Technical University Delft

Literature Review

Interactive Wave Simuations

Author:
Akash Mittal

Supervisor:
Prof. Kees Vuik

January 27, 2014

Contents

Chapter 1

Introduction

1.1 Background

Simulation of ocean waves can be categorized into two major groups. First one is based on the physical models whereas the other generates the ocean waves based on either geometrical shapes or oceanography spectrums. Even though the later method group requires less computational effort, the waves modeled are less realistic in nature.

Currently MARIN (Maritime Research Institute Netherlands) provides ship manoevering simulators. Computation of the wave field is based on the wave spectra (Fourier theory). Being deterministic in time and space; they are easy to implement on distributed simulation systems. Also the ship movements are realistic. However, this model is not interactive, that is impact of the ship movement on the wave field is not taken into account. Also, from a visualization point of view, the model is limited.

In theory, for fluid simulation, the non-linear Navier Stokes equations along with the equation of continuity are solved. Boussinesq's approximation for water waves is applied for weakly non-linear and fairly long waves. The essential idea in the Boussinesq approximation is the eliminiation of the vertical coordinate from the flow structure, while still retaining some of the influence of the vertical structure of the flow. This is possible because wave propagates in the horizontal direction, whereas it has a different non-wave like behavior in the vertical direction. In Boussinesq-like models, an approximate vertical structure of the flow is used to eliminate the vertical space.

Frequency dispersion: Water waves exhibit frequency dispersion, i.e. water waves of different wave lengths travel with different phase speeds. Classical Boussinesq models are limited to long waves - having wavelengths much longer than the water depth.

The variational Bousinnesq model (developed by Gert Klopman [2]) can deal with deep waters with varying depth, as the vertical structure is treated as a function of depth, surface elevation and other parameters. The big advantage over the deterministic model is that it can incorporate the influence of the ship movement, thus making it an interactive wave model. Owing to the variation in the vertical structure and its interactive nature, it is much more computational intensive, and much work has been done in order to improve the running time of the solver. Two previous master thesis by Van't Wout [3] and De Jong [1] have focussed on creating a fast linear solver and increasing the efficiency by GPU programming.

1.2 Earlier Work

1.2.1 Elwin van't Wout: Fast solver and model explanation

Elwin's work [3] comprised of improving the linear solver that is used in a real-time ship simulator. The underlying wave model is the variational Boussinesq model as suggested by Gert Klopman [2]. Elwin starts with derivation of the Variational Boussinesq Model (VBM). Starting point of the derivation of the model equations are the Euler equations which are valid for motions of ideal fluids and are a special case of the Navier-Stokes equations (inviscid incompressible flow). In order to reduce the problem from 3 dimensions to 2 dimensions, vertical shape functions are employed. The resulting model is linearized to reduce complexity. The linearized model is described in detail in Chapter 2.

Equations are then discretized (spatially) using a finite volume method with structured Cartesian grids. Time integration is performed using a leap-frog scheme. One of the equations results in a system of linear equations. Elwin provides a detailed description of various numerical methods available for solving linear system of equations. Based on the properties of the matrix involved, he selects the Conjugate Gradient method as solver. This iterative method is then combined with various preconditioners, namely diagoal scaling, modified incomplete Cholesky and repeated red black ordering. Elwin modified the original implementation of the repeated red black-preconditioner to a repeated red black preconditioner for a predefined number of levels, combined with a complete Cholesky decomposition on the maximum level. Elwin concludes that the repeated red black preconditioner is the method with the lowest amount of computation time, in most of the cases.

1.2.2 Martin De Jong: Developing a CUDA solver for large sparse matrices for MARIN

The C++ RRB solver developed by Elwin was able to solve the system of linear equations within 50 ms for domains no bigger than 100,000 to 200,000 nodes. The focus of Martijn's thesis was to increase the performance of above solver by utilizing GPU architecture. By carefully selecting the storage data structures, optimal CUDA programming parameters, Martijn was able to achieve a speedup of 30x as compared to the serial code (Performance of the linear solver). Martijn reports that the new solver can solve systems that have more than 1.5 million nodes within 50ms with ease.

1.3 Current Work and Organization Of the Report

Our main target is to explore different approaches to increase the performance of the current time dependent solver, and allow solution to large problems in a similar runtime. Literature review on the project will focus upon exploring implicit time integration approaches, and non-uniform grid formulation. The organization of the report is given below:

1. Chapter 2 describes the governing equations and discretization.
2. Chapter 3 discusses previous implementation by Martijn, the RRB solver and its key features.
3. Chapter 4 discusses the implicit time integration techniques.
4. Chapter 5 discusses linear solvers for non-symmetric matrices

5. Chapter 5 discusses the required research.

Chapter 2

Model Equations and Discretization

2.1 The Variational Boussinesq Model (VBM)

The basic idea of the model is to minimize the pressure in the whole fluid. Starting point of the derivation of the model equation comes from the physical model of the Navier-Stokes Equations. The Euler equations are a special case of the Navier Stokes equations where viscous forces are neglected. For ocean waves, this is considered to be a valid assumption as the Reynolds number is typically large, and thus surface waves in water are a superb example of a stationary and ergodic random process. Also, rotational effects are negligible in large water waves. The Euler equation is given by:

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

The fluid velocity is given by $\mathbf{u}$, ρ denotes the mass density, p the pressure and $\mathbf{g}$ the gravitational constant. Understanding physics behind the problem is important as it provides information on the numerical techniques required for the solution of the problem. Note that the inviscid and irrotational assumption is not necessarily valid near solid boundaries, where very small flow structures associated with turbulence result from the no-slip boundary condition. In the current model, this has been ignored and the assumptions are considered to be valid throughout the domain.

Equation 2.1 is converted to the instationary Bernoulli equation (fluid being assumed irrotational) , and then the pressure is integrated over the whole domain. The basic idea of the variational Boussinesq model is to minimize the total pressure p. For irrotational flows, velocity can be represented by a velcoity potential (ϕ). Another parameter ζ (water level) is also introduced while performing integration. The problem thus becomes finding these functions (ϕ and ζ). The vertical structure of the flow is often known. The velocity potential can be written as a series expansion in predefined vertical shape functions, thus reducing the 3D model to a 2D model.

2.1.1 Hamiltonian Description

Being an irrotational and incompressible flow, the system of equations has a Hamiltonian structure, where the force acting on the system is the gravitational and the pressure force. The Hamiltonian $H(\zeta, \varphi)$ is given by the sum of kinetic and potential Energy. In order to obtain

the minimal value of total pressure, the Hamiltonian function is minimized with respect to variables ζ and φ [3]. This results in :

$$\frac{\partial \zeta}{\partial t} = \nabla H_{\varphi}(\varphi, \zeta) \tag{2.2a}$$

$$\frac{\partial \varphi}{\partial t} = \nabla H_{\zeta}(\varphi, \zeta) \tag{2.2b}$$

where ∇H_{ζ} refers to the partial derivative of H with respect to ζ. The structure of the above equations will play an important role in the discussion of temporal integration which is discussed in Chapter 4.

2.1.2 Linearized Model Description

The variational Boussinesq model constructed based on the above simplifications is non-linear in nature. In order to simplify these equations and thus reducing the computational effort, the Hamiltionian equations given by Equation (2.2) is linearized. Detailed derivation from the Hamiltionian to linear set of equations is described in [3].

The final set of equations after performing the linearization are given by:

$$\frac{\partial \zeta}{\partial t} + \nabla.(\zeta \mathbf{U} + h\nabla\varphi - h\mathcal{D}_0\nabla\psi) = 0, \tag{2.3a}$$

$$\frac{\partial \varphi}{\partial t} + \mathbf{U}.\nabla\varphi + g\zeta = P_s, \tag{2.3b}$$

$$\mathcal{M}_0\psi + \nabla.(h\mathcal{D}_0\nabla\varphi - \mathcal{N}_0\nabla\psi) = 0, \tag{2.3c}$$

Notes:

1. Equations 2.3 are solved for three basic variables: water level ζ, surface velocity potential (2-D) φ and vertical structure ψ. Splitting of the velocity potential into the surface potential and the vertical shape function is given by:

$$\phi(x, y, z, t) = \varphi(x, y, t) + f(z)\psi(x, y, t) \tag{2.4}$$

2. Shape function f is chosen among either a parabolic or a cosine-hyperbolic shape. The model parameters (functionals $\mathcal{D}, \mathcal{M}$ and $\mathcal{N}$) are computed using the shape function f.

3. The water depth $h = h(x, y, t)$ is relative to the reference level $z = 0$. The bottom of a basin, river or sea is thus at level $z = -h$.

4. The total velocity can be split into the average current velocity and the velocity due to the wave front. $\mathbf{U} = \mathbf{U}(x, y, t)$ is the time average horizontal velocity of the current and is used as an external input in the model.

5. Equations 2.3 represent the motion of waves. The impact of a moving ship is not seen directly. In order to model a ship, a pressure pulse on the water surface is defined. In the pressure functional given by Equation (2.3b), P_s represented the source term with $P_s := -\frac{p_s}{\rho}$. The draft of a ship's hull is the vertical distance between the waterline and the bottom of the hull. Given the draft of a ship, p_s is computed as the hydrostatic pressure at the given depth. Let the draft be given as d_s, then $p_s = g d_s \alpha(x, y)$ with

$\alpha(x, y)$ a shape function with one in the middle of the ship and zero on the horizontal boundary of the ship. Alternatively, the shape of the ship's hull can be imported from an external surface description file.

