FUNWAVE-TVD

Fully Nonlinear Boussinesq Wave Model with TVD Solver

Documentation and User's Manual

(**Version 1.0)**

Fengyan Shi, James T. Kirby and Babak Tehranirad
Center for Applied Coastal Research, University of Delaware, Newark, DE 19716
Jeffrey C. Harris and Stephan Grilli
Department of Ocean Engineering, University of Rhode Island, Narragansett, RI 02882

July, 2011

Center for Applied Coastal Research
University of Delaware
Research Report NO. CACR-11-04

Acknowledgements

This model development was a part of the surfzone optics project sponsored by the Office of Naval Research, Coastal Geosciences Program through grant N00014-10-1-0088.

Abstract

The report documents a new version of the fully nonlinear Boussinesq wave model (FUNWAVE) initially developed by Kirby et al. (1998). The development of the present version was motivated by recent needs for modeling of surfzone-scale optical properties in a Boussinesq model framework, and modeling of Tsunami wave in both a regional/coastal scale for prediction of coastal inundation and a basin scale for wave propagation. This version features several theoretical and numerical improvements, including 1) a more complete set of fully nonlinear Boussinesq equations; 2) MUSCL-TVD solver with adaptive Runge-Kutta time stepping; 3) Shock-capturing wave breaking scheme; 4) wetting-drying moving boundary condition with incorporation of HLL construction method into the scheme; 5) an option for parallel computation. The documentation provides derivations of the conservation form of theoretical equations, re-arrangement of pressure gradient term in order to obtain a numerically well-balanced form, detailed numerical schemes, users' manual and examples.

Contents

List of Figures

List of Tables

1 Introduction

The Boussinesq wave model has been a useful tool for modeling surface waves from deep water to the swash zone, as well as wave-induced circulations inside the surfzone. FUNWAVE, the fully nonlinear Boussinesq model developed by the University of Delaware group (Kirby et al.,1998), has come into fairly wide usage in the coastal community. FUNWAVE was based on the fully nonlinear Boussinesq equations derived by Wei et al. (1995) and used the high-order finite difference numerical method.

Since the initial version of FUNWAVE was developed, there were several updates in both theory and numerics for the fully nonlinear Boussinesq model. Gobbi et al. (2000) extended Wei et al. (1995) to higher order, which improved predictions of near-bed kinematics in deeper water. Chen (2006) pointed out missing terms in Wei et al. (1995) which represent the vertical vorticity on a sloping bed. Kennedy et al. (2001) introduced a concept of reference elevation in the derivation of the extended Boussinesq equations to improve nonlinear performance. Shi et al. (2001) extended the fully nonlinear Boussinesq equations into a non-Cartesian coordinate system.

In the aspect of numerics, Wei and Kirby (1995) initially described a numerical scheme in which time stepping is treated using a fourth-order Adams-Bashforth-Moulton scheme, while spatial differencing is handled using a mixed-order scheme, employing fourth-order accurate centered differences for first derivatives and second-order accurate differences for third derivatives. The choice is made in order to move leading truncation errors to one order higher than the $O(\mu^2)$ dispersive terms, while maintaining the tridiagonal structure of spatial derivatives within time-derivative terms. A non-staggered grid system was used in Wei and Kirby (1995). Shi et al (2001) used similar numerical schemes but a staggered grid approach, which has less apparent sensitivity to treatment of boundary conditions. The staggered grid scheme has become the preferred approach in later developments of the Delaware Boussinesq models such as in FUNWAVE 2 (Long and Kirby, 2006).

The moving shoreline condition in FUNWAVE was treated using a so-called slot technique (Kennedy et al., 2000, Chen et al., 2000). In the slot method, deep and narrow slots are added to each grid row, extending down at least to the lowest elevation that will be experienced during shoreface rundown. Recent model tests on solitary wave runup performed at Oregon State University's O.H. Hinsdale Wave Research Laboratory (ISEC/NEES Workshop, Oregon State University, July, 2009) have raised a concern about considerable errors induced by the slot method. Slots, which are too wide relative to the model grid spacing, admit too much fluid before filling during runup, and cause both a reduction in amplitude and a phase lag in modeled runup events. At the other extreme, slots, which are too narrow, tend to induce a great deal of numerical noise, leading to the need for intermittent or even fairly frequent filtering of swash zone solutions.

