mean number of DICCG-iterations at open sea for constant number of subdomains.

For zero subdomains, the results of Table 2 for $\omega = 0$ are obtained. Considering more subdomains, gives a reduction in the number of CG-iterations. Applying Richardson's extrapolation to 20^2 and 40^2 subdomains gives values of approximately one. So the number of iterations is $\mathcal{O}(h^{-1})$, which implies a spectral condition number of $\mathcal{O}(h^{-2})$. From the results it is therefore concluded that taking a constant number of subdomain deflation vectors does reduce the number of CG-iterations, but does not reduce the order of convergence.

Another way of comparing the results is by taking the number of nodes per subdomain constant with mesh-refinement. For example, a 100×100 grid is divided in 5^2 subdomains of 20×20 nodes, than the 200×200 grid is divided in 10^2 subdomains of 20×20 nodes. The size of the subdomains remains constant, but the number of deflation vectors increases for smaller mesh sizes.

	20^2 nodes	10^2 nodes	5^2 nodes
100×100	7.932	7.768	6.773
200×200	15.500	13.128	9.662
400×400	25.867	18.006	12.571
800×800	35.170	22.755	15.299

Table 15: Mean number of DICCG-iterations at open sea for constant number of nodes per subdomain.

In Table 15 results are shown for $\omega = 0$. Applying Richardson's extrapolation does not give useful results. But one can see that for two times smaller mesh size, the number of CG-iterations is less than two times more, which implies an spectral condition number smaller than $\mathcal{O}(h^{-2})$.

14.7 The deflated RRB method

In previous sections, results have been given for the deflation method combined with diagonal scaling and the relaxed incomplete Cholesky preconditioner. The RRB preconditioner can also be combined with deflation, however, this is not implemented because of the reasons presented below.
The deflated RRB method is quite difficult to implement, because of the relatively difficult RRB-ordering of nodes and because the CG-iteration is performed on the first Schur-complement only. This implies that the implementation of the deflation method for diagonal scaling and the RIC-preconditioner can not be directly used to the RRB-method.
Besides implementational issues, an improvement of the performance is not expected. Deflation reduces the spectral condition number. However, the RRB-method has already a small

spectral condition number and needs only a very few CG-iterations. A smaller condition number will therefore hardly give a reduction in CG-iterations for the RRB-preconditioner.
In Section 14.6 it is explained that the deflation method hardly improves the MIC-preconditioner, because the lowest eigenvalues of a MIC preconditioned matrix are clustered around one (see Figure 6). In [6] it has been proven that for the discrete Poisson equation, all eigenvalues of a RRB-k preconditioned matrix are larger than one, i.e.,

$$\lambda_{M^{-1}S} \geq 1.$$

The discrete model equation (8.1) is similar to the discrete Poisson equation. Because of the lower bound of the spectrum, it is likely that the smallest eigenvalues are clustered. Therefore, the spectrum of the RRB-k preconditioner will probably have the same structure as the spectrum of MIC.
Summarizing, the deflated RRB method is not implemented because implementation is quite difficult. Moreover, it is likely that deflation will not improve the convergence of RRB enough to reduce the computation time considerable.

14.8 Parallel RIC-preconditioner

To speed up the solvers, one can choose to parallelize the method over several computers. Expressions like inner products can be parallelized without any changes to the algorithm. But most computing time of the PCG-method is solving the preconditioned system. In general, this is not easily parallelized. However, one can change the preconditioner such that it is easier to parallelize, at the expense of some extra CG-iterations.
Let's consider the RIC-preconditioner $M = LL^T$, with L an lower triangular matrix. Solving the preconditioner is done with forward and backward substitution, which are recursively and thus not parallelizable. A similar preconditioner is the block-RIC preconditioner, for which L is a block matrix. For the case of two blocks we have $L = \begin{bmatrix} L_1 & 0 \\ 0 & L_2 \end{bmatrix}$, with L_1 and L_2 lower triangular matrices. Solving a system $Lx = b$ reduces to $L_1 x_1 = b_1$ and $L_2 x_2 = b_2$ which are independent of each other and can therefore be simultaneously solved on two different computers.
The RIC-preconditioner uses a decomposition LL^T on the whole grid. To parallelize this, the decomposition will be done on subdomains. The same rectangular subdomains as in the deflation method will be used. On each subdomain, an incomplete Cholesky decomposition is performed, thus ignoring the coupling between subdomains in the preconditioner. Solving the preconditioner can than be done for each subdomain separately, thus paralellizable. For the correct numbering of nodes, this reduces to the block incomplete Cholesky decomposition.
Because this preconditioner ignores the coupling between subdomains, it will be less accurate and the number of CG-iterations will increase. One can try to restore this coupling with the deflation method. The deflation method will require extra computation time per CG-iteration, but will reduce the number of CG-iterations. Most components of the deflation method can be parallelized, except of solving the coarse grid matrix.
The RIC-preconditioner per subdomain is implemented and in Table 16 some results are shown. In the undeflated version, it is clearly seen that the number of CG-iterations increases for an increasing number of subdomains. Note that in the limit, so 400^2 subdomains, the block-RIC preconditioner reduces to diagonal scaling, which requires 105.5 CG-iterations.

# subd.	# CG-iter.	CPU-time	# CG-iter.	CPU-time	# CG-iter.	CPU-time	# CG-iter.	CPU-time
	$\omega = 0$		$\omega = 1$		$\omega = 0$		$\omega = 1$	
0×0	32.90	975.6	17.14	501.3	32.90	945.8	17.14	520.7
10×10	42.00	1280.5	46.00	1147.6	39.97	1569.2	46.35	1828.7
100×100	56.67	1717.5	55.20	1371.3	16.23	902.9	16.32	916.4
200×200	74.33	2435.7	75.01	2021.7	10.64	1848.5	10.79	1870.2

Table 16: Results of RIC per subdomain, at open sea of 400×400 nodes; left without deflation, right with deflation.

In combination with deflation, the number of CG-iterations increases initially, but at large numbers of subdomains, less iterations are needed. Although every iteration becomes more expensive with deflation, the CPU-time is in some cases less than in the undeflated method. However, at large numbers of subdomains, solving the coarse grid matrix in the deflation method becomes more important and will reduce the ability of parallelization.
In practice, parallelization is done over a few computers. Numbers like 100^2 are too large, and therefore it is not needed to make a block preconditioner on such large numbers of subdomains. Deflation has only a positive effect on large numbers of subdomains. The number of subdomains used in the preconditioner should thus be smaller than in the deflation method. This can be done by making a block preconditioner on, for example, 2^2 subdomains, and applying the deflation method to 50^2 subdomains.
The presented results show that the parallel preconditioner may lead to faster calculations. Whether this is really the case and how to choose the parameters will be future research.

14.9 Termination criterium

To solve the discrete model equation (8.1), the CG-method is used with several preconditioners. Because it is an iterative method, a termination criterium has to be specified. This can be done in several ways, some of them will be discussed in this section.
Solving a system $Sx = b$ with an iterative method results in a sequence $x_0, x_1, \ldots, x_n$ of consecutive approximations of the solution x. A good estimate of the error at iteration i is the residual $||b - Sx_i||$, with $|| \cdot ||$ some norm. In the CG-method, the residual r_i is updated every iteration and approximately the same as the exact residual, i.e., $r_i \approx b - Sx_i$. Because r_i is readily available at every iteration, termination criteriums are based on this residual.

14.9.1 Absolute and relative criteria

A common choice for the termination criterium is

$$||r_i|| \quad \leq \quad \epsilon,$$

which is an *absolute termination criterium* and $\epsilon > 0$ denoting the *termination value* or *tolerance*. A drawback of this choice is that it is not scaling invariant. To explain this, consider a wave field at a grid with mesh size h and the same wave field at a grid with $2h$ mesh size. The norm of the wave height vector is approximately four times larger for h than

for $2h$, altough the same wave field is used. The same holds for residuals, and therefore, mesh refinement will give an unwanted change in termination criterium.
This can be solved by considering *relative termination criteria*, for instance

$$||r_i|| \quad \leq \quad \frac{\epsilon}{h^2},$$

with h denoting the characteristic mesh size. Other common choices are

$$||r_i|| \quad \leq \quad \epsilon\,||r_0||,$$

relative to the inital residual $r_0 = b - Sx_0$, and

$$||r_i|| \quad \leq \quad \epsilon\,||b||,$$

relative to the right hand side b. Note that b may depend on the mesh size h.
These relative termination criteria are scaling invariant, so mesh-refinement does not lead to more stringent criteria. However, the right hand side and also initial residual depends on the model variables in the wave model. As can be concluded from Equation (7.9b), the right hand side b explicitly depends on the velocity potential. A wave pattern with only a few waves therefore gives a small b, while a fully developed wave pattern results in a large norm of b. In order to make this more precise, the norms of the right hand side b and initial residual $b - Sx_0$ are calculated at the test problem of an open sea.

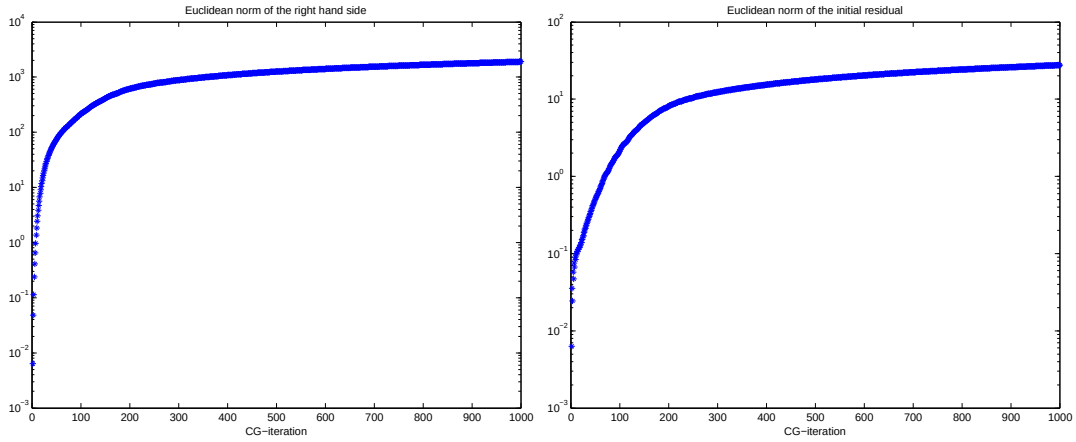

Figure 20: Euclidean norm of the right hand side and the initial residual at an open sea of 200×200 nodes.

The norms shown in Figure 20 are increasing in time. The zero initial state gives a small norm at the start of the simulation. Later on, the waves are propagated into the whole domain, resulting in a larger norm of the right hand side and initial residual. Observe that the differnce in the norms at the start and at the end is some orders of magnitude. A relative termination criterium will therefore be very stringent at the start of the simulation, which may lead to problems with rounding errors.
The initial solution needed for the CG-method is in the wave model given by the value on the previous time step, thus depending on the timestep used. Because of the real-time simulation, the timestep is quite small and will give a good initial estimate. An absolute criterium is independent on the initial solution, but the termination criterium relative to the initial residual may become unnecessary stringent.

14.9.2 Termination criterium based on the preconditioned residual

The vector r_i in iteration i of the PCG-method is related to the residual $\bar{r}_i = b - Sx_i$. Depending on the implementation (the two choices of PCG in Table 1), it represents the residual $\bar{r}_i$ or the preconditioned residual $P^{-1}\bar{r}_i$. In the first case, both r_i and $z_i = M^{-1}r_i$ are calculated in every CG-iteration, whereas in the second case only $P^{-1}r_i$ is available. For both cases, the value $\rho_i = \langle r_i, M^{-1}r_i\rangle_2 = \langle P^{-1}r_i, P^{-1}r_i\rangle_2 = \langle r_i, r_i\rangle_{M^{-1}} = ||r_i||^2_{M^{-1}}$ is calculated (remember that $PP^T = M \simeq S$). Although $||r_i||_2$ can also be calculated, it is not directly available in the CG-method. It is therefore logical to consider termination criteria based on the preconditioned residual.
The absolute termination criterium becomes

$$||r_i||_{M^{-1}} \quad \leq \quad \epsilon.$$

The termination criterium clearly depends on the preconditioner M. This implies that for some preconditioners the termination criterium becomes more stringent than for other choices of preconditioner. In order to quantify this, in next section some residuals are calculated.

14.9.3 Different residual norms

In order to compare the residual for different preconditioners, some simulations have been performed on an open sea of 200×200 nodes.
The residual r_i in an CG-iteration should be the same as the residual $b - Sx_i$. For different preconditioners, it is checked whether $r_n = b - Sx_n$, with x_n the final estimate of PCG. In the simulations the difference between the two is very small, so r_i estimates the residual well. Because the termination criterium can be based on the preconditioned residual, the norm of it may not change a lot for different preconditioners.

	Dgs	IC	MIC	RRB
$\|\|r_n\|\|_{M^{-1}}$	1.8456e-06	1.2516e-06	7.823e-07	1.1571e-06
$\|\|r_n\|\|_2$	1.4369e-05	7.0015e-06	3.9281e-06	6.9868e-06

Table 17: Residual in two different norms at open sea of 200×200 nodes.

Let's consider an open sea of 200×200 nodes with different preconditioners. At time t_{500} the residual r_n is given, which is the first residual satisfying the termination criterium. The norm of both the preconditioned and the unpreconditioned residual is calculated and shown in Table 17. There is a clear difference between the two norms, however, the difference in value is not very large. Between the different preconditioners, the differences are also small.

14.10 Conluding remarks on the results

In Section 14.1, the results have been introduced with some assessment criteria. In the next sections 14.2 to 14.9, results have been discussed for several test problems and linear solvers. In this sections, some conclussions are briefly given.

Efficiency The CPU-time used by the linear solvers have been presented for several values of the different parameters. Because of the good initial solution and the fast convergence of the iterative methods, the solution of the linear system of equation is obtained in a small

amount of CPU-time. For the coarser grids, it is smaller than the time step of the wave model, giving a real-time method. At the open sea a grid of up to 200×200 nodes could be solved within this time limit.
In most cases the RRB-method gives the lowest amount of computation time, while the MICCG method is almost as fast. The deflation method is able to speed up the RICCG method at finer grids.
The storage of the preconditioner is quite small. Because of the sparse preconditioners, only a few vectors have to be stored. The deflation method requires more storage. The deflation vectors are cheaply stored as subdomains, but the coarse grid matrix is stored as a complete Cholesky decomposition, which may be relatively large for small subdomains.

Convergence with mesh-refinement The diagonally scaled CG-method has a spectral condition number of $\mathcal{O}(h^{-2})$. When considering a grid with a two times smaller mesh size, the number of CG-iterations will increase with a factor two. This method is therefore not well suitable for finer grids. The RICCG method has a spectral condition number between $\mathcal{O}(h^{-2})$ and $\mathcal{O}(h^{-1})$, depending on the relaxation parameter. This can be reduced with deflation. For the RRB method, the spectral condition number is $\mathcal{O}(h^{-1})$. With the complete Cholesky decomposition on the maximum level, it can be further decreased to approximately $\mathcal{O}(h^{-\frac{1}{2}})$, depending on the choice of maximum level. Although this order of convergence is not reached for the test problems, the results show that the convergence is better than $\mathcal{O}(h^{-1})$ and therefore the extra computational effort with mesh refinement is small.

