# **N**on-**H**ydrostatic **W**ave **M**odel **NHWAVE**
# Documentation and User's Manual
# (**Version 2.0**)

Gangfeng Ma
Department of Civil and Environmental Engineering
Old Dominion University, Norfolk, VA 23529

James T. Kirby and Fengyan Shi
Center for Applied Coastal Research
University of Delaware, Newark, DE 19716

November, 2014

Center for Applied Coastal Research
University of Delaware
Research Report NO. CACR-14-11

**Abstract**

This report documents a three-dimensional shock-capturing Non-Hydrostatic WAVE model NHWAVE of Ma et al. (2012), which solves the incompressible Navier-Stokes equations in terrain and surface-following sigma coordinates. The model predicts instantaneous surface elevation and 3D flow field, and is capable of resolving coastal wave processes (shoaling, refraction, diffraction, breaking etc.) as well as tsunami wave generation by landslide. The documentation provides a detailed description of governing equations, turbulence models and numerical schemes. A user's manual is provided and gives instructions on how to set up the simulations. Finally, several examples are documented.

# Contents

# List of Figures

# 1 Introduction

Wave propagation from deep water to coastal region is subject to wave refraction, diffraction, shoaling and breaking. Accurate prediction of these phenomena is crucial to studying nearshore hydrodynamics and solute transport in the coastal area. Boussinesq-type wave models with improved nonlinearity and dispersion characteristics provide an efficient and well-tested tool for the simulation of wave propagation, especially in shallow water regions (Madsen and Sørensen, 1992; Nwogu, 1993; Wei et al., 1995). Means for extending these models to higher order in dispersion have been developed (see Gobbi et al (2000), Lynett and Liu (2002) and Agnon et al (1999), for example), and more recently, extensions to the model formulation to account for turbulent structure of the flow field and the resulting effects on depth-averaged solute or contaminant transport have been developed (Kim et al, 2009; Kim and Lynett, 2011). All of these extensions lead to a great deal of complexity in the resulting model equations.

An alternative approach is to solve the Navier-Stokes equations directly with proper free surface tracking techniques, such as the marker-and-cell (MAC) method (Harlow and Welch, 1965), the volume-of-fluid (VOF) method (Hirt and Nichols, 1981) and the level-set method (Osher and Sethian, 1988). These approaches have wide applications on the simulations of wave shoaling and breaking in the surf zone; see, for example, Lin and Liu (1998a,b), Watanabe et al. (2005), Christensen (2006), Shi et al. (2010) and Ma et al. (2011). The main drawbacks of these types of models are: (1) They are computationally expensive, making applications to large-scale domains infeasible; (2) The free surface normally crosses the computational cell arbitrarily, which brings the difficulty of applying the pressure boundary condition precisely on the free surface and may eventually affect the accuracy of velocity computation (Lin and Li, 2002); and (3) The grid resolution in the surf zone and swash zone, where the water depth is relatively shallow, is usually poor due to the use of Cartesian grid system on most of applications.

A direct simplification of the above-mentioned approach is to assume that the free surface elevation is a single value function of the horizontal coordinates. By doing so, the free surface is always located at the upper computational boundary and can be determined by applying the free surface boundary conditions. It is computationally more efficient with the lack of free surface tracking. The pressure boundary condition at the free surface can be accurately prescribed with some proper treatments. This simplification leads to a new set of non-hydrostatic models, which are not only suitable for modeling short wave propagation but also for the simulation of turbulence and solute transport in the surf zone. To solve the non-hydrostatic equations, the pressure is decomposed into hydrostatic and non-hydrostatic components. The governing equations can be discretized by finite difference method (Casulli and Stelling, 1998; Casulli, 1999; Namin et al., 2001; Casulli and Zanolli, 2002; Lin and Li, 2002; Chen, 2003; Stelling and Zijlema, 2003; Zijlema and Stelling, 2005; Yuan and Wu, 2004; Lee et al., 2006; Young et al., 2007, 2009, 2010; Wu et al., 2010), finite element method (Walters, 2005) and finite volume method (Bradford, 2005; Fringer et al., 2006; Ai and Jin, 2010; Lai et al., 2010). A major concern addressed in recent developments of non-hydrostatic models is the accurate prediction of wave dispersion characteristics with relatively few vertical grid points. It has been recognized that $10 \sim 20$ vertical layers are nor-

mally required to describe wave dispersion up to an acceptable level with some simple treatments of pressure boundary conditions at the top layer, for example, Casulli and Stelling (1998), Casulli (1999), Casulli and Zanolli (2002), Li and Fleming (2001), Namin et al. (2001), Lin and Li (2002), Chen (2003). To address this issue, Stelling and Zijlema (2003) proposed the Keller-box method to replace the staggered grid in the vertical direction, which enables the pressure to be located at the cell faces rather than the cell centers. The pressure boundary condition at the free surface can be exactly assigned to zero without any approximation. Yuan and Wu (2004a,b) proposed an integral method to remove the top-layer hydrostatic assumption using a staggered grid framework. Young and Wu (2010) used the Boussinesq-type-like equations with the reference velocity to provide an analytical-based non-hydrostatic pressure distribution at the top layer. All of these methods significantly reduce the errors in dynamic pressure estimation and allow for use of a very small number of vertical layers for accurate simulation of dispersive waves.

It is non-trivial to apply non-hydrostatic models to the simulation of breaking waves in the surf zone and wave run-up in the swash region, because the numerical scheme involved must treat shock propagation adequately in order to model broken waves (Zijlema and Stelling, 2008). Shock-capturing schemes based on Godunov-type approach, which can deal with discontinuous flow, are well-suited for breaking wave simulations. These schemes are able to track actual location of wave breaking without requiring any criterion that tells the model when and where the wave breaking happens. An application of this approach to simulation of breaking waves in the surf zone was given by Bradford (2011). It was showed that the non-hydrostatic model with Godunov-type scheme can predict wave height distribution, turbulence and undertow under breaking waves at least as accurate as the VOF model. However, eight or more vertical layers are needed in his model to accurately predict the surface elevation around the outer surf zone as well as velocity profiles within the surf zone.

Recently, we have developed a new non-hydrostatic wave model (called NHWAVE) (Ma et al., 2012), which solves the incompressible Navier-Stokes equations in terrain and surface-following $\sigma$ coordinates. Bottom movement is included in order to simulate tsunami generation by three-dimensional underwater landslides. To apply Godunov-type scheme, the velocities are defined at cell centers. The dynamic pressure is defined at vertically-facing cell faces as in the Keller-box method, allowing the pressure boundary condition at the free surface to be precisely imposed. The hydrostatic equations are solved by a well-balanced finite volume method. The fluxes at cell faces are estimated by HLL Riemann approximation. To obtain second-order temporal accuracy, the nonlinear Strong Stability-Preserving (SSP) Runge-Kutta scheme (Gottlieb et al., 2001) is adopted for adaptive time stepping. The model is fully parallelized using Message Passing Interface (MPI) with non-blocking communication. The poisson equation is solved by the high performance pre-conditioner HYPRE software library (http://acts.nersc.gov/hypre/). A sediment transport module (Ma et al., 2014c) and a vegetation module (Ma et al., 2013a) have also been incorporated into NHWAVE. The model has been extensively used to study landslide-induced tsunami waves (Ma et al., 2013b), infragravity wave processes in coral reef systems (Ma et al., 2014a), wave interactions with porous structures (Ma et al., 2014b), wave-vegetation-sediment interactions (Ma et al., 2014d) as well as turbulence and sediment transport in a tidal inlet (Keshtpoor et al., 2014).

This documentation provides a detailed description of governing equations, turbulence models and numerical schemes. A user's manual is also provided and gives instructions on how to set up the simulations. Finally, several examples are documented.

# 2 Theoretical Background

## 2.1 Model Equations

The incompressible Navier-Stokes equations in Cartesian coordinates $(x_1^*, x_2^*, x_3^*)$, where $x_1^* = x^*$, $x_2^* = y^*$ and $x_3^* = z^*$ and time $t^*$ are given by

$$\frac{\partial u_i}{\partial x_i^*} = 0 \tag{1}$$

$$\frac{\partial u_i}{\partial t^*} + u_j \frac{\partial u_i}{\partial x_j^*} = -\frac{1}{\rho}\frac{\partial p}{\partial x_i^*} + g_i + \frac{\partial \tau_{ij}}{\partial x_j^*} \tag{2}$$

where $(i, j) = 1, 2, 3$, $u_i$ is velocity component in the $x_i^*$ direction, $p$ is total pressure, $\rho$ is water density, $g_i = -g\delta_{i3}$ is the gravitational body force and $\tau_{ij} = \nu_t(\partial u_i/\partial x_j^* + \partial u_j/\partial x_i^*)$ is turbulent stress with $\nu_t$ the turbulent kinematic viscosity.

In order to accurately represent bottom and surface geometry, a $\sigma$-coordinate developed by Phillips (1957) is adopted in this study.

$$t = t^* \quad x = x^* \quad y = y^* \quad \sigma = \frac{z^* + h}{D} \tag{3}$$

where $D(x, y, t) = h(x, y, t) + \eta(x, y, t)$, $h$ is water depth, $\eta$ is surface elevation. This coordinate transformation basically maps the varying vertical coordinate in the physical domain to a uniform transformed space where $\sigma$ spans from 0 to 1 (Lin and Li, 2002). Using the principle of chain differentiation, the partial differentiation of a variable $f = f(x^*, y^*, z^*, t^*)$ in the physical domain is transformed as follows.

$$\begin{aligned}
\frac{\partial f}{\partial t^*} &= \frac{\partial f}{\partial t} + \frac{\partial f}{\partial \sigma}\frac{\partial \sigma}{\partial t^*} \\
\frac{\partial f}{\partial x^*} &= \frac{\partial f}{\partial x} + \frac{\partial f}{\partial \sigma}\frac{\partial \sigma}{\partial x^*} \\
\frac{\partial f}{\partial y^*} &= \frac{\partial f}{\partial y} + \frac{\partial f}{\partial \sigma}\frac{\partial \sigma}{\partial y^*} \\
\frac{\partial f}{\partial z^*} &= \frac{\partial f}{\partial \sigma}\frac{\partial \sigma}{\partial z^*}
\end{aligned} \tag{4}$$

Plugging equation (4) into (1) and (2), we obtain the governing equations in the new coordinate $(x, y, \sigma)$ and time t.

$$\frac{\partial D}{\partial t} + \frac{\partial Du}{\partial x} + \frac{\partial Dv}{\partial y} + \frac{\partial \omega}{\partial \sigma} = 0 \tag{5}$$

$$\frac{\partial \mathbf{U}}{\partial t}+\frac{\partial \mathbf{F}}{\partial x}+\frac{\partial \mathbf{G}}{\partial y}+\frac{\partial \mathbf{H}}{\partial \sigma}=\mathbf{S}_h+\mathbf{S}_p+\mathbf{S}_\tau \tag{6}$$

where $\mathbf{U}=(Du,Dv,Dw)^T$. The fluxes are

$$\mathbf{F}=\begin{pmatrix} Duu+\frac{1}{2}gD^2 \\ Duv \\ Duw \end{pmatrix} \quad \mathbf{G}=\begin{pmatrix} Duv \\ Dvv+\frac{1}{2}gD^2 \\ Dvw \end{pmatrix} \quad \mathbf{H}=\begin{pmatrix} u\omega \\ v\omega \\ w\omega \end{pmatrix}$$

The source terms are given by

$$\mathbf{S}_h=\begin{pmatrix} gD\frac{\partial h}{\partial x} \\ gD\frac{\partial h}{\partial y} \\ 0 \end{pmatrix} \quad \mathbf{S}_p=\begin{pmatrix} -\frac{D}{\rho}(\frac{\partial p}{\partial x}+\frac{\partial p}{\partial \sigma}\frac{\partial \sigma}{\partial x^*}) \\ -\frac{D}{\rho}(\frac{\partial p}{\partial y}+\frac{\partial p}{\partial \sigma}\frac{\partial \sigma}{\partial y^*}) \\ -\frac{1}{\rho}\frac{\partial p}{\partial \sigma} \end{pmatrix} \quad \mathbf{S}_\tau=\begin{pmatrix} DS_{\tau_x} \\ DS_{\tau_y} \\ DS_{\tau_z} \end{pmatrix}$$

where the total pressure has been divided into two parts: dynamic pressure $p$ (use $p$ as dynamic pressure hereinafter for simplicity) and hydrostatic pressure $\rho g(\eta-z)$. $\omega$ is the vertical velocity in the $\sigma$ coordinate image domain, given by

$$\omega=D\left(\frac{\partial \sigma}{\partial t^*}+u\frac{\partial \sigma}{\partial x^*}+v\frac{\partial \sigma}{\partial y^*}+w\frac{\partial \sigma}{\partial z^*}\right) \tag{7}$$

with

$$\begin{aligned} \frac{\partial \sigma}{\partial t^*}&=\frac{1}{D}\frac{\partial h}{\partial t}-\frac{\sigma}{D}\frac{\partial D}{\partial t} \\ \frac{\partial \sigma}{\partial x^*}&=\frac{1}{D}\frac{\partial h}{\partial x}-\frac{\sigma}{D}\frac{\partial D}{\partial x} \\ \frac{\partial \sigma}{\partial y^*}&=\frac{1}{D}\frac{\partial h}{\partial y}-\frac{\sigma}{D}\frac{\partial D}{\partial y} \\ \frac{\partial \sigma}{\partial z^*}&=\frac{1}{D} \end{aligned} \tag{8}$$

Turbulent diffusion terms $S_{\tau_x}, S_{\tau_y}, S_{\tau_z}$ are given by

$$\begin{aligned} S_{\tau_x}&=\frac{\partial \tau_{xx}}{\partial x}+\frac{\partial \tau_{xx}}{\partial \sigma}\frac{\partial \sigma}{\partial x^*}+\frac{\partial \tau_{xy}}{\partial y}+\frac{\partial \tau_{xy}}{\partial \sigma}\frac{\partial \sigma}{\partial y^*}+\frac{\partial \tau_{xz}}{\partial \sigma}\frac{\partial \sigma}{\partial z^*} \\ S_{\tau_y}&=\frac{\partial \tau_{yx}}{\partial x}+\frac{\partial \tau_{yx}}{\partial \sigma}\frac{\partial \sigma}{\partial x^*}+\frac{\partial \tau_{yy}}{\partial y}+\frac{\partial \tau_{yy}}{\partial \sigma}\frac{\partial \sigma}{\partial y^*}+\frac{\partial \tau_{yz}}{\partial \sigma}\frac{\partial \sigma}{\partial z^*} \\ S_{\tau_z}&=\frac{\partial \tau_{zx}}{\partial x}+\frac{\partial \tau_{zx}}{\partial \sigma}\frac{\partial \sigma}{\partial x^*}+\frac{\partial \tau_{zy}}{\partial y}+\frac{\partial \tau_{zy}}{\partial \sigma}\frac{\partial \sigma}{\partial y^*}+\frac{\partial \tau_{zz}}{\partial \sigma}\frac{\partial \sigma}{\partial z^*} \end{aligned} \tag{9}$$

and the stresses in the transformed space are calculated as

$$\begin{aligned}
\tau_{xx} &= 2\nu_t(\frac{\partial u}{\partial x}+\frac{\partial u}{\partial \sigma}\frac{\partial \sigma}{\partial x^*}) \quad \tau_{xy}=\tau_{yx}=\nu_t(\frac{\partial u}{\partial y}+\frac{\partial u}{\partial \sigma}\frac{\partial \sigma}{\partial y^*}+\frac{\partial v}{\partial x}+\frac{\partial v}{\partial \sigma}\frac{\partial \sigma}{\partial x^*}) \\
\tau_{yy} &= 2\nu_t(\frac{\partial v}{\partial y}+\frac{\partial v}{\partial \sigma}\frac{\partial \sigma}{\partial y^*}) \quad \tau_{xz}=\tau_{zx}=\nu_t(\frac{\partial u}{\partial \sigma}\frac{\partial \sigma}{\partial z^*}+\frac{\partial w}{\partial x}+\frac{\partial w}{\partial \sigma}\frac{\partial \sigma}{\partial x^*}) \\
\tau_{zz} &= 2\nu_t(\frac{\partial w}{\partial \sigma}\frac{\partial \sigma}{\partial z^*}) \qquad \tau_{yz}=\tau_{zy}=\nu_t(\frac{\partial v}{\partial \sigma}\frac{\partial \sigma}{\partial z^*}+\frac{\partial w}{\partial y}+\frac{\partial w}{\partial \sigma}\frac{\partial \sigma}{\partial y^*})
\end{aligned} \tag{10}$$

The turbulent kinematic viscosity $\nu_t$ can be estimated by the Smagorinsky subgrid model or k-$\epsilon$ turbulence closure, which will be introduced in the next section. The Smagorinsky subgrid model is expressed as

$$\nu_t = (C_s\Delta)^2\sqrt{2S_{ij}S_{ij}} \tag{11}$$

where $C_s$ is the Smagorinsky coefficient, which is taken as 0.1$\sim$0.2, $\Delta$ is the filter width, which is calculated as $\Delta=(\Delta x\Delta y\Delta\sigma D)^{1/3}$, and $S_{ij}=\frac{1}{2}(\frac{\partial u_i}{\partial x_j^*}+\frac{\partial u_j}{\partial x_i^*})$ is the stress tensor.