Wave breaking in FUNWAVE was approached with the eddy viscosity method of Kennedy et al. (2000), following an early eddy viscosity model by Zelt (1991). Kennedy et al. used a model which involves a time history in order to allow the slope of the breaking wave crest to relax after the onset of breaking. Similar approaches were used by other Boussinesq model developers, such as Nwogu and Demirbilek (2001) who used a more sophisticated eddy viscosity model in which

the eddy viscosity is expressed in terms of turbulent kinetic energy and a length scale.

Recent progress in the development of Boussinesq-type wave models, using a hybrid method combining the finite-volume and finite-difference TVD-type schemes (Toro, 2009), have shown robust performance of the shock-capturing method in simulating breaking waves and coastal inundation (Tonelli and Petti, 2009, Roeber et al., 2010, Shiach and Mingham, 2009, Erduran et al., 2005, and others). The shock-capturing scheme makes the treatment of wave breaking straightforward, without the artificial viscosity adopted in some breaking wave models such as in Kennedy et al. (2000). The scheme is also able to capture the sharp wave front occurring in the swash zone. The combined finite-volume and finite difference scheme also makes it easy to implement a wetting-drying moving shoreline condition. Recent applications of using such a wetting-drying method have shown to be quite accurate in modeling of wave runup (e.g., Lynett et al., 2002).

This work was motivated by recent needs to model the surfzone and swash zone dynamics and associated breaking wave-induced processes, such as optical properties and sediment transport in a 10 km-scale computational domain, and to model tsunami waves in both a regional/coastal scale for prediction of coastal inundation and a basin scale for wave propagation. We are pursuing a version, which is stable and robust in model efficiency and accuracy in a long time wave simulation in a large computational domain.

In this version, we start with the more complete set of fully nonlinear Boussinesq equations developed by Chen (2006), extended to incorporate a moving reference elevation following Kennedy et al (2001). The use of a moving reference elevation is more consistent with a time-varying representation of elevation at a moving shoreline in modeling of a swash zone dynamics and coastal inundation. A conservative form of the equations is derived in order to use a hybrid numerical scheme. Dispersive terms were reorganized with the aim of constructing a tridiagonal structure of spatial derivatives within time-derivative terms. The surface elevation gradient term was also rearranged to obtain a numerically well-balanced form, which is suitable for any numerical order. In contrast to previous high-order temporal schemes, which usually require uniform time-stepping, we used adaptive time stepping based on the third-order Runge-Kutta method. Spatial derivatives were discretized using a combination of finite-volume and finite-difference methods. A high-order MUSCL reconstruction technique, which is accurate up to the fourth-order, was used in the Riemann solver. The wave breaking scheme followed the approach of Tonelli and Petti (2009), who used the ability of the nonlinear shallow water equations with a TVD solver to simulate moving hydraulic jumps. Wave breaking is modeled by switching Boussinesq to NSWE at cells where the Froude number exceeds a certain threshold. A wetting-drying scheme was used to model a moving shoreline.

The model was parallelized using the domain decomposition technique. The Message Passing Interface (MPI) with non-blocking communication is used for data communication between processors.

This report provides derivations of the conservation form of theoretical equations with a well-balanced pressure gradient term, numerical schemes, and users' manual. The last part of report illustrates the model's applications to problems of wave breaking and runup in the context of a standard suite of benchmark tests. In addition, we include a brief documentation of the spheri-

cal Boussinesq model used for Tsunami wave simulations. The spherical Boussinesq model was based on Kirby et al. (2004, 2011) and implemented in the same model framework. A detailed documentation for the spherical code will be reported separately.