2.2 Numerical Discretization

2.2.1 Computational Domain

Both Elwin and Martijn have assumed the domain to be rectangular, and divided the domain in a rectilinear Cartesian grid. The dimensions of the domain are $L_x \times L_y$. It is divided into $N_x \times N_y$ grid points. The outermost nodes represent the physical boundary. The mesh spacing in each direction is given as $\Delta x = \frac{L_x}{N_x - 1}$ and $\Delta y = \frac{L_y}{N_y - 1}$. An example is given in 2.2.1

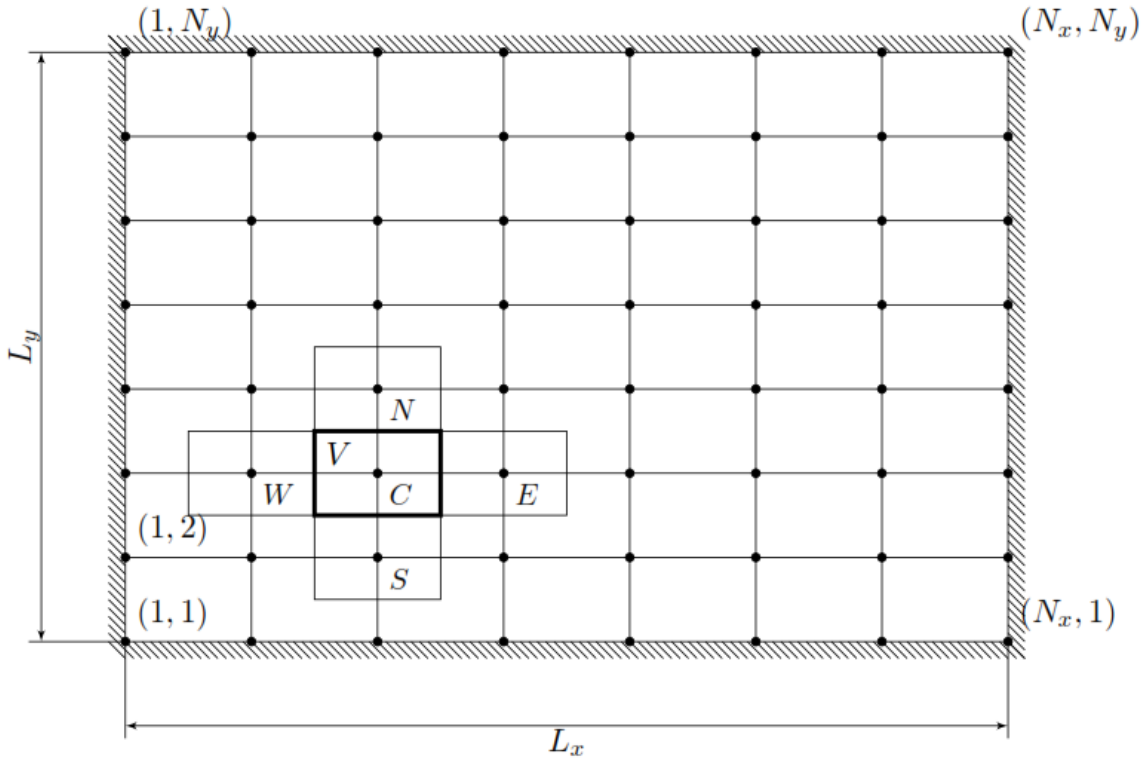

Figure 2.1: Physical domain and corresponding cartesian grid

2.2.2 Spatial Discretization

The model Equations 2.3 are discretized using a finite volume method. The variables are evaluated at the grid points and the finite volumes are rectangles of size $\Delta x \times \Delta y$ centered around the grid point. The derivatives are approximated with centered differences yielding a five-point stencil. For the grid point located at C (= center) the surrounding control volume V and its four nearest neighbors (N = north,E = east, S = south, W = west) are indicated in Section 2.2.1. On node (i, j), the discretized versions of the variables ζ, φ and ψ are given by ζ_{ij}, φ_{ij} and ψ_{ij}. In order to put the variables in matrix format, one dimensional ordering of the

variables is defined which gives the vector $\vec{\zeta}, \vec{\varphi}$ and $\vec{\psi}$. The spatial discretization can be written as :

$$\frac{\mathrm{d}}{\mathrm{dt}} \begin{bmatrix} \vec{\zeta} \\ \vec{\varphi} \\ 0 \end{bmatrix} + \begin{bmatrix} S_{\zeta\zeta} & S_{\zeta\varphi} & S_{\zeta\psi} \\ S_{\varphi\zeta} & S_{\varphi\varphi} & S_{\varphi\psi} \\ S_{\psi\zeta} & S_{\psi\varphi} & S_{\psi\psi} \end{bmatrix} \begin{bmatrix} \vec{\zeta} \\ \vec{\varphi} \\ \vec{\psi} \end{bmatrix} = \begin{bmatrix} 0 \\ P_s \\ 0 \end{bmatrix} \tag{2.5}$$

The matrix S's are given by five point stencils as follows:

$$S_{\zeta\zeta} : \begin{bmatrix} 0 & \frac{1}{2\Delta y}\overline{V_N} & 0 \\ -\frac{1}{2\Delta x}\overline{U_W} & \frac{1}{2\Delta y}\overline{V_N} - \frac{1}{2\Delta y}\overline{V_S} - \frac{1}{2\Delta x}\overline{U_W} + \frac{1}{2\Delta x}\overline{U_E} & \frac{1}{2\Delta x}\overline{U_E} \\ 0 & -\frac{1}{2\Delta y}\overline{V_S} & 0 \end{bmatrix} \tag{2.6}$$

where U and V denotes the current velocity in x and y direction respectively.

$$S_{\zeta\varphi} : \begin{bmatrix} 0 & -\frac{1}{\Delta y^2}\overline{h_N} & 0 \\ -\frac{1}{\Delta x^2}\overline{h_W} & \frac{1}{\Delta y^2}\overline{h_N} + \frac{1}{\Delta x^2}\overline{h_W} + \frac{1}{\Delta x^2}\overline{h_E} + \frac{1}{\Delta y^2}\overline{h_S} & -\frac{1}{\Delta x^2}\overline{h_E} \\ 0 & -\frac{1}{\Delta y^2}\overline{h_S} & 0 \end{bmatrix} \tag{2.7}$$

$$S_{\zeta\psi} : \begin{bmatrix} 0 & -\frac{1}{\Delta y^2}\overline{h_N D_N} & 0 \\ -\frac{1}{\Delta x^2}\overline{h_W D_W} & \frac{1}{\Delta y^2}(\overline{h_N D_N} + \overline{h_S D_S}) + \frac{1}{\Delta x^2}(\overline{h_W D_W} + \frac{1}{\Delta x^2}\overline{h_E D_E}) & -\frac{1}{\Delta x^2}\overline{h_E D_E} \\ 0 & -\frac{1}{\Delta y^2}\overline{h_S D_S} & 0 \end{bmatrix} \tag{2.8}$$

$$S_{\varphi\zeta} = g, \quad S_{\varphi\psi} = 0, \quad S_{\psi\zeta} = 0 \tag{2.9}$$

$$S_{\varphi\varphi} : \begin{bmatrix} 0 & \frac{1}{2\Delta y}\overline{V_N} & 0 \\ -\frac{1}{2\Delta x}\overline{U_W} & -(\frac{1}{2\Delta y}\overline{V_N} - \frac{1}{2\Delta y}\overline{V_S} - \frac{1}{2\Delta x}\overline{U_W} + \frac{1}{2\Delta x}\overline{U_E}) & \frac{1}{2\Delta x}\overline{U_E} \\ 0 & -\frac{1}{2\Delta y}\overline{V_S} & 0 \end{bmatrix} \tag{2.10}$$

$$S_{\psi\varphi} : \Delta x \Delta y \begin{bmatrix} 0 & \frac{1}{\Delta y^2}\overline{h_N D_N} & 0 \\ \frac{1}{\Delta x^2}\overline{h_W D_W} & -(\frac{1}{\Delta y^2}\overline{h_N D_N} + \frac{1}{\Delta x^2}\overline{h_W D_W} + \frac{1}{\Delta x^2}\overline{h_E D_E} + \frac{1}{\Delta y^2}\overline{h_S D_S}) & \frac{1}{\Delta x^2}\overline{h_E D_E} \\ 0 & \frac{1}{\Delta y^2}\overline{h_S D_S} & 0 \end{bmatrix} \tag{2.11}$$

$$S_{\psi\psi} : \Delta x \Delta y \begin{bmatrix} 0 & -\frac{1}{\Delta y^2}\overline{\mathcal{N}_N} & 0 \\ -\frac{1}{\Delta x^2}\overline{\mathcal{N}_W} & \frac{1}{\Delta y^2}\overline{\mathcal{N}_N} + \frac{1}{\Delta x^2}\overline{\mathcal{N}_W} + \frac{1}{\Delta x^2}\overline{\mathcal{N}_E} + \frac{1}{\Delta y^2}\overline{\mathcal{N}_S} + \mathcal{M} & -\frac{1}{\Delta x^2}\overline{\mathcal{N}_E} \\ 0 & -\frac{1}{\Delta y^2}\overline{\mathcal{N}_S} & 0 \end{bmatrix} \tag{2.12}$$

The system can be written as :

$$\dot{\mathbf{q}} = L\mathbf{q} + f, \tag{2.13a}$$

$$S\vec{\psi} = \mathbf{b} \tag{2.13b}$$

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 $\mathbf{f} = -\begin{bmatrix} S_{\zeta\psi}\vec{\psi} \\ S_{\varphi\psi}\vec{\psi} \end{bmatrix}$.