Parallelization The diagonally scaled preconditioner can straightforwardly be parallelized. The RRB preconditioner uses several levels of red-black orderings. At each level, the algorithm can be parallelized. Due to recursion, parallelization is limited to one level each time. The RICCG method can be changed to a parallel equivalent, which uses the RIC decomposition on several subdomains. There is no coupling between the subdomains, giving good parallel properties. Results show that the increase in computation time is relatively small. In a parallel environment, the wall-clock time can therefore be reduced considerably. The deflation method also uses subdomains, which gives a straightforward parallel implementation. Only the coarse grid system is not parallelized easily, especially for a large number of deflation vectors. In practice, the number of deflation vectors is much smaller than the total number of nodes.
The implemented deflated RICCG method is inherently parallel. Only the communication has to be added to have a full parallel implementation. The performance results on a sequential computer are quite promising.

Robustness The linear solvers are applied to different test problems. In all cases, convergence is achieved without major problems. Also the domain decomposition methods, like deflation and the block IC preconditioner, were able to solve the test problems with changing bathymetries. It can therefore be concluded that the solvers are robust.

15 Conclusions

This master's thesis started with a literature study about the variational Boussinesq model, which has recently been developed at MARIN. The derivation of this wave model starts with basic equations in fluid mechanics. Minimizing the pressure results in a variational formulation of the fluid motions. By applying a vertical shape function, the three-dimensional model reduces to a two-dimensional problem. To reduce the computational effort further, the model equations have been linearized. By presenting the derivation of the variational Boussinesq model, this thesis gives a full description of the wave model.
The finite volume and leapfrog method have been used to discretize the variational Boussinesq model. Results of the model show realistic wave patterns for varying water depth and near coastal structures. Applying a simple model of the motions of the ship to the variational Boussinesq model results in realistic waves patterns around the ship.
After the review of the physical properties and assumptions of the wave model, the literature study has concentrated on the solution method of the model equations. The discrete equivalent of one of the model equations is a linear system of equations, which has to be solved in every time step of the variational Boussinesq model. The wave model will be used in a real-time ship simulator and therefore needs a very efficient linear solver. Because of the properties of the matrix, the conjugate gradient method has been used as linear solver. This iterative method has been combined with three different preconditioners, namely diagonal scaling, modified incomplete Cholesky and repeated red-black.
In order to improve the efficiency of the model, research has been carried out for improving the linear solver. Two of the preconditioners have been improved and a new method, namely deflation, is applied.
The existing modified incomplete Cholesky preconditioner has been extended to the relaxed incomplete Cholesky preconditioner. Analysis of this method shows that taking the relaxation parameter equal to one is the best choice.
The original implementation of the repeated red-black preconditioner uses recursively a red-black ordering on as many as possible levels. This method has been changed to a repeated red-black preconditioner for a predefined number of levels, combined with a complete Cholesky decomposition on the maximum level. Results for test problems with larger numbers of nodes show a good performance, only a small increase in the number of iterations occurs. With the extension of a variable maximum level in the red-black orderings, the computation time is reduced with 5 to 25% depending on the test problem. Also the increase in number of iterations for mesh-refinement is smaller.
The preconditioners for the conjugate gradient method are combined with the deflation method. The deflation method uses a projection of several vectors into the null-space. This gives a smaller number of iterations, but each iteration becomes more expensive. Subdomain deflation has been implemented in the code of the wave model, with rectangular subdomains and piecewise-constant deflation vectors. The combination of deflation and diagonal scaling reduces the computation time only for large linear systems.
Combining the deflation method with the relaxed incomplete Cholesky preconditioner has led to a remarkable relation between deflation and the relaxation parameter. Due to different structures of the spectra, at low values of the relaxation parameter, deflation reduces the number of iterations considerably, while for large values of the relaxation parameter deflation hardly improves the convergence. Because the undeflated method performs better at large relaxation parameters, intermediate choices of the relaxation parameter give optimal

convergence when combined with deflation. The subdomain deflation method uses domain decomposition, which makes the use of shared memory natural. The implemented combination of subdomain deflation and block incomplete Cholesky is suitable for a parallel environment. The conjugate gradient method with the three different preconditioners and the deflation method has been tested on different bathymetries. When the different methods are compared with each other, the repeated red-black preconditioner is in most cases the method with the lowest amount of computation time. The deflated relaxed incomplete Cholesky performs almost as well, but has better properties for parallelizing the linear solver.

16 Future research

The research of the thesis has been concentrated mainly on the improvement of the linear solvers in the variational Boussinesq model. Three preconditioners have been investigated: diagonal scaling, relaxed incomplete Cholesky and repeated red-black. They have been combined with the deflation method. Although these methods are performing quite well, there are some possibilities for further improving the performance. In this section some topics for future research are briefly discussed.

The main topic of future research is likely to be the parallelization of the methods, thus using several computing units during one iteration of the model. The matrix-vector multiplications and inner products can be parallelized within the current algorithm. However, most computation time is used in the preconditioning step. As is briefly explained for the incomplete Cholesky preconditioner, the preconditioner itself can be changed to make it better parallelizable. The considered domain decomposition makes the algorithm particularly suited for distributed memory parallel computers. Some results of this parallel preconditioner are given in the thesis, but the actual distribution over several systems still has to be done. Combined with deflation, the parallel preconditioner may lead to smaller wall-clock times.

The methods have been implemented in the programming language `C++` and calculated on the CPU (central processing unit) of a usual desktop system. In recent years, computational methods have been implemented on the GPU (graphics processing unit) with promising results. Using the GPU for computing the wave model requires a different implementation, but can improve the computation time.

The deflation method, as considered in this thesis, uses deflation vectors according to rectangular subdomains. One can use different kinds of subdomains, for example based on physical parameters like depth, which could take all dry nodes as one subdomain. The deflation vectors are chosen constant in each subdomain; using piecewise-linear vectors can be used too. Especially for the case of the modified incomplete Cholesky preconditioner, other deflation vectors should be chosen.

Another numerical method which can be applied for solving a linear system of equation is the multigrid method. The main advantage of the multigrid method is its good performance with mesh-refinement. The RRB-preconditioner already has a small order of convergence and performs therefore almost as well with mesh-refinement as multigrid methods. Although its good convergence properties, multigrid methods can be quite expensive and are more difficult to parallelize than the incomplete Cholesky preconditioners. Deflation methods are able to improve the convergence of preconditioners and can be parallelized relatively easy. For this reasons, the research in this thesis has concentrated on the deflation method combined with the given preconditioners. But multigrid methods are still promising and especially on much finer grids can possibly perform better than the given methods.

A List of symbols used in the variational Boussinesq model

Following a list containing the unit and name of most of the symbols used in the variational Boussinesq model.

symbol	unit	name
ζ	m	water level
h	m	water depth
ϕ	$\frac{m^2}{s}$	velocity potential
φ	$\frac{m^2}{s}$	surface velocity potential
ρ	$\frac{kg}{m^3}$	mass density
t	s	time
x, y	m	horizontal coordinates
z	m	vertical coordinate
$\mathbf{u} = (u, v, w)$	$\frac{m}{s}$	velocity
g	$\frac{m}{s^2}$	gravity
p	$Pa = \frac{kg}{ms^2}$	pressure
$\mathcal{P}$	$\frac{m^5}{s^2}$	total pressure
$\mathcal{H}$	$\frac{m^5}{s^2}$	Hamiltonian
$\mathcal{L}$	$\frac{m^5}{s}$	Lagrangian
$f_m^{(p)}$	m	vertical shape function for the parabolic model
$f_m^{(c)}$	–	vertical shape function for the cosine-hyperbolic model
$\psi_m^{(p)}$	$\frac{m}{s}$	horizontal shape function for the parabolic model
$\psi_m^{(c)}$	$\frac{m^2}{s}$	horizontal shape function for the cosine-hyperbolic model
κ_m	$\frac{1}{m}$	shape parameter
N_x, N_y	–	number of grid points in x- resp. y-direction
L_x, L_y	m	length of computational domain in x- resp. y-direction
$\Delta x, \Delta y$	m	mesh size in x- resp. y-direction
h	m	characteristic mesh size

B Basic assumptions of the variational Boussinesq model

The basic assumptions of the nonlinear variational Boussinesq model are briefly summarized in this section.

- All functions are sufficiently smooth.
- Inviscid flow.
- Irrotational flow: $\nabla \times \mathbf{u} = 0$.
- Constant mass density of fluid in whole spatial and time domain, so incompressible flow.
- No other external forces than gravity, e.g. no wind stress and no Coriolis.
- The mild slope assumption gives $\nabla h = 0$.

Besides this, it is assumed that the vertical flow of the fluid can be realistically modelled with an expansion in shape parameters, in particular the parabolic or cosine-hyperbolic vertical shape model.
To obtain the model equation, it is assumed that the linearization may be performed, so relatively small water heights.

C Variational calculus

A basic concept in variational calculus is the first variation of a functional L and a function $u \in \mathcal{U}$, defined by

$$\delta\mathcal{L}(u;\delta u) \quad := \quad \lim_{\epsilon\to 0}\frac{\mathcal{L}(u+\epsilon\delta u)-\mathcal{L}(u)}{\epsilon}, \tag{C.1}$$

with δu a variation, satisfying $(u+\epsilon\delta u)\in\mathcal{U}$ [44]. The variation δu thus satisfies at least the properties of u. Often one can write the first variation as an inner product $\delta\mathcal{L}(u;\delta u) = (\delta\mathcal{L}(u),\delta u)$. In our case of water waves, the inner product $(u,v)=\int_A uv\ da$ is used. A zero first variation is a necessary condition for a stationary point: $\delta\mathcal{L}(u;\delta u)=0$, or $(\delta\mathcal{L}(u),\delta u)=0$. For continuous variation this is equivalent to

$$\delta\mathcal{L}(u) \quad = \quad 0, \tag{C.2}$$

the *Euler-Lagrange equation*.
As an example, the first variation of $P(\phi)=\frac{1}{2}(\nabla\phi)^2$ will be calculated:

$$\begin{aligned}
\delta P(\phi,\delta\phi) &= \lim_{\epsilon\to 0}\frac{P(\phi+\epsilon\ \delta\phi)-P(\phi)}{\epsilon}\\
&= \lim_{\epsilon\to 0}\frac{\frac{1}{2}\left(\nabla(\phi+\epsilon\ \delta\phi)\right)^2-\frac{1}{2}(\nabla\phi)^2}{\epsilon}\\
&= \lim_{\epsilon\to 0}\frac{\frac{1}{2}\left(\nabla\phi+\epsilon\nabla\delta\phi\right)^2-\frac{1}{2}(\nabla\phi)^2}{\epsilon}\\
&= \lim_{\epsilon\to 0}\frac{\frac{1}{2}(\nabla\phi)^2+\epsilon\nabla\phi\cdot\nabla\delta\phi+\frac{1}{2}\epsilon^2(\nabla\delta\phi)^2-\frac{1}{2}(\nabla\phi)^2}{\epsilon}\\
&= \lim_{\epsilon\to 0}\left(\nabla\phi\cdot\nabla\delta\phi+\frac{1}{2}\epsilon(\nabla\delta\phi)^2\right)\\
&= \nabla\phi\cdot\nabla\delta\phi.
\end{aligned} \tag{C.3}$$

For functionals depending on more functions, we write $\delta_\phi P(\phi,\zeta)=\delta P(\phi,\zeta;\delta\phi)$ the first variation w.r.t. ϕ and $\delta_\zeta P(\phi,\zeta)=\delta P(\phi,\zeta;\delta\zeta)$ the first variation w.r.t. ζ. A stationary point now have to satisfy both $\delta_\phi P(\phi,\zeta)=0$ and $\delta_\zeta P(\phi,\zeta)=0$ [33].
An important property in variational calculus is that the variation operator commutes with both the operations of differentiation and integration [33]. This property comes down to an interchange of two limit processes and therefore the functional should be smooth. Because the pressure functional is assumed to be smooth, we may interchange variations and differentiation or integration:

$$\delta_\phi\frac{\partial P(\phi(x))}{\partial x} \quad = \quad \frac{\partial}{\partial x}\delta_\phi P(\phi(x)), \tag{C.4a}$$

$$\delta_\phi\int P(\phi(x))\ dx \quad = \quad \int\delta_\phi P(\phi(x))\ dx. \tag{C.4b}$$

When the bounds of integration depend on the function of variation, the variation and integration cannot be interchanged anymore in this way. Then Leibniz's rule have to be applied.

D Depth averaged velocity

The depth averaged velocity is defined by

$$\mathbf{U} \;:=\; \frac{1}{h+\zeta}\int_{-h}^{\zeta} \mathbf{u}\; dz, \tag{D.1}$$

for the velocity $\mathbf{u} = \nabla\phi$. The velocity potential ϕ is given by the series expansion (3.1). Both the parabolic and the cosine-hyperbolic model consider only one shape function, so $\phi = \varphi + f\psi$. Only the horizontal components of $\mathbf{U}$ will be considered in the following derivations.
In the x-direction, the depth averaged velocity is given by

$$\begin{aligned} U &= \frac{1}{h+\zeta}\int_{-h}^{\zeta} u\; dz \\ &= \frac{1}{h+\zeta}\int_{-h}^{\zeta}\left(\frac{\partial\varphi}{\partial x} + f\frac{\partial\psi}{\partial x} + \frac{\partial f}{\partial\zeta}\frac{\partial\zeta}{\partial x}\psi\right) dz \\ &= \frac{\partial\varphi}{\partial x} + \frac{1}{h+\zeta}\frac{\partial\psi}{\partial x}\int_{-h}^{\zeta} f\; dz + \frac{1}{h+\zeta}\psi\frac{\partial\zeta}{\partial x}\int_{-h}^{\zeta}\frac{\partial f}{\partial\zeta}dz. \end{aligned} \tag{D.2}$$

The integrals are defined and calculated in Appendix E for two different shape models. With Equation (E.14), one has

$$\begin{aligned} U^{(p)} &= \frac{\partial\varphi}{\partial x} - \frac{1}{h+\zeta}\frac{\partial\psi}{\partial x}\frac{1}{3}(h+\zeta)^2 - \frac{1}{h+\zeta}\psi\frac{\partial\zeta}{\partial x}\frac{2}{3}(h+\zeta) \\ &= \frac{\partial\varphi}{\partial x} - \frac{1}{3}(h+\zeta)\frac{\partial\psi}{\partial x} - \frac{2}{3}\psi\frac{\partial\zeta}{\partial x}, \end{aligned} \tag{D.3}$$

the depth averaged velocity in the x-direction for a parabolic vertical shape. Because of symmetry in the x- and y-direction, the horizontal depth averaged velocity for the parabolic model is given by

$$\mathbf{U}^{(p)} \;=\; \nabla\varphi - \frac{1}{3}(h+\zeta)\nabla\psi - \frac{2}{3}\psi\nabla\zeta \tag{D.4}$$

and will be used in Section 3.2.
For the cosine-hyperbolic case (3.17), the integrals (E.28) give

$$U^{(c)} \;=\; \frac{\partial\varphi}{\partial x} - \mathcal{D}\,\frac{\partial\psi}{\partial x} - \kappa\,\mathcal{S}\,\psi\,\frac{\partial\zeta}{\partial x}, \tag{D.5}$$

with $\mathcal{D}$ and $\mathcal{S}$ as in (E.27). Because of symmetry,

$$\mathbf{U}^{(c)} \;=\; \nabla\varphi - \mathcal{D}\,\nabla\psi - \kappa\,\mathcal{S}\,\psi\nabla\zeta. \tag{D.6}$$

the horizontal depth averaged velocity for the cosine-hyperbolic model, used in Section 3.3.

E Detailed calculations for the Hamiltonian

The velocity potential is written as a series in vertical shape functions. To derive the Hamiltonian system for this potential, the series has to be substituted in the Hamiltonian. The zero variations will lead to three equations in the variables φ, ζ and ψ_m, which will be presented in this chapter.