Integrating equation (5) from $\sigma=0$ to 1 and using the boundary conditions at the bottom and surface for $\omega$, we get the governing equation for free surface movement.

$$\frac{\partial D}{\partial t}+\frac{\partial}{\partial x}(D\int_0^1 ud\sigma)+\frac{\partial}{\partial y}(D\int_0^1 vd\sigma)=0 \tag{12}$$

## 2.2 Turbulence Model

We have implemented a nonlinear $k-\epsilon$ model (Lin and Liu, 1998; Ma et al., 2011) into NHWAVE to simulate turbulent flow. The turbulent eddy viscosity is calculated by

$$\nu_t = C_\mu\frac{k^2}{\epsilon} \tag{13}$$

The $k-\epsilon$ equations in conservative form are given by

$$\frac{\partial Dk}{\partial t}+\nabla\cdot(D\mathbf{u}k)=\nabla\cdot\left[D\left(\nu+\frac{\nu_t}{\sigma_k}\right)\nabla k\right]+D\left(P_s+P_b-\epsilon\right) \tag{14}$$

$$\frac{\partial D\epsilon}{\partial t}+\nabla\cdot(D\mathbf{u}\epsilon)=\nabla\cdot\left[D\left(\nu+\frac{\nu_t}{\sigma_\epsilon}\right)\nabla\epsilon\right]+\frac{\epsilon}{k}D\left(C_{1\epsilon}(P_s+C_{3\epsilon}P_b)-C_{2\epsilon}\epsilon\right) \tag{15}$$

where $\sigma_k,\sigma_\epsilon,C_{1\epsilon},C_{2\epsilon},C_\mu$ are empirical coefficients given by

$$\sigma_k=1.0,\quad \sigma_\epsilon=1.3,\quad C_{1\epsilon}=1.44,\quad C_{2\epsilon}=1.92,\quad C_\mu=0.09 \tag{16}$$

$P_s$ and $P_b$ are shear and buoyancy production, which are described as

$$P_s=-\overline{u_i'u_j'}\frac{\partial u_i}{\partial x_j^*} \tag{17}$$

and

$$P_b = \frac{g}{\rho_0}\frac{\nu_t}{D}\frac{\partial \rho_m}{\partial \sigma} \tag{18}$$

where the Reynolds stress $\overline{u_i' u_j'}$ is calculated by a nonlinear model proposed by Lin and Liu (1998), which is given by

$$\begin{aligned}
\overline{u_i' u_j'} = &- C_d \frac{k^2}{\epsilon}\left(\frac{\partial u_i}{\partial x_j^*} + \frac{\partial u_j}{\partial x_i^*}\right) + \frac{2}{3}k\delta_{ij} \\
&- C_1 \frac{k^3}{\epsilon^2}\left(\frac{\partial u_i}{\partial x_l^*}\frac{\partial u_l}{\partial x_j^*} + \frac{\partial u_j}{\partial x_l^*}\frac{\partial u_l}{\partial x_i^*} - \frac{2}{3}\frac{\partial u_l}{\partial x_k^*}\frac{\partial u_k}{\partial x_l^*}\delta_{ij}\right) \\
&- C_2 \frac{k^3}{\epsilon^2}\left(\frac{\partial u_i}{\partial x_k^*}\frac{\partial u_j}{\partial x_k^*} - \frac{1}{3}\frac{\partial u_l}{\partial x_k^*}\frac{\partial u_l}{\partial x_k^*}\delta_{ij}\right) \\
&- C_3 \frac{k^3}{\epsilon^2}\left(\frac{\partial u_k}{\partial x_i^*}\frac{\partial u_k}{\partial x_j^*} - \frac{1}{3}\frac{\partial u_l}{\partial x_k^*}\frac{\partial u_l}{\partial x_k^*}\delta_{ij}\right)
\end{aligned} \tag{19}$$

where $C_d$, $C_1$, $C_2$ and $C_3$ are empirical coefficients as given by Lin and Liu (1998)

$$\begin{aligned}
C_d &= \frac{2}{3}\left(\frac{1}{7.4 + 2S_{max}}\right), \quad C_1 = \frac{1}{185.2 + 3D_{max}^2} \\
C_2 &= -\frac{1}{58.5 + 2D_{max}^2}, \quad C_3 = \frac{1}{370.4 + 3D_{max}^2}
\end{aligned} \tag{20}$$

where

$$\begin{aligned}
S_{max} &= \frac{k}{\epsilon} max\left\{\left|\frac{\partial u_i}{\partial x_i^*}\right|\right\} \text{ (indices not summed)} \\
D_{max} &= \frac{k}{\epsilon} max\left\{\left|\frac{\partial u_i}{\partial x_j^*}\right|\right\}
\end{aligned} \tag{21}$$

The above coefficients may ensure the non-negativity of turbulent velocities and bounded Reynolds stress. They have been successfully applied to simulate breaking waves on plane beaches (Lin and Liu, 1998).

# 3 Numerical Method

A combined finite-volume and finite-difference scheme with a Godunov-type method was applied to discretize equations (6) and (12). It is straightforward to define all dependent variables at cell centers to solve Riemann problem. However, this treatment results in checkerboard solutions in which the pressure and velocity become decoupled when they are defined at the same location

(Patankar, 1980). Therefore, most existing models use a staggered grid in which the pressure is defined at the centers of computational cells and the velocities are defined at cell faces (Bradford, 2005). However, staggered grids do not lend themselves as easily as co-located grids to the use of Godunov-type schemes. Meanwhile, difficulty in treating the cell-centered pressure at the top layer may arise when applying the pressure boundary condition at the free surface (Yuan and Wu, 2004).

With these considerations, a different kind of staggered grid framework is introduced, in which the velocities are placed at the cell centers and the pressure is defined at the vertically-facing cell faces as shown in figure 1. The momentum equations are solved by a second-order Godunov-type finite volume method. The HLL approximate Riemann solver (Harten et al., 1983; Shi et al., 2012) is used to estimate fluxes at the cell faces. As in Stelling and Zijlema (2003), the pressure boundary condition at the free surface can be precisely assigned to zero.

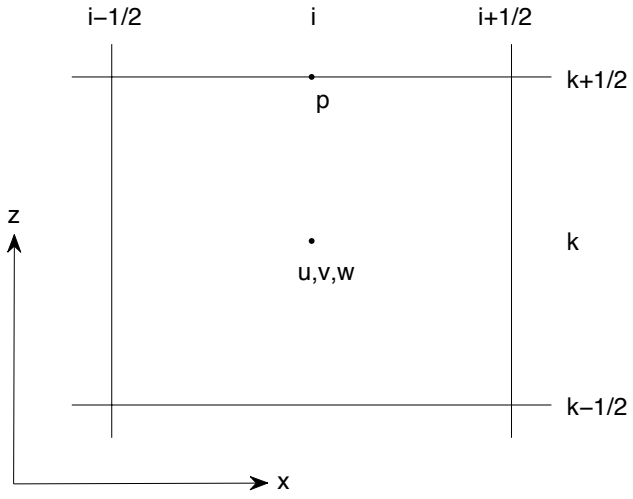


Figure 1: Layout of computational variables. Velocities $(u, v, w)$ are placed at cell center and dynamic pressure $(p)$ is defined at vertical cell face.

### 3.1 Time Stepping

To obtain second-order temporal accuracy, the two-stage second-order nonlinear Strong Stability-Preserving (SSP) Runge-Kutta scheme (Gottlieb et al., 2001) was adopted for time stepping. At the first stage, an intermediate quantity $\mathbf{U}^{(1)}$ is evaluated using a typical first-order, two-step projection

method given by

$$\frac{\mathbf{U}^* - \mathbf{U}^n}{\Delta t} = -\left(\frac{\partial \mathbf{F}}{\partial x} + \frac{\partial \mathbf{G}}{\partial y} + \frac{\partial \mathbf{H}}{\partial \sigma}\right)^n + \mathbf{S}_h^n + \mathbf{S}_\tau^n \tag{22}$$

$$\frac{\mathbf{U}^{(1)} - \mathbf{U}^*}{\Delta t} = \mathbf{S}_p^{(1)} \tag{23}$$

where $\mathbf{U}^n$ represents $\mathbf{U}$ value at time level $n$, $\mathbf{U}^*$ is the intermediate value in the two-step projection method, and $\mathbf{U}^{(1)}$ is the final first stage estimate. In the second stage, the velocity field is again updated to a second intermediate level using the same projection method, after which the Runge-Kutta algorithm is used to obtain a final value of the solution at the $n+1$ time level.

$$\frac{\mathbf{U}^* - \mathbf{U}^{(1)}}{\Delta t} = -\left(\frac{\partial \mathbf{F}}{\partial x} + \frac{\partial \mathbf{G}}{\partial y} + \frac{\partial \mathbf{H}}{\partial \sigma}\right)^{(1)} + \mathbf{S}_h^{(1)} + \mathbf{S}_\tau^{(1)} \tag{24}$$

$$\frac{\mathbf{U}^{(2)} - \mathbf{U}^*}{\Delta t} = \mathbf{S}_p^{(2)} \tag{25}$$

$$\mathbf{U}^{n+1} = \frac{1}{2}\mathbf{U}^n + \frac{1}{2}\mathbf{U}^{(2)} \tag{26}$$

Each stage of the calculation requires the specification of the nonhydrostatic component of the pressure force as expressed through the quantities $\mathbf{S}_p$. The pressure field needed to specify these is based on the solution of the Poisson equation described below. Also at each stage, the surface elevation is obtained by solving equation (12) explicitly. The time step $\Delta t$ is adaptive during the simulation, following the Courant-Friedrichs-Lewy (CFL) criterion

$$\Delta t = C\min\left[\min\frac{\Delta x}{|u_{i,j,k}| + \sqrt{gD_{i,j}}}, \min\frac{\Delta y}{|v_{i,j,k}| + \sqrt{gD_{i,j}}}, \min\frac{\Delta \sigma D_{i,j}}{|w_{i,j,k}|}\right] \tag{27}$$

where $C$ is the Courant number, which is taken to be 0.5 to ensure accuracy and stability in the current model.

### 3.2 Spatial Finite Volume Scheme

We discretize equation (22) and (24) using a second-order Godunov-type finite volume method. It is noticed that applying a standard finite volume Godunov-type scheme directly to the equation does not lead to an automatic preservation of steady state (Zhou et al., 2001; Kim et al, 2008; Liang and Marche, 2009). Therefore, It is desirable to reformulate the equation so that the flux and source terms can be automatically balanced at the discrete level in the steady state. In this study, the method by Liang and Marche (2009) is employed. Taking the $x$ component source term as an example, notice that the total water depth is $D = h + \eta$. The source term can be rewritten as

$$g(h+\eta)\frac{\partial h}{\partial x} = \frac{\partial}{\partial x}\left(\frac{1}{2}gh^2\right) + g\eta\frac{\partial h}{\partial x} \tag{28}$$

in which the first term in the right hand side can be combined together with the flux terms.

Based on this, the flux and source terms may be expressed as

$$\mathbf{F} = \begin{pmatrix} Duu + \frac{1}{2}g\eta^2 + gh\eta \\ Duv \\ Duw \end{pmatrix} \quad \mathbf{G} = \begin{pmatrix} Duv \\ Dvv + \frac{1}{2}g\eta^2 + gh\eta \\ Dvw \end{pmatrix} \quad \mathbf{S}_h = \begin{pmatrix} g\eta\frac{\partial h}{\partial x} \\ g\eta\frac{\partial h}{\partial y} \\ 0 \end{pmatrix}$$

The main advantage of the above formulation is that the flux and source terms are well-balanced so that no artificial flow due to bottom slope will be generated.

To solve equation (22) and (24), fluxes based on the conservative variables are required at the cell faces. In high-order Godunov-type methods, the values of the conservative variables within a cell are calculated using a reconstruction method based on the cell center data (Zhou et al., 2001). Usually a piecewise linear reconstruction is used, leading to a second order scheme. For $\mathbf{U}$ in the cell $i$, we have

$$\mathbf{U} = \mathbf{U}_i + (x - x_i)\Delta\mathbf{U}_i \tag{29}$$

where $\Delta\mathbf{U}_i$ is the gradient of $\mathbf{U}$, which is calculated by

$$\Delta\mathbf{U}_i = \text{avg}\left(\frac{\mathbf{U}_{i+1} - \mathbf{U}_i}{x_{i+1} - x_i}, \frac{\mathbf{U}_i - \mathbf{U}_{i-1}}{x_i - x_{i-1}}\right) \tag{30}$$

in which avg is a slope limiter which is used to avoid spurious oscillations in the reconstruction data at the cell faces. In this study, the van Leer limiter is adopted, which is given by

$$\text{avg}(a, b) = \frac{a|b| + |a|b}{|a| + |b|} \tag{31}$$

The left and right values of $\mathbf{U}$ at cell face $(i + \frac{1}{2})$ are given by

$$\mathbf{U}^L_{i+\frac{1}{2}} = \mathbf{U}_i + \frac{1}{2}\Delta x_i \Delta\mathbf{U}_i \quad \mathbf{U}^R_{i+\frac{1}{2}} = \mathbf{U}_{i+1} - \frac{1}{2}\Delta x_{i+1}\Delta\mathbf{U}_{i+1} \tag{32}$$

The flux $\mathbf{F}(\mathbf{U}^L, \mathbf{U}^R)$ is calculated by solving a local Riemann problem at each horizontally-facing cell face. In the present study, HLL Riemann solver is employed. The flux at the cell interface $(i + \frac{1}{2})$ is determined by

$$\mathbf{F}(\mathbf{U}^L, \mathbf{U}^R) = \begin{cases} \mathbf{F}(\mathbf{U}^L) & \text{if} \quad s_L \geq 0 \\ \mathbf{F}^*(\mathbf{U}^L, \mathbf{U}^R) & \text{if} \quad s_L < 0 < s_R \\ \mathbf{F}(\mathbf{U}^R) & \text{if} \quad s_R \leq 0 \end{cases} \tag{33}$$

where

$$\mathbf{F}^*(\mathbf{U}^L, \mathbf{U}^R) = \frac{s_R\mathbf{F}(\mathbf{U}^L) - s_L\mathbf{F}(\mathbf{U}^R) + s_L s_R(\mathbf{U}^R - \mathbf{U}^L)}{s_R - s_L} \tag{34}$$

with wave speed $s_L$ and $s_R$ defined by

$$s_L = \min(u^L - \sqrt{gD_L}, u_s - \sqrt{gD_s}) \tag{35}$$

$$s_R = \max(u^R + \sqrt{gD_R}, u_s + \sqrt{gD_s}) \tag{36}$$

where $u_s$ and $\sqrt{gD_s}$ are estimated by

$$u_s = \frac{1}{2}(u^L + u^R) + \sqrt{gD_L} - \sqrt{gD_R} \tag{37}$$

$$\sqrt{gD_s} = \frac{\sqrt{gD_L} + \sqrt{gD_R}}{2} + \frac{u^L - u^R}{4} \tag{38}$$

To obtain the non-hydrostatic velocity field, the dynamic pressure $p$ has to be calculated first. From equation (23) and (25), we get

$$u^{(k)} = u^* - \frac{\Delta t}{\rho}\left(\frac{\partial p}{\partial x} + \frac{\partial p}{\partial \sigma}\frac{\partial \sigma}{\partial x^*}\right)^{(k)} \tag{39}$$

$$v^{(k)} = v^* - \frac{\Delta t}{\rho}\left(\frac{\partial p}{\partial y} + \frac{\partial p}{\partial \sigma}\frac{\partial \sigma}{\partial y^*}\right)^{(k)} \tag{40}$$

$$w^{(k)} = w^* - \frac{\Delta t}{\rho}\frac{1}{D^{(k)}}\frac{\partial p^{(k)}}{\partial \sigma} \tag{41}$$

where $k = 1, 2$ represents the $k$th stage in the Runge-Kutta integration.