Elwin and Martijn focused on solving the system of equations represented by Equation (2.13b). One of the tasks of the current literature review is to study various time integration techniques for Equation (2.13a).

2.2.3 Temporal Integration :Leap Frog Scheme

In the current work by Elwin and Martijn, the leapfrog method has been used to integrate equation 2.13a. The Leapfrog method is one of the Symplectic Integration techniques designed for the numerical solution of Hamilton's equations given by Equation (2.2). More about Symplectic Integration is discussed in Chapter 4.

The method described in [3] is an explicit scheme and depends on two previous time steps (a so-called multistep) method. The exact solution to Equation (2.13a) is approximated at the time intervals $t_n = n\Delta t$, $n = 0, 1, 2, \ldots$ with $\Delta t > 0$ being the time step size. The numerical approximations are denoted by $\mathbf{q}^n \approx \mathbf{q}(t_n)$.

To keep the derivation short, we will first focus on the fixed constant step size $\Delta t := t_{n+1} - t_n$. A Taylor series expansion gives:

$$\mathbf{q}^{n+1} = \mathbf{q}^n + \Delta t\dot{\mathbf{q}}^n + \frac{1}{2}\Delta t^2\ddot{\mathbf{q}}^n + \frac{1}{6}\Delta t^3\dddot{\mathbf{q}}^n + \mathcal{O}(\Delta t^4), \tag{2.14a}$$

$$\mathbf{q}^{n-1} = \mathbf{q}^n - \Delta t\dot{\mathbf{q}}^n + \frac{1}{2}\Delta t^2\ddot{\mathbf{q}}^n - \frac{1}{6}\Delta t^3\dddot{\mathbf{q}}^n + \mathcal{O}(\Delta t^4) \tag{2.14b}$$

Subtracting the second equation from first, we obtain:

$$\mathbf{q}^{n+1} - \mathbf{q}^{n-1} = 2\Delta t\dot{\mathbf{q}}^n + \mathcal{O}(\Delta t^3), \tag{2.15}$$

which reduces to

$$\dot{\mathbf{q}}^n = \frac{\mathbf{q}^{n+1} - \mathbf{q}^{n-1}}{2\Delta t} + \mathcal{O}(\Delta t^2), \tag{2.16}$$

The explicit nature limits the size of the time-step for stability reasons. For example, given the equation $\dot{y} = \lambda y$ (λ arises as an eigenvalue of a local Jacobian and could be complex), the leapfrog method is stable only for $|\lambda\Delta t \leq 1|$. In order to approximate the interaction between the ship and the waves, a grid size of the order of half a meter or smaller is required. This limits the time step to 0.01 seconds maximum, while the manoeuvering simualtor at MARIN often can use time steps as large as 0.1 seconds. This provides the motivation to explore other time integration methods like the implicit methods, which would allow us to use larger time steps, without causing unstablity.

Chapter 3

Linear Solver

In [3] and [1], methods are given to solve the system of equations represented by Equation (2.13b). In current literature review, the solver and the properties of the matrix will be presented, as it is intended to use the same solver with minimum modification for the implicit time integration. We will study the implicit time integration techniques in Chapter 4, and whether the CUDA solver for Equation (2.13b) can be directly employed.

3.1 Properties of the Matrix

Numerical discretization of Equation (2.3c) gives Equation (2.13b). S is given by the five point stencil given in 2.12. When using lexicographical ordering of the grid points, this leads to a pentadiagonal matrix. Since the model parameters (functionals $\mathcal{D}, \mathcal{M}$ and $\mathcal{N}$) are positive [3], the center term in the stencil becomes positive and the outer elements are negative. The center term is equal to the sum of the absolute value of the outer terms plus one additional term which is also positive as $h > 0$ and $\mathcal{M}_{0C} > 0$. For a row of the matrix formed by the five point stencil, the diagonal entry is given by the central point of the five point stencil, and the off-diagonal entries are given by the remaining four entries of the stencil. As the diagonal entry is greater than the sum of absolute value of other entries in a row of the matrix, the matrix is strictly diagonally dominant.

To verify the symmetry of the matrix, one has to compare the outer diagonals with each other. For example the contribution of the west term and the east term should be the same. Thus it is required that $\left(\frac{\Delta y}{\Delta x}\bar{\mathcal{N}}_W\right)_{i,j} = \left(\frac{\Delta y}{\Delta x}\bar{\mathcal{N}}_E\right)_{i-i,j}$. In the case of non-uniform mesh, $(\Delta x)_{i,j}$ will be different from $(\Delta x)_{i-1,j}$. Only with the uniform meshes, mesh sizes are the same for different grid elements. With the overbar notation, then the requirement can be rewritten as $(\mathcal{N}_W + \mathcal{N}_C)_{i,j} = (\mathcal{N}_C + \mathcal{N}_E)_{i-1,j} \Leftrightarrow (\mathcal{N}_{i-1,j} + \mathcal{N}_{i,j}) = (\mathcal{N}_{i-1,j} + \mathcal{N}_{i,j})$, which is clearly satisfied. Applying the same reasoning to the north and the south neghbours, we conclude that the matrix is symmetrical. A symmetrical diagonally dominant matrix is positive definite and special methods (Conjugate Gradient) can be used to solve Equation (2.13b).

3.2 Conjugate Gradient Method

The Conjugate Gradient(CG) method is an element of the class of Krylov subspace methods. The Krylov subspace is created by repeatedly applying a matrix to a vector. These are iterative

methods, i.e., for a starting vector x^0, the method generates a series of vectors $x^1, x^2, \cdots$ converging to $x = S^{-1}\mathbf{b}$.

A detailed description and derivation of the CG method is provided in [1], [3]. Here we will describe the convergence characteristics and the requirements for the method, along with the algorithm pseudo-code.

The CG method can only be applied to the matrices which are symmetric and positive definite. Convergence of the CG Method for solving the linear system of equations depends on the wavelength spectrum of the matrix S. The improvement in the error is bounded by the Condition Number of the matrix which is given as $k(S) = \frac{\lambda_{max}}{\lambda_{min}}$, where λ_{max} and λ_{min} are the maximum and minimum eigenvalues of the matrix S. Higher the condition number, slower the convergence of the method. It can be derived [3] that for the given system represented by the stencil given in 2.12, the number of iterations required for the convergence of the CG method is inversely proportional to the mesh size.

3.3 Preconditioners

If the condition number of the matrix $(k(S))$ is large, preconditioning is used to replace the original system $S\psi - \mathbf{b} = 0$ by $\mathbf{M}^{-1}(S\psi - \mathbf{b} = 0)$ so that $(k(M^{-1}S))$ gets smaller than $(k(S))$. In most cases, preconditioning is necessary to ensure fast convergence of the CG method. Algorithm 3.1 provides the pseudo-code for the PCG method.

Algorithm 3.1 Preconditioned CG Algorithm to solve $Sx = \mathbf{b}$

Input x (Start Vector), r (Right Hand Side) , S (Matrix),
M (Preconditioner), ϵ (Tolerance)

```
r = b − Sx
solve : Mz = r
ρ_new = r^T z
i = 0
while ρ_new > ε^2 ||b|| do
    i = i + 1
    if i = 1 then
        p = z
    else
        β = ρ_new / ρ_old
        p = z + βp
    end if
    q = Ap
    σ = p^T q
    α = ρ_new / σ
    x = x + αp
    r = r − αq
    solve : Mz = r
    ρ_old = ρ_new
    ρ_new = r^T z
end while
```

The choice of the preconditioner is important to achieve a fast solver. Using a preconditioner should ultimately lead to a reduction in work coming from a reduction in required number of iterations for CG to converge. This implies that solving the system $Mz = r$ should be

relatively cheap to solve in the case of PCG. Martijn [1] worked on two preconditioners and translated them in an effective CUDA solver. We will describe the RRB (Repeated Red Black) preconditioner in brief here. The other preconditioner (Incomplete Poisson) is specific to the SPD (Symmetric Positive Definite) matrices, and may not be applicable for non-symmetric matrices (for example the matrix given by the stencil in Equation (2.6)).

3.3.1 Repeated Red Black (RRB) ordering as preconditioner

The RRB method uses a renumbering of the grid points according to a red-black ordering. Red-black ordering is the classic way of parallelizing the Gauss Seidel iterative solver by removing the dependencies between the adjacent cells.

What the RRB- method basically does is making the following decomposition:

$$A = LDL^T + R \tag{3.1}$$

where L is a lower triangular matrix, D a block diagonal matrix and R a matrix that contains the adjustments made during the lumping procedure.