E.1 Calculations for the general series model

The Hamiltonian for the general series model is given by (3.4):

$$\begin{aligned} H(\varphi,\zeta,\psi_m) &= \int_{-h}^{\zeta} \frac{1}{2}\Big(\nabla(\varphi+\sum_{m=1}^{M} f_m\psi_m)\Big)^2 dz + \frac{1}{2}g(\zeta^2-h^2) \\ &= \frac{1}{2}\int_{-h}^{\zeta}(\nabla\varphi)^2\,dz + \int_{-h}^{\zeta}\nabla\varphi\cdot\nabla\big(\sum_{m=1}^{M} f_m\psi_m\big)\,dz \\ &\quad +\frac{1}{2}\int_{-h}^{\zeta}\Big(\nabla\big(\sum_{m=1}^{M} f_m\psi_m\big)\Big)^2 dz + \frac{1}{2}g(\zeta^2-h^2). \end{aligned} \tag{E.1}$$

The three integrals will be discussed separately.
Because φ is independent of z, we have

$$\frac{1}{2}\int_{-h}^{\zeta}(\nabla\varphi)^2\,dz = \frac{1}{2}(h+\zeta)(\nabla\varphi)^2. \tag{E.2}$$

For the second integral, the derivative to z inside the gradient vanishes, because φ independent of z. So $\nabla\varphi = (\frac{\partial\varphi}{\partial x},\frac{\partial\varphi}{\partial y},\frac{\partial\varphi}{\partial z}) = (\frac{\partial\varphi}{\partial x},\frac{\partial\varphi}{\partial y},0) = (\nabla\varphi,0)$. For the horizontal gradient, we get

$$\begin{aligned} \int_{-h}^{\zeta}\nabla\varphi\cdot\nabla\Big(\sum_{m=1}^{M} f_m\psi_m\Big)\,dz &= \int_{-h}^{\zeta}\nabla\varphi\cdot\sum_{m=1}^{M}\nabla(f_m\psi_m)\,dz \\ &= \int_{-h}^{\zeta}\nabla\varphi\cdot\sum_{m=1}^{M}\Big(\frac{\partial f_m}{\partial\zeta}(\nabla\zeta)\psi_m + f_m\nabla\psi_m\Big)\,dz \\ &= \nabla\varphi\cdot\nabla\zeta\int_{-h}^{\zeta}\sum_{m=1}^{M}\frac{\partial f_m}{\partial\zeta}\psi_m\,dz + \nabla\varphi\cdot\int_{-h}^{\zeta}\sum_{m=1}^{M} f_m\nabla\psi_m\,dz \\ &= \nabla\varphi\cdot\nabla\zeta\sum_{m=1}^{M}\psi_m\int_{-h}^{\zeta}\frac{\partial f_m}{\partial\zeta}\,dz + \nabla\varphi\cdot\sum_{m=1}^{M}\nabla\psi_m\int_{-h}^{\zeta} f_m\,dz \\ &=: \nabla\varphi\cdot\nabla\zeta\sum_{m=1}^{M}\psi_m Q_m + \nabla\varphi\cdot\sum_{m=1}^{M}(\nabla\psi_m)P_m. \end{aligned} \tag{E.3}$$

The third integral of (E.1) contains a gradient, which will be split into a horizontal and a

vertical part. The part with the horizontal gradient becomes

$$
\begin{aligned}
\frac{1}{2}\int_{-h}^{\zeta}\Big(\nabla(\sum_{m=1}^{M} f_m\psi_m)\Big)^2 dz &= \frac{1}{2}\int_{-h}^{\zeta}\Big(\sum_{m=1}^{M}\nabla(f_m\psi_m)\Big)^2 dz \\
&= \frac{1}{2}\int_{-h}^{\zeta}\left(\sum_{m=1}^{M}(f_m\nabla\psi_m)+\sum_{n=1}^{M}\Big(\frac{\partial f_n}{\partial\zeta}(\nabla\zeta)\psi_n\Big)\right)^2 dz \\
&= \frac{1}{2}\int_{-h}^{\zeta}\Big(\sum_{m=1}^{M} f_m\nabla\psi_m\Big)^2 dz \\
&\quad +\int_{-h}^{\zeta}\Big(\sum_{m=1}^{M} f_m\nabla\psi_m\Big)\cdot\Big(\nabla\zeta\sum_{n=1}^{M}\frac{\partial f_n}{\partial\zeta}\psi_n\Big)dz \\
&\quad +\frac{1}{2}\int_{-h}^{\zeta}\Big(\nabla\zeta\sum_{n=1}^{M}\frac{\partial f_n}{\partial\zeta}\psi_n\Big)^2 dz \\
&= \frac{1}{2}\int_{-h}^{\zeta}\Big(\sum_{m,n=1}^{M} f_m f_n\nabla\psi_m\nabla\psi_n\Big)dz \\
&\quad +\nabla\zeta\cdot\int_{-h}^{\zeta}\Big(\sum_{m,n=1}^{M} f_m\nabla\psi_m\frac{\partial f_n}{\partial\zeta}\psi_n\Big)dz \\
&\quad +\frac{1}{2}(\nabla\zeta)^2\int_{-h}^{\zeta}\Big(\sum_{m,n=1}^{M}\frac{\partial f_m}{\partial\zeta}\frac{\partial f_n}{\partial\zeta}\psi_m\psi_n\Big)dz \\
&= \frac{1}{2}\sum_{m,n=1}^{M}\Big(\nabla\psi_m\nabla\psi_n\int_{-h}^{\zeta} f_m f_n\ dz\Big) \\
&\quad +\nabla\zeta\cdot\sum_{m,n=1}^{M}\Big((\nabla\psi_m)\psi_n\int_{-h}^{\zeta} f_m\frac{\partial f_n}{\partial\zeta}\ dz\Big) \\
&\quad +\frac{1}{2}(\nabla\zeta)^2\sum_{m,n=1}^{M}\Big(\psi_m\psi_n\int_{-h}^{\zeta}\frac{\partial f_m}{\partial\zeta}\frac{\partial f_n}{\partial\zeta}\ dz\Big) \\
&=: \frac{1}{2}(\nabla\zeta)^2\sum_{m,n=1}^{M}\psi_m\psi_n G_{mn} \\
&\quad +\nabla\zeta\cdot\sum_{m,n=1}^{M}(\nabla\psi_m)\psi_n R_{mn} \\
&\quad +\frac{1}{2}\sum_{m,n=1}^{M}\nabla\psi_m\nabla\psi_n F_{mn}. \qquad \text{(E.4)}
\end{aligned}
$$

For the z-direction the integral reads[25]

$$\begin{aligned}
\frac{1}{2}\int_{-h}^{\zeta}\Big(\frac{\partial}{\partial z}\Big(\sum_{m=1}^{M} f_m\psi_m\Big)\Big)^2 dz &= \frac{1}{2}\int_{-h}^{\zeta}\Big(\sum_{m=1}^{M}\frac{\partial}{\partial z}(f_m\psi_m)\Big)^2 dz \\
&= \frac{1}{2}\int_{-h}^{\zeta}\Big(\sum_{m=1}^{M}\psi_m\frac{\partial f_m}{\partial z}\Big)^2 dz \\
&= \frac{1}{2}\int_{-h}^{\zeta}\sum_{m=1}^{M}\sum_{n=1}^{M}\psi_m\psi_n\frac{\partial f_m}{\partial z}\frac{\partial f_n}{\partial z}\ dz \\
&= \frac{1}{2}\sum_{m,n=1}^{M}\Big(\psi_m\psi_n\int_{-h}^{\zeta}\frac{\partial f_m}{\partial z}\frac{\partial f_n}{\partial z}\ dz\Big) \\
&=: \frac{1}{2}\sum_{m,n=1}^{M}\psi_m\psi_n K_{mn}. \qquad \text{(E.5)}
\end{aligned}$$

The Hamiltonian density now equals

$$\begin{aligned}
H(\varphi,\zeta,\psi_m) &= \frac{1}{2}(h+\zeta)(\nabla\varphi)^2 + \frac{1}{2}\sum_{m,n=1}^{M} F_{mn}\nabla\psi_m\cdot\nabla\psi_n + \frac{1}{2}(\nabla\zeta)^2\sum_{m,n=1}^{M} G_{mn}\psi_m\psi_n \\
&\quad + \frac{1}{2}\sum_{m,n=1}^{M} K_{mn}\psi_m\psi_n + \nabla\varphi\cdot\sum_{m=1}^{M} P_m\nabla\psi_m + \nabla\varphi\cdot\nabla\zeta\sum_{m=1}^{M} Q_m\psi_m \\
&\quad + \nabla\zeta\cdot\sum_{m,n=1}^{M} R_{mn}\psi_n\nabla\psi_m + \frac{1}{2}g(\zeta^2-h^2). \qquad \text{(E.6)}
\end{aligned}$$

This Hamiltonian density can be substituted into the Hamiltonian system (2.35),(3.3c).
The continuity equations becomes

$$\begin{aligned}
\int\iint\frac{\partial\zeta}{\partial t}\delta\varphi\ dx\,dy\ dt &= \int\iint\delta_\varphi H(\varphi,\zeta,\psi_m)\ dx\,dy\ dt \\
&= \int\iint(h+\zeta)\nabla\varphi\cdot\nabla\delta\varphi\ dx\,dy\ dt + \int\iint\nabla\delta\varphi\cdot\sum_{m=1}^{M} P_m\nabla\psi_m\ dx\,dy\ dt \\
&\quad + \int\iint\nabla\delta\varphi\cdot\nabla\zeta\sum_{m=1}^{M} Q_m\psi_m\ dx\,dy\ dt \\
&= -\int\iint\nabla\cdot((h+\zeta)\nabla\varphi)\delta\varphi\ dx\,dy\ dt - \int\iint\delta\varphi\ \nabla\cdot(\sum_{m=1}^{M} P_m\nabla\psi_m)\ dx\,dy\ dt \\
&\quad - \int\iint\delta\varphi\ \nabla\cdot(\nabla\zeta\sum_{m=1}^{M} Q_m\psi_m)\ dx\,dy\ dt. \qquad \text{(E.7)}
\end{aligned}$$

The last equality holds by applying Green's theorem, and using the fact that the variation

[25] $(\sum_{m=1}^{M} f_m)^2 = \sum_{m=1}^{M}\sum_{n=1}^{M} f_m f_n$

vanishes at the boundaries. This equation is equivalent with

$$\frac{\partial \zeta}{\partial t} + \nabla \cdot \left((h+\zeta)\nabla\varphi + \sum_{m=1}^{M} P_m \nabla \psi_m + \nabla\zeta \sum_{m=1}^{M} Q_m \psi_m \right) = 0 \tag{E.8}$$

In a similar way the Bernoulli equation becomes

$$\frac{\partial \varphi}{\partial t} + \frac{1}{2}(\nabla\varphi)^2 + g\zeta + \mathcal{R} = 0, \tag{E.9}$$

with the non-hydrostatic term

$$\begin{aligned} \mathcal{R} := & \frac{1}{2}\sum_{m,n=1}^{M} F'_{mn} \nabla\psi_m \cdot \nabla\psi_n + \frac{1}{2}\sum_{m,n=1}^{M} \left((\nabla\zeta)^2 G'_{mn} + K'_{mn} \right) \psi_m \psi_n \\ & + \nabla\varphi \cdot \sum_{m=1}^{M} \left(P'_m \nabla\psi_m + Q'_m \psi_m \nabla\zeta \right) + \nabla\zeta \cdot \sum_{m,n=1}^{M} R'_{mn} \psi_n \nabla\psi_m \\ & - \nabla \cdot \left(\sum_{m=1}^{M} Q_m \psi_m \nabla\varphi + \sum_{m,n=1}^{M} \left(R_{mn}\psi_n \nabla\psi_m + G_{mn}\psi_m\psi_n \nabla\zeta \right) \right). \end{aligned} \tag{E.10}$$

The prime means a variation to ζ, so $F'_{mn} = \frac{\delta_\zeta F_{mn}}{\delta\zeta}$. Note that with Leibniz's rule and using $f|_{z=\zeta} = 0$, we get $P'_m = Q_m$ and $F'_{mn} = R_{mn} + R_{nm}$.
The zero first variation of the Hamiltonian to ψ_ℓ can be written as

$$\begin{aligned} & -\nabla \cdot \sum_{m=1}^{M} F_{\ell m} \nabla\psi_m + (\nabla\zeta)^2 \sum_{m=1}^{M} G_{\ell m}\psi_m + \sum_{n=1}^{M} K_{\ell m}\psi_m - \nabla\cdot(P_\ell \nabla\varphi) \\ & \qquad + Q_\ell \, \nabla\varphi\cdot\nabla\zeta + \nabla\zeta \cdot \sum_{m=1}^{M} R_{m\ell}\nabla\psi_m - \nabla\cdot\sum_{n=1}^{M} R_{\ell n}\psi_n\nabla\zeta = 0, \end{aligned} \tag{E.11}$$

or equivalently

$$\begin{aligned} & Q_\ell \, \nabla\varphi\cdot\nabla\zeta + \sum_{m=1}^{M} \left(K_{\ell m} + (\nabla\zeta)^2 G_{\ell m} \right)\psi_m + \nabla\zeta\cdot\sum_{m=1}^{M} R_{m\ell}\nabla\psi_m \\ & \qquad - \nabla\cdot\left(\sum_{m=1}^{M} F_{\ell m}\nabla\psi_m + P_\ell\nabla\varphi + \sum_{m=1}^{M} R_{\ell n}\psi_n\nabla\zeta \right) = 0, \end{aligned} \tag{E.12}$$

for $\ell = 1, 2, \ldots, M$.