2 Theory

In this section, we describe the development of a set of Boussinesq equations which are accurate to $O(\mu^2)$ in dispersive effects. Here, μ is a parameter characterizing the ratio of water depth to wave length, and is assumed to be small in classical Boussinesq theory. We retain dimensional forms below but will refer to the apparent $O(\mu^2)$ ordering of terms resulting from deviations from hydrostatic behavior in order to identify these effects as needed. The model equations used here follow from the work of Chen (2006). In this and earlier works starting with Nwogu (1993), the horizontal velocity is written as

$$\mathbf{u} = \mathbf{u}_\alpha + \mathbf{u}_2(z) \tag{1}$$

Here, $\mathbf{u}_\alpha$ denotes the velocity at a reference elevation $z = z_\alpha$, and

$$\mathbf{u}_2(z) = (z_\alpha - z)\nabla A + \frac{1}{2}\left(z_\alpha^2 - z^2\right)\nabla B \tag{2}$$

represents the depth-dependent correction at $O(\mu^2)$, with A and B given by

$$\begin{aligned} A &= \nabla \cdot (h\mathbf{u}_\alpha) \\ B &= \nabla \cdot \mathbf{u}_\alpha \end{aligned} \tag{3}$$

The derivation follows Chen (2006) except for the additional effect of letting the reference elevation z_α vary in time according to

$$z_\alpha = \zeta h + \beta\eta \tag{4}$$

where h is local still water depth, η is local surface displacement and ζ and β are constants, as in Kennedy et al (2001). This addition does not alter the details of the derivation, which are omitted below.

2.1 Governing equations

The equations of Chen (2006) extended to incorporate a possible moving reference elevation follow. The depth-integrated volume conservation equation is given by

$$\eta_t + \nabla \cdot \mathbf{M} = 0 \tag{5}$$

where

$$\mathbf{M} = H\left\{\mathbf{u}_\alpha + \overline{\mathbf{u}}_2\right\} \tag{6}$$

is the horizontal volume flux. $H = h+\eta$ is the total local water depth and $\overline{\mathbf{u}}_2$ is the depth averaged $O(\mu^2)$ contribution to the horizontal velocity field, given by

$$\overline{\mathbf{u}}_2 = \frac{1}{H}\int_{-h}^{\eta} \mathbf{u}_2(z)dz = \left(\frac{z_\alpha^2}{2} - \frac{1}{6}(h^2 - h\eta + \eta^2)\right)\nabla B + \left(z_\alpha + \frac{1}{2}(h-\eta)\right)\nabla A \tag{7}$$

The depth-averaged horizontal momentum equation can be written as

$$\mathbf{u}_{\alpha,t} + (\mathbf{u}_\alpha \cdot \nabla)\mathbf{u}_\alpha + g\nabla\eta + \mathbf{V}_1 + \mathbf{V}_2 + \mathbf{V}_3 + \mathbf{R} = 0 \tag{8}$$

where g is the gravitational acceleration and $\mathbf{R}$ represents diffusive and dissipative terms including bottom friction and subgrid lateral turbulent mixing. $\mathbf{V}_1$ and $\mathbf{V}_2$ are terms representing the dispersive Boussinesq terms given by

$$\mathbf{V}_1 = \left\{\frac{z_\alpha^2}{2}\nabla B + z_\alpha \nabla A\right\}_t - \nabla\left[\frac{\eta^2}{2}B_t + \eta A_t)\right] \tag{9}$$

$$\mathbf{V}_2 = \nabla\left\{(z_\alpha - \eta)(\mathbf{u}_\alpha\cdot\nabla)A + \frac{1}{2}(z_\alpha^2 - \eta^2)(\mathbf{u}_\alpha\cdot\nabla)B + \frac{1}{2}[A+\eta B]^2\right\} \tag{10}$$

The form of (9) allows for the reference level z_α to be treated as a time-varying elevation, as suggested in Kennedy et al (2001). If this extension is neglected, the terms reduce to the form given originally by Wei et al (1995). The expression (10) for $\mathbf{V}_2$ was also given by Wei et al (1995), and is not altered by the choice of a fixed or moving reference elevation.