In a two dimensional grid, the red nodes are given by the points $x_{i,j}, y_{i,j}$ with $i + j$ as even, and the black nodes with $i + j$ as odd. First the black nodes are numbered in lexicographical ordering and then the red points. For the RRB ordering, the red nodes are again split up into red and black nodes, and we repeat the procedure. When the black node elimination is repeated k times, the method is named RRB-k method. With an elimination process on the finest level, a stencil is obtained on the next coarser level, which is then simplified by lumping some outer elements (Converting the 9 point stencil on the coarser level back to 5 point stencil). The process of elimination and lumping is then repeatedly applied on the resulting stencil, until the grid is coarse enough to use a direct solver on it.

3.3.2 Implementation of RRB preconditioner

Martijn [1] has taken great efforts in describing the implementation of the RRB solver both in C++ programming, and CUDA kernels. Instead of digressing into minor details, the main points which will later impact the implicit time integration are described in brief here.

3.3.3 Effect of RRB ordering on sparsity pattern

The matrix S in the system $S\psi = b$ is given by a five point stencil, see Section 2.2.2. This results in a matrix whose structure is pentadiagonal. The sparsity pattern changes when RRB ordering is applied.

3.3.4 RRB-k method

The maximum number of levels of the RRB ordering could be computed given the dimensions of the grid in x and y direction. Though, it is not required to go upto the coarsest level. It is possible to stop at any level of coarsening. Such level is named k, and hence the RRB-k method. On level k, the nodes are numbered naturally (not in red-black fashion). From these nodes a matrix E is formed, which is a symmetric pentadiagonal matrix. The idea of stopping earlier is that we want to solve the remaining nodes accurately and exactly. In order to keep the computational cost lower, it is important to keep the size of the matrix E small.

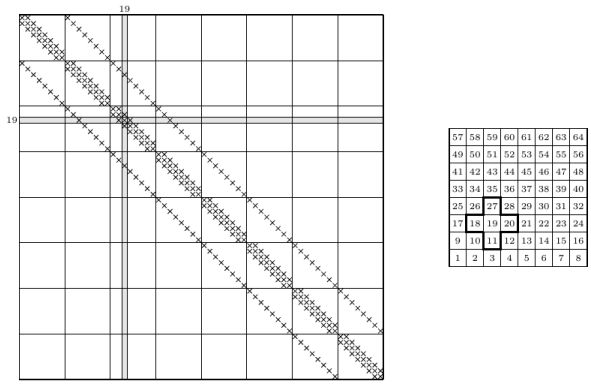

Figure 3.1: Sparsity pattern of $S \in \mathcal{R}^{64x64}$ before RRB ordering [1]

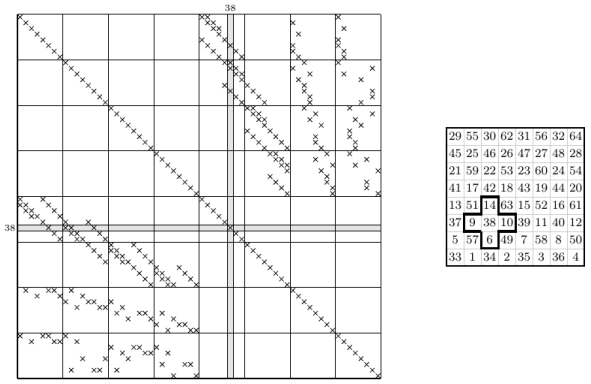

Figure 3.2: Sparsity pattern of $S \in \mathcal{R}^{64x64}$ after RRB ordering [1]

On level k, we have to solve the system of like $Ex = b$, where x is formed by level k nodes of z and b by the level k nodes of r in the problem $Mz = r$. It is possible to use direct solvers (Complete Cholesky factorization) or Gaussian elimination for such small problem.

3.3.5 The Schur complement and the 9-point Stencil

With the basic red-black numbering, the matrix S can be written in a block matrix format. The system $S\psi = b$ can then be written in the form:

$$\begin{bmatrix} D_b & S_{br} \\ S_{rb} & D_r \end{bmatrix} \begin{bmatrix} \psi_b \\ \psi_r \end{bmatrix} = \begin{bmatrix} b_b \\ b_r \end{bmatrix}$$

Subscript b and r denotes the black and red nodes respectively. D_r and D_b are the diagonal matrices, and for a symmetric matrix S, $S_{rb} = S_{br}^T$ (matrices with 4 off-diagonals).

Next, the black nodes are eliminated by applying Gaussian elimination. This yields:

$$\begin{bmatrix} D_b & S_{br} \\ 0 & D_r - S_{rb}D_b^{-1}S_{br} \end{bmatrix} \begin{bmatrix} \psi_b \\ \psi_r \end{bmatrix} = \begin{bmatrix} b_b \\ b_r - S_{rb}D_b^{-1}S_{br} \end{bmatrix}$$

The matrix $S_1 := D_r - S_{rb}D_b^{-1}S_{br}$ is called the *1st Schur complement* and is given by a 9-point stencil, the vector $b_1 := b_r - S_{rb}D_b^{-1}S_{br}$ is the corresponding right-hand side. Using the solution of red nodes, it is thus possible to compute the solution of black nodes. The catch here is that instead of the five point stencil of S, more expensive 9-point stencil of S_1 has to be solved.

3.3.6 Constructing the preconditioning matrix M

The preconditioning matrix $M = LDL^T$ is constructed in following four steps:

- Elimination of black nodes: 9-point stencil is generated from the 5-point stencil (creation of the Schur complement). In the case of C++ code, all the five diagonals are stored separately, whereas in CUDA code only three out of five diagonals are stored. West and South stencil are used to access the East and North stencil respectively (utilizing the symmetry of the matrix).
- Lumping from the 9-point stencil to the 5-point stencil for the remaining red nodes. This ensures removing stencil dependencies by adding the respective coefficients to other coefficients.
- Elimination of the first level red nodes using the lumped 5-point stencil (creating the second level red nodes with 9-point stencil).
- Lumping on the second level. The resulting matrix on the coarse grid is pentadiagonal and has the same properties as the original matrix S.

The above procedure is only one RRB iteration. It consists of an elimination $S = L_1S_1L_1^T$, lumping $S_1 = S_1 + R_1$ and again an elimination $S_1 = L_2S_2L_2^T$ and lumping $S_2 = S_2 + R_2$. Combined we have,

$$\begin{aligned} S &= L_1L_2S_2L_2^TL_1^T + L_1L_2R_2L_2^TL_1^T + L_1R_1L_1^T \\ &= LDL^T + R. \end{aligned}$$

3.3.7 Solving $Mz = r$

As given in Algorithm 3.1, in each iteration the preconditioning step $Mz = r \implies LDL^Tz = r$ has to be solved for z. This is done in three steps:

- $La = r$ is solved using forward-substitution. This is done level-wise going from finer grid to the coarser grid.
- $b = D^{-1}a$. As D is block diagonal matrix, its inverse is easy to compute.
- $L^Tz = b$ is solved using back-substitution. This is done by going from coarser grids to fine grids level-wise.

3.3.8 Efficient CUDA and C++ Code Implementation

In order to boost performance, Martijn has implemented efficient kernels for the following:

- Matrix Vector product

- Dot product : Performing mass reduction on GPU and using Kahan summation algorithm on CPU. Kahan summation algorithm greatly reduces error in floating point additionns (It makes possible to sum n numbers with an error that only depends on the floating-point precisions)
- Vector Updates

These optimized codes could be reused even if the PCG algorithm is changed.

Chapter 4

Temporal Discretization

The system of equations to be advanced in time are given in Equation (2.13a). Expanding the equation we get:

$$\dot{\vec{\zeta}} = -S_{\zeta\zeta}\vec{\zeta} - S_{\zeta\varphi}\vec{\varphi} - S_{\zeta\psi}\vec{\psi} \tag{4.1a}$$

$$\dot{\vec{\varphi}} = -S_{\varphi\zeta}\vec{\zeta} - S_{\varphi\varphi}\vec{\varphi} - S_{\varphi\psi}\vec{\psi} + P_s \tag{4.1b}$$

The equation for φ can be further simplified taking into account the structure of the stencils.

$$\dot{\vec{\varphi}} = -S_{\varphi\varphi}\vec{\varphi} - g\vec{\zeta} + P_s \tag{4.1c}$$

where g is the gravitational constant. Above equations in the form of Equation (2.13a) are solved using the explicit Leap-Frog scheme in [1] and [3]. The method is described in 2.2.3.

The equation for variable ψ, which does not contain the time derivative, but still depends on the other variables is given by

$$S_{\psi\psi}\vec{\psi} = -S_{\psi\varphi}\vec{\varphi} \tag{4.1d}$$

4.1 Explicit Time Stepping

In explicit schemes, the fluxes and the sources are computed at the n^{th} time level and their contribution is added to the current value of the variable. In [3] and [1], explicit leap frog method has been used which has a second order accuracy in time, and is only conditionally stable. Explicit time integration is simple and fast, and is easily parallelized for parallel computers.

