User Manual for the Hybrid 2DH-3D Program and its Constituents

William Pringle
Department of Urban Management, Graduate School of Engineering,
Kyoto University

July 11, 2014

Contents

Chapter 1

Introduction

1.1 Background

The 2011 off the Pacific Coast of Tohoku Earthquake Tsunami (2011 TET) induced massive damage. We note in particular the extent to which structures and infrastructure were damaged not only from wave impact forces but also floating debris. Damage to coastal structures was a predominant feature and their lack of resilience an important factor in the scale of the tsunami devastation. On the other hand, the lack of damage to large concrete structures and the cover they provided for saving human lives was also evident. Thus, the interaction of the tsunami flow with both coastal and land-based structures as well as objects that may become floating debris play an extremely important role in how such huge disasters end up impacting inhabited areas.

With the above in mind, we wish to analyse tsunami behaviour in greater detail, as it interacts with structures and inundates on land. Through numerical models this may be achievable but traditional horizontal two-dimensional (2DH) tsunami solvers may breakdown if we consider interactions against steep topography or vertical faces, particularly structures. More specifically, if the problem becomes more flow-like rather than wave-like the assumptions of the 2DH solvers may be limiting. For a more detailed assessment of flows, turbulence and forces, 3D models are better suited. In our study we thus make use of a RANS based 3D solver that uses the VOF method for free surface calculation and $k - \epsilon$ model for turbulence modelling [9]. Considering two-way flow interaction, we tightly couple this with a typical shallow water equation based 2DH solver for modelling tsunami propagation and typical near-shore behaviour, to obtain a single Hybrid 2DH-3D model.

The Hybrid 2DH-3D model was applied to Kamaishi Bay, Japan to simulate the 2011 TET and was verified with survey data of the maximum inundation heights around the coast in a previous study [6]. However, although our results were promising, the uncertainty and reliance on the quality of topographical and structural data involved with field applications made a detailed assessment difficult. In the future we wish to test the model under more fundamental conditions, and try to find areas where the 2DH and 3D model results differ so as to determine suitable locations for immersing the 3D domain in the 2DH domain. Furthermore, we extend the model so that a number of domain configurations may be considered that was not possible under the previous model presented in [6].

1.2 Manual Overview

This manual was designed to help the user with both the 2DH model and the Hybrid 2DH-3D model. It begins by introducing the governing equations and a brief overview of the calculation schemes of both the 2DH and 3D models. The coupling procedure for the Hybrid 2DH-3D model is then described. Once such formalities are out of the way, the actual procedure for using the programs are specified in detail. This procedure includes; 1) the generation of mesh and land-form data, 2) preparation of input waves on the boundary or initial water level conditions, 3) preparation for getting data output at the desired locations, 4) preparation of the input text file, and 5) post-analysis. This manual aims to be exhaustive (eventually) and any options or problems and issues that arise that you think is not covered in this manual, please alert me. With your help we can document such situations and the solution for those problems to help with future users (including me and you).

1.3 2DH Model

This section describes the 2DH model based on the shallow water equations. Firstly, the governing equations are introduced. Secondly, the actual calculation scheme is described and discussed.

1.3.1 Governing Equations

The 2DH model is governed by the so called shallow water or Saint-venant equations (SWE). They are characterised by the assumption that the wave height, H is much smaller than the mean water depth, h and the wave length, $L >> h$, so that a hydrostatic pressure assumption becomes reasonable as well as neglecting the vertical acceleration of the particles. Both of these assumptions can be easily derived from linear wave theory. In its conservative form, which is derived from the conservation of mass and conservation momentum, we describe the motion through momentum fluxes. In its non-conservative form we write the equations in terms of velocities. It is the conservative form that we use in this model so that in Cartesian Coordinates we end up with:

$$\frac{\partial \eta}{\partial t} + \frac{\partial M}{\partial x} + \frac{\partial N}{\partial y} = 0 \tag{1.1}$$

$$\frac{\partial M}{\partial t} + \frac{\partial}{\partial x}\left(\frac{M^2}{D}\right) + \frac{\partial}{\partial y}\left(\frac{MN}{D}\right) + gD\frac{\partial \eta}{\partial x} + \frac{gn^2 M\sqrt{M^2+N^2}}{D^{7/3}} = 0 \tag{1.2}$$

$$\frac{\partial N}{\partial t} + \frac{\partial}{\partial y}\left(\frac{N^2}{D}\right) + \frac{\partial}{\partial x}\left(\frac{NM}{D}\right) + gD\frac{\partial \eta}{\partial y} + \frac{gn^2 N\sqrt{M^2+N^2}}{D^{7/3}} = 0 \tag{1.3}$$

where, η is the free surface elevation, $M(= uD)$ and $N(= vD)$ are the momentum fluxes in the x and y directions respectively. Here, $D(= \eta + h)$ is the total water depth, and u and v are the respective depth-averaged velocity components. Finally, g is the acceleration due to gravity and n is Manning's friction coefficient for the bed stress term. Eqn. 1.1 is the continuity equation, and Eqns. 1.2 and 1.3 are the conservation momentum equations. Within Eqn. 1.2 we see from left to right; the unsteady acceleration term, the two nonlinear advection terms, the pressure gradient term and the bed stress term given in terms of Manning's n roughness coefficients. The same is true for Eqn. 1.3.

For the propagation of tsunami in deep water, not only does the "shallow water" restriction hold because of its immense wave length, H/h becomes extremely small. In such a case we can use the "linearised" form of the SWE (LSW) by ignoring the two nonlinear advection terms in each of Eqns. 1.2 and 1.3 and linearising the pressure gradient term. This is preferable since the computation time can be significantly reduced with negligible effects on the solution. However, as a tsunami propagates up onto the continental shelf and shoals it is usually better to use the full nonlinear form of the SWE especially when considering inundation. This program gives some options for choosing whether or not to use the linear or nonlinear forms. This is described in detail in section 2.5.

Finally, if we are interested in a very wide area of calculation that covers the scale of oceans or a significant part thereof, we might not only be prepared to use just the LSW, but also the Coriolis force may become important (at least more than the nonlinear components). This importance can be indicated by the Rossby number, $Ro = U/Lf$, where U is the characteristic velocity $(= \sqrt{gh})$, L is the characteristic length, and f is the Coriolis force defined below. If the Rossby number is small then the Coriolis forces become significant. Clearly this happens when L gets larger. In such a case it becomes convenient to write our equations in Spherical Coordinates. The LSW equations in such coordinates including the Coriolis force then become:

$$\frac{\partial \eta}{\partial t} + \frac{1}{R\cos\phi}\left[\frac{\partial M}{\partial \psi} + \frac{\partial(\cos\phi N)}{\partial \phi}\right] = 0 \tag{1.4}$$

$$\frac{\partial M}{\partial t} + \frac{gh}{R\cos\phi}\frac{\partial \eta}{\partial \psi} - fN + \Phi_x = 0 \tag{1.5}$$

$$\frac{\partial N}{\partial t} + \frac{gh}{R}\frac{\partial \eta}{\partial \phi} + fQ + \Phi_y = 0 \tag{1.6}$$

where, $R(= 6\ 378\ 137\ \text{m})$ is the equatorial radius of the earth, ϕ is latitude, ψ is longitude (in radians), and f is the Coriolis force equal to $2\Omega\sin\phi$ where $\Omega(= 7.292 \times 10^{-5}$ rad/s) is the angular rotation of the Earth. Φ is the dispersion correction term that is introduced in Section 1.3.2 under the dispersion correction heading. One may realise that we have also dropped our bed stress term as well, which is negligible for propagation in the deep ocean.

One extra phenomenon that we have yet to consider is dispersion. Since tsunami waves are very long dispersion can usually be neglected, however over very long distances of travel (e.g. traversing across an entire ocean) such dispersion effects accumulate and become more significant. We might then want to make use of a Boussinesq formulation where dispersion effects can be considered. However, in such a case higher-order terms are introduced and a higher-order scheme must typically be formulated (different to the scheme shown here) so that the computational load is again significantly increased. Such models that include dispersion effects are (e.g. [3, 4, 5]). However, interestingly [1] has introduced a nice way to deal with this for the leapgfrog method (the scheme we adopt here). It so happens that if we make use of the numerical dispersion in the model to match that of the actual dispersion of the linear Boussinesq equations then we can model the physical dispersion properly. We will go over how the dispersion correction is achieved through the numerical scheme in Section 1.3.2.