The term $\mathbf{V}_3$ in (8) represents the $O(\mu^2)$ contribution to the expression for $\boldsymbol{\omega}\times\mathbf{u} = \omega\mathbf{i}^z \times \mathbf{u}$ (with $\mathbf{i}^z$ the unit vector in the z direction) and may be written as

$$\mathbf{V}_3 = \omega_0\mathbf{i}^z \times \overline{\mathbf{u}}_2 + \omega_2\mathbf{i}^z \times \mathbf{u}_\alpha \tag{11}$$

where

$$\omega_0 = (\nabla\times\mathbf{u}_\alpha)\cdot\mathbf{i}^z = v_{\alpha,x} - u_{\alpha,y} \tag{12}$$

$$\omega_2 = (\nabla\times\overline{\mathbf{u}}_2)\cdot\mathbf{i}^z = z_{\alpha,x}(A_y + z_\alpha B_y) - z_{\alpha,y}(A_x + z_\alpha B_x) \tag{13}$$

Following Nwogu (1993), z_α is usually chosen in order to optimize the apparent dispersion relation of the linearized model relative to the full linear dispersion in some sense. In particular, the choice $\alpha = (z_\alpha/h)^2/2 + z_\alpha/h = -2/5$ recovers a Padé approximant form of the dispersion relation, while the choice $\alpha = -0.39$, corresponding to the choice $z_\alpha = -0.53h$, minimizes the maximum error in wave phase speed occurring over the range $0 \le kh \le \pi$. Kennedy et al (2001) showed that, allowing z_α to move up and down with the passage of the wave field, allowed a greater degree of flexibility in optimizing nonlinear behavior of the resulting model equations. In the examples chosen here, where a great deal of our focus is on the behavior of the model from the break point landward, we adopt Kennedy et al's "datum invariant" form

$$z_\alpha = -h + \beta H = (\beta - 1)h + \beta\eta = \zeta h + (1+\zeta)\eta \tag{14}$$

with $\zeta = -0.53$ as in Nwogu (1993) and $\beta = 1 + \zeta = 0.47$ This corresponds in essence to a σ coordinate approach, which places the reference elevation at a level 53% of the total local depth below the local water surface. This also serves to keep the model reference elevation within the actual water column over the entire wetted extent of the model domain.

2.2 Treatment of the surface gradient term

The hybrid numerical scheme requires a conservative form of continuity equation and momentum equations, thus requiring a modification of the leading order pressure term in the momentum equation. A numerical imbalance problem occurs when the surface gradient term is conventionally split into an artificial flux gradient and a source term that includes the effect of the bed slope for a non-uniform bed. To eliminate errors introduced by the traditional depth gradient method (DGM), a so-called surface gradient method (SGM) proposed by Zhou et al. (2001) was adopted in the TVD based-Boussinesq models in the recent literatures. Zhou et al. discussed an example of SGM in 1-D and verified that the slope-source term may be canceled out by part of the numerical flux term associated with water depth, if the bottom elevation at the cell center is constructed using the average of bottom elevations at two cell interfaces. Zhou et al. also showed a 2D application, but without explicitly describing 2D numerical schemes. Although this scheme can be extended into 2D following the same procedure as in 1D, it was found that the 2D extension may not be trivial in terms of the bottom construction for a 2D arbitrary bathymetry. Kim et al. (2008) pointed out that the water depth in the slope-source term should be written in a discretized form rather than the value obtained using the bottom construction, implying that their revised SGM is valid for general 2D applications.

For the higher-order schemes, such as the fourth-order MUSCL-TVD scheme (Yamamoto and Daiguji, 1993, Yamamoto et al., 1998) used in the recent Boussinesq applications, the original SGM and the revised SGM may not be effective in removing the artificial source. This problem was recently noticed by some authors, such as Roeber et al. (2010), who kept a first-order scheme (second-order for normal conditions) for the numerical flux term and the slope-source term in order to ensure a well-balanced solution, without adding noise for a rapidly varying bathymetry.

In fact, the imbalance problem can be solved by a reformulation of this term in terms of deviations from an unforced but separately specified equilibrium state (see general derivations in Rogers et al., 2003 and recent application in Liang and Marche, 2009). Using this technique, the surface gradient term may be split as

$$gH\nabla\eta = \nabla\left[\frac{1}{2}g(\eta^2 + 2h\eta)\right] - g\eta\nabla h \tag{15}$$

which is well-balanced for any numerical order under an unforced stationary condition (still water condition).