The main disadvantage is that the time step is limited by the numerical stability requirements. For the equations represented by Equation (2.13a). the stability condition requires that the time step is shorter than the crossing time of the grid cells by the faster wave:

$$\Delta t < \frac{\Delta x_i}{c_i^{max}} \tag{4.2}$$

for all grid cells and all directions $i = x, y$ and c represents the velocity of the wave. This is the famous Courant-Friedrich-Levy (CFL) condition valid for explicit time integration of arbitrary hyperbolic PDE's. In the current implementation, with the uniform-grid, Δx is constant for each cell, and average current velocity ($\vec{U}$) is chosen as the maximum wave velocity to determine the

time step such that it satisfies the CFL condition. A safety margin is added to ensure stability without needing to check the stability criteria at each time step.

The intention to use lower mesh sizes to capture details of wave interaction with ships at finer levels will require a further reduction in the time step, hence increasing the computation time. Thus it becomes more and more important to explore time integration methods which are more stable, hence allowing us to use larger time steps, and still giving good accuracy.

4.2 Implicit Time Stepping

4.2.1 Fully Implicit Scheme

Stability of the time integration can be significantly improved by employing fully implicit time integration techniques. The most common and simplest of the fully implicit scheme is the Backward Euler scheme. For the set of equations given by Equation (2.13a), we get:

$$\frac{q_{n+1} - q_n}{dt} = (Lq_{n+1} - S_{\zeta\psi}\vec{\psi}_{n+1}) \tag{4.3a}$$

where q = $\begin{bmatrix}\vec{\zeta}\\ \vec{\varphi}\end{bmatrix}$. The method is first order accurate in time (can be derived from the Taylor series expansion) and is unconditionally stable.

While the stability of the implicit time discretization is a great improvement over the explicit schemes, one has to solve the implicit equations for the unknowns q_{n+1} and $\vec{\psi}_{n+1}$, which requires solving a system of equations (Differential Algebraic Equations) represented by :

$$(I - L)q_{n+1} = q_n - S_{\zeta\psi}\vec{\psi}_{n+1} \tag{4.4a}$$

$$S_{\psi\psi}\vec{\psi}_{n+1} = -S_{\psi\varphi}\vec{\varphi}_{n+1} \tag{4.4b}$$

As the method is first order accurate in time, and the spatial discretization was second order accurate, overall accuracy of method is only first order. Such methods are usually good to achieve steady-state results, but not very accurate. The disadvantages of using the fully implicit scheme includes:

- Moving from explicit to implicit schemes incurs extra computational efforts as it requires solving the linear system of equations given by Equation (4.4a). For the system represented above, the iterative linear solvers are used, which are described more in detail in Chapter 5.
- First order implicit methods simply under-relax more to maintain the stability of the iterative solution. It is this increased damping, with the increase in time-step size, which induces more inaccuracies in transient behavior.
- Spatial discretization plays an important role in the stability of the numerical solutions of unsteady state hyperbolic equations. Generally for the Euler equations, the central difference scheme is more accurate than the first order upwinding schemes. Also stability in the case of the central differencing scheme is not an issue as the diffusive forces are not active. In the case flux limiters are used for spatial discretization which have the capability of capturing shocks, discontinuities or sharp changes in the solution domain, the implicit scheme results in the formation of non-linear system of equations, which then requires more computational effort (computation of Jacobi.

4.2.2 β - Implicit Scheme

Semi-implicit methods try to combine the stability of the implicit methods with the efficiency of the explicit methods. The system of equations is discretized using a one parameter (β) implicit scheme to advance in time:

$$q_{n+1} = q_n + \Delta t(\beta(Lq_{n+1} - S_{\zeta\psi}\vec{\psi}_{n+1}) + (1-\beta)(Lq_n - S_{\zeta\psi}\vec{\psi}_n)) \tag{4.5a}$$

The parameter β can vary between 0 and 1. For $\beta = 1$, we get the Backward-Euler method (fully implicit) and the so-called trapezoidal scheme for $\beta = 0.5$

For $\beta = 0.5$, we can rewrite the equations as:

$$q_{n+1} = q_n + 0.5\Delta t(\dot{q_{n+1}} + \dot{q_n}) \tag{4.6a}$$
$$= q_n + \Delta t\dot{q_n} + 0.5\Delta t(\dot{q_{n+1}} - \dot{q_n}) \tag{4.6b}$$
$$= q_n + \Delta t\dot{q_n} + \frac{1}{2}\Delta t^2\frac{d\dot{q}}{dt} + O(\Delta t^3) \tag{4.6c}$$

which gives the trapezoidal method of second order temporal accuracy.

4.2.3 Stability

The Scalar Test Equation

To understand the stability of time integration methods, we consider the scalar, complex test equation :

$$w^{'}(t) = \lambda w(t) \tag{4.7}$$

where $\lambda \in C$.

Application of the time integration scheme (Explicit or Implicit) gives :

$$w_{n+1} = R(\Delta t\lambda)w_n, \tag{4.8}$$

$R(\Delta t\lambda)$ is called the Stability Function. Let $z = \Delta t\lambda$. For the explicit schemes of Order s, $R(z)$ is a polynomial of degree $\leq s$. For implicit methods it is a rational function with degree of both numerator and denominator $\leq s$. The stability region is defined in terms of $R(z)$ as:

$$S = z \in C : |R(z)| \leq 1 \tag{4.9}$$

The scheme that has the property that S contains entire left half plane $C^- = x \in C : Rez \leq 0$ is called A-Stable. A scheme is said to be Strongly A-Stable if it is A-stable with $|R(\infty)| < 1$, and it is said to be L-stable if in addition $|R(\infty)| = 0$.

For the semi-implicit scheme with parameter β, the stability function is $R(z) = \dfrac{1+(1-\beta)z}{1-\beta z}$. The implicit trapezoidal rule given by $\beta = 0.5$ is A-stable, whereas the fully implicit Backward Euler method is L-stable with $\beta = 1$.

Stability for Linear Systems

Let the linear system of equations (m equations, m unknowns) given as:

$$w^{'}(t) = Aw(t) + g(t) \tag{4.10}$$

with $A \in \mathcal{R}^{mxm}$. Application of the semi-implicit scheme with parameter β gives:

$$w_{n+1} = R(\Delta tA)w_n + (I - \beta\Delta tA)^{-1}\Delta t g_{n+\beta} \tag{4.11}$$

where

$$R(\Delta tA) = (I - \beta\Delta tA)^{-1})(I + (1-\beta)\Delta tA) \tag{4.12}$$

and $g_{n+\beta} = (1-\beta)g(t_n) + \beta g(t_{n+1})$. Starting from initial solution of w_0, we obtain:

$$w_n = R(\Delta tA)^n w_0 + \Delta t \sum_{i=0}^{n-1} R(\Delta tA)^{n-i-1}(I - \beta\Delta tA)^{-1} g_{i+\beta} \tag{4.13}$$

If we perturb the initial solution $\hat{w}_0$, we get the formula for the perturbed solution at n^{th} time step as:

$$\hat{w}_n - w_n = R(\Delta tA)^n(\hat{w}_0 - w_0) \tag{4.14}$$

Hence, the powers $R(\Delta tA)^n$ determine the growth of the initial errors.

Let λ_j with $1 \leq j \leq m$ denote the eigenvalues of the matrix A, and let A be diagonizable such that $A = U\Lambda U^{-1}$ where $\Lambda = diag(\lambda_j)$ and condition number of $U \leq K$. [7]

Then for $\Delta t\lambda_j \in S, 1 \leq j \leq m \implies ||R(\Delta tA)^n|| \leq K \ \forall n \geq 1$ where S represents the stability region given above in the scalar test equation case.

Stability of the semi-implicit methods thus requires a moderate bound for these powers. Thus, unfortunately this trapezoidal scheme is only marginally stable for linear advection problems.

4.2.4 Backward Differentiation Formula

Another approach to achieve second order temporal accuracy is by using information from multiple previous time steps. This gives rise to a two-parameter three-level time integration scheme:

$$q_{n+1} = q_n + \Delta t \left[\alpha \frac{q_n - q_{n-1}}{\Delta t} - \alpha \dot{q_n} + \beta \dot{q_{n+1}} + (1-\beta)\dot{q_n}\right] \tag{4.15}$$

where $\dot{(}q_n$ represents the derivative of q at the n^{th} time interval.

This scheme is three-level whenever the parameter $\alpha \neq 0$. When $\alpha = 1/3$ and $\beta = 2/3$, we obtain the second order Backward Differentiation Formula (BDF2) for constant time step Δt.

The analysis of the accuracy for a multi-step method is presented below. Let us assume that q_n was computed from q_{n-1} with a second order temporal accuracy. This implies

$$\frac{q_n - q_{n-1}}{\Delta t} = \dot{q_n} - (\Delta t/2)\frac{d\dot{q}}{dt} + O(\Delta t^2) \tag{4.16}$$

Substituting this into Equation (4.15)

$$q_{n+1} = q_n + \Delta t \left[\dot{q_n} - \alpha \frac{\Delta t}{2} \frac{d\dot{q}}{dt} + \beta \Delta t \frac{d\dot{q}}{dt} + O(\Delta t^2) \right] \tag{4.17a}$$

$$= q_n + \Delta t \dot{q_n} + \Delta t^2 (\beta - \frac{\alpha}{2}) \frac{d\dot{q}}{dt} + O(\Delta t^3) \tag{4.17b}$$

The method is second order temporal accurate when $2\beta - \alpha = 1$. At the start of simulation, one could use trapezoidal scheme for second order accuracy, or more stable backward Euler method (as the value at $n - 1^{th}$ time step is not available). The BDF2 method has better stability properties than trapezoidal rule, and is regularly used for stiff problems. It can also be viewed as an implicit counterpart of the explicit Leap-Frog scheme.