1.3.2 Calculation Scheme

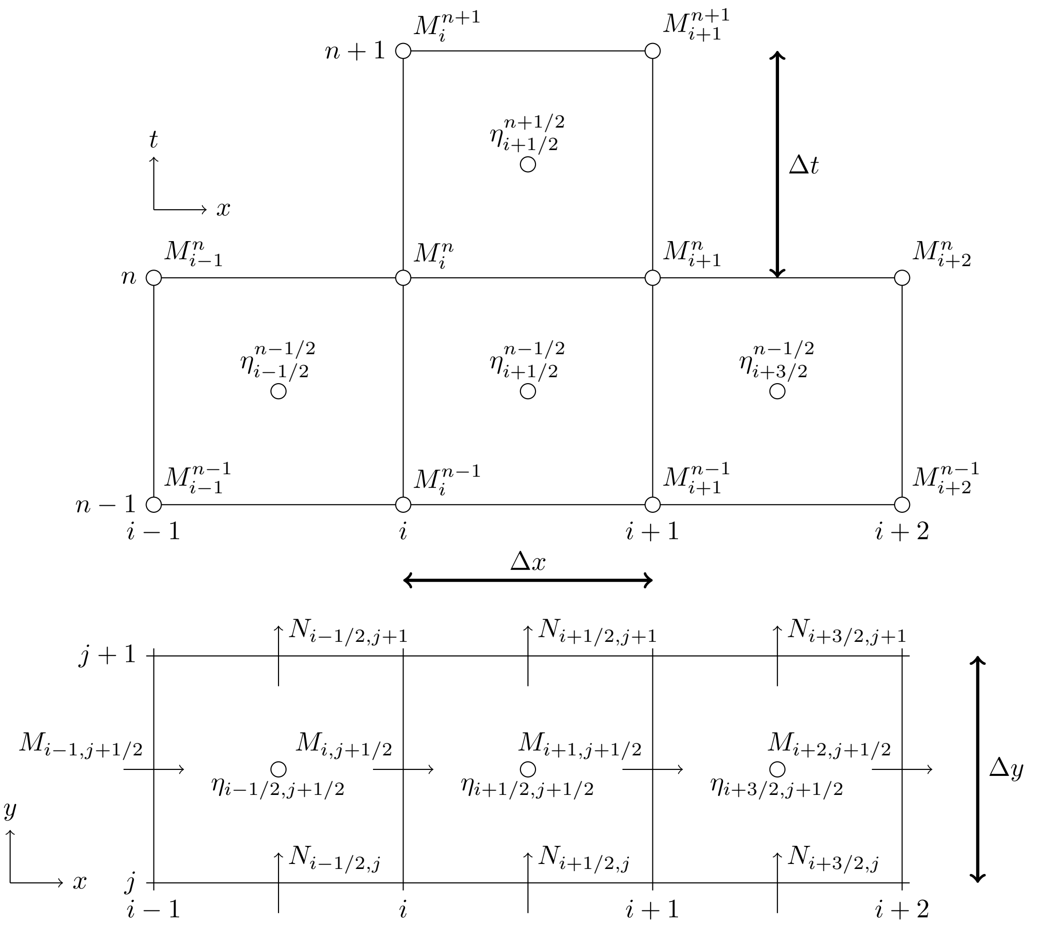


Figure 1.1: Staggered arrangement of the free surfaces and momentum fluxes in both time and space. Both the $x-t$ and $x-y$ planes are shown

The calculation scheme is the staggered leapfrog method. This means that we stagger our momentum fluxes and free surface values in both space and time. Momentum fluxes are defined at the cell boundaries and the free surface is defined in the cell center. Water depths at the cell boundaries are thus found by interpolation of the free surfaces. Fig. 1.1 shows the staggered arrangement of the fluxes in both the $x-t$ plane and the $x-y$ plane. Note of course that there of course would be an equivalent $y-t$ plane for the full 2DH fomulation but it is analogous to $x-t$ so it is not included here. The staggered leapfrog has the advantage that even by using the most simple form of discretization (1st order Euler), the scheme automatically becomes 2nd order. This is because the staggering allows the 1st order Euler to actually become analagous to the 2nd order central difference and Taylor expansions around the relevant variable will prove this. At least this is true for the linearised form of the equations, and in fact using the dispersion corrected scheme it even becomes 3rd order in space and time [1]. When we consider the nonlinear form, the accuracy of the advection terms may undermine the scheme if a 1st order discretization is used because the staggering is not useful in the calculation of these terms. However, typically the 1st order upwind method has been used for the nonlinear terms and I also tentatively adopt this method. I have also made a formulation for the 3rd order upwind, but it seems unstable particularly for inundation and when the wave face becomes steep. This should be investigated further but I have a feeling that if we would like greater order of accuracy a different calculation scheme should be used.

With the above in mind, the formulation for the continuity and momentum equations (Eqns. 1.1, 1.2 and 1.3 respectively) is given as follows:

$$\eta^{n+1/2}_{i+1/2,j+1/2} = \eta^{n-1/2}_{i+1/2,j+1/2} - \frac{\Delta t}{\Delta x}\left(M^n_{i+1,j+1/2} - M^n_{i,j+1/2}\right) + \frac{\Delta t}{\Delta y}\left(N^n_{i+1/2,j+1} - N^n_{i+1/2,j}\right) \quad (1.7)$$

$$\begin{aligned} M^{n+1}_{i,j+1/2} &= M^n_{i,j+1/2} - gD^{n+1/2}_{i,j+1/2}\frac{\Delta t}{\Delta x}\left(\eta^{n+1/2}_{i+1/2,j+1/2} - \eta^{n+1/2}_{i-1/2,j+1/2}\right) \\ &- \frac{\Delta t}{\Delta x}\left(\lambda_{11}\frac{(M^n_{i+1,j+1/2})^2}{D^n_{i+1,j+1/2}} + \lambda_{12}\frac{(M^n_{i,j+1/2})^2}{D^n_{i,j+1/2}} + \lambda_{13}\frac{(M^n_{i-1,j+1/2})^2}{D^n_{i-1,j+1/2}}\right) \\ &- \frac{\Delta t}{\Delta y}\left(\lambda_{21}\frac{M^n_{i,j+3/2}N^n_{i,j+3/2}}{D^n_{i,j+3/2}} + \lambda_{22}\frac{M^n_{i,j+1/2}N^n_{i,j+1/2}}{D^n_{i,j+1/2}} + \lambda_{23}\frac{M^n_{i,j-1/2}N^n_{i,j-1/2}}{D^n_{i,j-1/2}}\right) \\ &- gn^2\frac{\Delta t}{2}\frac{(M^{n+1}_{i,j+1/2} + M^n_{i,j+1/2})\sqrt{(M^n_{i,j+1/2})^2 + (N^n_{i,j+1/2})^2}}{(D^n_{i,j+1/2})^{7/3}} \end{aligned} \quad (1.8)$$

$$\begin{aligned} N^{n+1}_{i+1/2,j} &= N^n_{i+1/2,j} - gD^{n+1/2}_{i+1/2,j}\frac{\Delta t}{\Delta y}\left(\eta^{n+1/2}_{i+1/2,j+1/2} - \eta^{n+1/2}_{i+1/2,j-1/2}\right) \\ &- \frac{\Delta t}{\Delta y}\left(\lambda_{31}\frac{(N^n_{i+1/2,j+1})^2}{D^n_{i+1/2,j+1}} + \lambda_{32}\frac{(N^n_{i+1/2,j})^2}{D^n_{i+1/2,j}} + \lambda_{33}\frac{(N^n_{i+1/2,j-1})^2}{D^n_{i+1/2,j-1}}\right) \\ &- \frac{\Delta t}{\Delta x}\left(\lambda_{41}\frac{M^n_{i+3/2,j}N^n_{i+3/2,j}}{D^n_{i+3/2,j}} + \lambda_{42}\frac{M^n_{i+1/2,j}N^n_{i+1/2,j}}{D^n_{i+1/2,j}} + \lambda_{43}\frac{M^n_{i-1/2,j}N^n_{i-1/2,j}}{D^n_{i-1/2,j}}\right) \\ &- gn^2\frac{\Delta t}{2}\frac{(N^{n+1}_{i+1/2,j} + N^n_{i+1/2,j})\sqrt{(M^n_{i+1/2,j})^2 + (N^n_{i+1/2,j})^2}}{(D^n_{i+1/2,j})^{7/3}} \end{aligned} \quad (1.9)$$

where the values of λ are given by the first-order upwind method:

$$\begin{aligned} M^n_{i,j+1/2} \geq 0: &\quad \lambda_{11} = 0, \lambda_{12} = 1, \lambda_{13} = -1 \qquad & M^n_{i,j+1/2} < 0: &\quad \lambda_{11} = 1, \lambda_{12} = -1, \lambda_{13} = 0 \\ N^n_{i,j+1/2} \geq 0: &\quad \lambda_{21} = 0, \lambda_{22} = 1, \lambda_{23} = -1 \qquad & N^n_{i,j+1/2} < 0: &\quad \lambda_{21} = 1, \lambda_{22} = -1, \lambda_{23} = 0 \\ M^n_{i+1/2,j} \geq 0: &\quad \lambda_{31} = 0, \lambda_{32} = 1, \lambda_{33} = -1 \qquad & M^n_{i+1/2,j} < 0: &\quad \lambda_{31} = 1, \lambda_{32} = -1, \lambda_{33} = 0 \\ N^n_{i+1/2,j} \geq 0: &\quad \lambda_{41} = 0, \lambda_{42} = 1, \lambda_{43} = -1 \qquad & N^n_{i+1/2,j} < 0: &\quad \lambda_{41} = 1, \lambda_{42} = -1, \lambda_{43} = 0 \end{aligned}$$

Furthermore, the values of D, M and N that are specified in different locations to their definitions in Fig. 1.1 are estimated through linear interpolation (assuming constant Δx, Δy and Δt for simplicity. However, the numerical code is actually formulated for non-constant values):

$$D^{n-1/2}_{i,j+1/2} = h_{i,j+1/2} + 0.5(\eta^{n-1/2}_{i+1/2,j+1/2} + \eta^{n-1/2}_{i-1/2,j+1/2}) \tag{1.10}$$

$$D^{n}_{i,j+1/2} = 0.5(D^{n+1/2}_{i,j+1/2} + D^{n-1/2}_{i,j+1/2}) \tag{1.11}$$

$$M^{n}_{i+1/2,j} = 0.25(M^{n}_{i+1,j+1/2} + M^{n}_{i,j+1/2} + M^{n}_{i+1,j-1/2} + M^{n}_{i,j-1/2}) \tag{1.12}$$

$$N^{n}_{i,j+1/2} = 0.25(N^{n}_{i+1/2,j+1} + N^{n}_{i+1/2,j} + N^{n}_{i-1/2,j+1} + N^{n}_{i-1/2,j}) \tag{1.13}$$

Moving Boundary Algorithm

In the scheme we also consider the possibility or runup and rundown on originally dry land (where $h < 0$). To consider this we need an algorithm to model the moving boundary. The algorithm we adopt is simple, where we assume a staircase landform so that water may only move through onto the dry cell when the water level is higher than the ground level of the dry cell as illustrated in Fig. 1.2. When the water level is lower or equal to the ground level of the dry cell the flux at the cell boundary is set to zero so that for a dry cell all the surrounding fluxes at the cell boundary are zero. When the water level becomes larger than the ground level of the dry cell, the flux may be calculated in the normal way, however we set the inundation depth shown in Fig. 1.2 equal to: $D_{i+1} = \eta_{i+1/2} + h_{i+3/2}$. Furthermore, in reality it is usually necessary to set a minimum allowable water depth so that unrealistic velocities do not arise with tiny depths particularly because of the bottom friction term. This of course may also save computing time by excluding such dry regions from the calculation. Thus, for the example in Fig. 1.2, $\eta_{i+1/2} + h_{i+3/2} > D_{min}$ must be satisfied before inundation can occur. Usually, D_{min} may be set to approximately 1 mm in a large scale tsunami calculation but smaller values may be used with smaller scale problems.