Applying equation (3) and (4), the continuity equation (1) is transformed as

$$\frac{\partial u}{\partial x} + \frac{\partial u}{\partial \sigma}\frac{\partial \sigma}{\partial x^*} + \frac{\partial v}{\partial y} + \frac{\partial v}{\partial \sigma}\frac{\partial \sigma}{\partial y^*} + \frac{1}{D}\frac{\partial w}{\partial \sigma} = 0 \tag{42}$$

Substituting equation (39) $\sim$ (41) into (42), we obtain the Poisson equation in $(x, y, \sigma)$ coordinate system

$$\begin{aligned}
&\frac{\partial}{\partial x}\left[\frac{\partial p}{\partial x} + \frac{\partial p}{\partial \sigma}\frac{\partial \sigma}{\partial x^*}\right] + \frac{\partial}{\partial y}\left[\frac{\partial p}{\partial y} + \frac{\partial p}{\partial \sigma}\frac{\partial \sigma}{\partial y^*}\right] + \frac{\partial}{\partial \sigma}\left(\frac{\partial p}{\partial x}\right)\frac{\partial \sigma}{\partial x^*} + \\
&\frac{\partial}{\partial \sigma}\left(\frac{\partial p}{\partial y}\right)\frac{\partial \sigma}{\partial y^*} + \left[\left(\frac{\partial \sigma}{\partial x^*}\right)^2 + \left(\frac{\partial \sigma}{\partial y^*}\right)^2 + \frac{1}{D^2}\right]\frac{\partial}{\partial \sigma}\left(\frac{\partial p}{\partial \sigma}\right) = \\
&\frac{\rho}{\Delta t}\left(\frac{\partial u^*}{\partial x} + \frac{\partial u^*}{\partial \sigma}\frac{\partial \sigma}{\partial x^*} + \frac{\partial v^*}{\partial y} + \frac{\partial v^*}{\partial \sigma}\frac{\partial \sigma}{\partial y^*} + \frac{1}{D}\frac{\partial w^*}{\partial \sigma}\right)
\end{aligned} \tag{43}$$

The above equation is discretized with the second-order space-centered finite difference method. The velocities of $(u^*, v^*, w^*)$ at vertical cell faces are interpolated from adjacent cell-centered values. The resulting linear equation is given by

$$\begin{aligned}
&a_1 p_{i,j-1,k-1} + a_2 p_{i-1,j,k-1} + a_3 p_{i,j,k-1} + a_4 p_{i+1,j,k-1} + a_5 p_{i,j+1,k-1} + \\
&a_6 p_{i,j-1,k} + a_7 p_{i-1,j,k} + a_8 p_{i,j,k} + a_9 p_{i+1,j,k} + a_{10} p_{i,j+1,k} + a_{11} p_{i,j-1,k+1} + \\
&a_{12} p_{i-1,j,k+1} + a_{13} p_{i,j,k+1} + a_{14} p_{i+1,j,k+1} + a_{15} p_{i,j+1,k+1} = R_p
\end{aligned} \tag{44}$$

where

$$a_1 = -\left(\frac{(\sigma_y)_{i,j-1,k}}{2\Delta y(\Delta\sigma_k + \Delta\sigma_{k-1})} + \frac{(\sigma_y)_{i,j,k}}{2\Delta y(\Delta\sigma_k + \Delta\sigma_{k-1})}\right)$$

$$a_2 = -\left(\frac{(\sigma_x)_{i-1,j,k}}{2\Delta x(\Delta\sigma_k + \Delta\sigma_{k-1})} + \frac{(\sigma_x)_{i,j,k}}{2\Delta x(\Delta\sigma_k + \Delta\sigma_{k-1})}\right)$$

$$a_3 = -\frac{(\sigma_x^2 + \sigma_y^2 + \frac{1}{D^2})_{i,j,k}}{0.5(\Delta\sigma_k + \Delta\sigma_{k-1})\Delta\sigma_{k-1}}$$

$$a_4 = \frac{(\sigma_x)_{i+1,j,k}}{2\Delta x(\Delta\sigma_k + \Delta\sigma_{k-1})} + \frac{(\sigma_x)_{i,j,k}}{2\Delta x(\Delta\sigma_k + \Delta\sigma_{k-1})}$$

$$a_5 = \frac{(\sigma_y)_{i,j+1,k}}{2\Delta y(\Delta\sigma_k + \Delta\sigma_{k-1})} + \frac{(\sigma_y)_{i,j,k}}{2\Delta y(\Delta\sigma_k + \Delta\sigma_{k-1})}$$

$$a_6 = a_{10} = -\frac{1}{\Delta y^2} \quad a_7 = a_9 = -\frac{1}{\Delta x^2}$$

$$a_8 = \frac{2}{\Delta x^2} + \frac{2}{\Delta y^2} + \frac{(\sigma_x^2 + \sigma_y^2 + \frac{1}{D^2})_{i,j,k}}{0.5(\Delta\sigma_k + \Delta\sigma_{k-1})\Delta\sigma_k} + \frac{(\sigma_x^2 + \sigma_y^2 + \frac{1}{D^2})_{i,j,k}}{0.5(\Delta\sigma_k + \Delta\sigma_{k-1})\Delta\sigma_{k-1}}$$

$$a_{11} = \frac{(\sigma_y)_{i,j-1,k}}{2\Delta y(\Delta\sigma_k + \Delta\sigma_{k-1})} + \frac{(\sigma_y)_{i,j,k}}{2\Delta y(\Delta\sigma_k + \Delta\sigma_{k-1})}$$

$$a_{12} = \frac{(\sigma_x)_{i-1,j,k}}{2\Delta x(\Delta\sigma_k + \Delta\sigma_{k-1})} + \frac{(\sigma_x)_{i,j,k}}{2\Delta x(\Delta\sigma_k + \Delta\sigma_{k-1})}$$

$$a_{13} = -\frac{(\sigma_x^2 + \sigma_y^2 + \frac{1}{D^2})_{i,j,k}}{0.5(\Delta\sigma_k + \Delta\sigma_{k-1})\Delta\sigma_k}$$

$$a_{14} = -\left(\frac{(\sigma_x)_{i+1,j,k}}{2\Delta x(\Delta\sigma_k + \Delta\sigma_{k-1})} + \frac{(\sigma_x)_{i,j,k}}{2\Delta x(\Delta\sigma_k + \Delta\sigma_{k-1})}\right)$$

$$a_{15} = -\left(\frac{(\sigma_y)_{i,j+1,k}}{2\Delta y(\Delta\sigma_k + \Delta\sigma_{k-1})} + \frac{(\sigma_y)_{i,j,k}}{2\Delta y(\Delta\sigma_k + \Delta\sigma_{k-1})}\right)$$

$$R_p = -\frac{\rho}{\Delta t}\left(\frac{\partial u^*}{\partial x} + \frac{\partial u^*}{\partial \sigma}\frac{\partial \sigma}{\partial x^*} + \frac{\partial v^*}{\partial y} + \frac{\partial v^*}{\partial \sigma}\frac{\partial \sigma}{\partial y^*} + \frac{1}{D}\frac{\partial w^*}{\partial \sigma}\right)$$

where $\sigma_x = \frac{\partial \sigma}{\partial x^*}$ and $\sigma_y = \frac{\partial \sigma}{\partial y^*}$.

Uniform gridding is used in the horizontal direction while gridding in the vertical direction is generalized to be non-uniform in order to capture the bottom and surface boundary layers when desired. The coefficient matrix is asymmetric and has a total of 15 diagonal lines. The linear system is solved using the high performace preconditioner HYPRE software library. With $p$ solved, the non-hydrostatic velocities at each stage can be updated from equation (39) to (41).

### 3.3 Boundary Conditions

To solve the governing equations, boundary conditions are required for all the physical boundaries. At the free surface, the continuity of normal and tangential stresses is enforced. With wind effects absent, the tangential stress equals zero, resulting in

$$\frac{\partial u}{\partial \sigma}|_{z=\eta} = \frac{\partial v}{\partial \sigma}|_{z=\eta} = 0 \tag{45}$$

The vertical velocity $w$ at the ghost cells is obtained to ensure that $w$ at free surface satisfies the kinematic boundary condition.

$$w|_{z=\eta} = \frac{\partial \eta}{\partial t} + u\frac{\partial \eta}{\partial x} + v\frac{\partial \eta}{\partial y} \tag{46}$$

The zero pressure condition on the free surface is applied when the Poisson equation is solved.

$$p|_{z=\eta} = 0 \tag{47}$$

At the bottom, the normal velocity and the tangential stress are prescribed. The normal velocity $w$ is imposed through the kinematic boundary condition.

$$w|_{z=-h} = -\frac{\partial h}{\partial t} - u\frac{\partial h}{\partial x} - v\frac{\partial h}{\partial y} \tag{48}$$

For the horizontal velocities, either free-slip boundary conditions

$$\frac{\partial u}{\partial \sigma}|_{z=-h} = \frac{\partial v}{\partial \sigma}|_{z=-h} = 0 \tag{49}$$

or bottom shear stresses are considered.

$$\nu_t \frac{\partial \mathbf{u}}{\partial \sigma}|_{z=-h} = Dc_d|\mathbf{u}_b|\mathbf{u}_b \tag{50}$$

where $c_d$ is the bed drag coefficient, which can be computed from the law of the wall for fully rough, turbulent flow as $c_d = 0.16[ln^2(15\Delta z_1/k_s)]^{-2}$, $\Delta z_1 = D\Delta\sigma_1$ is the thickness of the cell above the bed, $k_s$ is the bottom roughness height. $\mathbf{u}_b$ is velocity at the cell above the bed.

The Neumann boundary condition is used for dynamic pressure, which is directly obtained from the governing equation for $w$.

$$\frac{\partial p}{\partial \sigma}|_{z=-h} = -\rho D\frac{dw}{dt}|_{z=-h} \tag{51}$$

where $w$ at $z = -h$ is given by (48). In the application to an underwater landslide in section 4.6 below, we linearize the resulting boundary condition which gives

$$\frac{\partial p}{\partial \sigma}|_{z=-h} = \rho D\frac{\partial^2 h}{\partial t^2} \tag{52}$$

At the closed boundaries or vertical walls, free-slip boundary conditions are imposed, so that the normal velocity and the tangential stress are set to zero. The normal pressure gradient is zero. At inflow, both free surface and velocities calculated from the analytical solutions are specified. In the lateral direction, periodic boundary conditions can be applied. To facilitate the parallel implementation, we used two ghost cells at each boundaries. The boundary conditions are specified at the ghost cells.

### 3.4 Wetting-Drying Treatment

It is straightforward to use a wetting-drying scheme for modeling moving boundaries. In the present study, a simple wetting-drying scheme is implemented. The wet and dry cells are judged by total water depth $D$. If a cell has the total water depth $D$ greater than $D_{min}$, it's a wet cell with $\text{Mask}_{i,j} = 1$. Otherwise it's a dry cell with $\text{Mask}_{i,j} = 0$. $D_{min}$ is the minimum water depth allowed for computation. The surface elevation in the dry cells are defined as $\eta_{i,j} = D_{min} - h_{i,j}$. For a dry cell surrounded by wet cells, $\text{Mask}_{i,j}$ has to reevaluated as

$$\begin{aligned} \text{Mask}_{i,j} &= 1 \quad \text{if} \quad \eta_{i,j} \leq \eta_{neighbor} \\ \text{Mask}_{i,j} &= 0 \quad \text{if} \quad \eta_{i,j} > \eta_{neighbor} \end{aligned} \tag{53}$$

In the dry cells, the normal flux at cell faces are set to zero. The wave speed of equation (35) and (36) for a dry bed are modified as (Zhou et al., 2001)

$$s_L = u^L - \sqrt{gD_L} \quad s_R = u^L + 2\sqrt{gD_L} \quad \text{(right dry bed)} \tag{54}$$

$$s_L = u^R - 2\sqrt{gD_R} \quad s_R = u^R + \sqrt{gD_R} \quad \text{(left dry bed)} \tag{55}$$

## 4 User's Manual

### 4.1 Installation and Compilation

NHWAVE is distributed in a compressed fie. Use

> tar -xzf *.tar.gz

to extract files from the package. The exacted files will be distributed in three different directories: /source, /examples and /doc. Directory /source contains all the source codes. The main program is in nhwave.F. The program mkdep.f90 can be used to generate water depth file depth.txt for the examples. Directory /examples contains the examples that are introduced in this manual.

NHWAVE requires the installation of *hypre*, as its poisson solver. *hypre* is a library of high performance preconditioners that features parallel multigrid methods for both structured and unstructured grid problems. It is being developed by the Center for Applied Scientific Computing (CASC) at Lawrence Livermore National Laboratory. The source of the *hypre* library can be obtained from http://acts.nersc.gov/hypre/. After unpacking the *hypre* distribution, it must be configured for the same compiler that will be used for NHWAVE. To do this, set the F77 environment variable to the fortran compiler that will be used to build NHWAVE. Then run the configure script

> ./configure –without-fei –prefix=/home/gma/hypre/parallel

where the value assigned to the command line option –prefix refers to the location that the parallel *hypre* library will be installed. Then compile and install the *hypre* library by typing

> make

> make install

After installing *hypre* library, go to /source and modify Makefile to compile NHWAVE. There are several necessary flags in Makefile needed to specify as shown below.

-DDOUBLE_PRECISION: use double precision, default is single precision.

-DPARALLEL: use parallel mode, default is serial mode.

-DLANDSLIDE: include a landslide model for tsunami wave generation

-DSALINITY: run salinity module

-DBUBBLE: run bubble module

-DSEDIMENT: run sediment module

-DVEGETATION: run vegetation module

-DINTEL: use intel fortran compiler

CPP: path to CPP directory.