4.2.5 Predictor Corrector Methods

Simply stating, a predictor-corrector method is an algorithm that proceeds in two steps. First, the prediction step computes a rough approximation of the desired quantity using not so expensive algorithms like explicit method. Second, the corrector step refines the initial approximation using other means for example an implicit method.

There are various variants of a predictor-corrector method, depending upon how is the corrector algorithm applied, and how many times.

- A predictor formula is used to get a first estimate of the next value of the dependent variables, and the corrector formula is applied iteratively until convergence is obtained. In this case, the stability properties of the algorithm are completely determined by the corrector formula alone and the predictor formula only influences the number of iterations required.
- The values of the dependent variables obtained from one application of the corrector formula are regarded as the final values. The predicted and corrected values are compared to obtain an estimate of the truncation error associated with the integration step. Based on the allowable error limits, the corrected value is either accepted, or the time step reduced starting from the last accepted point. For such method, stability analysis of the corrector equation alone is not sufficient. The analysis must include the predictor equation, the corrector equation and the manner in which they are used.

The application of predictor corrector method here may require another set of iteration, which is the computation of $\vec{\psi}$ by solving the linear system of equation given by Equation (4.4a).

4.2.6 Minimum Residual Predictor Corrector (MR-PC) Time Stepping [4]

One time step by the MR-PC scheme consists of an explicit predictor step (with a time step larger than normally allowed by the CFL condition) and a corrector step where the linear system of an implicit scheme is solved by a few minimum residual-type (e.g. GMRES or BiCGSTAB) iterations with the initial guess from the predictor step. A nice aspect of the GMRES method is that it constructs implicitly an integration polynomial of which the coefficients are adjusted to the specific right-hand side of the system of equations (generated from the implicit scheme). If after some time steps, components of high frequencies in the solution have not damped out, then they are present in the right-hand side. As soon as they become large, they are automatically damped out by the GMRES polynomial, provided that the time step is not too large with respect to the number of GMRES iterations. [5]

Any of the implicit schemes described above can be taken for the correction step. To obtain good temporal accuracy, the choice of the predictor step is important. For second order accuracy, an initial guess from an explicit second order scheme (e.g. Leap-Frog/ trapezoidal) can be used. If the corrector step is also second order, the overall MR-PC will also be second order.

The stability region of the MR-PC is wider than that of the explicit schemes discussed. For example, even with a single iteration in the corrector step, a second order MR-PC with BDF2 corrector has an effective time step (time step accounted for additional computational effort) three times that of the explicit scheme. Also, it is possible to control the time step to ensure stability based on the information received from the GMRES process.

The use of residual iterations is mainly to smoothen the error generated by the explicit scheme. Also, no preconditioning is required, as we do not intend to solve the implicit set of equations.

We will present this idea for the fully implicit backward Euler method. It is given by

$$q_{n+1B} - q_n = \Delta t(Lq_{n+1B} - S_{\psi\varphi}\vec{\psi}_{n+1B}) \tag{4.18}$$
$$S_{\psi\psi}\vec{\psi}_{n+1B} = -S_{\psi\varphi}\vec{\varphi}_{n+1B} \tag{4.19}$$

In order to solve this equation by MR-PC, we apply the Euler forward (explicit) method instead to get the initial guess.

$$q_{n+1F} - q_n = \Delta t(Lq_n - S_{\psi\varphi}\vec{\psi}_n) \tag{4.20a}$$
$$S_{\psi\psi}\vec{\psi}_{n+1F} = -S_{\psi\varphi}\vec{\varphi}_{n+1F} \tag{4.20b}$$

And let $\Delta q = q_{n+1B} - q_{n+1F}$ and $\Delta\vec{\psi} = \vec{\psi}_{n+1B} - \vec{\psi}_{n+1F}$. Thus we obtain:

$$q_{n+1F} + \Delta q = q_n + \Delta t(L(q_{n+1F} + \Delta q) - S_{\psi\varphi}(\vec{\psi}_{n+1F} + \Delta\vec{\psi})) \tag{4.21a}$$
$$S_{\psi\psi}\Delta\vec{\psi} = -S_{\psi\varphi}\Delta\vec{\varphi} \tag{4.21b}$$

This leads to

$$\Delta q = q_n + \Delta t(Lq_{n+1F} + L\Delta q) - q_{n+1F} - \Delta t S_{\psi\varphi}(\vec{\psi}_{n+1F} + \Delta\vec{\psi}) \tag{4.22a}$$
$$(I - L\Delta t)\Delta q = q_n - (I - L\Delta t)q_{n+1F} - \Delta t S_{\psi\varphi}(\vec{\psi}_{n+1F} + \Delta\vec{\psi}) \tag{4.22b}$$
$$S_{\psi\psi}\Delta\vec{\psi} = -S_{\psi\varphi}\Delta\vec{\varphi} \tag{4.22c}$$

We intend to perform GMRES iterations on the Equation (4.22b). After performing one explicit step (Equation (4.20)) and obtaining most of the terms of the right-hand side, we are still left with $\Delta\vec{\psi}$, which is linked to Δq through Equation (4.22c). On the first pass, we could assume $\Delta\vec{\psi}$ to be zero, perform few iterations of GMRES and then solve for Equation (4.22c). Based on the value of $\Delta\vec{\psi}$ obtained, we could decide if another round of correction is required or not.

On similar ground, a higher order method (2nd order Trapezoidal order) can be constructed.

GMRES iterations

The iterations are applied to the linear system with matrix $L^{'} = I - L\Delta t$ for the Euler backward scheme. The matrix is non-symmetric and general in nature, and thus we cannot apply the

Conjugate Gradient method for the iterations. More general Krylov space methods (GMRES, BiConujgate Stab) can be used to perform the minimal residual iterations.

For k steps of GMRES iterations, k matrix-vector multiplication with $L^{'}$ and $\frac{k(k+1)}{2} + k$ innerproducts will be required. It might be possible to use the existing CUDA or C++ methods of computation of matrix-vector product with minimal adaptation.

4.2.7 Semi-Implicit schemes

Any implicit method requires solving a linear system of equation. In [1], linear solver has been developed for the case when the system of equation is represented by a pentadioganal matrix. (The matrix is also SPD, but that we will deal with later). From the Equation (4.4a), the linear system of equations formed for the variables q_{n+1} is block pentadiagonal instead of pentadiagonal as $q = \begin{bmatrix} \vec{\zeta} \\ \vec{\varphi} \end{bmatrix}$, and each $\vec{\zeta}$ and $\vec{\varphi}$ are represented by their own five point stencils.

As previously mentioned, it is our intention to make use of the linear solver in [1] with minimum modifications possible. For this reason, we split the Equation (4.4a) into Equation (4.1a) and derive the corresponding equations for various implicit time integration procedures.

In the first approach, we implicitly advance variable $\vec{\zeta}$, and use the previous time step values of variables $\vec{\varphi}$ and $\vec{\psi}$. After advancing $\vec{\zeta}$, equation for $\vec{\varphi}$ is advanced implicitly, and then the linear system of equations for $\vec{\psi}$ is solved. The equations for the same are given below:

$$\vec{\zeta}_{n+1} = \vec{\zeta_n} + \Delta t(-S_{\zeta\zeta}\vec{\zeta}_{n+1} - S_{\zeta\varphi}\vec{\varphi_n} - S_{\zeta\psi}\vec{\psi_n}) \tag{4.23a}$$

$$(\frac{I}{\Delta t} + S_{\zeta\zeta})\vec{\zeta}_{n+1} = \frac{1}{\Delta t}\vec{\zeta_n} - (S_{\zeta\varphi}\vec{\varphi_n} + S_{\zeta\psi}\vec{\psi_n}) \tag{4.23b}$$

Please note that variables φ and ψ here are treated explicitly.

$$\vec{\varphi}_{n+1} = \vec{\varphi}_n + \Delta t(-S_{\varphi\zeta}\vec{\zeta}_{n+1} - S_{\varphi\varphi}\vec{\varphi}_{n+1} - S_{\varphi\psi}\vec{\psi_n}) \tag{4.24a}$$

$$(\frac{I}{\Delta t} + S_{\varphi\varphi})\vec{\varphi}_{n+1} = \frac{1}{\Delta t}\vec{\varphi}_n - (S_{\varphi\zeta}\vec{\zeta}_{n+1} + S_{\varphi\psi}\vec{\psi_n}) \tag{4.24b}$$

Only variable ψ here is treated explicitly. As the value of ζ at $n + 1^{th}$ time step is available, we use it. As it is a combination of implicit and explicit scheme, Stability of the method is not guaranteed.

Another possibility here is to use a predictor-corrector kind of scheme, described below:

- Advance $\vec{\varphi}$ using an explicit time integration scheme.
- Based on the new value of $\vec{\varphi}$, compute $\vec{\psi}$ by solving linear system of equations.
- Implicitly advance $\vec{\zeta}$ where the values of $\vec{\varphi}$ and $\vec{\psi}$ on the right-hand side of Equation (4.23) are substituted by above computed values.
- Perform implicit correction of $\vec{\varphi}$ and compute $\vec{\psi}$.