E.2 Calculations for the parabolic shape function

The parabolic shape function (3.12) is given by

$$f^{(p)} = \frac{1}{2}(z-\zeta)\left(1 + \frac{h+z}{h+\zeta}\right). \tag{E.13}$$

The integrals are given by (see [21] or use `Maple`)

$$
\begin{aligned}
F^{(p)} &= \frac{2}{15}(h+\zeta)^3, && \text{(E.14a)}\\
G^{(p)} &= \frac{7}{15}(h+\zeta), && \text{(E.14b)}\\
K^{(p)} &= \frac{1}{3}(h+\zeta), && \text{(E.14c)}\\
P^{(p)} &= -\frac{1}{3}(h+\zeta)^2, && \text{(E.14d)}\\
Q^{(p)} &= -\frac{2}{3}(h+\zeta), && \text{(E.14e)}\\
R^{(p)} &= \frac{1}{5}(h+\zeta)^2. && \text{(E.14f)}
\end{aligned}
$$

The variations to ζ are

$$
\begin{aligned}
F'^{(p)} &= \frac{2}{5}(h+\zeta)^2, && \text{(E.15a)}\\
G'^{(p)} &= \frac{7}{15}, && \text{(E.15b)}\\
K'^{(p)} &= \frac{1}{3}, && \text{(E.15c)}\\
P'^{(p)} &= -\frac{2}{3}(h+\zeta), && \text{(E.15d)}\\
Q'^{(p)} &= -\frac{2}{3}, && \text{(E.15e)}\\
R'^{(p)} &= \frac{2}{5}(h+\zeta). && \text{(E.15f)}
\end{aligned}
$$

To get the parabolic Hamiltonian, these functionals can be substituted in the general series Hamiltonian (E.6). With some rearranging, the parabolic Hamiltonian density $H^{(p)}$ becomes[26]

$$
\begin{aligned}
H^{(p)} &= \frac{1}{2}(h+\zeta)(\nabla\varphi)^2 + \frac{1}{2}F^{(p)}(\nabla\psi)^2 + \frac{1}{2}(\nabla\zeta)^2 G^{(p)}\psi^2 + \frac{1}{2}K^{(p)}\psi^2\\
&\quad + P^{(p)}\nabla\varphi\cdot\nabla\psi + Q^{(p)}\psi\nabla\varphi\cdot\nabla\zeta + \nabla\zeta\cdot R^{(p)}\psi\nabla\psi + \frac{1}{2}g(\zeta^2 - h^2)\\
&= \frac{1}{2}(h+\zeta)(\nabla\varphi)^2 + \frac{1}{2}F^{(p)}(\nabla\psi)^2 + \frac{1}{2}(\nabla\zeta)^2 G^{(p)}\psi^2 + \frac{1}{2}K^{(p)}\psi^2\\
&\quad + P^{(p)}\nabla\varphi\cdot\nabla\psi + Q^{(p)}\psi\nabla\varphi\cdot\nabla\zeta + \nabla\zeta\cdot R^{(p)}\psi\nabla\psi + \frac{1}{2}g(\zeta^2 - h^2)\\
&= \frac{1}{2}(h+\zeta)\left((\nabla\varphi)^2 + \frac{2}{15}(h+\zeta)^2(\nabla\psi)^2 + (\nabla\zeta)^2\frac{7}{15}\psi^2\right) + \frac{1}{6}(h+\zeta)\psi^2\\
&\quad - \frac{1}{3}(h+\zeta)^2\nabla\varphi\cdot\nabla\psi - \frac{2}{3}(h+\zeta)\psi\nabla\varphi\cdot\nabla\zeta + \frac{1}{5}(h+\zeta)^2\psi\nabla\zeta\cdot\nabla\psi + \frac{1}{2}g(\zeta^2 - h^2)
\end{aligned}
$$

[26] $(a-b-c)^2 = a^2+b^2+c^2-2ab-2ac+2bc \Rightarrow \frac{1}{2}(a^2+b^2+c^2) = \frac{1}{2}(a-b-c)^2 + ab + ac - bc$

$$
\begin{aligned}
&= \frac{1}{2}(h+\zeta)\left((\nabla\varphi)^2+\left(\frac{2}{3}\psi\nabla\zeta\right)^2+\left(\frac{1}{3}(h+\zeta)\nabla\psi\right)^2\right)+\frac{1}{6}(h+\zeta)\psi^2\\
&\quad+\frac{1}{2}(h+\zeta)\left(\left(\frac{7}{15}-(\frac{2}{3})^2\right)(\psi\nabla\zeta)^2+\left(\frac{2}{15}-(\frac{1}{3})^2\right)((h+\zeta)\nabla\psi)^2\right)\\
&\quad-\frac{1}{3}(h+\zeta)^2\nabla\varphi\cdot\nabla\psi-\frac{2}{3}(h+\zeta)\psi\nabla\varphi\cdot\nabla\zeta+\frac{1}{5}(h+\zeta)^2\psi\nabla\zeta\cdot\nabla\psi+\frac{1}{2}g(\zeta^2-h^2)\\
&= \frac{1}{2}(h+\zeta)\left(\nabla\varphi-\frac{2}{3}\psi\nabla\zeta-\frac{1}{3}(h+\zeta)\nabla\psi\right)^2+\frac{1}{6}(h+\zeta)\psi^2\\
&\quad+(h+\zeta)\left(\frac{2}{3}\psi\nabla\varphi\cdot\nabla\zeta+\frac{1}{3}(h+\zeta)\nabla\varphi\cdot\nabla\psi-\frac{2}{9}(h+\zeta)\psi\nabla\zeta\cdot\nabla\psi\right)\\
&\quad+\frac{1}{2}(h+\zeta)\left(\frac{1}{45}(\psi\nabla\zeta)^2+\frac{1}{45}((h+\zeta)\nabla\psi)^2\right)\\
&\quad-\frac{1}{3}(h+\zeta)^2\nabla\varphi\cdot\nabla\psi-\frac{2}{3}(h+\zeta)\psi\nabla\varphi\cdot\nabla\zeta+\frac{1}{5}(h+\zeta)^2\psi\nabla\zeta\cdot\nabla\psi+\frac{1}{2}g(\zeta^2-h^2)\\
&= \frac{1}{2}(h+\zeta)\left(\nabla\varphi-\frac{2}{3}\psi\nabla\zeta-\frac{1}{3}(h+\zeta)\nabla\psi\right)^2+\frac{1}{6}(h+\zeta)\psi^2\\
&\quad+\frac{1}{90}(h+\zeta)\left((\psi\nabla\zeta)^2+((h+\zeta)\nabla\psi)^2\right)+\left(\frac{1}{5}-\frac{2}{9}\right)(h+\zeta)^2\psi\nabla\zeta\cdot\nabla\psi+\frac{1}{2}g(\zeta^2-h^2)\\
&= \frac{1}{2}(h+\zeta)\left(\nabla\varphi-\frac{2}{3}\psi\nabla\zeta-\frac{1}{3}(h+\zeta)\nabla\psi\right)^2\\
&\quad+\frac{1}{90}(h+\zeta)\left((\psi\nabla\zeta)^2-2(h+\zeta)\psi\nabla\zeta\cdot\nabla\psi+((h+\zeta)\nabla\psi)^2\right)\\
&\quad+\frac{1}{6}(h+\zeta)\psi^2+\frac{1}{2}g(\zeta^2-h^2)\\
&= \frac{1}{2}(h+\zeta)\Big(\nabla\varphi-\frac{2}{3}\psi\nabla\zeta-\frac{1}{3}(h+\zeta)\nabla\psi\Big)^2+\frac{1}{6}(h+\zeta)\psi^2\\
&\quad+\frac{1}{90}(h+\zeta)\Big(\psi\nabla\zeta-(h+\zeta)\nabla\psi\Big)^2+\frac{1}{2}g(\zeta^2-h^2)
\end{aligned}
\tag{E.16}
$$

For the Hamiltonian system, the continuity equation (E.8) becomes

$$
\begin{aligned}
\frac{\partial\zeta}{\partial t}+\nabla\cdot\left((h+\zeta)\nabla\varphi+P^{(p)}\nabla\psi+Q^{(p)}\psi\nabla\zeta\right) &= 0,\\
\frac{\partial\zeta}{\partial t}+\nabla\cdot\left((h+\zeta)\nabla\varphi-\frac{1}{3}(h+\zeta)^2\nabla\psi-\frac{2}{3}(h+\zeta)\psi\nabla\zeta\right) &= 0,\\
\frac{\partial\zeta}{\partial t}+\nabla\cdot\left((h+\zeta)\left(\nabla\varphi-\frac{2}{3}\psi\nabla\zeta-\frac{1}{3}(h+\zeta)\nabla\psi\right)\right) &= 0.
\end{aligned}
\tag{E.17}
$$

The Bernoulli-like equation (E.9) becomes

$$
\begin{aligned}
\frac{\partial\varphi}{\partial t}+\frac{1}{2}(\nabla\varphi)^2+g\zeta+\frac{1}{2}F'^{(p)}(\nabla\psi)^2+\frac{1}{2}\left((\nabla\zeta)^2G'^{(p)}+K'^{(p)}\right)\psi^2 \\
+\nabla\varphi\cdot\left(P'^{(p)}\nabla\psi+Q'^{(p)}\psi\nabla\zeta\right)+\nabla\zeta\cdot R'^{(p)}\psi\nabla\psi \\
-\nabla\cdot\left(Q^{(p)}\psi\nabla\varphi+R^{(p)}\psi\nabla\psi+G^{(p)}\psi^2\nabla\zeta\right) &= 0, \\
\frac{\partial\varphi}{\partial t}+\frac{1}{2}(\nabla\varphi)^2+g\zeta+\frac{1}{2}\frac{2}{5}(h+\zeta)^2(\nabla\psi)^2+\frac{1}{2}\left((\nabla\zeta)^2\frac{7}{15}+\frac{1}{3}\right)\psi^2 \\
+\nabla\varphi\cdot\left(-\frac{2}{3}(h+\zeta)\nabla\psi-\frac{2}{3}\psi\nabla\zeta\right)+\nabla\zeta\cdot\frac{2}{5}(h+\zeta)\psi\nabla\psi \\
-\nabla\cdot\left(-\frac{2}{3}(h+\zeta)\psi\nabla\varphi+\left(\frac{1}{5}(h+\zeta)^2\psi\nabla\psi+\frac{7}{15}(h+\zeta)\psi^2\nabla\zeta\right)\right) &= 0, \\
\frac{\partial\varphi}{\partial t}+\frac{1}{2}\left((\nabla\varphi)^2+\frac{2}{5}(h+\zeta)^2(\nabla\psi)^2+\frac{7}{15}\psi^2(\nabla\zeta)^2\right) \\
+\frac{1}{6}\psi^2-\frac{2}{3}(h+\zeta)\nabla\varphi\cdot\nabla\psi-\frac{2}{3}\psi\nabla\varphi\cdot\nabla\zeta+\frac{2}{5}(h+\zeta)\psi\nabla\zeta\cdot\nabla\psi \\
+\nabla\cdot\left((h+\zeta)\left(\frac{2}{3}\nabla\varphi-\frac{7}{15}\psi\nabla\zeta-\frac{1}{5}(h+\zeta)\nabla\psi\right)\psi\right)+g\zeta &= 0, \\
\frac{\partial\varphi}{\partial t}+\frac{1}{2}\left((\nabla\varphi)^2+\left(\frac{2}{3}(h+\zeta)\nabla\psi\right)^2+\left(\frac{2}{3}\psi\nabla\zeta\right)^2\right) \\
+\left(\frac{1}{2}\frac{2}{5}-\frac{1}{2}\left(\frac{2}{3}\right)^2\right)(h+\zeta)^2(\nabla\psi)^2+\left(\frac{1}{2}\frac{7}{15}-\frac{1}{2}\left(\frac{2}{3}\right)^2\right)\psi^2(\nabla\zeta)^2 \\
+\frac{1}{6}\psi^2-\frac{2}{3}(h+\zeta)\nabla\varphi\cdot\nabla\psi-\frac{2}{3}\psi\nabla\varphi\cdot\nabla\zeta+\frac{2}{5}(h+\zeta)\psi\nabla\zeta\cdot\nabla\psi \\
+\nabla\cdot\left((h+\zeta)\left(\frac{2}{3}\nabla\varphi-\frac{7}{15}\psi\nabla\zeta-\frac{1}{5}(h+\zeta)\nabla\psi\right)\psi\right)+g\zeta &= 0, \\
\frac{\partial\varphi}{\partial t}+\frac{1}{2}\left(\nabla\varphi-\frac{2}{3}(h+\zeta)\nabla\psi-\frac{2}{3}\psi\nabla\zeta\right)^2 \\
+\frac{2}{3}(h+\zeta)\nabla\varphi\cdot\nabla\psi+\frac{2}{3}\psi\nabla\varphi\cdot\nabla\zeta-\frac{2}{3}\frac{2}{3}(h+\zeta)\psi\nabla\psi\cdot\nabla\zeta \\
-\frac{1}{45}(h+\zeta)^2(\nabla\psi)^2+\frac{1}{90}\psi^2(\nabla\zeta)^2 \\
+\frac{1}{6}\psi^2-\frac{2}{3}(h+\zeta)\nabla\varphi\cdot\nabla\psi-\frac{2}{3}\psi\nabla\varphi\cdot\nabla\zeta+\frac{2}{5}(h+\zeta)\psi\nabla\zeta\cdot\nabla\psi \\
+\nabla\cdot\left((h+\zeta)\left(\frac{2}{3}\nabla\varphi-\frac{7}{15}\psi\nabla\zeta-\frac{1}{5}(h+\zeta)\nabla\psi\right)\psi\right)+g\zeta &= 0, \\
\frac{\partial\varphi}{\partial t}+\frac{1}{2}\left(\nabla\varphi-\frac{2}{3}(h+\zeta)\nabla\psi-\frac{2}{3}\psi\nabla\zeta\right)^2 \\
-\frac{1}{45}(h+\zeta)^2(\nabla\psi)^2+\frac{1}{90}\psi^2(\nabla\zeta)^2 \\
+\frac{1}{6}\psi^2+\left(\frac{2}{5}-\frac{2}{3}\frac{2}{3}\right)(h+\zeta)\psi\nabla\zeta\cdot\nabla\psi \\
+\nabla\cdot\left((h+\zeta)\left(\frac{2}{3}\nabla\varphi-\frac{7}{15}\psi\nabla\zeta-\frac{1}{5}(h+\zeta)\nabla\psi\right)\psi\right)+g\zeta &= 0,
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial\varphi}{\partial t}+\frac{1}{2}\left(\nabla\varphi-\frac{2}{3}(h+\zeta)\nabla\psi-\frac{2}{3}\psi\nabla\zeta\right)^2 & \\
+\frac{1}{6}\psi^2-\frac{1}{45}(h+\zeta)^2(\nabla\psi)^2-\frac{2}{45}(h+\zeta)\psi\nabla\zeta\cdot\nabla\psi+\frac{1}{90}\psi^2(\nabla\zeta)^2 & \\
+\nabla\cdot\left((h+\zeta)\Big(\frac{2}{3}\nabla\varphi-\frac{7}{15}\psi\nabla\zeta-\frac{1}{5}(h+\zeta)\nabla\psi\Big)\psi\right)+g\zeta &= 0,\\
\frac{\partial\varphi}{\partial t}+\frac{1}{2}\left(\nabla\varphi-\frac{2}{3}(h+\zeta)\nabla\psi-\frac{2}{3}\psi\nabla\zeta\right)^2+\frac{1}{6}\psi^2+\Big(\frac{1}{90}+\frac{1}{45}\Big)\psi^2(\nabla\zeta)^2 & \\
-\frac{1}{45}\Big((h+\zeta)^2(\nabla\psi)^2+2(h+\zeta)\psi\nabla\zeta\cdot\nabla\psi+\psi^2(\nabla\zeta)^2\Big) & \\
+\nabla\cdot\left((h+\zeta)\Big(\frac{2}{3}\nabla\varphi-\frac{7}{15}\psi\nabla\zeta-\frac{1}{5}(h+\zeta)\nabla\psi\Big)\psi\right)+g\zeta &= 0,\\
\frac{\partial\varphi}{\partial t}+\frac{1}{2}\left(\nabla\varphi-\frac{2}{3}(h+\zeta)\nabla\psi-\frac{2}{3}\psi\nabla\zeta\right)^2 & \\
+\frac{1}{6}\Big(1+\frac{1}{5}(\nabla\zeta)^2\Big)\psi^2-\frac{1}{45}\Big((h+\zeta)\nabla\psi+\psi\nabla\zeta\Big)^2 & \\
+\nabla\cdot\left((h+\zeta)\Big(\frac{2}{3}\nabla\varphi-\frac{7}{15}\psi\nabla\zeta-\frac{1}{5}(h+\zeta)\nabla\psi\Big)\psi\right)+g\zeta &= 0. \qquad \text{(E.18)}
\end{aligned}
$$

The elliptic equation (E.12) becomes

$$
\begin{aligned}
Q^{(p)}\nabla\varphi\cdot\nabla\zeta+\Big(K^{(p)}+(\nabla\zeta)^2G^{(p)}\Big)\psi+\nabla\zeta\cdot R^{(p)}\nabla\psi & \\
-\nabla\cdot\Big(F^{(p)}\nabla\psi+P^{(p)}\nabla\varphi+R^{(p)}\psi\nabla\zeta\Big) &= 0,\\
-\frac{2}{3}(h+\zeta)\nabla\varphi\cdot\nabla\zeta+\frac{1}{3}(h+\zeta)\psi+\frac{7}{15}(h+\zeta)\psi(\nabla\zeta)^2+\frac{1}{5}(h+\zeta)^2\nabla\zeta\cdot\nabla\psi & \\
-\nabla\cdot\left(\frac{2}{15}(h+\zeta)^3\nabla\psi-\frac{1}{3}(h+\zeta)^2\nabla\varphi+\frac{1}{5}(h+\zeta)^2\psi\nabla\zeta\right) &= 0,\\
(h+\zeta)\psi+\left(\frac{1}{3}+\frac{7}{15}(\nabla\zeta)^2\right)-\left(\frac{2}{3}(h+\zeta)\nabla\varphi+\frac{1}{5}(h+\zeta)^2\nabla\zeta\right)\cdot\nabla\psi & \\
+\nabla\cdot\left(\frac{1}{3}(h+\zeta)^2\nabla\varphi-\frac{1}{5}(h+\zeta)^2\psi\nabla\zeta-\frac{2}{15}(h+\zeta)^3\nabla\psi\right) &= 0. \qquad \text{(E.19)}
\end{aligned}
$$

E.3 Calculations for the cosine-hyperbolic shape function

The cosine-hyperbolic vertical shape function (3.17) is given by

$$
f^{(c)} \;:=\; \cosh(\kappa(h+z))-\cosh(\kappa(h+\zeta)), \qquad \text{(E.20)}
$$

with $\kappa(x,y)$ denoting a shape parameter.