FC: Fortran compiler.

LIBS: path to the hypre library

INCS: path to the hypre head files

After modifying Makefile, then execute

> make

After compilation, a executable file 'nhwave' will be generated. Note: use 'make clean' after modifying Makefile.

To run the model, go to /examples. Modify input.txt and depth.txt if needed and run. The format of these two files will be introduced below.

## 4.2 Program Outline

The code was written using Fortran 90 with the c preprocessor (cpp) statements for separation of the source code. Arrays are dynamically allocated at runtime. Precision is selected using the *selected_real_kind* Fortran intrinsic function defined in the makefile. The default precision is single.

The present version of NHWAVE includes a number of options including (1) choice of serial or parallel code; (2) rigid/deformable landslides; (3) salinity module; (4) bubble module; (5) cohesive/non-cohesive sediment transport module and (6) vegetation module. These options are controlled by Makefile as presented above.

## 4.3 Subroutines and Functions

**read_input:** read parameters from input.txt, called by MAIN

**index:** define work index, called by MAIN

**allocate_variables:** allocate variables, called by MAIN

**generate_grid:** calculate coordinate x, xc, y, yc, sig and sigc based on dx and dy

**read_bathymetry:** read water depth from depth.txt

**initial:** initialize all the variables, read initial conditions for $\eta, u, v, w$

**estimate_dt:** estimate time step dt based on Courant number and viscous number.

**update_bathymetry:** update water depth for each time step. Only used if the landslide module is turned on

**update_wave_bc:** calculate $\eta, u, v, w$ for the flux boundaries based on wave theories

**update_mask:** update Mask for wetting-drying

**update_wind:** update wind speed at the current time level

**update_var:** save computational variables at previous time level

**source_terms:** calculate all source terms in the momentum equations, including internal wave-maker

**fluxes:** calculate fluxes using HLL or HLLC scheme

**eval_duvw:** update all computational variables ($d, \eta, u, v, w$)

**spong_damping:** add sponge layer

**eval_turb:** update turbulence quantities and eddy viscosities

**eval_sali:** update salinity

**eval_dens:** update water density based on salinity or suspended sediment concentration

**settling_velocity:** calculate sediment settling velocity

**eval_sedi:** update suspended sediment concentration

**eval_bedload:** calculate bed load sediment flux

**update_bed:** update bed elevation

**wave_average:** calculate wave averaged quantities

**statistics:** calculate statistical quantities

**probes:** output time series of variables ($\eta, u, v, w$) at stations to files

**preview:** output field data for both 2D and 3D variables to files

**projection_corrector:** update DU, DV, DW at corrector step with new dynamic pressure, called by **eval_duvw**

**poisson_solver:** solve poisson equation and update dynamic pressure, called by **eval_duvw**

**hypre_pres_solver:** call *hypre* library to solve poisson equation, called by **poisson_solver**

**generate_coef_rhs:** generate coefficient matrix and right-hand-side term for poisson solver

**driver:** add external forcing

**interpolate_velocity_to_faces**: interpolate variables at cell centers to vertical cell faces

**get_Omega:** solve continuity equation to obtain $\omega$ velocity

**get_UVW**: calculate u,v,w based on DU, DV, DW

**construction:** construct left and right fluxes at cell faces

**delxyzFun:** calculate derivatives with respect to x, y and z. It is called by **fluxes**.

**delxFun_2D:** calculate derivatives with respect to x for 2D variables. It is called by **delxyzFun**.

**delyFun_2D:** calculate derivatives with respect to y for 2D variables. It is called by **delxyzFun**.

**delxFun_3D:** calculate derivatives with respect to x for 3D variables. It is called by **delxyzFun**.

**delyFun_3D:** calculate derivatives with respect to y for 3D variables. It is called by **delxyzFun**.

**delzFun_3D:** calculate derivatives with respect to $\sigma$ for 3D variables. It is called by **delxyzFun**.

**flux_bc:** boundary conditions for fluxes

**fluxes_at_faces_HLLC:** calculate fluxes using HLLC scheme

**fluxes_at_faces_HLL:** calculate fluxes using HLL scheme

**HLL:** calculate fluxes using HLL scheme, called by **fluxes_at_faces_HLL**

**HLLC:** calculate fluxes using HLLC scheme, called by **fluxes_at_faces_HLLC**

**wave_speed:** calculate gravity wave speed

**LIMITER:** add van Leer limiter

**putfile2D:** save 2D variables to files

**putfile3D:** save 3D variables to files

**vel_bc:** impose boundary conditions for velocities

**wl_bc:** impose boundary conditions for $\eta$ and d

**phi_2D_coll:** collect data into ghost cells

**diffusion:** calculate horizontal diffusion terms

**adv_scalar_hlpa:** calculate convection terms for scalar transport using HLPA scheme

**kepsilon_3D:** k-epsilon turbulence model for both horizontal and vertical eddy viscosities

## 4.4 Permanent Variables

Some of the variables defined in *mod_global.F* are described below.

x: x coordinate at grid node
xc: x coordinate at cell center
y: y coordinate at grid node
yc: y coordinate at cell center
sig: $\sigma$ coordinate at vertical face
sigc: $\sigma$ coordinate at vertical cell center
H: still water depth $h$ at grid node
Hc: still water depth $h$ at cell center
Hfx: still water depth $h$ at x cell face
Hfy: still water depth $h$ at y cell face
Eta: surface elevation, for dry point, Eta() = MinDep - Depth(), MinDep is specified in input.txt.
Eta0: $\eta$ at previous time level
D: total water depth, D = Hc+Eta
D0: total water depth at previous time level
DeltH: $\partial H/\partial t$ at cell center
DelxH: $\partial H/\partial x$ at cell center
DelyH: $\partial H/\partial y$ at cell center
Mask: 1 - wet, 0 - dry
Mask_Struct: 0 - permanent dry point
U: velocity in x coordinate at current time level
U0: velocity in x coordinate at previous time level
Uf: u velocity at vertical cell faces
V: velocity in y coordinate at current time level
V0: velocity in y coordinate at previous time level
Vf: v velocity at vertical cell faces
W: velocity in z coordinate at current time level

W0: velocity in z coordinate at previous time level
Wf: w velocity at vertical cell faces
Omega: velocity in $\sigma$ coordinate
P: dynamic pressure
DU: D*U
DV: D*V
DW: D*W
CmuHt: horizontal eddy viscosity
CmuVt: vertical eddy viscosity
Rho: water density
Tke: turbulence kinetic energy
Eps: turbulent dissipation rate
DTke: D*Tke
DEps: D*Eps
Prod_s: shear production
Prod_b: bupyancy production
Richf: Richardson number
EtaxL: left value of $\eta$ at x-oriented cell face
EtaxR: right value of $\eta$ at x-oriented cell face
UxL: left value of U velocity at x-oriented cell face
UxR: right value of U velocity at x-oriented cell face
VxL: left value of V velocity at x-oriented cell face
VxR: right value of V velocity at x-oriented cell face
WxL: left value of W velocity at x-oriented cell face
WxR: right value of W velocity at x-oriented cell face
DUxL: left value of DU velocity at x-oriented cell face
DUxR: right value of DU velocity at x-oriented cell face
DVxL: left value of DV velocity at x-oriented cell face
DVxR: right value of DV velocity at x-oriented cell face
DWxL: left value of DW velocity at x-oriented cell face
DWxR: right value of DW velocity at x-oriented cell face
EtayL: left value of $\eta$ at y-oriented cell face
EtayR: right value of $\eta$ at y-oriented cell face
UyL: left value of U velocity at y-oriented cell face
UyR: right value of U velocity at y-oriented cell face
VyL: left value of V velocity at y-oriented cell face
VyR: right value of V velocity at y-oriented cell face
WyL: left value of W velocity at y-oriented cell face
WyR: right value of W velocity at y-oriented cell face
DUyL: left value of DU velocity at y-oriented cell face
DUyR: right value of DU velocity at y-oriented cell face

DVyL: left value of DV velocity at y-oriented cell face
DVyR: right value of DV velocity at y-oriented cell face
DWyL: left value of DW velocity at y-oriented cell face
DWyR: right value of DW velocity at y-oriented cell face
UzL: left value of U velocity at z-oriented cell face
UzR: right value of U velocity at z-oriented cell face
VzL: left value of V velocity at z-oriented cell face
VzR: right value of V velocity at z-oriented cell face
WzL: left value of W velocity at z-oriented cell face
WzR: right value of W velocity at z-oriented cell face
OzL: left value of $\omega$ velocity at z-oriented cell face
OzR: right value of $\omega$ velocity at z-oriented cell face
DelxU: $\partial U/\partial x$ at cell center
DelyU: $\partial U/\partial y$ at cell center
DelzU: $\partial U/\partial \sigma$ at cell center
DelxV: $\partial V/\partial x$ at cell center
DelyV: $\partial V/\partial y$ at cell center
DelzV: $\partial V/\partial \sigma$ at cell center
DelxW: $\partial W/\partial x$ at cell center
DelyW: $\partial W/\partial y$ at cell center
DelzW: $\partial W/\partial \sigma$ at cell center
DelxDU: $\partial DU/\partial x$ at cell center
DelyDU: $\partial DU/\partial y$ at cell center
DelxDV: $\partial DV/\partial x$ at cell center
DelyDV: $\partial DV/\partial y$ at cell center
DelxDW: $\partial DW/\partial x$ at cell center
DelyDW: $\partial DW/\partial y$ at cell center
ExL = DUxL
ExR = DUxR
FxL = DUxL*UxL+0.5*g*(EtaxL*EtaxL+2*EtaxL*Hfx)
FxR = DUxR*UxR+0.5*g*(EtaxR*EtaxR+2*EtaxR*Hfx)
GxL = DxL*UxL*VxL
GxR = DxR*UxR*VxR
HxL = DxL*UxL*WxL
HxR = DxR*UxR*WxR
EyL = DVyL
EyR = DVyR
FyL = DyL*UyL*VyL
FyR = DyR*UyR*VyR
GyL = DVyL*VyL+0.5*g*(EtayL*EtayL+2*EtayL*Hfy)
GyR = DVyR*VyR+0.5*g*(EtayR*EtayR+2*EtayR*Hfy)

HyL = DyL*VyL*WyL
HyR = DyR*VyR*WyR

## 4.5 Program Input

The following describes parameters in input.txt. (**NOTE:** The names of the parameters are capital sensitive).

### 4.5.1 General parameters

TITLE: title of your case, only used for log file.

RESULT_FOLDER: the name of the directory where the outputs are saved

Mglob: cell number in x

Nglob: cell number in y

Kglob: the number of vertical layers
**NOTE:** Typically, 3-5 vertical layers are sufficient to simulate wave processes. More vertical layers would be needed to simulate turbulent flow.

PX: processor numbers in x

PY : processor numbers in y
**NOTE:** PX and PY must be consistency with number of processors defined in mpirun command, e.g., mpirun -np n (where n = px $\times$ py)

SIM_STEPS: total time steps of the simulation
**NOTE:** Simulation will be terminated if the time step is greater than SIM_STEPS.

TOTAL_TIME: total simulation time in seconds
**NOTE:** Simulation will be terminated if the computational time is greater than TOTAL_TIME.

PLOT_START: start time for output to files in seconds

PLOT_INTV: time interval for output to files in seconds

SCREEN_INTV: time interval for screen output in seconds

DX: grid size(m) in x direction.

DY: grid size(m) in y direction.

IVGRD: type of vertical grid. IVGRD = 1, uniform grid with $d\sigma = 1.0/Kglob$. IVGRD = 2, grid size is exponentially increasing from bottom to surface.

GRD_R: the ratio of grid size increment. only used if IVGRD = 2.

DT_INI: initial time step in seconds

DT_MIN: the minimum time step allowable. Simulation will be terminated if dt is less than DT_MIN.

DT_MAX: the maximum time step allowable. If dt is greater than DT_MAX, DT_MAX will be used in the simulation.

DEPTH_TYPE: depth input type. DEPTH_TYPE = CELL_CENTER if water depth is defined at cell center. DEPTH_TYPE = CELL_GRID if water depth is defined at grid points. In this case, the depth file should contain (Mglob+1) $\times$ (Nglob+1) data.

ANA_BATHY: If ANA_BATHY = T, specify water depth in the subroutine read_bathymetry. Otherwise, read water depth from depth.txt. The format of depth.txt is given as follows if DEPTH_TYPE = CELL_CENTER.

DO J=1,Nglob

READ(1,*)(Depth(I,J),I=1,Mglob)

ENDDO

INITIAL_EUVW: If INITIAL_EUVW = T, read initial conditions of $\eta$, $u$, $v$ and $w$ from eta0.txt and uvw0.txt. Otherwise, the simulation has a cold start. The initial conditions for $\eta$, $u$, $v$ and $w$ are zero.

INITIAL_SALI: If INITIAL_SALI = T, read initial condition for salinity from sali0.txt. Needed if salinity module is turned on.

HIGH_ORDER: spatial scheme option, SECOND for the second-order (suggested)

TIME_ORDER: temporal scheme option, FIRST for the first-order, SECOND for the second-order (suggested)

CONVECTION: scalar convection scheme. CONVECTION = HLPA

HLLC: If HLLC = T, HLLC scheme is used to estimate fluxes at cell faces. Otherwise, HLL scheme is used.

Ibot: If Ibot = 1, bottom friction is calculated by a specified drag coefficient Cd0. If Ibot = 2, bottom friction is calculated by a specified bottom roughness Zob

Cd0: drag coefficient

Zob: bottom roughness ($\approx 5.5 d_{50}$)

Iws: If Iws = 1, wind speed is temporally and spatially constant specified by WindU and WindV. If Iws = 2, wind speed is spatially varying given in an input file wind.txt

WindU: wind speed in x direction

WindV: wind speed in y direction

slat: latitude for calculating Coriolis forcing

BAROTROPIC: BAROTROPIC = T for barotropic run. Otherwise, it's a baroclinic run.

NON_HYDRO: If NON_HYDRO = T, it is a non-hydrostatic simulation. Otherwise, it's a hydrostatic simulation.

CFL: Courant number for estimating time step dt (0.5 is suggested).

TRAMP: time to ramp up the simulation (not used)

VISCOUS_FLOW: If VISCOUS_FLOW = T, turbulence model is turned on. In this case, more vertical layers are needed.

IVTURB: option for turbulence closure used to calculate vertical eddy viscosity.
**NOTE:** IVTURB = 1, constant vertical eddy viscosity Cvs; IVTURB = 3, k-$\epsilon$ turbulence closure only considering vertical shear; IVTURB = 10, k-$\epsilon$ turbulence closure considering 3D shear; IVTURB = 20, large-eddy simulation with Smagorinsky subgrid model.

IHTURB: option for turbulence closure used to calculate horizontal eddy viscosity
**NOTE:** IHTURB = 1, constant horizontal eddy viscosity Chs; IHTURB $\geq$ 10, the horizontal eddy viscosity equals the vertical eddy viscosity.

VISCOSITY: laminar viscosity

Schmidt: Schmidt number (=1.0)

Chs: horizontal eddy viscosity if IHTURB = 1

Cvs: vertical eddy viscosity if IVTURB = 1

VISCOUS_NUMBER: The viscous number used to estimate time step dt (=0.1666667).

MinDep: minimum water depth for wetting-drying

ISOLVER: option for different preconditioners (only for serial simulation)

ITMAX: the maximum number of iterations (only for serial simulation)

TOL: stopping criterion for poisson solver (only for serial simulation)

PERIODIC_X: If PERIODIC_X = T, use periodic condition in x direction. In this case, Mglob must be power-of-two.