This will definitely demand more number computational effort, but will be more stable than Equation (4.23). Similar derivations can be done for higher order methods.

4.3 Symplectic integration

The Symplectic integrators are designed for the numerical solution of Hamiltonian equations given by Equation (2.2). Symplectic integrators ensures time-reversibility and preservation of the symplectic (Ergodic) nature of the equation.

The so-called symplectic Euler method (first order) can be constructed as follows:

$$\zeta_{n+1} = \zeta_n + \Delta t \nabla H_\varphi(\zeta_{n+1}, \varphi_n) \tag{4.25a}$$

$$\varphi_{n+1} = \varphi_n + \Delta t \nabla H_\zeta(\zeta_{n+1}, \varphi_n) \tag{4.25b}$$

The methods are implicit for general Hamiltonian systems. However if $H(\zeta, \varphi)$ is separable as $H(\zeta, \varphi) = T(\zeta) + U(\varphi)$, it turns out to be explicit.

The Stormer-Verlet schemes are the Symplectic methods of order 2. They are composed of the two symplectic Euler methods with step size $\frac{\Delta t}{2}$.

$$\zeta_{n+1/2} = \zeta_n + \frac{\Delta t}{2} \nabla H_\varphi(\zeta_{n+1/2}, \varphi_n) \tag{4.26a}$$

$$\varphi_{n+1} = \varphi_n + \frac{\Delta t}{2} (\nabla H_\zeta(\zeta_{n+1/2}, \varphi_n) + \nabla H_\zeta(\zeta_{n+1/2}, \varphi_{n+1}) \tag{4.26b}$$

$$\zeta_{n+1} = \zeta_{n+1/2} + \frac{\Delta t}{2} \nabla H_\varphi(\zeta_{n+1/2}, \varphi_{n+1}) \tag{4.26c}$$

Implicit mid-point rule:

For a fully implicit method, two variables are combined in one equation , representesd by variable q, the implicit mid-point rule is given by:

$$q_{n+1} = q_n + \Delta t \nabla H(\frac{q_{n+1} + q_n}{2}) \tag{4.27}$$

Chapter 5

Solvers for Non-Symmetric Matrices

As we have seen, most of the implicit time integration methods described in Chapter 4 (except MR-PC) will require solving a set of equations with the matrices given by the stencils in Section 2.2.2. The matrices involved are not symmetrical as compared to the matrix given by Stencil $S_{\psi\psi}$. For this reason, the PCG method as described in Chapter 3 cannot be applied directly to solve the linear system of equations arising from the implicit time integration.

The objective is therefore to determine the solution vector for a large, sparse and linear system of equations which is not symmetric. An overview of these methods can be found in [6], which forms the basis for this Chapter.

As mentioned before in Chapter 3, the idea of some iterative methods is to project a n-dimensional problem into a lower-dimensional Krylov subspace. Given a matrix A and a vector b, the associated Krylov sequence is the set of vectors $b, Ab, A^2b, A^3b, \cdots$, which can be computed by matrix-vector multiplications in the form $b, Ab, A(Ab), A(A(Ab))), \cdots$. The corresponding Krylov subspaces are the spaces spanned by successively larger group of these vectors.

5.1 From Symmetric to Non-Symmetric Matrices

Figure 5.1 shows the classification of Krylov Subspace methods as we move from symmetric matrices to non-symmetric matrices. The Conjugate Gradient (CG) method results in the tridiagonal orthogonalization of the original matrix, which can be described as $A = QTQ*$, where Q is the Unitary matrix and T is a tridiagonal matrix. When A is non-symmetric, this result cannot be obtained from a CG iteration. Two approaches are followed :

- Use of the so-called Arnoldi Iteration, a process of Hessenberg orthogonalization. This results in $A = QHQ*$ where Q is the Unitary matrix and H is the Hessenberg matrix. (An upper Hessenberg matrix has zero entries below the first sub-diagonal).
- Bi-orthogonalization methods are based on the opposite choice. If we insist on obtaining a tridiagonal result, then we have to give up the unitary transformations, which gives us tridiagonal biorthogonalization : $A = VTV^{-1}$, where V is non-singular but generally not Unitary. The term 'biorthogonal' refers to the fact that though all the columns of V are not orthogonal to each other, they are orthogonal to the columns of $V^{-1}*$.

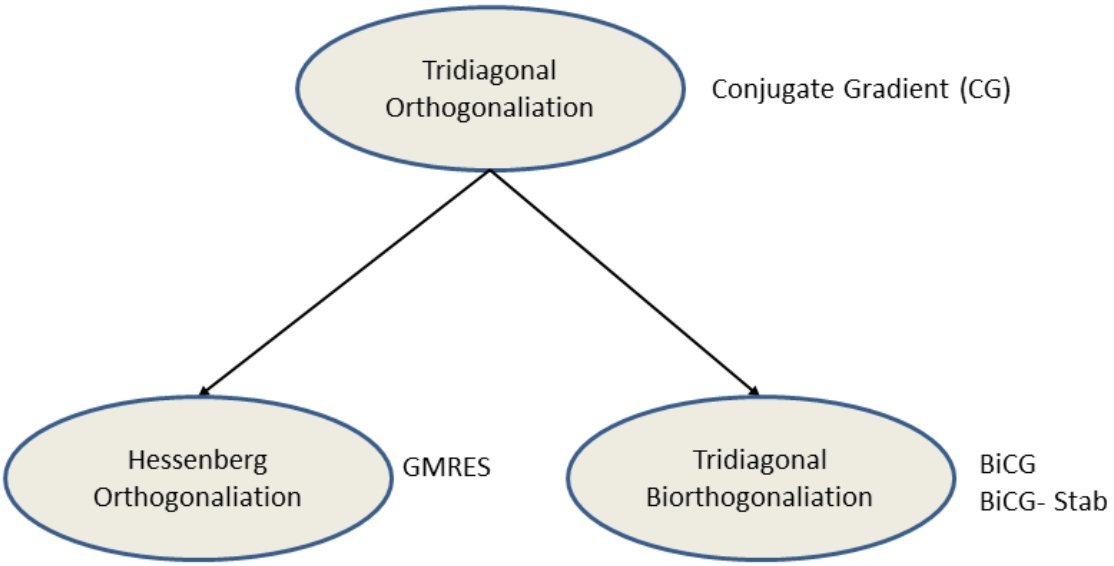

Figure 5.1: Classification of Krylov Subspace iterations

5.2 Arnoldi Iteration and GMRES

Arnolid iteration can be understood as the analogue of Gram-Schmidt type iteration for similary transformations to Hessenberg form. It has the advantage that it can be stopped part-way, leaving one with a partial reduction to hessenberg form that is exploited when dimensions upto Krylov subspace are considered. A simple algorithm of Arnoldi iteration is given below:

Algorithm 5.1 Arnoldi Iteration

b = arbitrary, $q_1 = b/||b||$
for $n = 1, 2, 3, \cdots$ **do**
  $v = Aq_n$
  for $j = 1$ **do**
    $h_{jn} = q_j^* v$
    $v = v - h_{jn} q_j$
  end for
  $h_{n+1,n} = ||v||$
  $q_{n+1} = v/h_{n+1,n}$
end for

The above algorithm can be condensed in the following form:

- The matrices Q_n generated by the Arnoldi iteration are reduced QR factors of the Krylov matrix:
$$K_m = Q_n R_n \tag{5.1}$$
- The Hessenberg matrices H_n are the corresponding projections : $H_n = Q_n^* A Q_n$
- The successive iterates are related by the formula: $AQ_n = Q_{n+1} H_n^{'}$, where $H_n^{'}$ is the $(n+1)xn$ upper-left section of H

The idea of GMRES is one-liner. At step n, the exact solution ($x_* = A^{-1}b$) is approximated by the vector $x_n \in K_n$ that minimizes the norm of the residual $r_n = b - Ax_n$, hence the name : Generalized Minimal Residuals (GMRES). It was proposed in 1986 and is applicable to system of equations when the matrix is general (Non-singular) square matrix. Arnolidi iteration is used to construct a sequence of Krylov matrice Q_n whose columns $q,q_2, \cdots$ successively span the Krylov subspace K_n. Thus we can write $x_n = Qny$, where y represents the vector such that

$$||AQ_n y - b|| = \text{minimum} \tag{5.2}$$

Using the similarity transform, this equation can be written as:

$$||Q_{n+1}H_n^{'}y - b|| = \text{minimum} \tag{5.3}$$

Multiplication by Unitary matrix does not change the norm, thus we can rewrite above equation as:

$$||Q_{n+1}^* Q_{n+1} H_n^{'} y - Q_{n+1}^* b|| = \text{minimum} ||H_n^{'} y - Q_{n+1}^* b|| = \text{minimum} \tag{5.4}$$

Finally, by construction of the Krylov matrices Q_n, $Q_{n+1}^* b = ||b||e_1$ where $e_1 = (1, 0, 0, \cdots)*$. Thus we obtain:

$$||H_n^{'} y - ||b||e_1|| = \text{minimum} \tag{5.5}$$

The GMRES algorithm (unPreconditioned) can be written as :

Algorithm 5.2 GMRES

$q_1 = b/||b||$
for $n = 1, 2, 3, \cdots$ **do**
  ¡step n of Arnoldi iteration, Algorithm 5.1¿
  Find y to minimize $||H_n^{'} y - ||b||e_1||(= ||r_n||)$
  $x_n = Q_n y$
end for

in order to find y, QR factorization can be used which required $O(n^2)$ flops, because of the hessenberg structure of the matrix $H^{'}$. Also it is possible to get the QR factorization of $H_n^{'}$ from that of $H_{n-1}^{'}$ by using Given's Rotation.