2.3 Conservative form of fully nonlinear Boussinesq equations

For Chen's (2006) equations or the minor extension considered here, $H\mathbf{u}_\alpha$ can be used as a conserved variable in the construction of a conservative form of Boussinesq equations, but this results in a source term in the mass conservation equation, such as in Shiach and Mingham (2009) and Roeber et al. (2010). An alternative approach is to use $\mathbf{M}$ as a conserved variable in terms of the physical meaning of mass conservation. In this study, we used $\mathbf{M}$, instead of $H\mathbf{u}_\alpha$, in the following derivations of the conservative form of the fully nonlinear Boussinesq equations.

Using $\mathbf{M}$ from (6) together with the vector identity

$$\nabla \cdot (\mathbf{uv}) = \nabla \mathbf{u} \cdot \mathbf{v} + (\nabla \cdot \mathbf{v})\mathbf{u} \tag{16}$$

allows (8) to be rearranged as

$$\begin{aligned} &\mathbf{M}_t + \nabla \cdot \left(\frac{\mathbf{MM}}{H} \right) + gH\nabla\eta \\ &= H\left\{ \overline{\mathbf{u}}_{2,t} + \mathbf{u}_\alpha \cdot \nabla \overline{\mathbf{u}}_2 + \overline{\mathbf{u}}_2 \cdot \nabla \mathbf{u}_\alpha - \mathbf{V}_1 - \mathbf{V}_2 - \mathbf{V}_3 - \mathbf{R} \right\} \end{aligned} \tag{17}$$

Following Wei et al. (1995), we separate the time derivative dispersion terms in $\mathbf{V}_1$ according to

$$\mathbf{V}_1 = \mathbf{V}'_{1,t} + \mathbf{V}''_1 \tag{18}$$

where

$$\mathbf{V}'_1 = \frac{z_\alpha^2}{2}\nabla B + z_\alpha \nabla A - \nabla \left[\frac{\eta^2}{2} B + \eta A \right] \tag{19}$$

and

$$\mathbf{V}''_1 = \nabla \left[\eta_t (A + \eta B) \right] \tag{20}$$

Using (15), (19) and (20), the momentum equation can be rewritten as

$$\begin{aligned} &\mathbf{M}_t + \nabla \cdot \left[\frac{\mathbf{MM}}{H} \right] + \nabla \left[\frac{1}{2} g(\eta^2 + 2h\eta) \right] = \\ &= H\left\{ \overline{\mathbf{u}}_{2,t} + \mathbf{u}_\alpha \cdot \nabla \overline{\mathbf{u}}_2 + \overline{\mathbf{u}}_2 \cdot \nabla \mathbf{u}_\alpha - \mathbf{V}'_{1,t} - \mathbf{V}''_1 - \mathbf{V}_2 - \mathbf{V}_3 - \mathbf{R} \right\} + g\eta\nabla h \end{aligned} \tag{21}$$

A difficulty usually arises in applying the adaptive time-stepping scheme to the time derivative dispersive terms $\bar{u}_{2,t}$ and $\mathbf{V}'_{1,t}$, which was usually calculated using values stored in several time levels in the previous Boussinesq codes such as in Wei et al. (1995) and Shi et al. (2001). To prevent this, the equation can be re-arranged by merging the time derivatives on the right hand side into the time derivative term on the left hand side, giving

$$\begin{aligned} \mathbf{V}_t \quad &+\nabla \cdot \left[\frac{\mathbf{MM}}{H} \right] + \nabla \left[\frac{1}{2} g(\eta^2 + 2h\eta) \right] = \eta_t (\mathbf{V}'_1 - \overline{\mathbf{u}}_2) \\ &+H\left(\mathbf{u}_\alpha \cdot \overline{\mathbf{u}}_2 + \overline{\mathbf{u}}_2 \cdot \nabla \mathbf{u}_\alpha - \mathbf{V}''_1 - \mathbf{V}_2 - \mathbf{V}_3 - \mathbf{R} \right) + g\eta\nabla h \end{aligned} \tag{22}$$

where

$$\mathbf{V} = H(\mathbf{u}_\alpha + \mathbf{V}_1') \tag{23}$$

In (23) η_t can be calculated explicitly using (5) as in Roeber et al. (2010). Equations (5) and (22) are the governing equations solved in this study. As $\mathbf{V}$ is obtained, the velocity $\mathbf{u}_\alpha$ can be found by solving a system of equation with tridiagonal matrix formed by (23), in which all cross-derivatives are moved to the right-hand side of the equation.