PERIODIC_Y: If PERIODIC_Y = T, use periodic condition in y direction. In this case, Nglob must be power-of-two.

EXTERNAL_FORCING: If EXTERNAL_FORCING = T, external forcing can be added in subroutine ***driver***.

BC_X0: option for boundary type at left boundary

BC_Xn: option for boundary type at right boundary

BC_Y0: option for boundary type at front boundary

BC_Yn: option for boundary type at back boundary

BC_Z0: option for boundary type at bottom boundary

BC_Zn: option for boundary type at upper boundary
**NOTE:** bc_type = 1: free-slip; 2: no-slip; 3: influx; 4: outflux; 5: bottom friction; 6: radiation bc

WAVEMAKER: wavemaker type listed as follows

LEF_SOL: left boundary solitary wave, need AMP, DEP

LEF_LIN: left boundary linear wave, need AMP, PER, DEP

LEF_CON: left boundary cnoidal wave, need AMP, PER, DEP

LEF_STK: left boundary stokes wave, need AMP, PER, DEP

LEF_TID: left boundary tide wave, has to specify in subroutine ***tidal_wave_left_boundary***

LEF_JON: left boundary for JONSWAP spectrum, need JONSWAP parameters

INI_ETA: initial surface elevation specified in subroutine ***initial***

INT_LIN: internal wavemaker for linear wave, need AMP, PER, DEP and internal wavemaker parameters

INT_CON: internal wavemaker for cnoidal wave

INT_SOL: internal wavemaker for solitary wave

INT_JON: internal wavemaker for JONSWAP spectrum

INT_SPC: internal wavemaker for 2D spectrum (need spc2d.txt)

FLUX_LR: impose flux at both left and right boundaries

AMP: wave height of regular waves

PER: wave period of regular waves

DEP: water depth at wavemaker

THETA: incident wave angle

Xsource_West: location of west boundary of the internal wavemaker

Xsource_East: location of east boundary of the internal wavemaker

Ysource_Suth: location of south boundary of the internal wavemaker

Ysource_Nrth: location of north boundary of the internal wavemaker
**NOTE:** The width of the internal wavemaker can be 1-2 grid cells. The internal wavemaker should cover the whole width of the computational domain.

Hm0: spectral significant wave height for JONSWAP spectrum

Tp: peak wave period for JONSWAP spectrum

Freq_Min: low frequency cutoff (1/s) in JONSWAP spectrum

Freq_Max: high frequency cutoff (1/s) in JONSWAP spectrum

NumFreq: numer of frequency discretizations

SPONGE_ON: logical parameter, T - sponge layer, F - no sponge layer.

Sponge_West_Width: width (m) of sponge layer at west boundary.

Sponge_East_Width: width (m) of sponge layer at east boundary.

Sponge_South_Width: width (m) of sponge layer at south boundary.

Sponge_North_Width: width (m) of sponge layer at north boundary

WAVE_AVERAGE_ON: If WAVE_AVERAGE_ON = T, calculate wave averaged quantities.

WAVE_AVERAGE_START: start time for wave-averaging

WAVE_AVERAGE_END: end time for wave-averaging

WaveheightID: options for calculating wave height (under development)

NSTAT: output at stations. If NSTAT $\geq$ 1, the locations (x,y) of the stations should be given in stat.txt

PLOT_INTV_STAT: time interval for probe output

OUT_H: logical parameter, T - output water depth

OUT_E: logical parameter, T - output surface elevation

OUT_U: logical parameter, T - output velocity in x direction

OUT_V: logical parameter, T - output velocity in y direction

OUT_W: logical parameter, T - output velocity in z direction

OUT_P: logical parameter, T - output dynamic pressure

OUT_K: logical parameter, T - output turbulent kinetic energy

OUT_D: logical parameter, T - output turbulent dissipation rate

OUT_S: logical parameter, T - output shear production

OUT_C: logical parameter, T - output eddy viscosity

OUT_B: logical parameter, T - output bubble void fraction

OUT_A: logical parameter, T - output Reynolds stress

OUT_T: logical parameter, T - output bottom shear stress

OUT_F: logical parameter, T - output sediment concentration

OUT_G: logical parameter, T - output bed elevation

OUT_I: logical parameter, T - output salinity

### 4.5.2 Parameters for sediment module

The following parameters are only used when the sediment module is turned on.

Sed_Type: sediment type, cab be 'COHESIVE' or 'NONCOHESIVE'

BED_LOAD: logical parameter, T - bed load, F - no bed load

COUPLE_FS: logical parameter, T - two-way coupling between suspended sediment and water, F - one-way coupling

Af: parameter to account for sediment effect on bottom drag (= 5.5)

D50: median diameter of sediment

ntyws: option for calculating sediment settling velocity. ntyws = 1, constant settling velocity Sedi_Ws

Sedi_Ws: sediment settling velocity

Shields_c: critical shields number

Tau_ce: critical bed shear stress for erosion

Tau_cd: critical bed shear stress for deposition

Erate: coefficient for sediment erosion flux

Mud_Visc: viscosity of mud

Tim_Sedi: time to start sediment simulation

MorDt: time step for updating morphology

BED_CHANGE: logical parameter, T - include morphology model, F - bed elevation is not changed

### 4.5.3 Parameters for vegetation module

The following parameters are only used when the vegetation module is turned on.

Veg_Type: vegetation type, can be 'RIGID' or 'FLEXIBLE' (Flexible vegetation is under development.)

Veg_X0: location of west edge of vegetation canopy

Veg_Xn: location of east edge of vegetation canopy

Veg_Y0: location of south edge of vegetation canopy

Veg_Yn: location of north edge of vegetation canopy

VegH: vegetation height

VegDens: vegetation canopy density ($1/m^2$)

Vegbv: stem size (m)

VegDrag: drag coefficient for vegetation

VegVM: virtual mass coefficient for vegetation

EI: Youngs module for flexible vegetation

### 4.5.4 Parameters for landslide module

The following parameters are only used when the landslide module is turned on.

SlideType: landslide type, can be 'RIGID', 'DEFORMABLE' and 'TWOLAYER' (Two-layer granular landslide model is under development)

SlideT: landslide thickness

SlideL: landslide length

SlideW: landslide width

SlideAngle: slide angle

SlopeAngle: angle of bottom slope

SlideX0: x coordinate of initial landslide location

SlideY0: y coordinate of initial landslide location

SlideUt: parameter $u_t$ for Enet and Grilli (2005) landslide model

SlideA0: parameter $a_0$ for Enet and Grilli (2005) landslide model

SlideDens: density of landslide material for deformable and two-layer landslides

SlideVisc: viscosity of landslide material for deformable and two-layer landslides

SlideLambda: parameter $\lambda$ for two-layer landslide

SlideIniU: initial velocity of two-layer landslide

Cf_ul: drag coefficient for two-layer landslide

PhiInt: internal friction angle

PhiBed: Coulomb bed friction angle

RHEOLOGY_ON: logical parameter, T - rheology for deformable landslide, F - no rheology

Yield_Stress: yield stress for deformable landslide

Plastic_Visc: plastic viscosity for deformable landslide

### 4.6 Program Output

The output files are saved in a directory defined by RESULT_FOLDER in input.txt. For outputs in ASCII, a file name is a combination of variable name and an output series number (5 digits) such as eta_00001, eta_00002, .... The format and read/write algorithm can be found in subroutines ***putfile2D*** and ***putfile3D***. The variable name can be found in subroutine ***preview***. Output for stations is a series of numbered files (4 digits) such as probe_0001, probe_0002 ....

## 5 Examples

The model has been validated extensively using laboratory experiments for wave transformation and breaking. Some of these benchmarks are given below. In addition, the landslide-induced tsunami benchmarks have been reported by Tehranirad et al. (2012) and Shi et al. (2012).

### 5.1 Periodic wave over a submerged bar

The experimental data by Beji and Battjes (1993) is used to validate our non-hydrostatic model. This case has been used to verify a number of non-hydrostatic free surface models including Casulli (1999), Lin and Li (2002), Chen (2003), Stelling and Zijlema (2003), Yuan and Wu (2004) and Bradford (2005). The data has also frequently been used as a test of Boussinesq models, as the case falls outside the range of typical $O(\mu^2)$ models such as Wei et al (1995), but is handled by various higher order approaches such as Gobbi and Kirby (1999) or Lynett and Liu (2002).

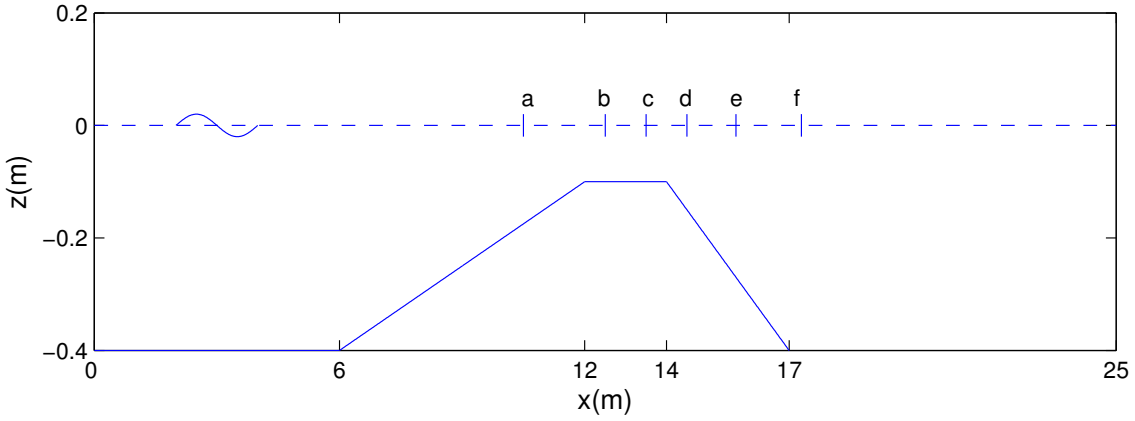


Figure 2: Bottom geometry and location of wave gauges used in the computation (a) $x = 10.5\ m$; (b) $x = 12.5\ m$; (c) $x = 13.5\ m$; (d) $x = 14.5\ m$; (e) $x = 15.7\ m$; (f) $x = 17.3\ m$.

The model setup and bottom geometry is shown in figure 2. The wave flume has a length of 30 $m$. The still water depth is 0.4 $m$, which is reduced to 0.1 $m$ at the bar. The offshore slope of the bar is $1/20$ and the onshore slope is $1/10$. Periodic waves with period 2.02 $s$ and amplitude

1.0 $cm$ are incident at the left boundary. The computational domain is 35 $m$ long with 10 $m$ of sponge layer at the right end. The sponge layer technique introduced by Larsen and Dancy (1983) is employed. This technique has been widely used to absorbing shortwaves (Chen et al., 1999). To discretize the computational domain, 1750 constant horizontal grids and three vertical layers are used to ensure that the free higher harmonics can be properly calculated.

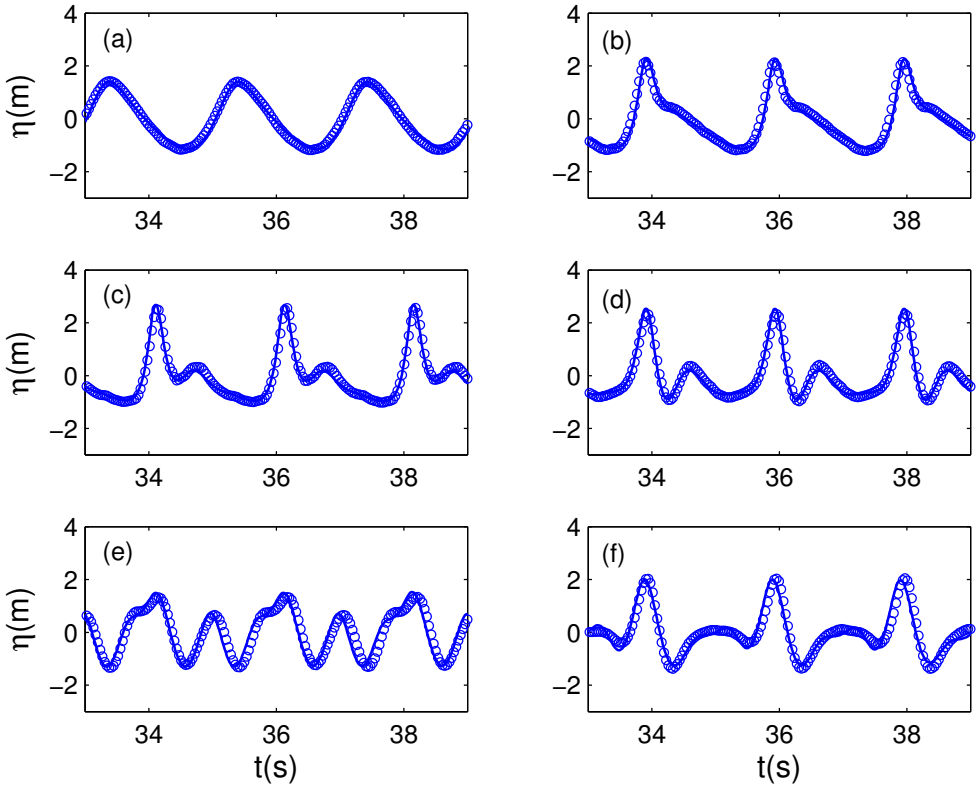


Figure 3: Comparisons between numerical (solid line) and experimental (circles) surface elevations at (a) $x = 10.5\ m$; (b) $x = 12.5\ m$; (c) $x = 13.5\ m$; (d) $x = 14.5\ m$; (e) $x = 15.7\ m$; (f) $x = 17.3\ m$.

Figure 3 shows the comparisons of free surface elevation at six measurement locations between numerical results and experimental data. Wave shoaling at station a and propagation over the bar at station b are well simulated by the model. The bound higher harmonics generated by the nonlinear shoaling wave on the upward slope of the bar become free on the downward slope, resulting in irregular wave pattern at station c to f. The model generally predicts free surface evolution at these stations well, indicating that the dispersion for higher frequency components is well simulated with three vertical layers.