The integrals (3.6) can be calculated[27] by substituting the shape function:

$$\begin{aligned}
F_{mn}^{(c)} &= \int_{-h}^{\zeta} f_m^{(c)} f_n^{(c)}\, dz \\
&= \int_{-h}^{\zeta} (\cosh(\kappa(h+z)) - \cosh(\kappa(h+\zeta)))^2\, dz \\
&= \int_{-h}^{\zeta} (\cosh(\kappa(h+z)))^2\, dz - 2\int_{-h}^{\zeta} \cosh(\kappa(h+z))\cosh(\kappa(h+\zeta))\, dz + \int_{-h}^{\zeta} (\cosh(\kappa(h+\zeta)))^2\, dz \\
&= \left[\frac{1}{2\kappa}\sinh(\kappa(h+z))\cosh(\kappa(h+z)) + \frac{h+z}{2}\right]_{z=-h}^{\zeta} - 2\cosh(\kappa(h+\zeta))\left[\frac{1}{\kappa}\sinh(\kappa(h+z))\right]_{z=-h}^{\zeta} \\
&\quad +(h+\zeta)(\cosh(\kappa(h+\zeta)))^2 \\
&= \frac{1}{2\kappa}\sinh(\kappa(h+\zeta))\cosh(\kappa(h+\zeta)) + \frac{1}{2}(h+z) - \frac{2}{\kappa}\cosh(\kappa(h+\zeta))\sinh(\kappa(h+\zeta)) \\
&\quad +(h+\zeta)(\cosh(\kappa(h+\zeta)))^2 \\
&= -\frac{3}{2\kappa}\sinh(\kappa(h+\zeta))\cosh(\kappa(h+\zeta)) + \frac{1}{2}(h+z) + (h+\zeta)(\cosh(\kappa(h+\zeta)))^2; \qquad \text{(E.21)}
\end{aligned}$$

$$\begin{aligned}
G_{mn}^{(c)} &= \int_{-h}^{\zeta} \frac{f_m^{(c)}}{\partial\zeta}\frac{f_n^{(c)}}{\partial\zeta}\, dz \\
&= \int_{-h}^{\zeta} (-\kappa\sinh(\kappa(h+\zeta)))^2\, dz \\
&= \kappa^2(h+\zeta)(\sinh(\kappa(h+\zeta)))^2; \qquad \text{(E.22)}
\end{aligned}$$

$$\begin{aligned}
K_{mn}^{(c)} &= \int_{-h}^{\zeta} \frac{f_m^{(c)}}{\partial z}\frac{f_n^{(c)}}{\partial z}\, dz \\
&= \int_{-h}^{\zeta} (\kappa\sinh(\kappa(h+z)))^2\, dz \\
&= \kappa^2\left[\frac{1}{2\kappa}\sinh(\kappa(h+z))\cosh(\kappa(h+z)) - \frac{h+z}{2}\right]_{z=-h}^{\zeta} \\
&= \frac{1}{2}\kappa\sinh(\kappa(h+\zeta))\cosh(\kappa(h+\zeta)) - \frac{1}{2}\kappa^2(h+\zeta); \qquad \text{(E.23)}
\end{aligned}$$

[27] The following holds: $\int \sinh(x)dx = \cosh(x) + \mathrm{c}$, $\int \sinh(x)dx = \cosh(x) + c$, $\int(\sinh(ax))^2 dx = \frac{1}{2a}\sinh(ax)\cosh(ax) - \frac{x}{2} + c$ and $\int(\cosh(ax))^2 dx = \frac{1}{2a}\sinh(ax)\cosh(ax) + \frac{x}{2} + c$.

$$
\begin{aligned}
P_m^{(c)} &= \int_{-h}^{\zeta} f_m^{(c)}\, dz \\
&= \int_{-h}^{\zeta} \left(\cosh(\kappa(h+z)) - \cosh(\kappa(h+\zeta))\right) dz \\
&= \left[\frac{1}{\kappa}\sinh(\kappa(h+z))\right]_{z=-h}^{\zeta} - (h+\zeta)\cosh(\kappa(h+\zeta)) \\
&= \frac{1}{\kappa}\sinh(\kappa(h+\zeta)) - (h+\zeta)\cosh(\kappa(h+\zeta)) \\
&= -(h+\zeta)\left(\cosh(\kappa(h+\zeta)) - \frac{\sinh(\kappa(h+\zeta))}{\kappa(h+\zeta)}\right); \qquad \text{(E.24)}
\end{aligned}
$$

$$
\begin{aligned}
Q_m^{(c)} &= \int_{-h}^{\zeta} \frac{f_m^{(c)}}{\partial\zeta}\, dz \\
&= \int_{-h}^{\zeta} -\kappa\sinh(\kappa(h+\zeta))\, dz \\
&= -\kappa(h+\zeta)\sinh(\kappa(h+\zeta)); \qquad \text{(E.25)}
\end{aligned}
$$

$$
\begin{aligned}
R_{mn}^{(c)} &= \int_{-h}^{\zeta} \frac{f_m^{(c)}}{\partial\zeta} f_n^{(c)}\, dz \\
&= \int_{-h}^{\zeta} -\kappa\sinh(\kappa(h+\zeta))\left(\cosh(\kappa(h+z)) - \cosh(\kappa(h+\zeta))\right)\, dz \\
&= \left[-\sinh(\kappa(h+\zeta))\sinh(\kappa(h+z))\right]_{z=-h}^{\zeta} + \kappa(h+\zeta)\sinh(\kappa(h+\zeta))\cosh(\kappa(h+z)) \\
&= \kappa(h+\zeta)\sinh(\kappa(h+\zeta))\left(\cosh(\kappa(h+\zeta)) - \frac{\sinh(\kappa(h+\zeta))}{\kappa(h+\zeta)}\right). \qquad \text{(E.26)}
\end{aligned}
$$

The following abbreviations were introduced during the calculations:

$$
\begin{aligned}
\mathcal{D} &:= \cosh(\kappa(h+\zeta)) - \frac{\sinh(\kappa(h+\zeta))}{\kappa(h+\zeta}, && \text{(E.27a)} \\
\mathcal{S} &:= \sinh(\kappa(h+\zeta)), && \text{(E.27b)} \\
\mathcal{C} &:= \cosh(\kappa(h+\zeta)). && \text{(E.27c)}
\end{aligned}
$$

The results can now be summarized as

$$
\begin{aligned}
F^{(c)} &= -\frac{3}{2}\frac{1}{\kappa}\,\mathcal{S}\,\mathcal{C} + \frac{1}{2}(h+\zeta) + (h+\zeta)\,\mathcal{C}^2, && \text{(E.28a)} \\
G^{(c)} &= \kappa^2\,(h+\zeta)\,\mathcal{S}^2, && \text{(E.28b)} \\
K^{(c)} &= \frac{1}{2}\,\kappa\,\mathcal{S}\,\mathcal{C} - \frac{1}{2}\,\kappa^2\,(h+\zeta), && \text{(E.28c)} \\
P^{(c)} &= -(h+\zeta)\,\mathcal{D}, && \text{(E.28d)} \\
Q^{(c)} &= -\kappa\,(h+\zeta)\,\mathcal{S}, && \text{(E.28e)} \\
R^{(c)} &= \kappa\,(h+\zeta)\,\mathcal{S}\,\mathcal{D}. && \text{(E.28f)}
\end{aligned}
$$

The variations to ζ of these six functionals have to be calculated also. We have $((h+\zeta)\,\mathcal{D})' = \mathcal{C} + (h+\zeta)\,\kappa\,\mathcal{S} - \mathcal{C} = \kappa\,(h+\zeta)\,\mathcal{S}$, $\mathcal{S}' = \kappa\,\mathcal{C}$ and $\mathcal{C}' = \kappa\,\mathcal{S}$. Then we get

$$
\begin{aligned}
F'^{(c)} &= 2\,\kappa\,(h+\zeta)\,\mathcal{S}\,\mathcal{D}, && \text{(E.29a)}\\
G'^{(c)} &= \kappa^2\,\mathcal{S}^2 + 2\,\kappa^3\,(h+\zeta)\,\mathcal{C}\;\mathcal{S}, && \text{(E.29b)}\\
K'^{(c)} &= \kappa^2\,\mathcal{S}^2, && \text{(E.29c)}\\
P'^{(c)} &= -\kappa\,(h+\zeta)\,\mathcal{S}, && \text{(E.29d)}\\
Q'^{(c)} &= -\kappa^2\,(h+\zeta)\,\mathcal{C} - \kappa\,\mathcal{S}, && \text{(E.29e)}\\
R'^{(c)} &= \kappa^2\,(h+\zeta)\left(\mathcal{C}^2+\mathcal{S}^2\right) - \kappa\,\mathcal{C}\,\mathcal{S}. && \text{(E.29f)}
\end{aligned}
$$

Equation (3.7) for mass-conservation is derived using the parameters (E.28) as

$$
\begin{aligned}
\frac{\partial\zeta}{\partial t} + \nabla\cdot\Big((h+\zeta)\nabla\varphi + P^{(c)}\nabla\psi + Q^{(c)}\psi\nabla\zeta\Big) &= 0,\\
\frac{\partial\zeta}{\partial t} + \nabla\cdot\Big((h+\zeta)\nabla\varphi - (h+\zeta)\,\mathcal{D}\,\nabla\psi - \kappa\,(h+\zeta)\,\mathcal{S}\,\psi\nabla\zeta\Big) &= 0,\\
\frac{\partial\zeta}{\partial t} + \nabla\cdot\Big((h+\zeta)\Big(\nabla\varphi - \mathcal{D}\,\nabla\psi - \kappa\,\mathcal{S}\,\psi\nabla\zeta\Big)\Big) &= 0,\\
\frac{\partial\zeta}{\partial t} + \nabla\cdot\Big((h+\zeta)\,\mathbf{U}\Big) &= 0, && \text{(E.30)}
\end{aligned}
$$

with $\mathbf{U}$ the depth averaged velocity (D.6).