Seawalls, breakwaters and banks

The 2DH program may also consider the presence of walls such as breakwaters or seawalls at the cell boundaries as illustrated at the bottom of Fig. 1.2. Here if the water level on both sides of the wall is lower than the wall the flux will be set to zero. If this is not the case, the flux will be calculated via Homma's overtopping weir formula instead of the momentum equations (Eqns. 1.8 and 1.9):

$$M_i = 0.35 D^L_i \sqrt{2 g D^L_i} \qquad D^R_i \leq \frac{2}{3} D^L_i \tag{1.14}$$

$$M_i = 0.91 D^L_i \sqrt{2 g (D^L_i - D^R_i)} \qquad D^R_i > \frac{2}{3} D^L_i \tag{1.15}$$

When $D^L_i > D^R_i$. Thus with this method, the actual water depth at the cell centers are not

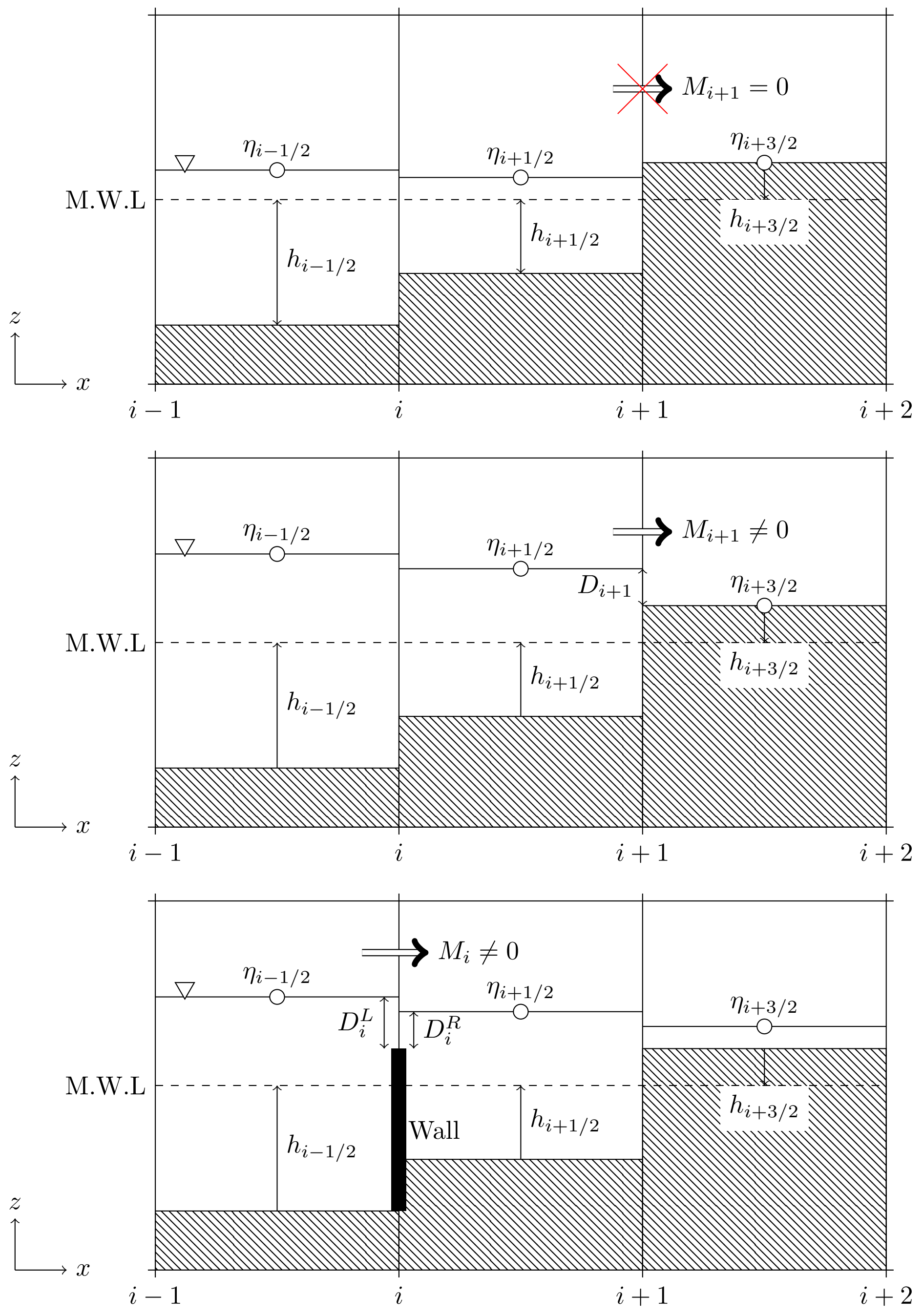


Figure 1.2: Sketch of the moving boundary algorithm and consideration of walls at cell boundaries

affected by the wall while we can easily use the normal landform and add in walls later on (see Section 2.2.2).

Dispersion Correction

In Section 1.3.1 we mentioned the possibility of using the numerical dispersion in the leapfrog method to mimic the actual dispersion from the linear Boussinesq equations. [2] actually first showed that if the still water depth, cell size and step size was properly matched $-(\Delta x)^2 = 4h^2 + gh(\Delta t)^2-$ then the leapfrog scheme automatically satisfies the linear Boussinesq dispersion properties. However, this is only true for 1DH wave propagation and not for wave propagation along the diagonal. To include the dispersion properties for the diagonal as well we need to add an extra term to the governing equations. Furthermore, we may realise if h is not constant, then Δx should be changed to satisfy the requirement. Thus, [1] gives us the solutions to these problems so that we can use a constant Δx for varying h by the following equations and set the dispersion correction term, Φ in Eqns. 1.5 and 1.6 as:

$$\Phi_x = \frac{\alpha}{12}(\Delta x)^2 gh\frac{\partial^3\eta}{\partial x^3} + \frac{\gamma}{12}(\Delta y)^2 gh\frac{\partial^3\eta}{\partial y^2 x} \tag{1.16}$$

$$\Phi_y = \frac{\alpha}{12}(\Delta y)^2 gh\frac{\partial^3\eta}{\partial y^3} + \frac{\gamma}{12}(\Delta x)^2 gh\frac{\partial^3\eta}{\partial x^2 y} \tag{1.17}$$

$$\alpha = \frac{4h^2 + gh(\Delta t)^2 - (\Delta x)^2}{(\Delta x)^2} \tag{1.18}$$

$$\gamma = \alpha + 1 \tag{1.19}$$

Thus for a given h we can calculate α and γ at each cell and this will stay constant for the calculation as long as Δt does (it should for the leapfrog scheme which would become technically incorrect if Δt is changed mid-calculation unless some interpolation is used). There is however a limit to the applicability of this technique. It so happens that the range of α is from -1 to 1. Moreover, Δt should be chosen to satisfy the stability condition, which changes with α. For example, when $\alpha = 0$ the Courant number should be less than 0.866 (when $\gamma, \alpha = 0$ in [2] scheme $Cr_{lim} = 0.707$). In fact the limit of the Courant number, Cr_{lim} is given by:

$$Cr_{lim} = \sqrt{\frac{3}{4(1-\alpha)}} = \frac{\sqrt{gh}\Delta t}{\Delta x} \tag{1.20}$$

Thus, it is difficult provide exactly a requirement for Δx but $\alpha = -1$ corresponds to zero depth and is non-physical so taking the limit when $\alpha = 1$ we obtain the requirement from Eqn. 1.18 that:

$$\Delta x \geq \sqrt{2h_{max}^2 + 0.5gh_{max}(\Delta t)^2} \tag{1.21}$$

Taking the limit when $\Delta t = 0$ gives us the absolute minimum requirement for Δx for this scheme to be possible: $\Delta x_{min} = \sqrt{2} h_{max}$.

Thus, we can already see that for this scheme to work the cell size should be fairly large. But fortunately as mentioned before, the dispersion effects are usually only important for propagation in the open ocean over large distances but typically not nearshore (where we should in any case be looking to balance the dispersion effects with higher-order nonlinear terms found in full Boussinesq models anyway). Typically we do in fact use large cell sizes in these regions so here we should be able to apply our scheme. For example, if the water depth in the ocean was 1500 m, then the cell size should be greater than 2120 m. Note that if we assume that our cell size is constant we need to use the maximum value of h in the computational domain (where we use the dispersion correction scheme), written as h_{max} in Eqn. 1.21. If we cannot satisfy this equation then too little dispersion will be added to the scheme, but it may be better than none. In any case, some care should be taken in determing Δx. After we determine that the scheme may work and the cell size determined, we should ensure Δt is chosen to satisfy all requirements as shown in the equations above everywhere in the calculation domain.

Stability limit

As already hinted in the dispersion correction subsection, the stability of the calculation scheme depends on the C.F.L condition (Courant Fliedlichs Law). For the nonlinear equation we should also consider the flow speed to be part of the characteristic velocity so that the Courant number becomes equal to:

$$Cr = \max \left[\frac{(|u| + \sqrt{gD})_{max} \Delta t}{\Delta x}, \frac{(|v| + \sqrt{gD})_{max} \Delta t}{\Delta y} \right] \tag{1.22}$$

Where the values of u, v and D are calculated at every cell and the maximum value throughout the entire domain is adopted as Cr. If the dispersion correction scheme is not used then for the linear scheme the limit of the Courant number is equal to $Cr_{lim} = 0.707$. Furthermore, considering the nonlinear terms, the leading order truncation error from the first order upwind discretizations actually increases as the Courant number becomes smaller as shown in [8]. Thus, it is generally desirable that we keep Cr as close as possible to Cr_{lim}. This can be challenging though when we do not know the maximum u, v and D *a priori* that might occur in the calculation. One way is just to estimate or even set a maximum allowable value of u and v for the calculation, while D can usually be estimated from the problem at hand. A conservative guess would be to apply $|u|$ or $|v|$ equal to 10 m/s in the region where we would expect the maximum D to occur and estimate Cr that way. The other method is to consider the possibility of changing Δt throughout the calculation. However, this requires a slight reformulation of the scheme for it to be technically correct. Furthermore, as mentioned for the dispersion correction, we would need to change our values of α and γ during the calculation in such a case. As such, if possible it is best to avoid changing Δt throughout the calculation.