One of the disadvantage of the GMRES method is the storage requirements. As it requires storing the whole sequence of the Krylov subspace, a large amount of storage is required as compares to the Conjugate Gradient method. For this reason, restarted versions of this method are used, where computational and storage costs are limited by specifying a fixed number of vectors to be generated.

5.3 BiConjugate Gradient methods

The Biconjugate gradient method (BCG) is the other extension for the non-symmetric matrices. As we saw in the previous section, the principle of GMRES is to pick the vector x_n such that the residual corresponding to x_n is orthogonal to the n^{th} Krylov subspace. The principle of BCG algorithm is to pick x_n in the same sub-space, i.e. $x_n \in K_n$, but to enforce that the residual is orthogonal to $< w_1, A * w_1, \cdots, (A*)^{n-1} w_1 >$, where $w_1 \in R^m$ is an arbitrary vector

satisfying $w_1 * v_1 = 1$. Its advantage is that it can be implemented with three-term recurrences rather than the (n+1) - term recurrences of GMRES (Difference arising from Hessenberg form of matrix vs. Triangular form).

There are two major problems with BiCG method :

- Convergence is slower as compared to GMRES and often erratic. Also, it may have the consequence of reducing the ultimately attainable accuracy because of rounding errors.
- It required multiplication with $A*$ (transpose) as well as A. Computing the transpose brings serialization to the code and thus is not preferred.

To address this problem, other variants of BiCG method were developed. One of them is stabilized BiCG method (Bi-CGSTAB). As for the Conjugate Gradient method, any other Krylov subspace method needs a good preconditioner to ensure fast and robust convergence. Algorithm for the Preconditioned BiCGSTAB method is given below. Converting this algorithm to adjust to the PCG algorithm used in [1] will be one of the future tasks.

Algorithm 5.3 BiCGSTAB

Solve the system of equation given by $Ax = b$ BiCGSTAB
Compute $r^0 = b - Ax^0$ for some initial guess x^0.
Choose $\tilde{r}$ (for example $\tilde{r} = r^0$)
for $i = 1, 2, 3, \cdots$ **do**
  $\rho_{i-1} = isimportanttohavetilder^T r^{i-1}$
  if $\rho_{i-1} = 0$ **then**
    Method Fails
  end if
  if $i = 1$ **then**
    $p^i = r^{i-1}$
  else
    $\beta_{i-1} = \frac{\rho_{i-1}}{\rho_{i-2}} \frac{\alpha_{i-1}}{\omega_{i-1}}$
    $p_i = r^{i-1} + \beta_{i-1}(p^{i-1} - \omega_{i-1} v^{i-1}$
  end if
  Solve $M\tilde{p} = p^i$
  $v^i = A\tilde{p}$
  $\alpha_i = \frac{\rho^{i-1}}{\tilde{r}^T} v^i$
  $s = r^{i-1} - \alpha_i v^i$
  Check norm of s; if small enough, set $x^i = x^{i-1} + \alpha_i \tilde{p}$
  Solve $M\tilde{s} = s$
  $t = A\tilde{s}$
  $\omega_i = \frac{t^T s}{t^T t}$
  $x^i = x^{i-1} + \alpha_i \tilde{p} + \omega_i \tilde{s}$
  $r^i = s - \omega_i t$
  Check Convergence, continue if necessary
  For continuation it is required that $\omega_i \neq 0$
end for

Chapter 6

Test Problems and Further Research

6.1 Test problem for time discretization schemes

To assess the stability of various implicit time integration procedures, we shall consider a simplified problem in one dimension given below as the test problem.

$$\frac{\partial \zeta}{\partial t} + \nabla.(\zeta \mathbf{U} + h\nabla\varphi) = 0, \tag{6.1a}$$

$$\frac{\partial \varphi}{\partial t} + \mathbf{U}.\nabla\varphi + g\zeta = 0, \tag{6.1b}$$

Here, we have neglected the impact of the vertical structure ψ and pressure puls P_s. This is now a coupled initial boundary value problem, where both the equations are hyperbolic in nature and also represents a Hamiltonian Set. Also, the boundary conditions for both ζ and φ are taken as periodic.

Assuming no spatial variation of mean current velocity U and depth h, the above equations can be written as:

$$\frac{\partial \zeta}{\partial t} + \mathbf{U}\frac{\partial \zeta}{\partial x} + h\frac{\partial^2 \varphi}{\partial x^2} = 0, \tag{6.2a}$$

$$\frac{\partial \varphi}{\partial t} + \mathbf{U}\frac{\partial \varphi}{\partial x} + g\zeta = 0, \tag{6.2b}$$

Implicit time integration procedures will require solving a system of linear equations. We will use MATLAB to solve the system after performing spatial discretization and applying the boundary conditions. Idea here is to assess the stability and accuracy of various methods and not the performance.

6.2 Test problem for generalized Krylov Subspace methods

In order to validate and analyze the performance of the generalized Krylov subspace methods, we will test the method with Poisson equation given by:

$$-\Delta u = f(x,y) \quad \text{on} \quad \Omega = (0,1) \times (0,1) \tag{6.3a}$$

$$u(x,y) = 0 \qquad \text{on} \quad \delta\Omega. \tag{6.3b}$$

The discretization of the Poisson equation results in a symmetric matrix where both CG and generalized Kyrlov Subspace methods can be applied. The RRB-k PCG method developed in [1] has been tested for the Poisson equation, and we will compare the performance and the storage requirements of RRB-k PCG method with the generalized Krylov Subspace methods.

6.3 Realistic problems Ijssel, Plymouth

Two realistic problems are obtained from MARIN's database and previous simulations by Martijn in [1]. Testing will be carried out with respect to the validated models in [1]. A very small time step (much lower than then allowable limit) will be used to first carry out simulations with the current model which uses explicit leap frog time integration. Results from implicit time integration will then be compared with the results from explicit time integration.

6.3.1 The Gelderse IJssel

The Gelderse IJssel, a small river, is a branch from the Rhine in the Dutch provinces Gelder-land and Overijssel. The river flows from Westervoort and discharges in the IJsselmeer. In Figure 6.1 a (small) part of the river is shown. From this part several test problems are extracted. This is done by extracting small regions out of the the displayed region . For the discretization an equidistant 2 m by 2 m grid is used. During later part of the thesis, non-uniform grids will be explored and the discretization modified accordingly.

Figure 6.1: The Gelderse IJssel (Google Maps)

6.3.2 The Plymouth Sound

Plymouth Sound is a bay located at Plymouth, a town in the South shore region of England, United Kingdom. The Plymouth Break- water is a dam in the centre of the Plymouth Sound which protects anchored ships in the Northern part of the Plymouth Sound against south-western storms. From this region also test problems are extracted, see Figure 6.2. For the discretization an equidistant 5 m by 5 m grid is used. For the non-uniform grid structure, discretization will be modified accordingly.

Figure 6.2: The Plymouth Sound (Google Maps)

6.3.3 Research Questions

Following research questions and goals, are the topics that will be treated during the rest of the research:

- Implement generalized Kyrlov subspace method.
  - Analyze the impact on the performance (speed-up) and storage requirements for the generalized Kyrlov Subspace methods for CUDA and C++ code.
- Implement various time integration schemes.
  - Analyze the stability and accuracy of various time integration schemes and assess the storage and performance requirements.
  - From the current time step of 0.01 seconds, how much improvement without loss of accuracy and performance can be achieved.
- If time integration schemes do not yield significant improvement in stability, analyze the possibility of using the non-uniform mesh.
  - Provide a framework for the use of either Adaptive mesh refinement or moving mesh method.

References

[1] Jong, M. de, Developing a CUDA solver for large sparse matrices for MARIN, MSc Thesis Applied Mathematics, Delft University of Technology, Delft, February 2012.

[2] Klopman, G., Variational Boussinesq Modelling of Surface Gravity Waves over Bathymetry, Phd Thesis, University of Twenty, Twente, May 2010.

[3] Wout, E. van 't, Improving the Linear Solver used in the Interactive Wave Model of a Real-time Ship Simulator, MSc Thesis Applied Mathematics, Delft University of Technology, Delft, August 2009.

[4] Keppens R., Tth G., Botchev M. A, and Ploeg A. van der, Implicit and semi-implicit schemes: algorithms, International Journal for Numerical Methods in Fluids, 30, 335-352, 1999.

[5] Botchev M. A, Sleijpen G. L. G. and Vorst H. A. van der, Low-dimensional Krylov subspace iterations for enhancing stability of time-step integration schemes, Preprint 1004, Dept. of Mathematics, Utrecht University, the Netherlands , (1997).

[6] Trefethen L.N, Bau D. III , Numerical Linear Algebra , Society for Industrial and Applied, 1997

[7] Hundsdorfer W. , Verwer J.G., Numerical Solution of Time-Dependent Advection-Diffusion Reaction Equations, Springer, 2003