The Bernoulli-like equation (3.8) becomes

$$
\begin{aligned}
\frac{\partial\varphi}{\partial t}+\frac{1}{2}(\nabla\varphi)^2+g\zeta+\frac{1}{2}F'^{(c)}(\nabla\psi)^2+\frac{1}{2}G'^{(c)}\psi^2(\nabla\zeta)^2+\frac{1}{2}K'^{(c)}\psi^2 \\
+P'^{(c)}\nabla\varphi\cdot\nabla\psi+Q'^{(c)}\psi\nabla\varphi\cdot\nabla\zeta+R'^{(c)}\psi\nabla\zeta\cdot\nabla\psi \\
-\nabla\cdot\Big(Q^{(c)}\psi\nabla\varphi+R^{(c)}\psi\nabla\psi+G^{(c)}\psi^2\nabla\zeta\Big) &= 0,\\
\frac{\partial\varphi}{\partial t}+\frac{1}{2}(\nabla\varphi)^2+g\zeta+\kappa\,(h+\zeta)\,\mathcal{S}\,\mathcal{D}(\nabla\psi)^2+\frac{1}{2}\left(\kappa^2\,\mathcal{S}^2+2\,\kappa^3\,(h+\zeta)\,\mathcal{C}\;\mathcal{S}\right)\psi^2(\nabla\zeta)^2 \\
+\frac{1}{2}\kappa^2\,\mathcal{S}^2\psi^2-\kappa\,(h+\zeta)\,\mathcal{S}\nabla\varphi\cdot\nabla\psi-\left(\kappa^2\,(h+\zeta)\,\mathcal{C}+\kappa\,\mathcal{S}\right)\psi\nabla\varphi\cdot\nabla\zeta \\
+\left(\kappa^2\,(h+\zeta)\left(\mathcal{C}^2+\mathcal{S}^2\right)-\kappa\,\mathcal{C}\,\mathcal{S}\right)\psi\nabla\zeta\cdot\nabla\psi \\
-\nabla\cdot\Big(-\kappa\,(h+\zeta)\,\mathcal{S}\psi\nabla\varphi+\kappa\,(h+\zeta)\,\mathcal{S}\,\mathcal{D}\psi\nabla\psi+\kappa^2\,(h+\zeta)\,\mathcal{S}^2\psi^2\nabla\zeta\Big) &= 0,\\
\frac{\partial\varphi}{\partial t}+\frac{1}{2}(\nabla\varphi)^2+\frac{1}{2}\mathcal{D}^2(\nabla\psi)^2+\frac{1}{2}\kappa^2\,\mathcal{S}^2\psi^2(\nabla\zeta)^2+g\zeta-\frac{1}{2}\mathcal{D}^2(\nabla\psi)^2 \\
+\kappa\,(h+\zeta)\,\mathcal{S}\,\mathcal{D}(\nabla\psi)^2+\kappa^3\,(h+\zeta)\,\mathcal{C}\;\mathcal{S}\psi^2(\nabla\zeta)^2 \\
+\frac{1}{2}\kappa^2\,\mathcal{S}^2\psi^2-\kappa\,(h+\zeta)\,\mathcal{S}\nabla\varphi\cdot\nabla\psi-\kappa^2\,(h+\zeta)\,\mathcal{C}\psi\nabla\varphi\cdot\nabla\zeta-\kappa\,\mathcal{S}\psi\nabla\varphi\cdot\nabla\zeta \\
+\kappa^2\,(h+\zeta)\left(\mathcal{C}^2+\mathcal{S}^2\right)\psi\nabla\zeta\cdot\nabla\psi-\kappa\,\mathcal{C}\,\mathcal{S}\psi\nabla\zeta\cdot\nabla\psi \\
+\nabla\cdot\Big(\kappa\,(h+\zeta)\,\mathcal{S}\psi\Big(\nabla\varphi-\mathcal{D}\nabla\psi-\kappa\,\mathcal{S}\psi\nabla\zeta\Big)\Big) &= 0,\\
\frac{\partial\varphi}{\partial t}+\frac{1}{2}\left(\nabla\varphi-\mathcal{D}\nabla\psi-\kappa\,\mathcal{S}\psi\nabla\zeta\right)^2+\mathcal{D}\nabla\varphi\cdot\nabla\psi+\kappa\mathcal{S}\psi\nabla\varphi\cdot\nabla\zeta-\kappa\mathcal{S}\mathcal{D}\psi\nabla\psi\cdot\nabla\zeta+g\zeta \\
-\frac{1}{2}\mathcal{D}^2(\nabla\psi)^2+\kappa\,(h+\zeta)\,\mathcal{S}\,\mathcal{D}(\nabla\psi)^2+\kappa^3\,(h+\zeta)\,\mathcal{C}\;\mathcal{S}\psi^2(\nabla\zeta)^2 \\
+\frac{1}{2}\kappa^2\,\mathcal{S}^2\psi^2-\kappa\,(h+\zeta)\,\mathcal{S}\nabla\varphi\cdot\nabla\psi-\kappa^2\,(h+\zeta)\,\mathcal{C}\psi\nabla\varphi\cdot\nabla\zeta-\kappa\,\mathcal{S}\psi\nabla\varphi\cdot\nabla\zeta \\
+\kappa^2\,(h+\zeta)\mathcal{C}^2\psi\nabla\zeta\cdot\nabla\psi+\kappa^2\,(h+\zeta)\mathcal{S}^2\psi\nabla\zeta\cdot\nabla\psi-\kappa\,\mathcal{C}\,\mathcal{S}\psi\nabla\zeta\cdot\nabla\psi \\
+\nabla\cdot\Big(\kappa\,(h+\zeta)\,\mathcal{S}\psi\Big(\nabla\varphi-\mathcal{D}\nabla\psi-\kappa\,\mathcal{S}\psi\nabla\zeta\Big)\Big) &= 0,\\
\frac{\partial\varphi}{\partial t}+\frac{1}{2}\left(\nabla\varphi-\mathcal{D}\nabla\psi-\kappa\,\mathcal{S}\psi\nabla\zeta\right)^2+g\zeta+\frac{1}{2}\kappa^2\,\mathcal{S}^2\psi^2+\frac{1}{2}\mathcal{D}^2(\nabla\psi)^2 \\
-\kappa\,(h+\zeta)\,\mathcal{S}\nabla\varphi\cdot\nabla\psi+\mathcal{D}\nabla\varphi\cdot\nabla\psi+\kappa\,(h+\zeta)\,\mathcal{S}\,\mathcal{D}(\nabla\psi)^2-\mathcal{D}^2(\nabla\psi)^2 \\
+\kappa^2\,(h+\zeta)\mathcal{S}^2\psi\nabla\zeta\cdot\nabla\psi-\kappa\mathcal{S}\mathcal{D}\psi\nabla\psi\cdot\nabla\zeta \\
-\kappa^2\,(h+\zeta)\,\mathcal{C}\psi\nabla\varphi\cdot\nabla\zeta+\kappa^2\,(h+\zeta)\mathcal{C}\left(\mathcal{C}-\frac{\mathcal{S}}{\kappa\,(h+\zeta)}\right)\psi\nabla\zeta\cdot\nabla\psi+\kappa^3\,(h+\zeta)\,\mathcal{C}\;\mathcal{S}\psi^2(\nabla\zeta)^2 \\
+\nabla\cdot\Big(\kappa\,(h+\zeta)\,\mathcal{S}\psi\Big(\nabla\varphi-\mathcal{D}\nabla\psi-\kappa\,\mathcal{S}\psi\nabla\zeta\Big)\Big) &= 0,
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial\varphi}{\partial t}+\frac{1}{2}\left(\nabla\varphi-\mathcal{D}\nabla\psi-\kappa\,\mathcal{S}\psi\nabla\zeta\right)^2+g\zeta+\frac{1}{2}\kappa^2\,\mathcal{S}^2\psi^2+\frac{1}{2}\mathcal{D}^2(\nabla\psi)^2 \\
-\kappa\,(h+\zeta)\,\mathcal{S}\left(\nabla\varphi-\frac{\mathcal{D}\nabla\varphi}{\kappa\,(h+\zeta)\,\mathcal{S}}-\mathcal{D}\nabla\psi+\frac{\mathcal{D}^2\nabla\psi}{\kappa\,(h+\zeta)\,\mathcal{S}}-\kappa\mathcal{S}\psi\nabla\zeta+\frac{\kappa\,\mathcal{D}\,\mathcal{S}\psi\nabla\psi}{\kappa\,(h+\zeta)\,\mathcal{S}}-\kappa\,\mathcal{S}\psi^2\nabla\zeta\right)\cdot\nabla\psi \\
-\kappa^2\,(h+\zeta)\,\mathcal{C}\psi\,(\nabla\varphi-\mathcal{D}\nabla\psi-\kappa\mathcal{S}\psi\nabla\zeta)\cdot\nabla\zeta \\
+\nabla\cdot\left(\kappa\,(h+\zeta)\,\mathcal{S}\psi\Big(\nabla\varphi-\mathcal{D}\nabla\psi-\kappa\,\mathcal{S}\psi\nabla\zeta\Big)\right) &= 0, \\
\frac{\partial\varphi}{\partial t}+\frac{1}{2}\mathbf{U}^2+g\zeta+\frac{1}{2}\kappa^2\mathcal{S}^2\psi^2+\frac{1}{2}\mathcal{D}^2(\nabla\psi)^2 \\
-\kappa(h+\zeta)\mathbf{U}\cdot\left(\Big(\mathcal{S}-\frac{\mathcal{D}}{\kappa(h+\zeta)}\Big)\nabla\psi+\kappa\,\mathcal{C}\psi\nabla\zeta\right)+\nabla\cdot\big(\kappa(h+\zeta)\mathcal{S}\mathbf{U}\psi\big) &= 0.
\end{aligned}
\tag{E.31}
$$

The elliptic equation (3.10) becomes

$$
\begin{aligned}
Q^{(c)}\ \nabla\varphi\cdot\nabla\zeta+K^{(c)}\psi+G^{(c)}\psi(\nabla\zeta)^2+R^{(c)}\nabla\zeta\cdot\nabla\psi \\
-\nabla\cdot\left(F^{(c)}\nabla\psi+P^{(c)}\nabla\varphi+R^{(c)}\psi\nabla\zeta\right) &= 0, \\
-\kappa\,(h+\zeta)\,\mathcal{S}\ \nabla\varphi\cdot\nabla\zeta+\left(\frac{1}{2}\,\kappa\,\mathcal{S}\,\mathcal{C}-\frac{1}{2}\,\kappa^2\,(h+\zeta)\right)\psi \\
+\kappa^2\,(h+\zeta)\,\mathcal{S}^2\psi(\nabla\zeta)^2+\kappa\,(h+\zeta)\,\mathcal{S}\,\mathcal{D}\nabla\zeta\cdot\nabla\psi \\
-\nabla\cdot\left(\left(-\frac{3}{2}\frac{1}{\kappa}\,\mathcal{S}\,\mathcal{C}+\frac{1}{2}(h+\zeta)+(h+\zeta)\,\mathcal{C}^2\right)\nabla\psi-(h+\zeta)\,\mathcal{D}\nabla\varphi+\kappa\,(h+\zeta)\,\mathcal{S}\,\mathcal{D}\psi\nabla\zeta\right) &= 0, \\
-\kappa\,(h+\zeta)\,\mathcal{S}\nabla\varphi\cdot\nabla\zeta+\kappa\,(h+\zeta)\,\mathcal{S}\,\mathcal{D}\nabla\zeta\cdot\nabla\psi+\kappa^2\,(h+\zeta)\,\mathcal{S}^2\psi(\nabla\zeta)^2 \\
+\frac{1}{2}\,\kappa\,(\mathcal{S}\,\mathcal{C}-\kappa\,(h+\zeta))\,\psi+\nabla\cdot((h+\zeta)\,\mathcal{D}\nabla\varphi-\kappa\,(h+\zeta)\,\mathcal{S}\,\mathcal{D}\psi\nabla\zeta) \\
+\nabla\cdot\left(\left(-(h+\zeta)\,\mathcal{C}^2+2\frac{1}{\kappa}\,\mathcal{S}\,\mathcal{C}-\frac{1}{2}\frac{1}{\kappa}\,\mathcal{S}\,\mathcal{C}-\frac{1}{2}(h+\zeta)\right)\nabla\psi\right) &= 0, \\
-\kappa\,(h+\zeta)\,\mathcal{S}\,(\nabla\varphi-\mathcal{D}\nabla\psi-\kappa\mathcal{S}\psi\nabla\zeta)\cdot\nabla\zeta \\
+\frac{1}{2}\,\kappa\,(\mathcal{S}\,\mathcal{C}-\kappa\,(h+\zeta))\,\psi+\nabla\cdot((h+\zeta)\,\mathcal{D}\,(\nabla\varphi-\kappa\,\mathcal{S}\psi\nabla\zeta)) \\
+\nabla\cdot\left(\left(-(h+\zeta)\left(\mathcal{C}-\frac{\mathcal{S}}{\kappa(h+\zeta)}\right)^2+\frac{\mathcal{S}^2}{\kappa^2(h+\zeta)}-\frac{1}{2}\frac{1}{\kappa}\,\mathcal{S}\,\mathcal{C}-\frac{1}{2}(h+\zeta)\right)\nabla\psi\right) &= 0, \\
-\kappa\,(h+\zeta)\,\mathcal{S}\left(\nabla\varphi-\mathcal{D}\nabla\psi-\kappa\mathcal{S}\psi\nabla\zeta\right)\cdot\nabla\zeta \\
+\frac{1}{2}\kappa\big(\mathcal{S}\,\mathcal{C}-\kappa(h+\zeta)\big)\psi+\nabla\cdot\big((h+\zeta)\mathcal{D}\,(\nabla\varphi-\kappa\mathcal{S}\psi\nabla\zeta)\,\big) \\
+\nabla\cdot\left(\frac{1}{\kappa}\Big(\frac{\mathcal{S}^2}{\kappa(h+\zeta)}-\mathcal{D}^2\kappa(h+\zeta)-\frac{1}{2}\kappa(h+\zeta)-\frac{1}{2}\mathcal{S}\mathcal{C}\Big)\nabla\psi\right) &= 0.
\end{aligned}
\tag{E.32}
$$

E.4 Calculations for the linearized Hamiltonian

In (4.13) the linearized Hamiltonian is given for the general series model (4.2) with $M = 1$. The integral over the vertical domain is evaluated as

$$\begin{aligned}
\mathcal{H}_0 &= \iint \left(\zeta \mathbf{U}\cdot\nabla\varphi + \frac{1}{2}g(\zeta^2 - h^2)\right) dx\,dy \\
&\quad + \iint\int_{-h}^{0} \left(\frac{1}{2}(\nabla\varphi)^2 + \nabla\varphi\cdot f\nabla\psi + \frac{1}{2}(f\nabla\psi)^2 + \frac{1}{2}(f'\psi)^2\right) dz\; dx\,dy \\
&= \iint \left(\zeta \mathbf{U}\cdot\nabla\varphi + \frac{1}{2}g(\zeta^2 - h^2)\right) dx\,dy \\
&\quad + \iint \left(\frac{1}{2}h(\nabla\varphi)^2 + \Big(\int_{-h}^{0} f\; dz\Big)\nabla\varphi\cdot\nabla\psi + \frac{1}{2}\Big(\int_{-h}^{0} f^2\; dz\Big)(\nabla\psi)^2 + \frac{1}{2}\Big(\int_{-h}^{0} f'^2\; dz\Big)\psi^2\right) dx\,dy \\
&=: \iint \left(\zeta \mathbf{U}\cdot\nabla\varphi + \frac{1}{2}g(\zeta^2 - h^2)\right) dx\,dy \\
&\quad + \iint \left(\frac{1}{2}h(\nabla\varphi)^2 - h\mathcal{D}_0\nabla\varphi\cdot\nabla\psi + \frac{1}{2}\mathcal{N}_0\;(\nabla\psi)^2 + \frac{1}{2}\mathcal{M}_0\psi^2\right) dx\,dy \\
&= \iint \left(\zeta \mathbf{U}\cdot\nabla\varphi + \frac{1}{2}g(\zeta^2 - h^2)\right) dx\,dy \\
&\quad + \iint \left(\frac{1}{2}h\left(\nabla\varphi - \mathcal{D}_0\nabla\psi\right)^2 - \frac{1}{2}h\mathcal{D}_0^2(\nabla\psi)^2 + \frac{1}{2}\mathcal{N}_0\;(\nabla\psi)^2 + \frac{1}{2}\mathcal{M}_0\psi^2\right) dx\,dy \\
&= \iint \left(\zeta \mathbf{U}\cdot\nabla\varphi + \frac{1}{2}h\left(\nabla\varphi - \mathcal{D}_0\nabla\psi\right)^2 + \frac{1}{2}\left(\mathcal{N}_0 - h\mathcal{D}_0^2\right)(\nabla\psi)^2 + \frac{1}{2}\mathcal{M}_0\psi^2 + \frac{1}{2}g(\zeta^2 - h^2)\right) dx\,dy.
\end{aligned} \tag{E.33}$$

Use has been made of the following functionals:

$$\mathcal{D}_0 := -\frac{1}{h}\int_{-h}^{0} f\; dz, \tag{E.34a}$$

$$\mathcal{M}_0 := \int_{-h}^{0} f'^2\; dz, \tag{E.34b}$$

$$\mathcal{N}_0 := \int_{-h}^{0} f^2\; dz. \tag{E.34c}$$

To write the linearized Hamiltonian system (4.14) in the basic variables, the variations of the linearized Hamiltonian (4.15) have to be calculated. The variation w.r.t. φ is given by

$$\begin{aligned}
\delta_\varphi\mathcal{H}_0 &= \iint \left(\zeta\mathbf{U}\cdot\nabla\delta\varphi + h\left(\nabla\varphi - \mathcal{D}_0\nabla\psi\right)\cdot\nabla\delta\varphi\right) dx\,dy \\
&= -\iint \left(\nabla\cdot\zeta\mathbf{U}\delta\varphi + \nabla\cdot h\left(\nabla\varphi - \mathcal{D}_0\nabla\psi\right)\delta\varphi\right) dx\,dy.
\end{aligned} \tag{E.35}$$

The last equality is obtained by applying Green's theorem. Because there are no boundaries specified yet in the horizontal plane, the variations at the horizontal boundaries are taken zero and therefore there are no boundary integrals. The variation w.r.t. ζ is given by

$$\delta_\zeta\mathcal{H}_0 = \iint \left(\mathbf{U}\cdot\nabla\varphi\;\delta\zeta + g\zeta\,\delta\zeta\right) dx\,dy. \tag{E.36}$$

And the variation w.r.t ψ will be

$$\begin{aligned}\delta_\psi \mathcal{H}_0 &= \iint \left(h\left(\nabla\varphi - \mathcal{D}_0\nabla\psi\right)\cdot\left(-\mathcal{D}_0\nabla\delta\psi\right) + \left(\mathcal{N}_0 - h\mathcal{D}_0^2\right)\nabla\psi\cdot\nabla\delta\psi + \mathcal{M}_0\psi\delta\psi\right)dx\,dy \\ &= \iint \left(\nabla\cdot h\mathcal{D}_0\left(\nabla\varphi - \mathcal{D}_0\nabla\psi\right)\delta\psi - \nabla\cdot\left(\left(\mathcal{N}_0 - h\mathcal{D}_0^2\right)\nabla\psi\right)\delta\psi + \mathcal{M}_0\psi\delta\psi\right)dx\,dy. \quad \text{(E.37)}\end{aligned}$$

Substituting these variations in the linearized Hamiltonian system (4.14) gives

$$\begin{aligned}\frac{\partial\zeta}{\partial t} + \nabla\cdot\left(\zeta\,\mathbf{U} + h\,\nabla\varphi - h\,\mathcal{D}_0\,\nabla\psi\right) &= 0, && \text{(E.38a)}\\ \frac{\partial\varphi}{\partial t} + \mathbf{U}\cdot\nabla\varphi + g\zeta &= 0, && \text{(E.38b)}\\ \mathcal{M}_0\,\psi + \nabla\cdot\left(h\,\mathcal{D}_0\,\nabla\varphi - \mathcal{N}_0\,\nabla\psi\right) &= 0, && \text{(E.38c)}\end{aligned}$$

the three basic equations for the linearized general variational Boussinesq model.