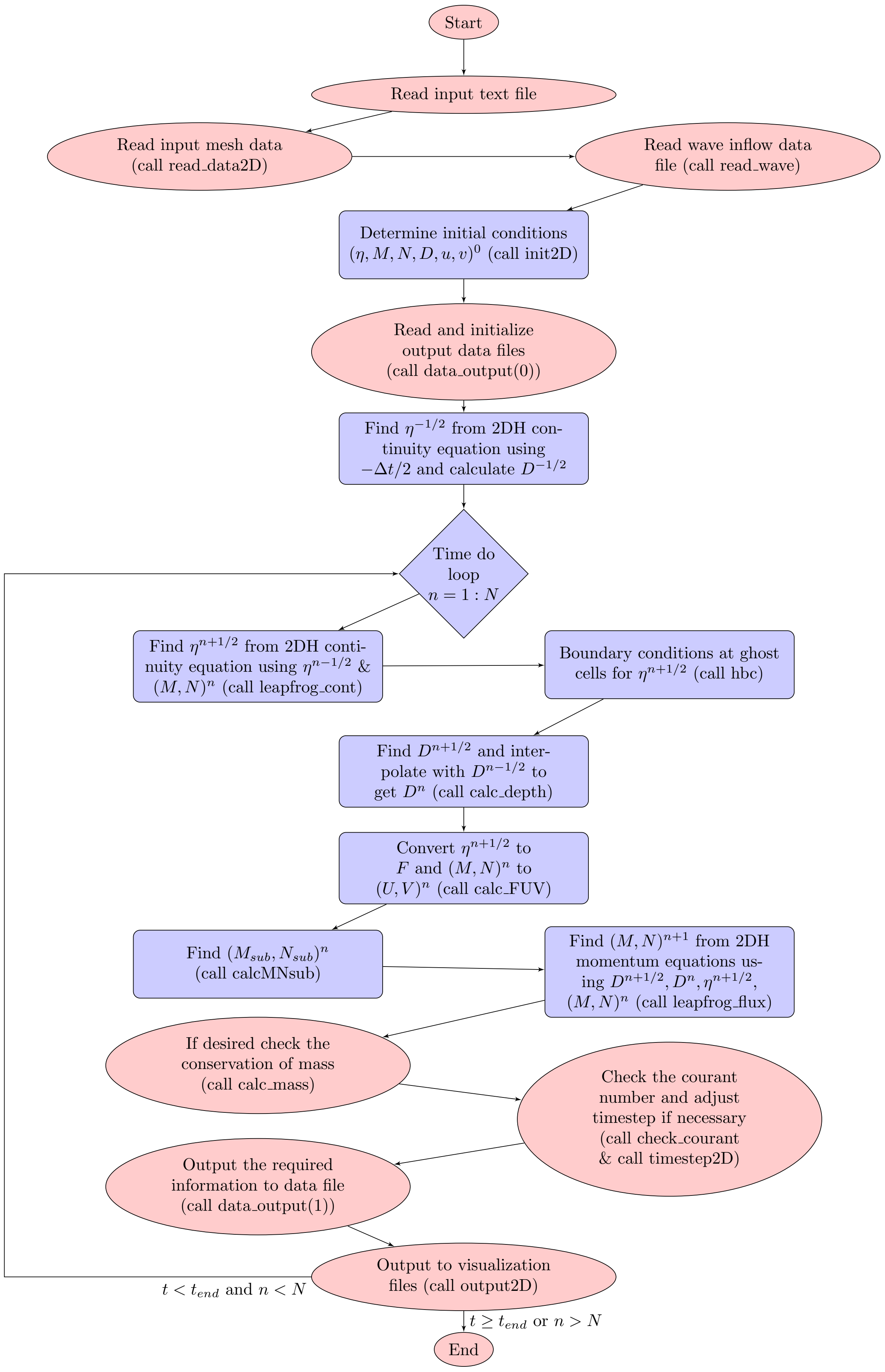

Start
Read input text file
Read input mesh data (call read_data2D)
Read wave inflow data file (call read_wave)
Determine initial conditions $(\eta, M, N, D, u, v)^0$ (call init2D)
Read and initialize output data files (call data_output(0))
Find $\eta^{-1/2}$ from 2DH continuity equation using $-\Delta t/2$ and calculate $D^{-1/2}$
Time do loop $n = 1 : N$
Find $\eta^{n+1/2}$ from 2DH continuity equation using $\eta^{n-1/2}$ & $(M,N)^n$ (call leapfrog_cont)
Boundary conditions at ghost cells for $\eta^{n+1/2}$ (call hbc)
Find $D^{n+1/2}$ and interpolate with $D^{n-1/2}$ to get D^n (call calc_depth)
Convert $\eta^{n+1/2}$ to F and $(M,N)^n$ to $(U,V)^n$ (call calc_FUV)
Find $(M_{sub}, N_{sub})^n$ (call calcMNsub)
Find $(M,N)^{n+1}$ from 2DH momentum equations using $D^{n+1/2}, D^n, \eta^{n+1/2}$, $(M,N)^n$ (call leapfrog_flux)
If desired check the conservation of mass (call calc_mass)
Check the courant number and adjust timestep if necessary (call check_courant & call timestep2D)
Output the required information to data file (call data_output(1))
Output to visualization files (call output2D)
$t < t_{end}$ and $n < N$
$t \geq t_{end}$ or $n > N$
End

Figure 1.3: Flowchart illustrating the detailed calculation procedure of the 2DH model

1.4 3D Model

This section describes the 3D model based on Reynolds averaged Navier-Stokes (RANS) equations. In terms of full 3D models that have few assumptions on fluid flow RANS models are among the most efficient. This is because we by taking the Reynolds average of the transport quantities the fluctuations of the fluid flow which are typically trivial but otherwise require an extremely fine mesh to resolve in direct numerical simulations (DNS) can be ignored and instead their effects are estimated through a turbulence model. In fact such a model can be more useful than DNS in some respects since it easy to split up certain variables and identify each as representing a certain quantity e.g. turbulence kinetic energy (TKE). Thus, we can make more sense of the overall picture of the flow. Another type of full 3D model available is what we call a large-eddy model (LES). LES is typically known as being more accurate than RANS models while naturally being more computationally expensive. The idea behind such a model is as its name suggests is to split up the eddy sizes into two sizes, large eddies greater in scale than the computational cell, and small eddies of sub-grid scale. Thus, we compute the full Navier-Stokes equations for the large eddy scales and use a sub-grid scale model to estimate the dissipation effects of the smaller eddies. I will refrain from going into any further detail, but fair to assume that for a simulation computed on any significant scale (such as a tsunami) a RANS model is more computationally acceptable than either LES or DNS. However, I will add that hybrid RANS-LES models are available and may be of interest in the coming future.

1.4.1 Governing Equations

The RANS equations are described below using Einstein's notation:

1.4.2 Calculation Scheme

1.5 Hybrid 2DH-3D Coupling Procedure

In this section I focus on a new procedure for coupling the two models. However, I must first clarify that a standard staggered grid approach is used in both models so that the velocities (u, v, w) and fluxes $(M(= uD), N(= vD)$, where D is the water depth) are defined on cell boundaries. Free surface levels, η and pressures, p are defined on cell centres.

Previously in [6], the scheme was devised so that only the horizontal fluxes from the 2DH model were converted to velocities and used as an input to the 3D model. Here the vertical velocity distribution was assumed to be constant based on the depth-averaged formulation of the shallow water equations. Following this, the newly calculated water surface and fluxes in the first column of the 3D model was used as a boundary condition for the 2DH model. The formulation was adequate for the calculation where we input a long tsunami wave travelling from the 2DH domain to the 3D domain. However, the problem arises that when flow retreats back from the 3D model the momentum normal to the boundary is not sufficiently transferred back through.

In the new version of the model I adjust the scheme so that in the 2DH water column (see Fig. 1.4) the horizontal velocities are input as an arbitrary velocity distribution. The velocities u_v, v_{v-1} and w_{v-1} are input into the 3D model based on the vertical velocity distribution profile in the water column of the 3D model at the previous timestep so that the sum of the normal fluxes and tangential fluxes on each cell boundary is equivalent to M_s and N_s respectively, and $w_{v-1,surf} = \partial\eta/\partial t = -(\partial M_s/\partial x + \partial N_s/\partial y)$ where $w_{v-1,surf}$ is the vertical velocity at the water surface. The first step in this process is to assume a distribution profile as shown in Fig. 1.5. Then for w_{v-1} since we already know $w_{v-1,surf}$ we can simply assign the w_{v-1} velocities at each cell based on the ratio a_k determined at the adjacent cell $_{vk}$ in the 3D domain for all cells k in the vertical direction. For u_v and v_{v-1}, we sum the $a_k F_k \Delta z_k$ fractions at the pertinent cell boundaries for all cells k in the vertical direction as shown in the denominators of Eqn. 1.23. The surface velocities, $u_{v,surf}$ and $v_{v-1,surf}$ are then found by dividing M_s and N_s respectively from the 2DH model by this sum:

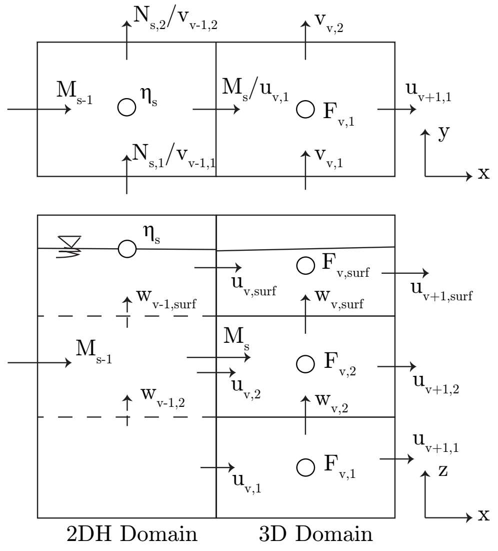


Figure 1.4: The arrangement of the various quantities at the 2DH-3D boundary. $_s$ and $_v$ indicate the variables from the 2DH model and the 3D model respectively

$$u_{surf} = \frac{M_s}{\sum_{k=1}^{surf} a_{kx} F_{vk} \Delta z_k} \quad , \quad v_{surf} = \frac{N_s}{\sum_{k=1}^{surf} a_{ky} F_{vk} \Delta z_k} \tag{1.23}$$

where, a_{surf} is equal to 1 and a_{kx} (a_{ky}) elsewhere is the ratio $u_{v+1,k}/u_{v+1,surf}$ ($v_{v,k}/v_{v,surf}$) found at the adjacent cell boundary in the 3D domain. Thus, with u_{surf} and v_{surf} found, u and v can be assigned at all cells k in the vertical direction as already done with the w velocities.