Some important parameters in the input file, input.txt, are

```
! ———————DIMENSION————————————

! cell numbers
Mglob = 1750
Nglob = 1
Kglob = 3

! ———————PROCESSOR NUMBER————————————-
PX = 5
PY = 1

! ———————————TIME————————————————-
! time: total computational time/ plot time / screen interval
! all in seconds
SIM_STEPS = 1000000000
TOTAL_TIME = 40.0
PLOT_START = 0.0
PLOT_INTV = 0.02
SCREEN_INTV = 0.02

! ———————————GRID————————————————-
! grid sizes
DX = 0.02
DY = 0.02

! ———————————VERTICAL GRID OPTION———————————
! IVGRD = 1: uniform; 2: exponential
IVGRD = 1
GRD_R = 1.1

! ———————————-TIME STEP———————————————
DT_INI = 0.10000
DT_MIN = 0.00001
DT_MAX = 0.10000

! ———————————-BATHYMETRY———————————————
! if analytical bathymetry, set ANA_BATHY = T
! DEPTH_TYPE = CELL_CENTER if the water depth is defined at
! cell center, otherwise, DEPTH_TYPE = CELL_GRID
DEPTH_TYPE = CELL_CENTER
```

```
ANA_BATHY = F

! ——————-NUMERICS———————-
! Scalar convection scheme: ”TVD” or ”HLPA”
HIGH_ORDER = SECOND
TIME_ORDER = SECOND
CONVECTION = HLPA
HLLC = F

! ——————BAROTROPIC———————
! if barotropic run, set BAROTROPIC = T
BAROTROPIC = T

! ——————-NON-HYDRO———————
! if non-hydrostatic simulation
NON_HYDRO = T

! ———————-BOUNDARY_TYPE————————
! bc_type=1: free-slip
! 2: no-slip
! 3: influx
! 4: outflux (specified eta)
! 5: bottom friction
! 6: radiation bc
BC_X0 = 3
BC_Xn = 1
BC_Y0 = 1
BC_Yn = 1
BC_Z0 = 1
BC_Zn = 1

! ——————WAVEMAKER———————
! wavemaker
! AMP - wave height; PER - wave period; DEP - incident water depth
! THETA - incident wave angle
! LEF_SOL - left boundary solitary wave, need AMP,DEP
! LEF_LIN - left boundary linear wave, need AMP,PER,DEP
! LEF_CON - left boundary cnoidal wave, need AMP,PER,DEP
! LEF_STK - left boundary stokes wave, need AMP,PER,DEP
! LEF_TID - left boundary tide wave, has to specify in subroutine
! LEF_JON - left boundary for JONSWAP spectrum
```

```
! RIG_LIN - right boundary linear wave, need AMP,PER,DEP,THETA
! INI_ETA - initial surface elevation specified in subroutine initial
! INT_LIN - internal wavemaker for linear wave
! INT_CON - internal wavemaker for cnoidal wave
! INT_SOL - internal wavemaker for solitary wave
! INT_JON - internal wavemaker for JONSWAP spectrum
! INT_SPC - internal wavemaker for 2D spectrum (need spc2d.txt)
! FLUX_LR - impose flux at both left and right boundaries
WAVEMAKER = LEF_LIN
AMP = 0.02
PER = 2.02
DEP = 0.40
THETA = 0.0

! ————————- SPONGE LAYER ——————————
! DHI type sponge layer
! need to specify widths of four boundaries and parameters
! set width=0.0 if no sponge
SPONGE_ON = T
Sponge_West_Width = 0.0
Sponge_East_Width = 10.0
Sponge_South_Width = 0.0
Sponge_North_Width = 0.0

! ————————————PROBE OUTPUT———————————————————
! output variables at stations which are given in file stat.txt
NSTAT = 6
PLOT_INTV_STAT = 0.01
```

## 5.2 Wave transformation over an elliptical shoal on a sloped bottom

This example is to test the model's capability of simulating wave refraction and diffraction over a 3D uneven bottom. The corresponding experiment was conducted by Berkhoff et al. (1982). The model setup and bottom geometry is shown in figure 4. An elliptical shoal is located on a plane beach with a slope of $1/50$. Let $(x', y')$ be the slope-oriented coordinates, which are related to $(x, y)$ coordinate system by means of rotation over $-20^o$. The still water depth without shoal is

given by

$$\begin{aligned} h &= 0.45 \quad x' < -5.84 \\ h &= max(0.07, 0.45 - 0.02(5.84 + x')) \quad x' \geq -5.84 \end{aligned} \tag{56}$$

Since the minimum water depth is 0.07 $m$, the wave is non-breaking. The boundary of the shoal is given by

$$\left(\frac{x'}{3}\right)^2 + \left(\frac{y'}{4}\right)^2 = 1 \tag{57}$$

where the thickness of the shoal is

$$d = -0.3 + 0.5\sqrt{1 - \left(\frac{x'}{3.75}\right)^2 - \left(\frac{y'}{5}\right)^2} \tag{58}$$

Regular wave with wave period of 1.0 $s$ and wave height of 4.64 $cm$ are incident at the left boundary $x = -12\ m$. At the right end, waves are completely absorbed by a sponge layer with $L = 5\ m$. Two walls are located at $y = -10\ m$ and 10 $m$, where free-slip boundary conditions are imposed.

To well simulate wave refraction and diffraction, a fine grid with $\Delta x = 0.025\ m$ and $\Delta y = 0.05\ m$ is used. Three vertical layers are used in the vertical direction. The time step is adjusted during the simulation, with courant number 0.5. The simulation period is 30 $s$. The final five waves are employed to estimate wave height. Figure 5 shows the comparisons of wave height between numerical results and experiment data at eight measurement sections. Due to refraction, wave focussing occurs behind the shoal with a maximum wave height of approximately 2.2 times the incident wave height (around $x = 5\ m$, $y = 0\ m$). The model slightly under-predicts the peak wave height at section 3, 4 and 5. However, the wave height variations along these sections are well captured. In other sections, the predictions agree quite well with the measurements. These results demonstrate that wave refraction and diffraction can be well simulated by the model.

Some important parameters in the input file, input.txt, are

```
! ————————————DIMENSION————————————————

! cell numbers
Mglob = 1200
Nglob = 400
Kglob = 3

! ————————————PROCESSOR NUMBER————————————-
PX = 8
PY = 1

! ————————————TIME————————————————-
! time: total computational time/ plot time / screen interval
```

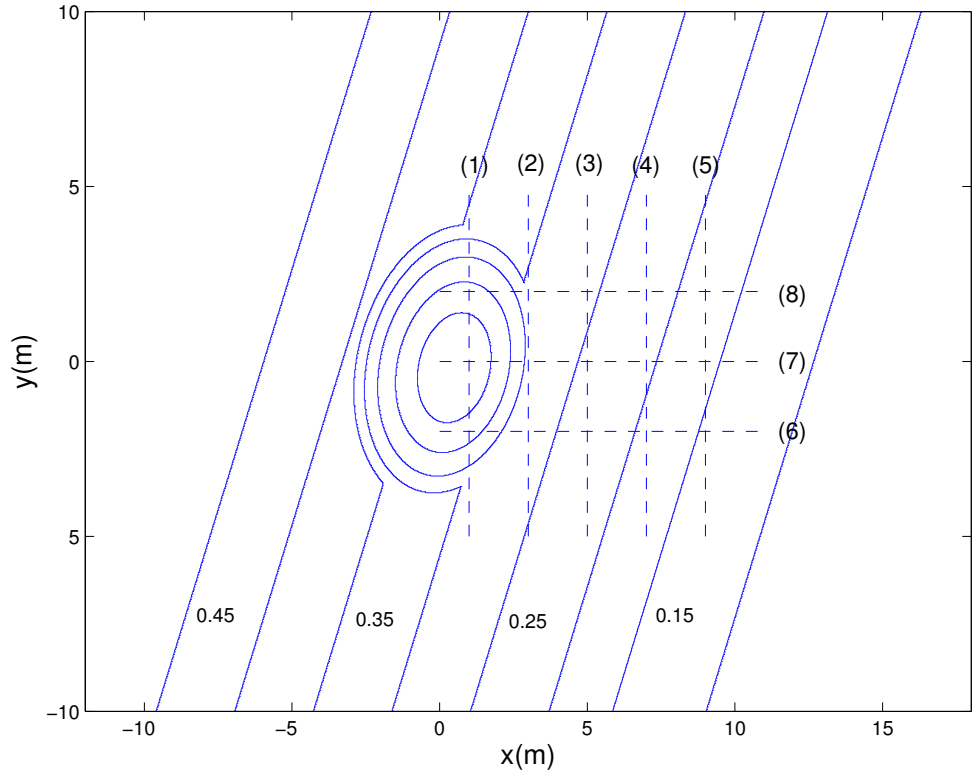


Figure 4: Bottom geometry for periodic wave propagation over an elliptical shoal, experimental setup by Berkhoff et al. (1982)

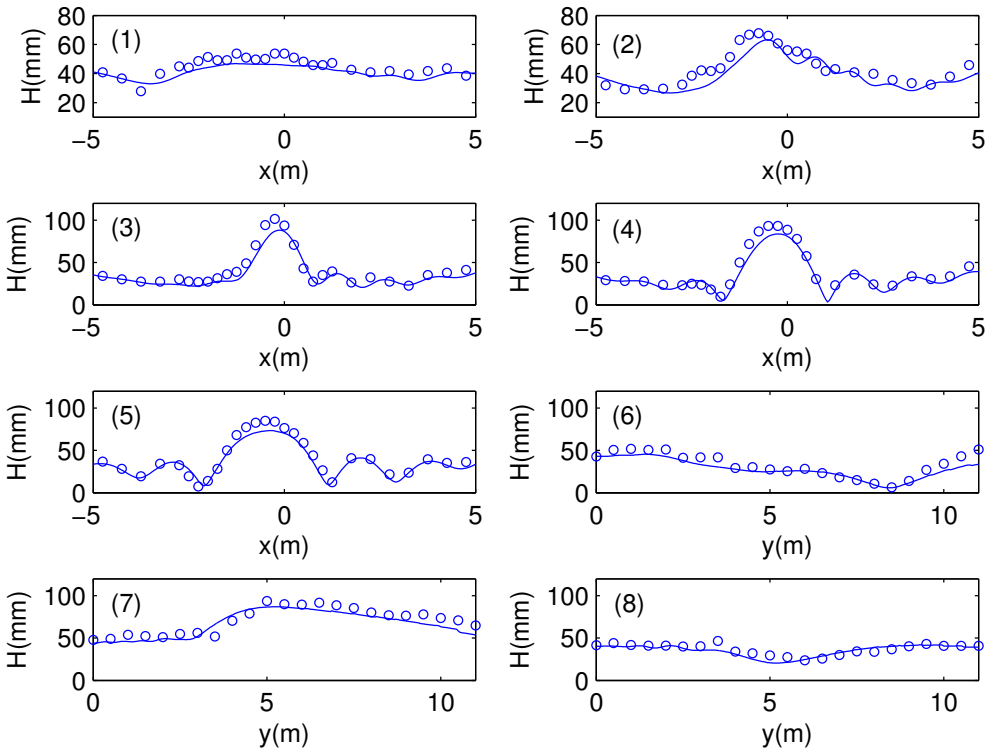


Figure 5: Comparisons between numerical (solid line) and experimental (circles) wave height at eight sections.

```
! all in seconds
SIM_STEPS = 1000000000
TOTAL_TIME = 30.0
PLOT_START = 20.0
PLOT_INTV = 0.02
SCREEN_INTV = 0.02

! ——————GRID————————————-
! grid sizes
DX = 0.025
DY = 0.05

! ——————VERTICAL GRID OPTION———————
! IVGRD = 1: uniform; 2: exponential
IVGRD = 1
GRD_R = 1.1

! ——————-TIME STEP———————————
DT_INI = 0.10000
DT_MIN = 0.00001
DT_MAX = 0.10000

! ——————-BATHYMETRY———————————
! if analytical bathymetry, set ANA_BATHY = T
! DEPTH_TYPE = CELL_CENTER if the water depth is defined at
! cell center, otherwise, DEPTH_TYPE = CELL_GRID
DEPTH_TYPE = CELL_CENTER
ANA_BATHY = F

! ——————-NUMERICS———————————-
! Scalar convection scheme: "TVD" or "HLPA"
HIGH_ORDER = SECOND
TIME_ORDER = SECOND
CONVECTION = HLPA
HLLC = F

! ——————-BOTTOM ROUGHNESS———————-
! Ibot = 1: given the drag coefficient Cd0
! Ibot = 2: given the bottom roughness height Zob
Ibot = 1
Cd0 = 0.006
```

```
Zob = 0.0001

! ——————BAROTROPIC———————
! if barotropic run, set BAROTROPIC = T
BAROTROPIC = T

! ——————-NON-HYDRO———————
! if non-hydrostatic simulation
NON_HYDRO = T

! ——————COURANT_NUMBER————————
CFL = 0.5

! ——————RAMP-UP———————
! time to ramp up simulation
TRAMP = 0.0

! ——————VISCOSITY———————
VISCOUS_FLOW = F
IVTURB = 10
IHTURB = 10
VISCOSITY = 1.e-6
Schmidt = 1.0
Chs = 0.001
Cvs = 0.001

! ——————VISCOUS NUMBER———————-
VISCOUS_NUMBER = 0.1666667

! ——————-WET-DRY———————-
! minimum depth for wetting-drying
MinDep = 0.005

! ——————PERIODIC BC———————
! periodic=.true. : periodic boundary condition in y direction
! Notice if periodic=.true., Nglob must be power-of-two.
! No periodic boundaries in serial run.
PERIODIC_X = F
PERIODIC_Y = F

! ——————EXTERNAL FORCING———————
```

```
EXTERNAL_FORCING = F

! ——————-BOUNDARY_TYPE————————————
! bc_type=1: free-slip
! 2: no-slip
! 3: influx
! 4: outflux (specified eta)
! 5: bottom friction
! 6: radiation bc
BC_X0 = 3
BC_Xn = 1
BC_Y0 = 1
BC_Yn = 1
BC_Z0 = 1
BC_Zn = 1

! ——————————WAVEMAKER———————————————
! wavemaker
! AMP - wave height; PER - wave period; DEP - incident water depth
! THETA - incident wave angle
! LEF_SOL - left boundary solitary wave, need AMP,DEP
! LEF_LIN - left boundary linear wave, need AMP,PER,DEP
! LEF_CON - left boundary cnoidal wave, need AMP,PER,DEP
! LEF_STK - left boundary stokes wave, need AMP,PER,DEP
! LEF_TID - left boundary tide wave, has to specify in subroutine
! LEF_JON - left boundary for JONSWAP spectrum
! RIG_LIN - right boundary linear wave, need AMP,PER,DEP,THETA
! INI_ETA - initial surface elevation specified in subroutine initial
! INT_LIN - internal wavemaker for linear wave
! INT_CON - internal wavemaker for cnoidal wave
! INT_SOL - internal wavemaker for solitary wave
! INT_JON - internal wavemaker for JONSWAP spectrum
! INT_SPC - internal wavemaker for 2D spectrum (need spc2d.txt)
! FLUX_LR - impose flux at both left and right boundaries
WAVEMAKER = LEF_LIN
AMP = 0.0464
PER = 1.0
DEP = 0.45
THETA = 0.0

! ——————— SPONGE LAYER ——————————————
```

```
! DHI type sponge layer
! need to specify widths of four boundaries and parameters
! set width=0.0 if no sponge
SPONGE_ON = T
Sponge_West_Width = 0.0
Sponge_East_Width = 3.0
Sponge_South_Width = 0.0
Sponge_North_Width = 0.0
```

### 5.3 Breaking solitary wave run-up

To show the model's capability of simulating breaking waves and wetting-drying, we applied the model to study breaking solitary wave run-up and run-down in a slope beach. The corresponding laboratory experiment was conducted by Synolakis (1987). The beach slope is $1/20$. The still water depth varies from 0.21 $m$ to 0.29 $m$. A solitary wave which has a wave height to still water depth ratio of 0.28 was incident on the left. Wave gauges were used to record the free surface displacement during the run-up and run-down.

In the numerical simulation, the solitary wave is generated from left boundary. The computational domain extends to a location beyond the maximum run-up point as seen in 6. The entire domain is discretized by 550 uniform grid in the horizontal with $\Delta x = 0.02\ m$. Three layers are used in the vertical direction. The minimum water depth is 5 $mm$, which determines wetting-and-drying of the computational cells.