F Positive model parameters

In the model equations, some parameters of the vertical shape models are used. In this chapter the sign of these parameters will be investigated.

For the cosine-hyperbolic model, the parameters (4.21) read

$$\mathcal{D}_0^{(c)} = \cosh(\kappa h) - \frac{\sinh(\kappa h)}{\kappa h}, \tag{F.1a}$$

$$\mathcal{M}_0^{(c)} = \frac{1}{2}\kappa \sinh(\kappa h)\cosh(\kappa h) - \frac{1}{2}\kappa^2 h, \tag{F.1b}$$

$$\mathcal{N}_0^{(c)} = -\frac{3}{2}\frac{1}{\kappa}\sinh(\kappa h)\cosh(\kappa h) + \frac{1}{2}h + h\left(\cosh(\kappa h)\right)^2. \tag{F.1c}$$

The Taylor series of the hyperbolic functions are given by

$$\sinh(x) = \sum_{n=0}^{\infty} \frac{x^{2n+1}}{(2n+1)!}, \tag{F.2a}$$

$$\cosh(x) = \sum_{n=0}^{\infty} \frac{x^{2n}}{(2n)!}. \tag{F.2b}$$

With the double angle formulas[28], we get

$$\begin{aligned}
\mathcal{N}_0^{(c)} &= -\frac{3}{4}\frac{1}{\kappa}\sinh(2\kappa h) + h + \frac{1}{2}h\cosh(2\kappa h) \\
&= h - \frac{3}{4}\frac{1}{\kappa}\sum_{n=0}^{\infty}\frac{(2\kappa h)^{2n+1}}{(2n+1)!} + \frac{1}{2}h\sum_{n=0}^{\infty}\frac{(2\kappa h)^{2n}}{(2n)!} \\
&= h + \sum_{n=0}^{\infty}\frac{-3\cdot 2^{2n+1}}{4(2n+1)!}\kappa^{2n}h^{2n+1} + \sum_{n=0}^{\infty}\frac{2^{2n}}{2(2n)!}\kappa^{2n}h^{2n+1} \\
&= h - \frac{3}{2}h + \frac{1}{2}h + \sum_{n=1}^{\infty}\left(\frac{2^{2n-1}}{(2n)!} - 3\frac{2^{2n-1}}{(2n+1)!}\right)\kappa^{2n}h^{2n+1} \\
&= \sum_{n=1}^{\infty}\left(1 - \frac{3}{2n+1}\right)\frac{2^{2n-1}}{(2n)!}\kappa^{2n}h^{2n+1}.
\end{aligned} \tag{F.3a}$$

Note that we have $h \geq 0$, so $h^{2n+1} \geq 0$. Because $n \in \mathbb{N}$ we have $\kappa^{2n} \geq 0$ and $\frac{2^{2n-1}}{(2n)!} \geq 0$. For the first term it holds that[29] $1 - \frac{3}{(2n+1)!} \geq 0$ for $n = 1, 2, 3, \ldots$. Therefore, we have $\mathcal{N}_0^{(c)} \geq 0$ with $\mathcal{N}_0^{(c)} = 0$ iff $\kappa h = 0$.

[28] The double angle identities are $\sinh(2x) = 2\sinh(x)\cosh(x)$ and $\cosh(2x) = 2\cosh^2(x) - 1$.

[29] For $n \geq 0$ we have $1 - \frac{3}{2n+1} \geq 0 \Leftrightarrow 1 \geq \frac{3}{2n+1} \Leftrightarrow 2n+1 \geq 3 \Leftrightarrow 2n \geq 2 \Leftrightarrow n \geq 1$.

Similarly,

$$\begin{aligned}
\mathcal{M}_0^{(c)} &= \frac{1}{4}\kappa\sinh(2\kappa h) - \frac{1}{2}\kappa^2 h \\
&= \frac{1}{4}\kappa\sum_{n=0}^{\infty}\frac{(2\kappa h)^{2n+1}}{(2n+1)!} - \frac{1}{2}\kappa^2 h \\
&= \sum_{n=1}^{\infty}\frac{1}{4}\kappa\frac{(2\kappa h)^{2n+1}}{(2n+1)!} + \frac{1}{4}\kappa\frac{1}{2}\kappa h - \frac{1}{2}\kappa^2 h \\
&= \sum_{n=1}^{\infty}\frac{2^{2n-1}}{(2n+1)!}\kappa^{2(n+1)}h^{2n+1}. \qquad \text{(F.4)}
\end{aligned}$$

We see that $\mathcal{M}_0^{(c)} \geq 0$ with $\mathcal{M}_0^{(c)} = 0$ iff $\kappa h = 0$.
Similarly the parameter $\mathcal{D}_0^{(c)}$ can be written as

$$\begin{aligned}
\mathcal{D}_0^{(c)} &= \sum_{n=0}^{\infty}\frac{(\kappa h)^{2n}}{(2n)!} + \frac{1}{\kappa h}\sum_{n=0}^{\infty}\frac{(\kappa h)^{2n+1}}{(2n+1)!} \\
&= \sum_{n=0}^{\infty}\left(1 - \frac{1}{2n+1}\right)\frac{(\kappa h)^{2n}}{(2n)!}. \qquad \text{(F.5)}
\end{aligned}$$

So we also have $\mathcal{D}_0^{(c)} \geq 0$ and $\mathcal{D}_0^{(c)} = 0$ iff $\kappa h = 0$.

G Pressure terms

The model starts with the Euler equations (2.1), which results in the model equations (5.1), which are related to the shallow water equations (3.11). In this chapter, the relation between the Euler equations and a part of the shallow water equations is presented. This will also give rise to the pressure pulse used for modelling ships (see Section 4.6).
Let's consider the third Euler equation:

$$\frac{\partial w}{\partial t} + u\frac{\partial w}{\partial x} + v\frac{\partial w}{\partial y} + w\frac{\partial w}{\partial z} = -\frac{1}{\rho}\frac{\partial p}{\partial z} - g. \tag{G.1}$$

Assuming slowly varying vertical velocity w , so zero derivatives of w, we have

$$\frac{\partial p}{\partial z} = -\rho g, \tag{G.2}$$

the *hydrostatic equation*. Integration from a depth d (so $-h < d < \zeta$) to the surface ζ gives

$$p(\zeta) - p(d) = -\rho g(\zeta - d). \tag{G.3}$$

Because one can take the atmospheric pressure zero (see Section 2.2), we have $p(\zeta) = 0$ and therefore

$$p(d) = \rho g(\zeta - d), \tag{G.4}$$

the *hydrostatic pressure* at depth d. Note that we have $\nabla p = \rho g \nabla \zeta$.
The Euler equations in the horizontal plane can now be rewritten to the shallow water equations. Assume $w\frac{\partial u}{\partial z} = w\frac{\partial v}{\partial z} = 0$ and substitute the hydrostatic pressure. This results in

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + g\frac{\partial \zeta}{\partial x} = 0, \tag{G.5}$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + g\frac{\partial \zeta}{\partial y} = 0, \tag{G.6}$$

which are (together with the continuity equation) the shallow water equations.
Let's consider the pressure as the sum of a hydrostatic pressure and a pressure pulse: $p = p_h + p_s$. Substituting this in the 2D-Euler equations gives

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -g\frac{\partial \zeta}{\partial x} - \frac{1}{\rho}\frac{\partial p_s}{\partial x}, \tag{G.7}$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = -g\frac{\partial \zeta}{\partial y} - \frac{1}{\rho}\frac{\partial p_s}{\partial y}. \tag{G.8}$$

Similar to the derivations in Chapter 4, applying the velocity potential (2.9) and the linearization results in

$$\frac{\partial}{\partial x}\left(\frac{\partial \varphi}{\partial t} + U\frac{\partial \varphi}{\partial x} + V\frac{\partial \varphi}{\partial y}\right) = -\frac{\partial}{\partial x}\left(g\zeta + \frac{p_s}{\rho}\right), \tag{G.9}$$

$$\frac{\partial}{\partial y}\left(\frac{\partial \varphi}{\partial t} + U\frac{\partial \varphi}{\partial x} + V\frac{\partial \varphi}{\partial y}\right) = -\frac{\partial}{\partial y}\left(g\zeta + \frac{p_s}{\rho}\right). \tag{G.10}$$

This can be written as

$$\frac{\partial \varphi}{\partial t} + \mathbf{U}\cdot\nabla\varphi + g\zeta = -\frac{p_s}{\rho}. \tag{G.11}$$

This equation is the Bernoulli equation (4.17b), but now with a source term consisting of the pressure pulse of the ship.

H Derivation of the incomplete Cholesky decomposition

In Section 10.2, a derivation of the incomplete Cholesky decomposition is presented. Detailed calculations used in the derivation will be shown in this chapter.
For convenience, we will look at an incomplete LU-decomposition instead of an incomplete Cholesky decomposition. The Cholesky decomposition is obtained by substituting $U = L^T$.
The incomplete decomposition M of the matrix S is given by

$$M = (D + L)D^{-1}(D + U), \tag{H.1}$$

with D a diagonal matrix, L a strictly lower triangular matrix and U a strictly upper triangular matrix. The sparsity pattern of $L + D + U$ equals the pentadiagonal sparsity pattern of S. So $D_{ij} = 0 \ \forall \ i \neq j$, $L_{ij} = 0 \ \forall \ i \notin \{j + 1, j + p\}$ and $U_{ij} = 0 \ \forall \ i \notin \{j - 1, j - p\}$, with p denoting half the bandwith of S.
Writing out the matrix multiplications and using the diagonal pattern of D^{-1} yields

$$M_{ij} \quad = \quad \sum_{k=1}^{n} \left((D_{ik} + L_{ik}) D_{kk}^{-1} (D_{kj} + U_{kj}) \right).$$

The banded structure of $D + L$ and $D + U$, i.e., $(D + L)_{ij} = 0$ for $i - j \in \{0, 1, p\}$ and $(D + U)_{ij} = 0$ for $j - i \in \{0, 1, p\}$, yields

$$M_{ij} \quad = \quad \sum_{k \in \{i-p, i-1, i, j, j-1, j-p\}} \left((D_{ik} + L_{ik}) D_{kk}^{-1} (D_{kj} + U_{kj}) \right).$$

The sparsity pattern of M is shown in Figure 5 and reads $M_{ij} = 0$ for all nodes (i, j) with $i - j \notin \{-p, -p + 1, -1, 0, 1, p - 1, p\}$. Therefore, M_{ij} will be writed out for $j \in \{i - p, i - p + 1, i - 1, i, i + 1, i + p - 1, i + p\}$ in the following calculations.

$$\begin{aligned}
M_{i,i-p} &= \sum_{k \in \{i-p, i-1, i, i-p, i-p-1, i-2p\}} \left((D_{ik} + L_{ik}) D_{kk}^{-1} (D_{k,i-p} + U_{k,i-p}) \right) \\
&= \sum_{k \in \{i-p\}} \left((D_{ik} + L_{ik}) D_{kk}^{-1} (D_{k,i-p} + U_{k,i-p}) \right) \\
&= L_{i,i-p} D_{i-p,i-p}^{-1} D_{i-p,i-p} \\
&= L_{i,i-p}
\end{aligned}$$

$$\begin{aligned}
M_{i,i-p+1} &= \sum_{k \in \{i-p, i-1, i, i-p+1, i-p, i-2p+1\}} \left((D_{ik} + L_{ik}) D_{kk}^{-1} (D_{k,i-p+1} + U_{k,i-p+1}) \right) \\
&= \sum_{k \in \{i-p\}} \left((D_{ik} + L_{ik}) D_{kk}^{-1} (D_{k,i-p+1} + U_{k,i-p+1}) \right) \\
&= L_{i,i-p} D_{i-p,i-p}^{-1} U_{i-p,i-p+1}
\end{aligned}$$

$$\begin{aligned}
M_{i,i-1} &= \sum_{k \in \{i-p, i-1, i, i-1, i-2, i-1-p\}} \left((D_{ik} + L_{ik}) D_{kk}^{-1} (D_{k,i-1} + U_{k,i-1}) \right) \\
&= \sum_{k \in \{i-1\}} \left((D_{ik} + L_{ik}) D_{kk}^{-1} (D_{k,i-1} + U_{k,i-1}) \right) \\
&= L_{i,i-1} D_{i-1,i-1}^{-1} D_{i-1,i-1} \\
&= L_{i,i-1}
\end{aligned}$$

$$
\begin{aligned}
M_{ii} &= \sum_{k\in\{i-p,i-1,i,i,i-1,i-p\}} \big((D_{ik}+L_{ik})D_{kk}^{-1}(D_{ki}+U_{ki})\big) \\
&= \sum_{k\in\{i-p,i-1,i\}} \big((D_{ik}+L_{ik})D_{kk}^{-1}(D_{ki}+U_{ki})\big) \\
&= L_{i,i-p}D_{i-p,i-p}^{-1}U_{i-p,i} + L_{i,i-1}D_{i-1,i-1}^{-1}U_{i-1,i} + D_{ii}D_{ii}^{-1}D_{ii} \\
&= L_{i,i-p}D_{i-p,i-p}^{-1}U_{i-p,i} + L_{i,i-1}D_{i-1,i-1}^{-1}U_{i-1,i} + D_{ii}
\end{aligned}
$$

$$
\begin{aligned}
M_{i,i+1} &= \sum_{k\in\{i-p,i-1,i,i+1,i,i+1-p\}} \big((D_{ik}+L_{ik})D_{kk}^{-1}(D_{k,i+1}+U_{k,i+1})\big) \\
&= \sum_{k\in\{i\}} \big((D_{ik}+L_{ik})D_{kk}^{-1}(D_{k,i+1}+U_{k,i+1})\big) \\
&= D_{ii}D_{ii}^{-1}U_{i,i+1} \\
&= U_{i,i+1}
\end{aligned}
$$

$$
\begin{aligned}
M_{i,i+p-1} &= \sum_{k\in\{i-p,i-1,i,i+p-1,i+p-2,i-1\}} \big((D_{ik}+L_{ik})D_{kk}^{-1}(D_{k,i+p-1}+U_{k,i+p-1})\big) \\
&= \sum_{k\in\{i-1\}} \big((D_{ik}+L_{ik})D_{kk}^{-1}(D_{k,i+p-1}+U_{k,i+p-1})\big) \\
&= L_{i,i-1}D_{i-1,i-1}^{-1}U_{i-1,i-1+p}
\end{aligned}
$$

$$
\begin{aligned}
M_{i,i+p} &= \sum_{k\in\{i-p,i-1,i,i+p,i+p-1,i\}} \big((D_{ik}+L_{ik})D_{kk}^{-1}(D_{k,i+p}+U_{k,i+p})\big) \\
&= \sum_{k\in\{i\}} \big((D_{ik}+L_{ik})D_{kk}^{-1}(D_{k,i+p}+U_{k,i+p})\big) \\
&= D_{ii}D_{ii}^{-1}U_{i,i+p} \\
&= U_{i,i+p}
\end{aligned}
$$

As explained in Section 10.2, $M_{ij} = S_{ij}$ for $j \in \{i-p, i-1, i+1, i+p\}$ is used. Than, the main diagonal is given by

$$
\begin{aligned}
M_{ii} &= S_{i,i-p}D_{i-p,i-p}^{-1}S_{i-p,i} + S_{i,i-1}D_{i-1,i-1}^{-1}S_{i-1,i} + D_{ii} \\
&= S_{i,i-p}^2D_{i-p,i-p}^{-1} + S_{i,i-1}^2D_{i-1,i-1}^{-1} + D_{ii}, \qquad \text{(H.2)}
\end{aligned}
$$

where S is assumed to be symmetric, so reducing to a Cholesky decomposition.
The fill-in elements are given by

$$
\begin{aligned}
M_{i,i-p+1} &= S_{i,i-p}D_{i-p,i-p}^{-1}S_{i-p,i-p+1}, \qquad \text{(H.3a)} \\
M_{i,i+p-1} &= S_{i,i-1}D_{i-1,i-1}^{-1}S_{i-1,i+p-1}. \qquad \text{(H.3b)}
\end{aligned}
$$

Now we have all elements of M as a function of S and D. How to choose the diagonal D has been explained in Section 10.2, resulting in the relaxed incomplete Cholesky preconditioner.