Concerning the free surface, η is defined every half-time step within the 2DH model leapfrog scheme while the free surface is calculated at every whole time step in the 3D model. This may cause us some problems since we might want to know η^{n+1} as an input to the 3D calculation before $\eta^{n+3/2}$ has even been calculated in the continuity equation thus making interpolation also impossible. We avoid such a problem the following way:

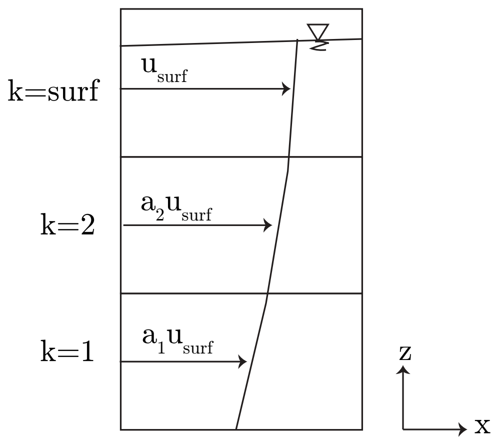


Figure 1.5: The vertical distribution of velocity, where the velocity at each cell is a scalar quantity of the surface velocity

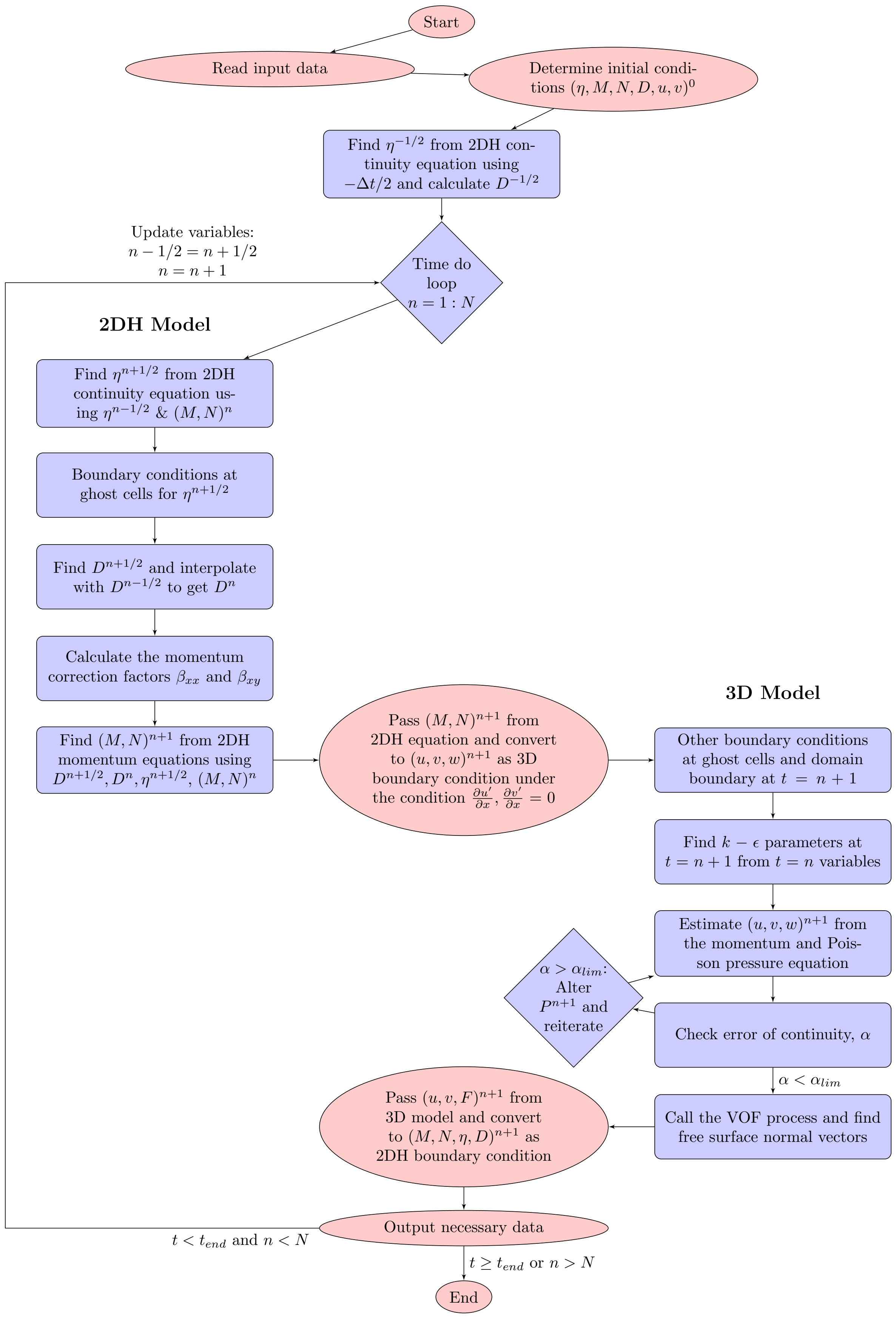
- As has already been mentioned we can input the velocities into the 3D model as described above without even knowing η^{n+1}. The important point is that the velocities are appropriate for input to the 3D model based on the 2DH momentum flux. It so happens that the F^n volume fraction values can be assumed at the interface to determine those velocities at $t = n + 1$ and it was proven to work well numerically. Such a boundary condition is adequate to describe the physics and the free surface at $t = n + 1$ thus does not need to provided and is simply calculated directly from the VOF calculation.

- For the 2DH model, the momentum flux and water depths at $t = n$ from the 3D domain is required for calculation of the momentum fluxes at the model interface. These are provided by the 3D model so we can directly sum the velocities from the 3D model to get the momentum flux as well as D^n which is used in the pressure gradient calculation of the 2DH model.

- Finally, the 2DH model also requires $\eta^{n+1/2}$ within the 3D domain which the 3D calculation does not provide. We work around this by firstly obtaining η^n as determined directly from the VOF calculation in the 3D domain. From this, $\eta^{n+1/2}$ is obtained by using the 2DH continuity equation utilising the momentum fluxes at $t = n$ from the 3D domain while simply substituting $\Delta t/2$ for Δt. The water depths $D^{n+1/2}$ at the next half-time step for use in the nonlinear advection and friction terms are thus also obtained.

The final point that needs to be mentioned concerns the $k - \epsilon$ model boundary conditions. By definition the 2DH model is inviscid and irrotational except for ad-hoc bed stress friction terms added into the equations. As such, it cannot model turbulence but may approximate energy loss due to roughness from the bed stress term. In any case, the model provides no information on turbulence boundary conditions and nor should it. Because of the assumptions of the 2DH model it is up to the user of the Hybrid 2DH-3D model to assume a proper position of the 2DH-3D interface *a priori* where turbulence effects are negligible, perhaps one of the main downsides to the model [7]. Thus, assuming such a position is correctly determined, the boundary conditions for the $k - \epsilon$ model are simply set to zero horizontal gradient.

Finally, Fig 1.6 shows the calculation procedure for the Hybrid 2DH-3D in the form of a flowchart.



Start
Read input data
Determine initial conditions $(\eta, M, N, D, u, v)^0$
Find $\eta^{-1/2}$ from 2DH continuity equation using $-\Delta t/2$ and calculate $D^{-1/2}$
Update variables: $n-1/2 = n+1/2$ $n = n+1$
Time do loop $n = 1 : N$
2DH Model
Find $\eta^{n+1/2}$ from 2DH continuity equation using $\eta^{n-1/2}$ & $(M,N)^n$
Boundary conditions at ghost cells for $\eta^{n+1/2}$
Find $D^{n+1/2}$ and interpolate with $D^{n-1/2}$ to get D^n
Calculate the momentum correction factors β_{xx} and β_{xy}
Find $(M,N)^{n+1}$ from 2DH momentum equations using $D^{n+1/2}, D^n, \eta^{n+1/2}, (M,N)^n$
Pass $(M,N)^{n+1}$ from 2DH equation and convert to $(u,v,w)^{n+1}$ as 3D boundary condition under the condition $\frac{\partial u'}{\partial x}, \frac{\partial v'}{\partial x} = 0$
3D Model
Other boundary conditions at ghost cells and domain boundary at $t = n+1$
Find $k-\epsilon$ parameters at $t = n+1$ from $t = n$ variables
Estimate $(u,v,w)^{n+1}$ from the momentum and Poisson pressure equation
$\alpha > \alpha_{lim}$: Alter P^{n+1} and reiterate
Check error of continuity, α
$\alpha < \alpha_{lim}$
Call the VOF process and find free surface normal vectors
Pass $(u,v,F)^{n+1}$ from 3D model and convert to $(M,N,\eta,D)^{n+1}$ as 2DH boundary condition
Output necessary data
$t < t_{end}$ and $n < N$
$t \geq t_{end}$ or $n > N$
End