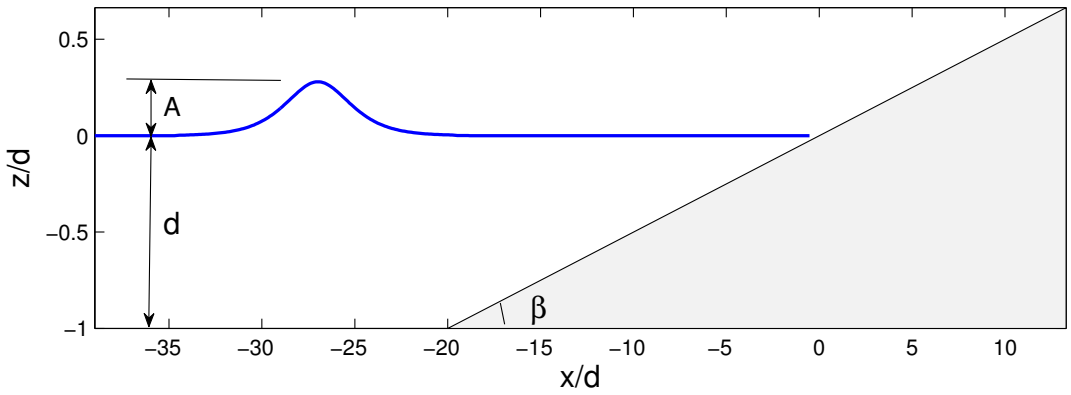


Figure 6: Computational domain and model setup. The beach slope is 1/20. The still water depth is 0.21 $m$. The amplitude of solitary wave is 0.0588 $m$.

The numerical results were compared with the experimental data after normalization. The length scale is normalized by the still water depth $d$ and the time scale is normalized by $\sqrt{g/d}$. Figure 7 shows comparisons of simulated and measured free surface profile during wave shoaling, breaking, run-up and run-down. Panels (a) and (b) show the shoaling process of the solitary wave. The wave becomes more asymmetric and the wave height increases as water depth decreases. Around $t\sqrt{g/d} = 20$, the wave starts to break as shown in panel (c), the surface profile is dramatically changed and the wave height is rapidly reduced. The wave continuously breaks as its turbulent front moves towards the shoreline. After the wave front passes the still-water shoreline, it collapses and the consequent run-up process commences. The run-up process is shown in the panel (d) and (e). After reaching the maximum run-up point, the front starts to run-down which is shown in the panel (f). The comparisons between the simulation and experiment data are fairly good. The shoaling, breaking, run-up and run-down processes of the solitary wave are well reproduced.

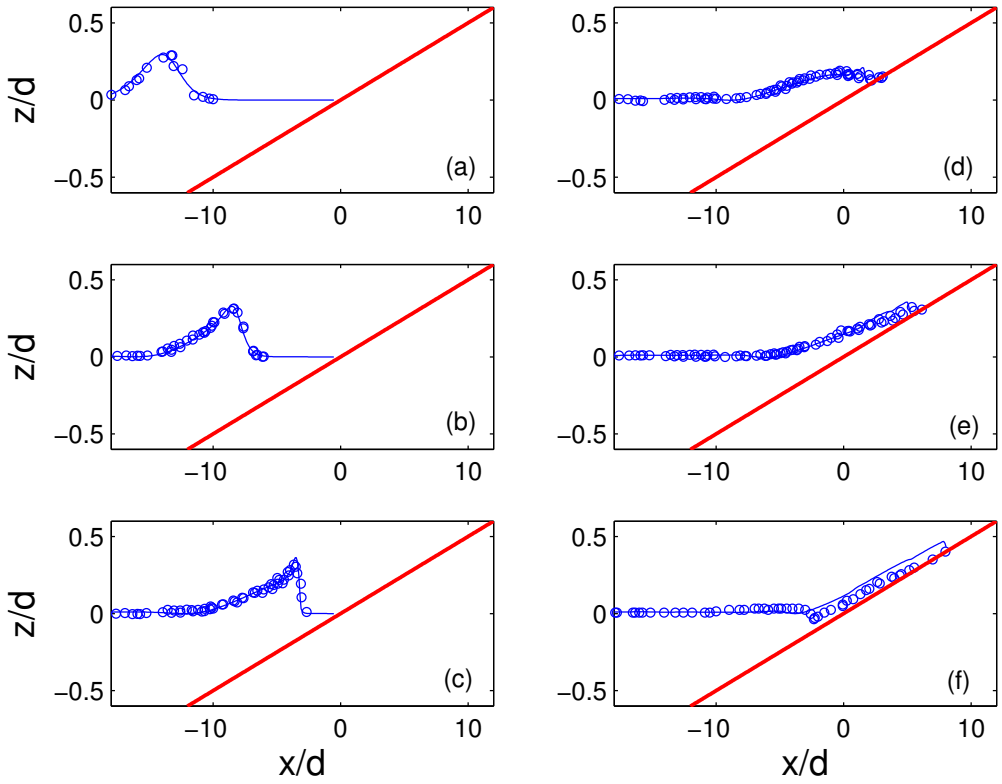


Figure 7: Comparisons between numerical (solid lines) and experimental (circles) free surface elevation for breaking solitary wave run-up and run-down at (a) $t\sqrt{g/d} = 10$; (b) $t\sqrt{g/d} = 15$; (c) $t\sqrt{g/d} = 20$; (d) $t\sqrt{g/d} = 25$; (e) $t\sqrt{g/d} = 30$; (f) $t\sqrt{g/d} = 50$ .

Some important parameters in the input file, input.txt, are

! ————————DIMENSION————————————

! cell numbers
Mglob = 550
Nglob = 1
Kglob = 3

! ————————PROCESSOR NUMBER————————-
PX = 5
PY = 1

```
! ——————TIME————————————-
! time: total computational time/ plot time / screen interval
! all in seconds
SIM_STEPS = 1000000000
TOTAL_TIME = 8.0
PLOT_START = 0.0
PLOT_INTV = 0.02
SCREEN_INTV = 0.02

! ——————GRID————————————-
! grid sizes
DX = 0.02
DY = 0.02

! ——————VERTICAL GRID OPTION——————
! IVGRD = 1: uniform; 2: exponential
IVGRD = 1
GRD_R = 1.1

! ——————-TIME STEP———————————
DT_INI = 0.10000
DT_MIN = 0.00001
DT_MAX = 0.10000

! ——————-BATHYMETRY———————————
! if analytical bathymetry, set ANA_BATHY = T
! DEPTH_TYPE = CELL_CENTER if the water depth is defined at
! cell center, otherwise, DEPTH_TYPE = CELL_GRID
DEPTH_TYPE = CELL_CENTER
ANA_BATHY = F

! ——————BAROTROPIC———————————
! if barotropic run, set BAROTROPIC = T
BAROTROPIC = T

! ——————-NON-HYDRO———————————
! if non-hydrostatic simulation
NON_HYDRO = T

! ——————VISCOSITY———————————
VISCOUS_FLOW = F
```

```
IVTURB = 10
IHTURB = 10
VISCOSITY = 1.e-6
Schmidt = 1.0
Chs = 0.001
Cvs = 0.001

! ——————-WET-DRY———————————-
! minimum depth for wetting-drying
MinDep = 0.005

! ——————-BOUNDARY_TYPE———————————
! bc_type=1: free-slip
! 2: no-slip
! 3: influx
! 4: outflux (specified eta)
! 5: bottom friction
! 6: radiation bc
BC_X0 = 3
BC_Xn = 1
BC_Y0 = 1
BC_Yn = 1
BC_Z0 = 1
BC_Zn = 1

! ——————WAVEMAKER———————————
! wavemaker
! AMP - wave height; PER - wave period; DEP - incident water depth
! THETA - incident wave angle
WAVEMAKER = LEF_SOL
AMP = 0.0588
PER = 1.0
DEP = 0.21
THETA = 0.0
```

## 5.4 Tsunami generation by a three-dimensional underwater landslide

Submarine landslides are one of the most dangerous mechanisms for tsunami generation in the coastal areas. In this section, we applied the model to simulate tsunami generation by an idealized

three-dimensional underwater landslides. Experiments have recently been performed by Enet and Grilli (2007) in a 3.7 $m$ wide, 1.8 $m$ deep and 30 $m$ long wave tank with a plane underwater slope with $\theta = 15^o$ angle. This data set has also been used recently by Fuhrman and Madsen (2009) to test the accuracy of a higher-order Boussinesq model.

The vertical cross section of the landslide is shown in figure 8. The geometry is defined using truncated hyperbolic secant functions.

$$\zeta = \frac{T}{1-\epsilon}[\mathrm{sech}(k_b x)\mathrm{sech}(k_w y) - \epsilon] \tag{59}$$

where $k_b = 2C/b$, $k_w = 2C/w$ and $C = acosh(1/\epsilon)$. The landslide has length $b = 0.395\ m$, width $w = 0.680\ m$ and thickness $T = 0.082\ m$. The truncation parameter $\epsilon = 0.717$. The landslide is initially located at the submergence depth $d$. The movement of the landslide is prescribed as

$$s(t) = s_0 \ln(\cosh \frac{t}{t_0}) \tag{60}$$

which closely approximates the landslide displacement measured in experiments. $s_0$ and $t_0$ are given by

$$s_0 = \frac{u_t^2}{a_0}, \quad t_0 = \frac{u_t}{a_0} \tag{61}$$

where $u_t$ and $a_0$ are the landslide terminal velocity and initial acceleration, respectively. To represent the landslide, the horizontal domain is discretized by a uniform grid with $\Delta x = 0.02\ m$ and $\Delta y = 0.02\ m$. Three vertical layers are employed in the simulation. The landslide parameters are $u_t = 1.70\ m/s$ and $a_0 = 1.12\ m/s^2$.

Three wave gauges are located at $(x, y)$ locations $(1469, 350)$, $(1929, 0)$ and $(1929, 500)$, where all distances are in mm and where $x$ denotes distance from the still water shoreline and $y$ denotes distances off the centerline axis of the sliding mass. Model results are presented as time series in comparison to measured data at each of the three gages, with two representative tests chosen. Figure 9 shows model/data comparisons for the case of an initial submergence of the landslide center of $d = 61$ mm. The model is seen to reproduce the amplitude and the phase structure of the generated wave train well. As would be expected, wave heights are highest at the gage lying along the axis of the landslide motion and drop off with distance away from the centerline axis.

Some important parameters in the input file, input.txt, are

! ————————DIMENSION————————————

! cell numbers
Mglob = 500
Nglob = 90
Kglob = 3

! ————————PROCESSOR NUMBER————————-

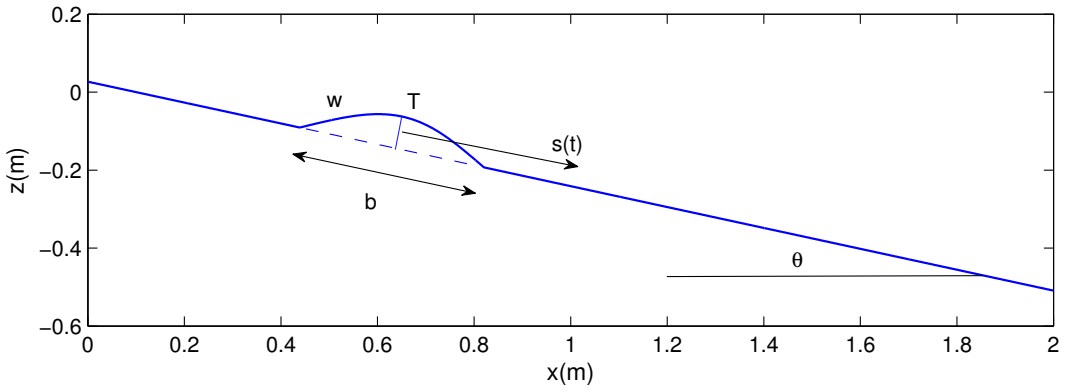


Figure 8: Vertical cross section for numerical setup of tsunami landslide. The gaussian shape landslide model has length $b = 0.395\ m$, width $w = 0.680\ m$ and thickness $T = 0.082\ m$ and is initially located at submergence depth $d$. The beach slope has an angle of $\theta = 15^o$.

PX = 5
PY = 1

! ——————TIME————————————-
! time: total computational time/ plot time / screen interval
! all in seconds
SIM_STEPS = 1000000000
TOTAL_TIME = 4.0
PLOT_START = 0.0
PLOT_INTV = 0.02
SCREEN_INTV = 0.02

! ——————GRID————————————-
! grid sizes
DX = 0.02
DY = 0.02

! ——————VERTICAL GRID OPTION——————
! IVGRD = 1: uniform; 2: exponential
IVGRD = 1
GRD_R = 1.1