I Flop count for the RRB-k preconditioner

In Section 11.2 the parameter k has been chosen such that the number of flops is minimal. The derivation of the number of flops needed for the RRB-k method will be presented in this chapter.
At each iteration of the PCG-method, the preconditioned residual $z = M^{-1}r$ has to be solved. For the RRB-k preconditioner, we have $M = LDL^T$, with the last block of D according to a complete Cholesky decomposition on the maximum level k.
A domain of $N_x \times N_y$ nodes is considered, with a maximum level of $N_x^k \times N_y^k$ nodes. The operations `+`, `-`, `*`, `/` all count for one flop.

First, the equation $Lx = y$ is solved with forward substitution. Because the first level black nodes are eliminated in all RRB-k methods, they don't have to be considered. The remaining red first level red nodes not in second level are solved with $4 \cdot 2$ flops for each node, so in total $4 \cdot 2 \cdot \frac{1}{2} \cdot \frac{1}{2} \cdot N_x^1 \cdot N_y^1$. For the levels p with $1 < p < k$, we need $4 \cdot 2$ flops for all black points and $4 \cdot 2$ flops for all red nodes not in the next level. In total, we get

$$\begin{aligned}
\#\text{flops} &= 2 \cdot N_x^1 \cdot N_y^1 + \sum_{p=2}^{k-1} \left(8 \cdot \frac{1}{2} + 8 \cdot \frac{1}{4} \right) N_x^p \cdot N_y^p \\
&= 2 \cdot N_x \cdot N_y + 6 \sum_{p=2}^{k-1} \left(\frac{1}{4} \right)^{p-1} \cdot N_x \cdot N_y \\
&= 2 \cdot N_x \cdot N_y + 6 \left(\frac{1 - (\frac{1}{4})^{k-1}}{1 - \frac{1}{4}} - 1 \right) \cdot N_x \cdot N_y \\
&= \left(4 - 8 \left(\frac{1}{4} \right)^{k-1} \right) \cdot N_x \cdot N_y.
\end{aligned}$$

The second part is to solve $Dx = y$ with D a diagonal matrix. This yields one flop for all nodes not in the maximum level. So $(\frac{1}{2}N_x^1 - N_x^k) \cdot (\frac{1}{2}N_y^1 - N_y^k) = (\frac{1}{2} - (\frac{1}{2})^{k-1})^2 \cdot N_x \cdot N_y$ flops. For the nodes of the maximum level, the solution from the Cholesky decomposition is substituted.
The third part is to solve $L^T x = y$ with backward substitution. This is similar to the first part. It also takes $(4 - 8(\frac{1}{4})^{k-1}) \cdot N_x \cdot N_y$ flops.
At the maximum level, a system $GG^T y = x$ has to be solved, with G lower triangular matrices from the complete Cholesky decomposition. The algorithm for solving $Gx = b$ with forward substitution is given by (neglecting the sharper bounds in the loops):

$$\begin{aligned}
&\textbf{for } i = 0, 1, 2, \ldots, N-1 \\
&\quad \textbf{for } k = i - b, \ldots, i - 1 \\
&\qquad x_i = x_i - s_{i,i-k} x_k \\
&\quad \textbf{end} \\
&\quad x_i = x_i / s_{i0} \\
&\textbf{end}
\end{aligned} \tag{I.1}$$

With $N = N_x^k \times N_y^k$ denoting the number nodes in the maximum level and $b = \min(N_x^k, N_y^k)$ denoting the bandwith of the last Schur complement, the number of flops is given by $(2b+1)N$.

Solving $G^T x = b$ with backward substitution uses the same number of flops. So together, it is $2N(b+1)$ flops for solving the Cholesky decomposition on the maximum level.
One can write $N_x = n$ and $N_x = an$ with $a \geq 1$, than $N_x^p = (\frac{1}{2})^{p-1} n$, $N_y^p = (\frac{1}{2})^{p-1} an$, $N = an^2$ and $b = n$. The number of flops used for the several parts of solving the preconditioned system can than be summarized as follows.

equation	#flops
$Lx = y$	$4an^2 - 8an_k^2$
$Dx = y$	$\frac{1}{2}an^2 - an_k^2$
$L^T x = y$	$4an^2 - 8an_k^2$
$Gx = y$	$an_k^2(2n_k + 1)$
$G^T x = y$	$an_k^2(2n_k + 1)$

Solving the RRB-part uses $8\frac{1}{2}an^2 - 17an_k^2$ and the Cholesky part $4an_k^3 + 2an_k^2$ flops. In total, the solve function uses $8\frac{1}{2}an^2 - 15an_k^2 + 4an_k^3$ flops.

To minimize to total amount of flops, we set the derivative w.r.t n_k to zero: $12an_k^2 - 30an_k = 0$, which has solutions $n_k = 0$ and $n_k = \frac{30}{12} = 2\frac{1}{2}$.
So, for minimal flops one has to choose $n_k = 2$, i.e. the maximum level must have size $2 \times m$, with $m \geq 2$ according to the domain.

J Influence of rounding errors on the deflation method

In Section 14.2, it has been shown that the residual is decreasing monotone in the PCG-method. For the deflation method, the same monotone decrease of residual is observed. However, in some tests of the deflation method an instability occured. The deflated ICCG method was applied to an open sea with a trench in it. The DICCG-method of the in `C++` implemented wave model resulted in a nonconverging residual. While a similar implementation in `Matlab` resulted in a converging method. When zero deflation vectors were chosen, no instability occured. So the instability is due to the use of deflation. In Figure 21 the norm of the residual is given, showing a clear difference between the two experiments.

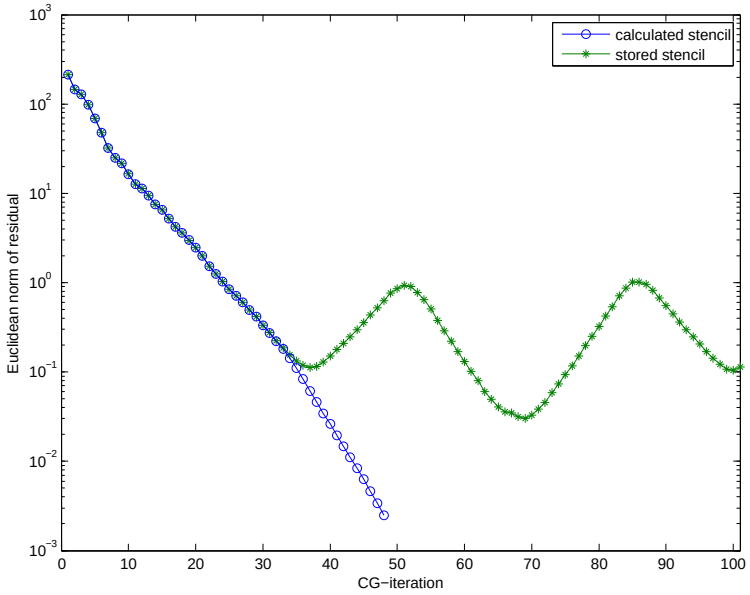

Figure 21: The residual in the DICCG-method with 4^2 subdomains, on an open sea with a trench and 128×128 nodes.

In most test problems, the bathymetry is given by an input file. The matrix is than calculated inside the wave model, as well as the right hand side and the initial solution. For this case, the calculated matrix and right hand side are stored in a file. In the simulation, the matrix and right hand side are loaded from this file. However, due to a small implementation error, the deflation method used the matrix calculated from the depth profile, while the preconditioner and CG-iteration used the loaded matrix. The values in the files are single precision, while the wave model calculates with double precision floating numbers. This implies a small rounding error in the calculated stencils and the loaded stencils of the matrix.
Aplying the stored stencil to both deflation and CG gives a monotone decreasing residual. The same is the case for applying the calculated stencil to the whole deflated CG-method. So the small difference in the matrix used by the deflation method and the matrix used in the PCG-method resulted in the instability.
The observed instability of the deflated ICCG-method is due to a small implementation error. But it shows that one should be carefull with rounding errors, which will always occur in the finite-precision calculations of a computer.

References

[1] R. A. Adams. *Calculus: A Complete Course*. Addison-Wesley Longman, Toronto, 2003.

[2] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide*. SIAM, Philadelphia, third edition, 1999.

[3] O. Axelsson and V. Eijkhout. The nested recursive two-level factorization method for nine-point difference matrices. *Journal on Scientific and Statistical Computing*, 12(6):1373–1400, 1991.

[4] O. Axelsson and G. Lindskog. On the eigenvalue distribution of a class of preconditioned methods. *Numerische Mathematik*, 48:479–498, 1986.

[5] G. K. Batchelor. *An Introduction to Fluid Dynamics*. Cambridge University Press, 2000.

[6] C. W. Brand. An incomplete-factorization preconditioning using repeated red-black ordering. *Numerische Mathematik*, 61:433–454, 1992.

[7] R. W. Cottle. Manifestations of the Schur complement. *Linear Algebra and its Applications*, 8:189–211, 1974.

[8] S. C. Eisenstat. Efficient implementation of a class of preconditioned conjugate gradient methods. *Journal on Scientific and Statistical Computing*, 2(1):1–4, 1981.

[9] J. Frank, W. Hundsdorfer, and J. G. Verwer. On the stability of implicit-explicit linear multistep methods. *Applied Numerical Mathematics*, 25:193–205, 1997.

[10] G. H. Golub and C. F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore, third edition, 1996.

[11] I. Gustafsson. A class of first order factorization methods. *BIT Numerical Mathematics*, 18(2):142–156, 1978.

[12] M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49(6):409–436, 1952.

[13] L. H. Holthuijsen. *Waves in Oceanic and Coastal Waters*. Cambridge University Press, 2007.

[14] P. Ciarlet Jr. Repeated red-black ordering: a new approach. *Numerical Algorithms*, 7:295–324, 1994.

[15] E. F. Kaasschieter. Preconditioned conjugate gradients for solving singular systems. *Journal of Computational and Applied Mathematics*, 24:265–275, 1988.

[16] G. Klopman. Temporal integration scheme and grid orientation. Technical report, Albatros Flow Research, September 2006.

[17] G. Klopman. Variational Boussinesq model for linear water waves on varying depth and current: the parabolic and cosh vertical-structure models. Technical report, Albatros Flow Research, July 2006.

[18] G. Klopman. Variational Boussinesq model for linear water waves on varying depth and current: harmonic wave propagation and incoming waves. Technical report, Albatros Flow Research, October 2007.

[19] G. Klopman. Variational Boussinesq model for linear water waves on varying depth and current: numerical approach. Technical report, Albatros Flow Research, September 2008.

[20] G. Klopman, M. W. Dingemans, and B. van Groesen. A variational model for fully non-linear water waves of Boussinesq type. In *Proceedings IWWWFB*, 2005.

[21] G. Klopman, B. van Groesen, and M. W. Dingemans. A variational approach to Boussinesq modelling of fully non-linear water waves. Under consideration for publication in J. Fluid. Mech.

[22] L. D. Landau and E. M. Lifschnitz. *Fluid Mechanics*. Butterworth-Heinemann, Oxford, 2nd edition, 2003.

[23] P. D. Lax. *Linear Algebra*. John Wiley & Sons, New York, 1997.

[24] J. C. Luke. A variational principle for a fluid with a free surface. *Journal of Fluid Mechanics*, 27(2):395–397, 1967.

[25] W. D. McComb. *Dynamics and Relativity*. Oxford University Press, 1999.

[26] 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. *Mathematics of Computation*, 31(137):148–162, 1977.

[27] R. A. Nicolaides. Deflation of conjugate gradients with applications to boundary value problems. *SIAM Journal on Numerical Analysis*, 24(2):355–365, 1987.

[28] Y. Notay and Z. Ould Amar. A nearly optimal preconditioning based on recursive red-black orderings. *Numerical Linear Algebra with Applications*, 4(5):369–391, 1997.

[29] D. F. Parker. *Fields, Flows and Waves: An Introduction to Continuum Models*. Springer, London, 2003.

[30] A. Reusken. Multigrid with matrix-dependent transfer operators for convection-diffusion problems. In *Multigrid Methods*, volume IV, pages 269–280, 1994.

[31] L. F. Richardson. The approximate arithmetical solution by finite differences of physical problems involving differential equations, with an application to the stresses in a masonry dam. *Philosophical Transactions of the Royal Society A*, 210:307–357, 1911.

[32] Y. Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, Philadelphia, 2nd edition, 2000.

[33] R. S. Schechter. *The Variational Method in Engineering*. Chemical Engineering Series. McGraw-Hill, New York, 1967.

[34] R. L. Seliger and G. B. Whitham. Variational principles in continuum mechanics. *Proceedings of The Royal Society A*, 305:1–25, 1968.

[35] L. F. Shampine. Stability of the leapfrog/midpoint method. Under consideration for publication in Applied Mathematics and Computation.

[36] J. M. Tang. *Two-Level Preconditioned Conjugate Gradient Methods with Applications to Bubbly Flow Problems*. PhD thesis, TU Delft, 2008.

[37] J. M. Tang and C. Vuik. Efficient deflation methods applied to 3-d bubbly flow problems. *Electronic Transactions on Numerical Analysis*, 26:330–349, 2007.

[38] J. M. Tang and C. Vuik. New variants of deflation techniques for pressure correction in bubbly flow problems. *Journal of Numerical Analysis, Industrial and Applied Mathematics*, 2(3):227–246, 2007.

[39] A. van der Ploeg. *Preconditioning for sparse matrices with applications*. PhD thesis, Rijksuniversiteit Groningen, 1994.

[40] A. van der Sluis and H. A. van der Vorst. The rate of convergence of conjugate gradients. *Numerische Mathematik*, 48:543–560, 1986.

[41] H. A. van der Vorst. The convergence behaviour of preconditioned cg and cg-s in the presence of rounding errors. *Lecture Notes in Mathematics*, 1457:127–136, 1990.

[42] H. A. van der Vorst. *Iterative Krylov Methods for Large Linear Systems*. Cambridge University Press, 2003.

[43] B. van Groesen. Variational Bousinesq model, part 1: Basic equations in Cartesian coordinates. Technical Report LMI-GeoMath-06/02, LabMath, 2006.

[44] B. van Groesen and J. Molenaar. *Continuum Modeling in the Physical Sciences*. SIAM, Philadelphia, 2007.

[45] J. van Kan, A. Segal, and F. Vermolen. *Numerical Methods in Scientific Computing*. VSSD, Delft, 2005.

[46] R. S. Varga. *Geršgorin and His Circles*. Springer, Berlin, 2004.

[47] P. Wesseling. *Principles of Computational Fluid Dynamics*. Springer, Berlin, 2001.