Figure 1.6: Flowchart illustrating the calculation procedure of the Hybrid 2DH-3D model

Chapter 2

Using the 2DH Model

2.1 Overview

This chapter describes all you need to know to go ahead and use the 2DH model. Fair to say that before tackling the Hybrid 2DH-3D model you should have some experience at using the 2DH model on its own and also have read this chapter. It would further obviously be preferable to have used the 3D model on its own that is presented in chapter 3. In comparison to the 3D model the 2DH is quite simple to use and you can have results to match in pretty quick time. However, in all three models the procedure for setting up the calculation is fairly similar and once you have got the hang of the the 2DH model it is trivial to set up most of the inputs for the 3D and Hybrid 2DH-3D models at least for the same problem. Probably the most important difference is the creation of the computational mesh and land form data that can be cumbersome at times for the 3D model. However, I am working to try and automate that process as much as possible. In any case mesh data and land form data are also needed for the 2DH model and that process is described in the first section. Following this, the preparation of the wave inflow data file, the output data file and the input text file is outlined. Actually, preparation of these files are almost identical for all the models so the process is described only in great detail in this chapter. In the remaining chapters for the 3D and Hybrid 2DH-3D models I skip the detailed discussion and just note the options available that exist for each respective model.

2.2 Mesh and Land Form Data Generation

For 2DH models to describe the bottom topography, we can only assume a single discrete value of the sea bed at each calculation cell. This is unlike the favor method for the 3D model which we can model as a full planar shape. Thus, technically our usage of a “polygon” is unnecessary to create the mesh data. However, I have worked on the solely on the project entitled “make_chikei_with_polygon” to develop the mesh generation and land form data for 2DH. There are a couple of preprocessor options to do a couple of different things. Furthermore, the program entitled “add_teibou” will add a wall such as a breakwater or seawall in the positions desired. On top of this it also inserts Manning's n friction coefficients for each computational cell if such data exists.

2.2.1 "make_chikei_with_polygon"

Inside "make_chikei_with_polygon" there should be the source files: cal_fluid_area, set_grid, set_chikei, Object_mod, chikei_mod and make_chikei_polygon. It is the non-module files that I have modified quite a lot. Plus you should use the library Lib_x64v12.lib.

Building the program

The main thing here is to consider the preprocessors necessary for building the project. Once the desired preprocessors are entered you should be able to go ahead and build the program. As normal, for debugging ensure that the preprocessor BUGWIN is set. The other thing to do is to type **TWOD** to indicate the 2DH mesh will be created, this is a necessity!

Creating input text file

Once the desired program has been properly "built" we can go ahead and create the input text file for creating a mesh and the ground data to be output as four files: .t0.000 (nf, nf and F data) .t0.000.fdata (ax, ay and fb data) .t0.000.xyz (x, y and z data) and t0.000.3d.godata (3D visualization). The text file we use may be used to create a full 3D mesh as well as a 2D mesh using the same basic structure, and must look like this:

1	:The number of polygons we use to create the data. Usually one
../INIT/polygonname.odata	:This is the name of the polygon. If more than two polygon data are used, write the other names underneath each on a new line. In the case where we do not create the data from a polygon, write any dummy name you like with odata at the end and the program will use this to name the ouput files **or** write the name of the .dat file (with odata as the extension instead) that contains the topography data to skip the need for creating a polygon
3 3	:is, js (usually 3 except in hybrid calc)
3 1 1	:Number of dx, dy and dz values. Must be the same number as written in the "Values for dx (or dy or dz)" line
0d0 -10000d0 20d0 100d0	:Start and end x values. Must be equal to "Number of dx values + 1". Defines the range of a certain value of dx. Insert -10000d0 in between two start and end values to make the dx value gradually change between the two values of dx defined at the start and end. Inserting -10000d0 at first or last start and end value will give the min or max value respectively from the polygon or .dat data
0d0 1d0	:Start and end y values. Must be equal to "Number of dy values + 1". (same as above)
-2d0 0.5d0	:Start and end z values. Must be equal to "Number of dz values + 1". (same as above)
1d0 -10000d0 0.2d0	:Values for dx. Defines the size of the dx in the range defined above. We also need to insert -10000d0 here to get the gradual change between the dx values defined at the start and end
1d0	:Values for dy. (same as above)
-10000d0	:Values for dz. Set to -10000d0 for 2D
0.0d0	:Initial water level (please set as 0.0)
2, 2, 2, 2	:North, South, East, West boundary openness. Set all to ($nfb =$) 2 for 2D
1	:Number of places to adjust water presence (set 0 if none and omit line below)
-0.125d0 75d0 80d0 0d0 1d0	:hadj, Xs,Xe,Ys,Ye (z value above where to remove water and the range of area to adjust water presence)
3	:Number of slopes (set 0 if none and omit lines below)
20d0 40d0 80d0	:Start of slopes
20d0 40d0 0d0	:β value of the slopes (1:β). Set to zero for a flat plateau

By following this template and reading the different options noted in the descriptions it is possible to create a variety of meshes with different dx and dy values at different places as well as gradual change of these values between specified regions. The algorithm used for the gradual change in cell size is described in the next subsection. Firstly, the program would look for the polygon named “polygonname.odata”. If it does not exist, it would look for “polygonname.dat” where you might have all the information of the bottom topography data. However, in this case where we have changing dx and/or dy values and the topograpy data does not correspond to these cell sizes we have a problem. Here, you would be advised to either only use the dx and dy values corresponding to the resolution of your data (generally recommended for tsunami problems where we are better off using nested grids to join data of high resolution with low resolution) or you should make a polygon and the program will automatically interpolate this data for any grid size (in theory).

In terms of the mesh generated when running this program we note that it would generate a 1DH mesh since the difference between the start and end y values is equal to size of dy. These numbers hence are arbitrary but the use of 1 m is recommended. For dz we should define the start value as negative (lowest sea bed elevation) and the end value as positive (some arbitrary cutoff point) so that the initial water elevation is zero. This is a requirement of the 2DH program. By setting dz = -10000d0, dz will be redefined as the difference between the start and end z values by the program. For the x direction in this case we note that the domain would stretch from 0 to 100 m. At 0 m (the first cell), dx = 1 m. Between 20 & 100 m dx = 0.2 m. Between 0 & 20 m, the cell size would gradually change from dx = 1 m to dx = 0.2 m following a power law described in the next subsection. We should note here though that in general we use a constant cell size for the 2DH program where instead we might employ nesting techniques. This gradual change in cell size should prove more useful for the 3D calculation.

There are also two options at the end of the file. We can omit these options by typing zero on the line for the “number of places to adjust water presence” and the “number of slopes”. We should also omit the lines below where we have set zero corresponding to the information required for each option.

The first option is an adjustment of the presence of water. This option was created for Kokuryo because for his calculation in Osaka, the ground level in the city was below sea level. This meant that initially in the calculation water would exist in the town, obviously unrealistic. But we could not simply get rid of the water in this region when rivers also exist. This option is designed to fix this problem by getting rid of the water in the town but keeping the water in the river. It works by specifying the region of the calculation that we want to remove the unwanted water. On top of this we choose a cutoff elevation, say -2m that means that if the ground level is above -2m we would have no water, but in the case where the elevation was below -2 m (i.e. hopefully in a river), we would preserve the water as normal. These values are specified in the input text file. In this example we choose one region to remove water above a certain ground level. The corresponding line will be read and in the region between x = 75 & 80 and y = 0 & 1, if the bed level is greater than z = -0.125 then the water will be removed by the program here. However the water level will be equal to zero elsewhere.

The second option is to easily make slopes. It was created because I often worked on a case concerning runup of a simple slope. Since it is tedious and unnecessary to create a polygon file etc for each slope condition we can define it in the input text file. A slope will be created based on our specified slope (1:β), and the x value from where to start the slope. We can make more than one slope so we can create composite beaches too. Typically we use this option in a 1DH analysis or at least if the slope is constant along the y-direction. In this example we would obtain three slopes, the toe of each begins at x = 20, 40 and 80 respectively. The slopes would be 1:20, 1:40 and horizontal respectively. I recommend using this option to also define a domain with a totally flat constant depth bottom that we may use sometimes for testing. This can be achieved by setting the start of the slope outside the domain or the β value equal to zero.

Exercise:

1. Run this file first setting the number of places to ajust water presence to zero and omitting the line below it.
2. Once we are convinced that indeed the designed slopes were created try running it with the water adjusting option included and see if the water is indeed removed or not.
3. Design your own set of slopes etc. Try changing the cell sizes.
4. Create a mesh from the provided dat file: "kamaishi_50mNPN.dat" (the cell sizes (resolution) are 50 m $\times$ 50 m and the size of the domain is 12 km $\times$ 10 km. The minimum elevation is equal to -135.35 m. Set any arbitrary maximum elevation as you like. The NPN in the name refers to the orientation of how we read the data in the x, y, z directions respectively. N indicates to read the data using a "negative" loop and P for a "positive loop" for the x, y directions. For the z direction, in this case the elevations are the *negative* of what we would like (above sea level is negative) so the program will automatically read the "N" and flip the data as required). Try changing the NPN to another combination and see the effect.
5. Create a mesh based on the provided polygon: "Kamaishi_50m.odata" (the polygon was created from the same data used above in exercise 4). Use -10000d0 to define the domain limits. See if we can make the topography using cell sizes smaller than the 50 m resolution of the data.