! ——————-TIME STEP———————————

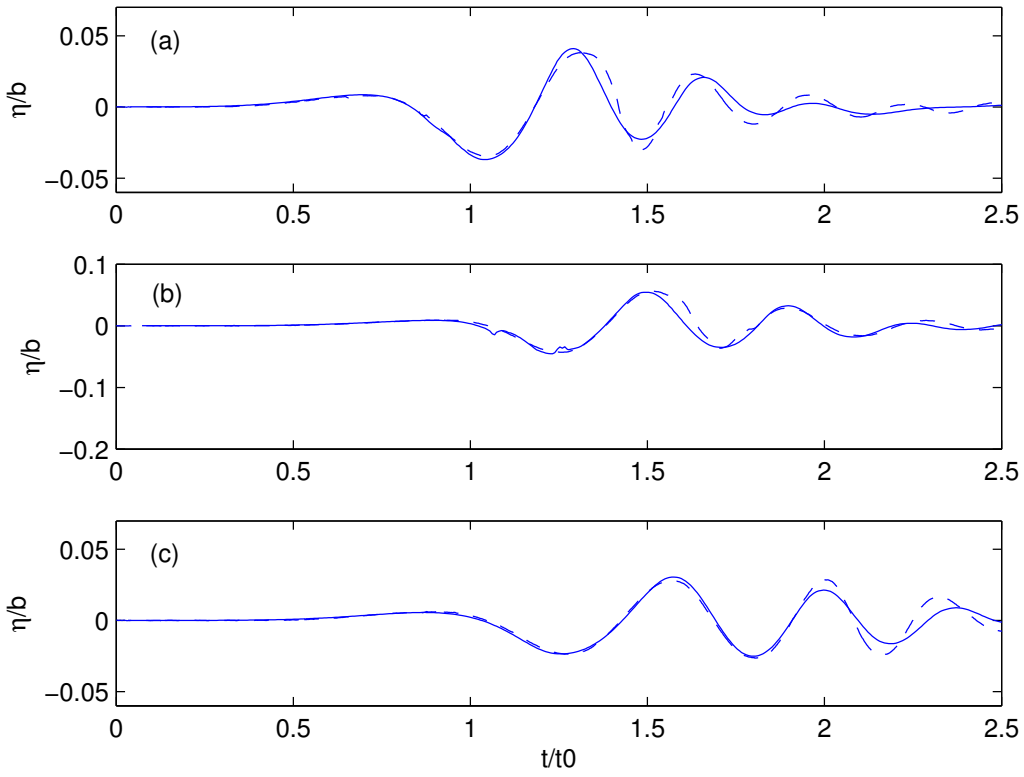


Figure 9: Comparisons between ]numerical results (solid lines) and experimental data (dashed lines) for free surface elevation for landslide-generated waves at three wave gauges with initial depth of submergence $d = 61$ mm. Gauge coordinates $(x, y)$: (a) (1469, 350) mm; (b) (1,929, 0) mm; (c) (1929, 500) mm, where $x$ is distance from shoreline and $y$ is perpendicular distance from the axis of the shore-normal slide trajectory.

```
DT_INI = 0.10000
DT_MIN = 0.00001
DT_MAX = 0.10000

! ——————-BATHYMETRY———————
! if analytical bathymetry, set ANA_BATHY = T
! DEPTH_TYPE = CELL_CENTER if the water depth is defined at
! cell center, otherwise, DEPTH_TYPE = CELL_GRID
DEPTH_TYPE = CELL_CENTER
ANA_BATHY = F

! ——————-INITIAL CONDITION———————
! if INITIAL_SALI = T, need file sali0.txt
INITIAL_EUVW = F
INITIAL_SALI = F

! ——————-NUMERICS———————-
! Scalar convection scheme: "TVD" or "HLPA"
HIGH_ORDER = SECOND
TIME_ORDER = SECOND
CONVECTION = HLPA
HLLC = F

! ——————BAROTROPIC———————
! if barotropic run, set BAROTROPIC = T
BAROTROPIC = T

! ——————-NON-HYDRO———————
! if non-hydrostatic simulation
NON_HYDRO = T

! ——————COURANT_NUMBER————————
CFL = 0.5

! ——————VISCOUS NUMBER———————-
VISCOUS_NUMBER = 0.1666667

! ——————-WET-DRY———————-
! minimum depth for wetting-drying
MinDep = 0.01
```

```
! ————————-BOUNDARY_TYPE————————————
! bc_type=1: free-slip
! 2: no-slip
! 3: influx
! 4: outflux (specified eta)
! 5: bottom friction
! 6: radiation bc
BC_X0 = 1
BC_Xn = 1
BC_Y0 = 1
BC_Yn = 1
BC_Z0 = 1
BC_Zn = 1

! ————————————WAVEMAKER————————————————
! wavemaker
! AMP - wave height; PER - wave period; DEP - incident water depth
! THETA - incident wave angle
! LEF_SOL - left boundary solitary wave, need AMP,DEP
! LEF_LIN - left boundary linear wave, need AMP,PER,DEP
! LEF_CON - left boundary cnoidal wave, need AMP,PER,DEP
! LEF_STK - left boundary stokes wave, need AMP,PER,DEP
! LEF_TID - left boundary tide wave, has to specify in subroutine
! LEF_JON - left boundary for JONSWAP spectrum
! RIG_LIN - right boundary linear wave, need AMP,PER,DEP,THETA
! INI_ETA - initial surface elevation specified in subroutine initial
! INT_LIN - internal wavemaker for linear wave
! INT_CON - internal wavemaker for cnoidal wave
! INT_SOL - internal wavemaker for solitary wave
! INT_JON - internal wavemaker for JONSWAP spectrum
! INT_SPC - internal wavemaker for 2D spectrum (need spc2d.txt)
! FLUX_LR - impose flux at both left and right boundaries
WAVEMAKER = NONE
AMP = 0.0464
PER = 1.0
DEP = 0.45
THETA = 0.0

! ————————-LANDSLIDE PARAMETERS————————————
! parameters for landslide module
! SlideType = 'RIGID' or 'DEFORMABLE'
```

```
! SlideT: thickness; SlideL: length; SlideW: width
! SlideAngle: slide angle
! SlopeAngle: bottom slope
! SlideX0,SlideY0: initial location
! SlideUt,SlideA0: rigid landslide kinematics
! SlideDens: deformable landslide density
SlideType = RIGID
SlideT = 0.082
SlideL = 0.395
SlideW = 0.680
SlideAngle = 0.0
SlopeAngle = 15.0
SlideX0 = 0.651
SlideY0 = 0.0
SlideUt = 1.70
SlideA0 = 1.12
SlideDens = 2104.0
SlideVisc = 0.00001
SlideLambda = 0.5
SlideIniU = 0.0
Cf_ul = 0.02
PhiInt = 41.0
PhiBed = 23.0

! ————————PROBE OUTPUT————————————
! output variables at stations which are given in file stat.txt
NSTAT = 3
PLOT_INTV_STAT = 0.01
```

# References


Agnon, Y., Madsen, P. A. and Schäffer, H. A., 1999, A new approach to high order Boussinesq models, *J. Fluid Mech.*, 399, 319-333.

Ai C. and Jin S., 2010, Non-hydrostatic finite volume method for non-linear waves interacting with structures, *Comp. Fluids*, 39, 2090-2100

Beji S. and Battjes J.A., 1993, Experimental investigation of wave propagation over a bar, *Coastal Engineering*, 19, 151-162

Berkhoff J.C.W., Booy N. and Radder A.C., 1982, Verification of numerical wave propagation models for simple harmonic linear water waves, *Coastal Engineering*, 6, 255-279

Bradford S.F., 2005, Godunov-based model for nonhydrostatic wave dynamics, *J. Waterway, Port, Coastal and Ocean Engineering*, 131, 226-238

Bradford S.F., 2011, Nonhydrostatic model for surf zone simulation, *J. Waterway, Port, Coastal and Ocean Engineering*, 137(4), 163-174

Casulli V. and Stelling G.S., 1998, Numerical simulation of 3D quasi-hydrostatic free-surface flows, *J. Hydr. Engnrg.*, 124, 678-686

Casulli V., 1999, A semi-implicit finite difference method for non-hydrostatic, free-surface flow, *Int. J. Numer. Meth. Fluids*, 30, 425-440

Casulli V. and Zanolli P., 2002, Semi-implicit numerical modeling of nonhydrostatic free-surface flows for environmental problems, *Math. Comp. Modeling*, 36, 1131-1149

Chen Q., Madsen P.A. and Basco D.R., 1999, Current effects on nonlinear interactions of shallow-water waves, *Journal of Waterway, Port, Coastal, and Ocean Engineering*, 125(4), 176-186

Chen Q., Kirby J.T., Dalrymple R.A., Shi F. and Thornton E.B., 2003, Boussinesq modeling of longshore currents, *J. Geophys. Res.*, 108, doi:10.1029/2002JC001308

Chen X., 2003, A fully hydrodynamic model for three-dimensional, free-surface flows, *Int. J. Numer. Meth. Fluids*, 42, 929-952

Christensen E.D., 2006, Large eddy simulation of spilling and plunging breakers, *Coastal Engineering*, 53, 463-485

Enet F. and Grilli S.T., 2007, Experimental study of tsunami generation by three-dimensional rigid underwater landslides, *J. Waterway, Port, Coast. Ocean Engrng.*, 133, 442-454.

Fringer O.B., Gerritsen M. and Street R.L., 2006, An unstructured-grid, finite-volume, nonhydrostatic, parallel coastal ocean simulator, *Ocean Modelling*, 14, 139-173

Fuhrman, D. R. and Madsen, P. A., 2009, Tsunami generation, propagation and run-up with a high-order Boussinesq model, *Coastal Engrng.*, 56, 747-758.

Gobbi, M. F. and Kirby, J. T., 1999, Wave evolution over submerged sills: tests of a high-order Boussinesq model, *Coastal Engrng.*, 37, 57-96.

Gobbi, M. F., Kirby, J. T. and Wei, G., 2000, A fully nonlinear Boussinesq model for surface waves. II. Extension to $O(kh^4)$, *J. Fluid Mech.*, 405, 181-210.

Gottlieb S., Shu C.-W. and Tadmor E., 2001, Strong stability-preserving high-order time discretization methods, *SIAM Review*, 43, 89-112

Harlow F.H. and Welch J.E., 1965, Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface, *Phys. Fluids*, 8, 2182-2189

Harten A., Lax P. and van Leer B., 1983, On upstream differencing and Godunov-type schemes for hyperbolic conservation laws, *SIAM Review*, 25, 35

Hirt C.W. and Nichols B.D., 1981, Volume of fluid (VOF) method for the dynamics of free boundaries, *J. Comp. Phys.*, 39, 201-225

Keshtpoor M., Puleo J.A., Shi F. and Ma G., 2014, 3D numerical simulation of turbulence and sediment transport within a tidal inlet, *Coastal Engineering*, in revision

Kim D. H., Cho Y.-S. and Kim H. J., 2008, Well-balanced scheme between flux and source terms for computation of shallow-water equations over irregular bathymetry, *J. Engnrng. Mech.*, 134, 277-290

Kim, D. H., Lynett, P. J. and Socolofsky, S. A., 2009, "A depth-integrated model for weakly dispersive, turbulent, and rotational fluid flows", *Ocean Modelling*, 27, 198-214.

Kim, D. H. and Lynett, P. J., 2011, Turbulent mixing and scalar transport in shallow and wavy flows, *Phys. Fluids*, 23, 016603.

Lai Z., Chen C., Cowles G.W. and Beardsley R.C., 2010, A nonhydrostatic version of FVCOM: 1. Validation experiments, *J. Geophys. Res.*, 115, doi:10.1029/2009JC005525

Larsen J. and Dancy H., 1983, Open boundaries in short-wave simulations – A new approach, *Coast. Engrg.*, 7(3), 285-297

Lee J.W., Terbner M.D., Nixon J.B. and Gill P.M., 2006, A 3-D non-hydrostatic pressure model for small amplitude free surface flows, *Int. J. Numer. Meth. Fluids*, 50, 649-672

Li B. and Fleming C., 2001, Three-dimensional model of Navier-Stokes equations for water waves, *Journal of Waterway, Port, Coastal, and Ocean Engineering*, 16-25

Liang Q. and Marche F., 2009, Numerical resolution of well-balanced shallow water equations with complex source terms, *Advances in Water Resource*, 32, 873-884

Lin P. and Li C.W., 2002, A $\sigma$-coordinate three-dimensional numerical model for free surface wave propagation, *Int. J. Numer. Meth. Fluids*, 38, 1045-1068

Lin P. and Liu P.L.-F., 1998, A numerical study of breaking waves in the surf zone, *J. Fluid Mech.*, 359, 239-264

Lin P. and Liu P.L.-F., 1998, Turbulent transport, vorticity dynamics, and solute mixing under plunging breaking waves in surf zone, *J. Geophy. Res.*, 103, 15677-15694

Lynett, P. and Liu, P. L.-F., 2002, A two-layer approach to wave modelling, *Proc. Roy. Soc. A*, 460, 2637-2669.

Ma G., Shi F. and Kirby J.T., 2011, A polydisperse two-fluid model for surf zone bubble simulation, *J. Geophys. Res.*, 116, doi:10.1029/2010JC006667

Ma G., Shi F. and Kirby J.T., 2012, Shock-capturing non-hydrostatic model for fully dispersive surface wave processes, *Ocean Modell.*, 43-44, 22-35

Ma G., Kirby J.T., Su S.-F., Figlus J. and Shi F., 2013a, Numerical study of turbulence and wave damping induced by vegetation canopies, *Coastal Eng.*, 80, 68-78

Ma G., Kirby J.T. and Shi F., 2013b, Numerical simulation of tsunami waves generated by deformable submarine landslides, *Ocean Modell.*, 69, 146-165

Ma G., Su S.-F., Liu S. and Chu J.-C., 2014a, Numerical simulation of infragravity waves in fringing reefs using a shock-capturing non-hydrostatic model, *Ocean Eng.*, 85, 54-64

Ma G., Shi F., Hsiao S.-C. and Wu Y.-T., 2014b, Non-hydrostatic modeling of wave interactions with porous structures, *Coastal Eng.*, 91, 84-98

Ma G., Chou Y.-J. and Shi F., 2014c, A wave-resolving model for nearshore suspended sediment transport, *Ocean Modell.*, 77, 33-49

Ma G., Han Y., Niroomandi A., Lou S. and Liu S., 2014d, Numerical study of sediment transport on the tidal flat with a patch of vegetation, *Ocean Dynamics*, revised and resubmitted

Madsen P.A. and Sørensen O.R., 1992, A new form of the Boussinesq equations with improved linear dispersion characteristics. Part A slowly-varying bathymetry, *Coastal Engineering*, 18, 183-204

Namin M., Lin B. and Falconer R., 2001, An implicit numerical algorithm for solving non-hydrostatic free-surface flow problems, *Int. J. Numer. Meth. Fluids*, 35, 341-356

Nwogu O., 1993, Alternative form of Boussinesq equations for nearshore wave propagation, *Journal of Waterway, Port, Coastal and Ocean Engineering*, 119, 618-638

Osher A. and Sethian J.A., 1988, Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations, *J. Comput. Phys.*, 79, 12-49

Patankar S.V., 1980, Numerical heat transfer and fluid flow, McGraw-Hill, New York

Phillips N.A., 1957, A coordinate system having some special advantages for numerical forecasting, *J. Meteor.*, 14, 184-185

Shi F., Kirby J.T. and Ma G., 2010, Modeling quiescent phase transport of air bubbles induced by breaking waves, *Ocean Modelling*. 35, 105-117

Shi F., Ma G., Kirby J.T. and Hsu T.-J., 2012, Application of a TVD solver in a suite of coastal engineering models, *Proc. 33rd Coast. Eng. Conf.*, ICCE2012, Santander, Spain, July 1-6

Shi F., Kirby J.T., Ma G. and Tehranirad B., 2012, Non-hydrostatic wave model NHWAVE user's guide for modeling submarine landslide tsunami (version 1.1), *Research Report No. CACR-12-04*, University of Delaware

Stelling G. and Zijlema M., 2003, An accurate and efficient finite-difference algorithm for non-hydrostatic free-surface flow with application to wave propagation, *Int. J. Numer. Meth. Fluids*, 43, 1-23

Tehranirad B., Kirby J.T., Ma G. and Shi F., 2012, Tsunami benchmark results for non-hydrostatic wave model NHWAVE version 1.1, *Research Report No. CACR-12-03*, University of Delaware

Visser P.J., 1991, Laboratory measurements of uniform longshore currents, *Coastal Engineering*, 15, 563-593

Watanabe Y., Saeki H. and Hosking R.J., 2005, Three-dimensional vortex structures under breaking waves, *J. Fluid Mech.*, 545, 291-328

Walters R.A., 2005, A semi-implicit finite element model for non-hydrostatic (dispersive) surface waves, *Int. J. Numer. Meth. Fluids*, 49, 721-737

Wei G., Kirby J.T., Grilli S.T. and Subramanya R., 1995, A fully nonlinear Boussinesq model for surface waves. Part 1. Highly nonlinear unsteady waves, *J. Fluid Mech.*, 294, 71-92

Wu C.H., Young C.-C., Chen Q. and Lynett P.J., 2010, Efficient nonhydrostatic modeling of surface waves from deep to shallow water, *J. Waterway, Port, Coastal, and Ocean Engineering*, 136, 104-118

Young C.-C., Wu C.-H., Kuo J.-T. and Liu W.-C., 2007, A higher-order $\sigma$-coordinate non-hydrostatic model for nonlinear surface waves, *Ocean Engineering*, 34, 1357-1370

Young C.-C., Wu C.-H., Liu W.-C. and Kuo J.-T., 2009, A higher-order non-hydrostatic $\sigma$ model for simulating nonlinear refraction-diffraction of water waves, *Coastal Engineering*, 56, 919-930

Young C.-C. and Wu C.-H., 2010, A $\sigma$-coordinate non-hydrostatic model with embedded Boussinesq-type-like equations for modeling deep-water waves, *Int. J. Numer. Meth. Fluids*, 63, 1448-1470

Yuan H. and Wu C.-H., 2004, A two-dimensional vertical non-hydrostatic $\sigma$ model with an implicit method for free-surface flows, *Int. J. Numer. Meth. Fluids*, 44, 811-835

Yuan H. and Wu C.-H., 2004, An implicit three-dimensional fully non-hydrostatic model for free-surface flows, *Int. J. Numer. Meth. Fluids*, 46, 709-733

Zhou J.G., Gauson D.M., Mingham C.G. and Ingram D.M., 2001, The surface gradient method for the treatment of source terms in the shallow-water equations, *J. Comp. Phys.*, 168, 1-25

Zijlema M. and Stelling G.S., 2005, Further experiences with computing non-hydrostatic free-surface flows involving water waves, *Int. J. Numer. Meth. Fluids*, 48, 169-197

Zijlema M. and Stelling G.S., 2008, Efficient computation of surf zone waves using the nonlinear shallow water equations with non-hydrostatic pressure, *Coastal Engineering*, 55, 780-790