Gradual change of cell size algorithm

Here we describe the algorithm that is used to gradually change the cell size according to a power law. As noted in the above section we define the start and end positions where the cell size is gradually changed between two defined values. Fig. 2.1 shows a sketch of an example where we have defined the first cell size Δx_0 in the section starting from x_0, as well as Δx_n in the *next* section starting from x_n. Assuming a power law relationship the two equations we obtain that describe this setup are as follows:

$$\Delta x_0 \alpha^n = \Delta x_n \tag{2.1}$$

$$\sum_{i=1}^{n} \Delta x_0 \alpha^{i-1} \equiv \Delta x_0 \frac{1-\alpha^n}{1-\alpha} = x_n - x_0 \tag{2.2}$$

where, n is the number of cells between x_0 and x_n, and α is the ratio of Δx between two adjacent cells in the x-direction. $\alpha < 0$ in the case sketched in Fig. 2.1 but we may also adopt the reverse of this case so that $\alpha > 0$.

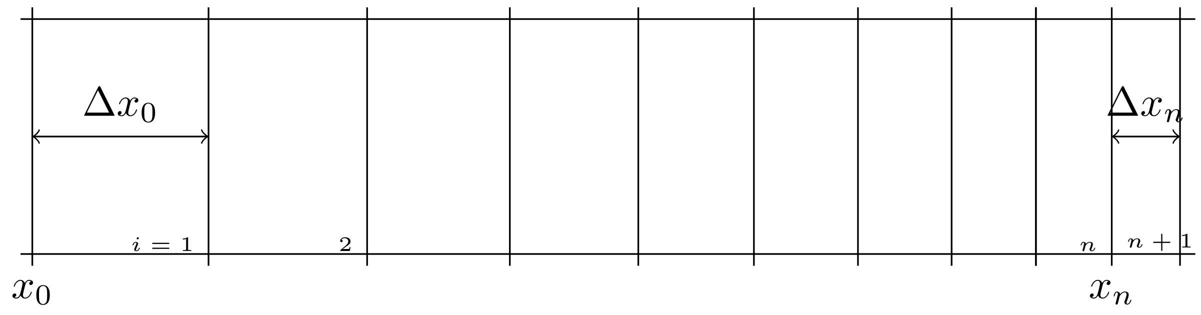


Figure 2.1: Sketch of the graduation in cell size between two prescribed locations

We can solve Eqns. 2.1 & 2.2 simultaneously to obtain the following equations for α and n respectively:

$$\alpha = 1 + \frac{\Delta x_n - \Delta x_0}{x_n - x_0} \tag{2.3}$$

$$n = \frac{\log \Delta x_n - \log \Delta x_0}{\log \alpha} \tag{2.4}$$

The number of cells, n must be an integer and if this requirement is met we have solved the problem. However Eqn. 2.4 does not guarantee that n will be calculated as an integer. In such a case we round n down to the nearest integer and re-calculate α to satisfy Eqn. 2.2 using the found integer, n. To solve this equation we use Newton Raphson iterations as described below:

$$\alpha_{i+1} = \alpha_i - \frac{(1-\alpha_i)(\Delta x_0(1-\alpha_i^n) + (1-\alpha_i)(x_0 - x_n)}{\Delta x_0(1-\alpha_i^n - n\alpha_i^{n-1}(1-\alpha_i))} \tag{2.5}$$

It is found that we quickly arrive at the new solution of α to within some desired accuracy in just a couple of iterations because our initial guess from Eqn. 2.3 is always close to the final solution. Furthermore, the algorithm gaurantees that $\Delta x_{i+1} < \Delta x_i$ is always satisfied so we judge it to be effective. The example in the previous section should show that this is indeed the case if you observe carefully the result. Also, x_n will be set to exactly what was prescribed and not what was calculated to avoid numerical errors. Lastly, note that we could also use the algorithm in reverse going from a small value through to a larger value of Δx.

2.2.2 “add_teibou”

Inside “add_teibou” there should be the source files: add_teibou_main, and chikei_mod plus you should use the library Lib_x64v12.lib.

Building the program

It should be trivial to build the program. No preprocessors are required except BUGWIN in debug mode as usual. It is designed to work for mainly the 2DH program but we can add walls by hand for the 3D mesh as well.

Creating input text file

Once the desired program has been properly "built" we can go ahead and create the input text file for adding in breakwaters, seawalls or rivers banks plus Manning's n friction data if available. "add_teibou' works by adjusting the .t0.000.fdata (ax, ay and fb data) file and it will output the adjusted file as tb.t0.000.fdata. The structure of the text file must look like this:

../INIT/name.t0.000	:Name of the mesh data already created. It will read all the data based on this name
../INIT/teibou.ext	:Name of the teibou data with any extension as desired. Type NON when not using such data and the program will skip this step for us
../INIT/mannings.ext	:Name of the Manning's n friction data with any extension as desired. Type NON when not using such data
NEG POS POS	:Direction/orientation when reading the data in the x, y, z directions respectively
1	:Number of teibous that we want to add by hand (defined below). Set 0 if none and omit line below.
-19.86d0, 6850.0d0, 6850.0d0, 6950.0d0, 7200.0d0	:Gives the crest height and Xs, Xe, Ys, Ye bounds of the 1st teibou. Write other teibou data underneath if more than 1 is set
1	:Number of bw values to change in the teibou.ext file. Set 0 if none and omit line below.
5.10 5.14d0	:Gives the crest height of the teibou defined in the teibou.ext file and the new crest height desired respectively

By following this template and reading the different options noted in the descriptions it is possible to create a number of different teibous and alter existing data as desired (for example if we want to simulate with a partially destroyed breakwater etc).

In this example the program reads name.t0.000 and alters the t0.000.fdata. For the teibou data it works by changing the ax and ay aperture ratios. The 2DH program recognises this and considers these ratios to be indicative of the crest height of a seawall, breakwater or river bank. "add_teibou' will read teibou.ext and the information set by hand to adjust these ratios. Furthermore we have to read the teibou data in the same way we created the mesh data in section 2.2.1 to get the orientation correct (e.g. NEG POS POS indicating negative or positive in x, y, z directions). For the teibou set by hand, it will create a breakwater of crest height defined in this example as -19.86. The breakwater will be defined between the bounds set thereafter (similar to the ADJ preprocessor line in section 2.2.1). Finally, Manning's friction data in mannings.ext will be read in the same way as teibou.ext where the values of az are altered to indicate the friction n values because az is not used in the 2DH program otherwise.

Exercise:

1. Use the result of our example of a slope with the SLOPE and ADJ preprocessors you should have created in exercise 2 in section 2.2.1 as name.t0.000. Here some of the water was removed at the top of the slope but if we run the 2DH program flow will occur here unless we define some kind of wall. Define a seawall at the water edge equal to the water level plus 0.1 m.

2. Use the result of our example you should have created in exercises 4 and 5 in section 2.2.1 as name.t0.000. Here, we have some teibou data called "kamaishi_50m.bdh" and Manning's friction data called "kamaishi_50m.fm". Run the template shown. By looking at the values of ax, ay, az observe that the breakwater and Manning's friction data was correctly added into the tb.t0.000.fdata.

3. Completely get rid of the Kamaishi Bay breakwater that is defined by the crest height of 5.1 m in the kamaishi_50m.bdh data

4. Partially omit some areas of the Kamaishi Bay breakwater to simulate partial destruction of the breakwater. Note, the "teibou by hand" values will overwrite the teibou.ext data.

2.3 Wave Inflow Data File

In this section I start by describing how the wave inflow data is read and input into the model through the "read_wave" subroutine. Then I will go over the format of the data file that must be used to input the data.

2.3.1 "read_wave" subroutine

"read_wave" is located the BSC.F90 file under the BNDCND module. BNDCND stands for boundary condition. As you might, expect the job of this subroutine is to read the data that corresponds to the wave input. There are a number of possibilities to choose from when inserting the wave data as listed below:

1. Solitary Wave (SW): Input a solitary wave into the domain through one of the boundaries by just specifying the wave amplitude, the still water depth and the input angle

2. Regular Wave (RW): Input a regular sinusoidal wave into the domain through one of the boundaries by just specifying the wave amplitude, the wave period and the input angle

3. Arbitrary Wave (AW): Input an arbitrary wave shape into the domain through one of the boundaries by specifying the time series of the free surface, the input angle and the depth of the water that the wave time series was measured at (if different from the input location)
4. Initial Condition (IC): Create an initial free surface profile and an optional specification of initial velocities by either specifying the free surface heights at each computational cell by inserting them in the data file or through some user specified equation (for example a solitary wave profile with initial velocities - this is already available for use).

The two letters in brackets refer to code that we use to input this condition in the data file. Also, we can actually create any number of simultaneous inputs from different locations or different types as desired although of course typically only one is used. Moreover, we can specify an exact location for the wave input, e.g. through just a small segment of a boundary or even within the calculation domain. Note that if we specify the input location within the calculation domain we cannot consider the changes in free surface at the wave source due to the calculated fluxes, the program just overwrites the calculated free surface with our wave input. To consider such changes we would have to use an internal source method that is not considered here.

2.3.2 Data file format

First example: Solitary and Regular Waves

Fig. 2.2 shows the first example of an input file for regular and solitary waves. In this case the input file is just three lines in length. The first line is just one integer and it tells the program how many different inputs there are. Although I have demonstrated 2 inputs here, usually this is just 1. The second line will tell the program to input a regular wave (RW) of amplitude, $a = 0.5$ m, with period, $T = 30$ s with no angle in through the south boundary. The third line will tell the program to input a solitary wave (SW) of amplitude, $a = 0.05$ m, with still water depth, $h =$ 1 m with no angle in through the west boundary. A little explanation is necessary to describe how we define the input location:

Input location definition

To define the input location we should specify the region by defining the i, j cell number numbers at the edge of the input region, i.e. istart, iend, jstart, jend in that order. In this way we can input the wave anywhere in the domain in the form of a rectangular source shape as illustrated in Fig. 2.2 under "general input region specification" heading. Typically though we input the wave through the boundary. To specify this, we should set either istart and iend to both the same value or jstart and jend to the same value. We can choose from 2 or -2. Furthermore to specify the wave over the whole boundary we should set the other two values (jstart and jend or istart and iend respectively) to 0. Specify some non-zero values to input a wave over a partial segment of the boundary. The combinations are listed below:

1. istart and iend = 2, jstart and jend = 0: Input wave through the entire west boundary

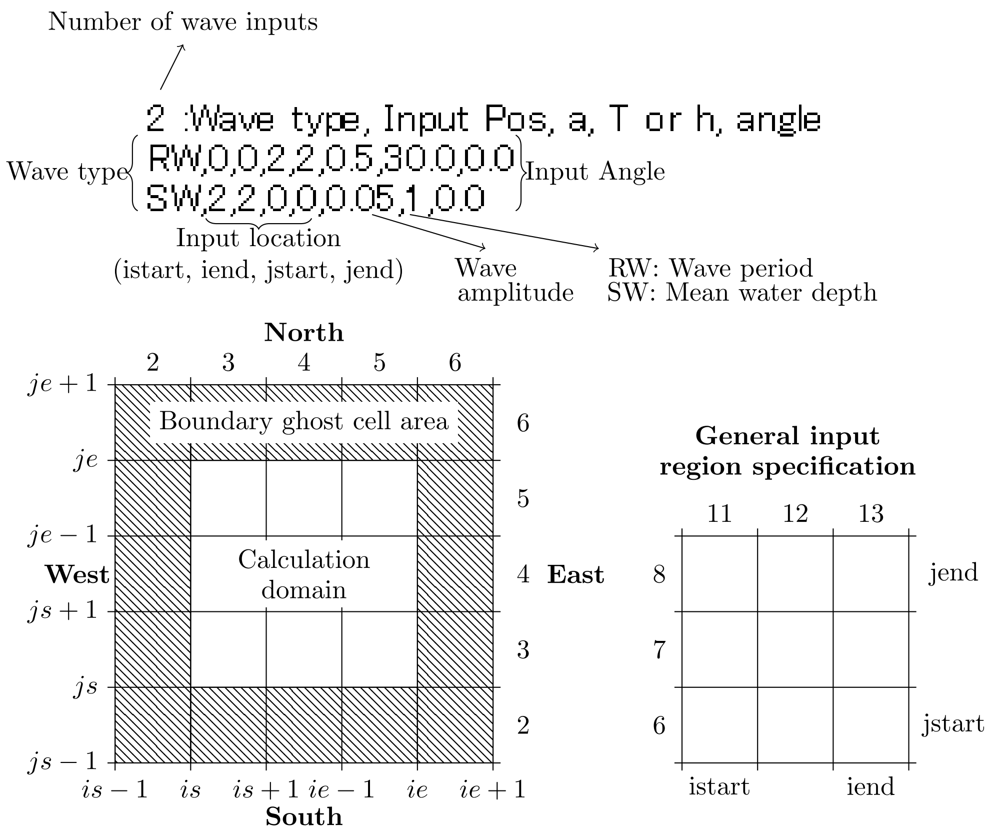


Figure 2.2: Example 1: Input of solitary and regular waves through the boundaries

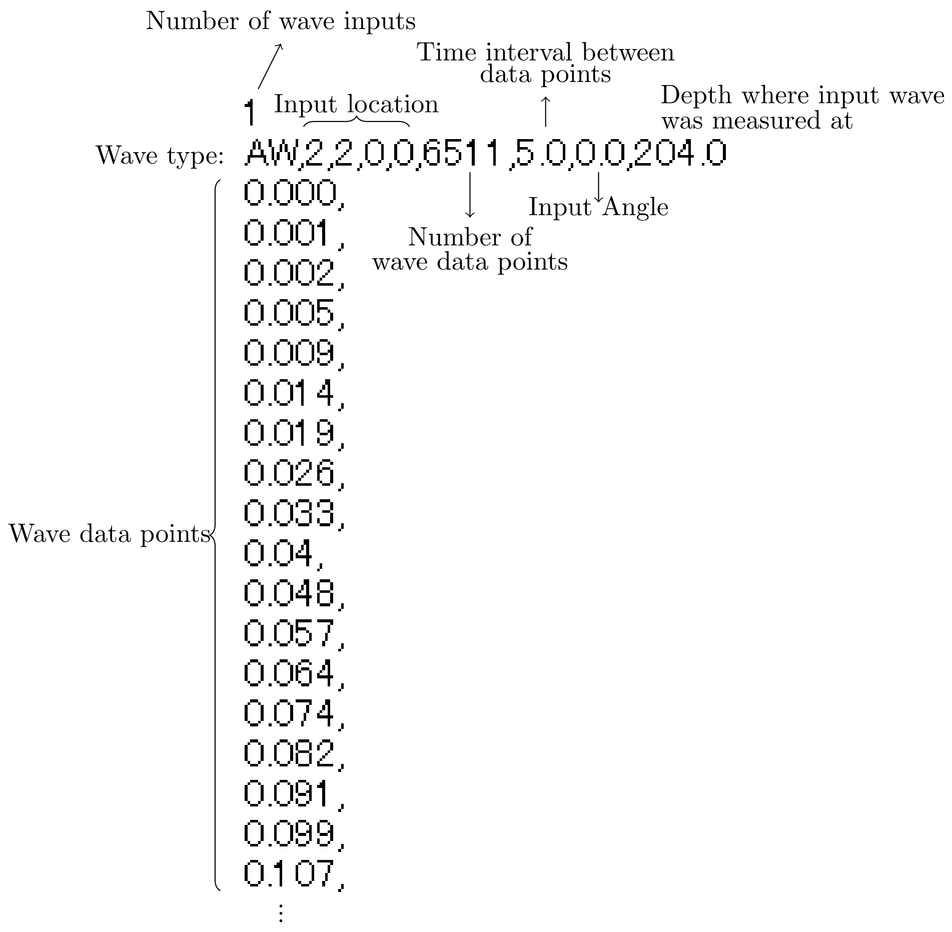
2. istart and iend = 2, jstart and jend $\neq$ 0: Input wave through a part of the west boundary
3. istart and iend = -2, jstart and jend = 0: Input wave through the entire east boundary
4. istart and iend = -2, jstart and jend $\neq$ 0: Input wave through a part of the east boundary
5. jstart and jend = 2, istart and iend = 0: Input wave through the entire south boundary
6. jstart and jend = 2, istart and iend $\neq$ 0: Input wave through a part of the south boundary
7. jstart and jend = -2, istart and iend = 0: Input wave through the entire north boundary
8. jstart and jend = -2, istart and iend $\neq$ 0: Input wave through a part of the north boundary

Second example: Arbitrary Wave

Fig. 2.3 shows the second example of an input file for an arbitrary wave. Here the wave heights for the time series are read in the input file below the second line. The first line says that there is only one input wave. The second line says that we want to put in an arbitrary wave through the west boundary at 0 angle, where we have 6511 data points to read in the time series with 5 s intervals. Furthermore, we specify that the time series of the data was measured at a depth of 204 m (by some buoy) so if the input boundary is at a different depth we will adjust the waveform using Green's Law:

$$\frac{\eta_{in}}{\eta_0} = \left(\frac{h_{in}}{h_0}\right)^{-1/4} \tag{2.6}$$

where h_0 and η_0 are the still water depth and free surface height at the location of origin (i.e. the wave buoy). Here, $h_0 = 204$ m, h_{in} will be the still water depth at the west boundary and finally η_{in} will be the final input through the boundary.



```
1
AW,2,2,0,0,6511,5.0,0.0,204.0
0.000,
0.001,
0.002,
0.005,
0.009,
0.014,
0.019,
0.026,
0.033,
0.04,
0.048,
0.057,
0.064,
0.074,
0.082,
0.091,
0.099,
0.107,
⋮
```

Number of wave inputs
Time interval between data points
Input location
Depth where input wave was measured at
Wave type:
Input Angle
Number of wave data points
Wave data points

Figure 2.3: Example 2: Input of arbitrary wave through the west boundary

2.4 Output Data File

2.5 Input Text File

Bibliography

[1] Yong-Sik Cho, Dae-Hee Sohn, and Seung Oh Lee. Practical modified scheme of linear shallow-water equations for distant propagation of tsunamis. *Ocean Engineering*, 34(11-12):1769–1777, August 2007.

[2] Chiaki Goto, Y Ogawa, Nobuo Shuto, and Fumihiko Imamura. IUGG/IOC Time Project: IOC Manuals and Guides No.35 - Numerical Method of Tsunami Simulation with the Leap-frog Scheme, 1997.

[3] James T. Kirby, Fengyan Shi, Babak Tehranirad, Jeffrey C. Harris, and Stephan T. Grilli. Dispersive tsunami waves in the ocean: Model equations and sensitivity to dispersion and Coriolis effects. *Ocean Modelling*, 62:39–55, 2013.

[4] F. Lø vholt, G. Pedersen, and G. Gisler. Oceanic propagation of a potential tsunami from the La Palma Island. *Journal of Geophysical Research*, 113(C9):C09026, September 2008.

[5] Geir Pedersen and Finn Lø vholt. Documentation of a global Boussinesq solver. Technical report, Dept. of Math, University of Oslo, January 2012.

[6] William Pringle and Nozomu Yoneyama. Development of Hybrid 2D-3D Numerical Analysis and its application to the Inundation of the 2011 off the Pacific Coast of Tohoku Earthquake Tsunami in Kamaishi Bay. In *35th IAHR World Congress*, number 2008, pages 1–11, Chengdu, 2013.

[7] K. I. Sitanggang and P. J. Lynett. Multi-scale simulation with a hybrid Boussinesq-RANS hydrodynamic model. *International Journal for Numerical Methods in Fluids*, page 34, 2009.

[8] Sangyoung Son, Patrick J. Lynett, and Dae-Hong Kim. Nested and multi-physics modeling of tsunami evolution from generation to inundation. *Ocean Modelling*, 38(1-2):96–113, January 2011.

[9] Nozomu Yoneyama, Masafumi Matsuyama, and Hiroyoshi Tanaka. Numerical analysis for locally high runup of the 1993 Hokkaido Nansei-oki tsunami. *Journal of Hydraulic, Coastal and Environmental Engineering, JSCE*, 705(II-59):139–150 (in Japanese), 2002.