3 Numerical schemes

3.1 Compact form of governing equations

We define

$$\begin{aligned}
&\mathbf{u}_\alpha = (u, v),\\
&\overline{\mathbf{u}}_2 = (U_4, V_4),\\
&\mathbf{M} = (P, Q) = H\left[u + U_4, v + V_4\right],\\
&\mathbf{V}_1' = (U_1', V_1'),\\
&\mathbf{V}_1'' = (U_1'', V_1''),\\
&\mathbf{V}_2 = (U_2, V_2),\\
&\mathbf{V} = (U, V) = H\left[(u + U_1'), (v + V_1')\right].
\end{aligned}$$

The generalized conservative form of Boussinesq equations can be written as

$$\frac{\partial \mathbf{\Psi}}{\partial t} + \nabla \cdot \mathbf{\Theta}(\mathbf{\Psi}) = \mathbf{S} \tag{24}$$

where $\mathbf{\Psi}$ and $\mathbf{\Theta}(\mathbf{\Psi})$ are the vector of conserved variables and the flux vector function, respectively, and are given by

$$\mathbf{\Psi} = \begin{pmatrix} \eta \\ U \\ V \end{pmatrix}, \qquad \mathbf{\Theta} = \begin{pmatrix} P\mathbf{i} + Q\mathbf{j} \\ \left[\frac{P^2}{H} + \frac{1}{2}g(\eta^2 + 2\eta h)\right]\mathbf{i} + \frac{PQ}{H}\mathbf{j} \\ \frac{PQ}{H}\mathbf{i} + \left[\frac{Q^2}{H} + \frac{1}{2}g(\eta^2 + 2\eta h)\right]\mathbf{j} \end{pmatrix}. \tag{25}$$

$$\mathbf{S} = \begin{pmatrix} 0 \\ g\eta\frac{\partial h}{\partial x} + \psi_x + HR_x \\ g\eta\frac{\partial h}{\partial y} + \psi_y + HR_y \end{pmatrix}, \tag{26}$$

where

$$\psi_x = \eta_t(U_1' - U_4) + H\left(uU_{4,x} + vU_{4,y} + U_4u_x + V_4u_y - U_1'' - U_2 - U_3\right) \tag{27}$$

$$\psi_y = \eta_t(V_1' - V_4) + H\left(uV_{4,x} + vV_{4,y} + U_4 v_x + V_4 v_y - U_1'' - V_2 - V_3\right) \tag{28}$$

The expanded forms of $(U_1', V_1'), (U_1'', V_1''), (U_2, V_2), (U_3, V_3)$ and (U_4, V_4) can be found in Appendix A. For the term $\mathbf{R}$, the bottom stress is approximated using a quadratic friction equation. A Smagorinsky (1963)-like subgrid turbulent mixing algorithm is implemented following Chen et al. (1999).

3.2 Spatial discretization

A combined finite-volume and finite-difference method was applied to the spatial discretization. For the flux terms and the first-order derivative terms, a high-order MUSCL-TVD scheme is implemented in the present model. The high-order MUSCL-TVD scheme can be written in a compact form including different orders of accuracy from the second- to the fourth-order, according to Erduran et al. (2005) who modified Yamamoto et al.'s (1995) fourth-order approach. In x-direction, for example, the combined form of the interface construction can be written as follows:

$$\phi^L_{i+1/2} = \phi_i + \frac{1}{4}\left[(1-\kappa_1)\chi(r)\Delta^*\phi_{i-1/2} + (1+\kappa_1)\chi(1/r)\Delta^*\phi_{i+1/2}\right] \tag{29}$$

$$\phi^R_{i-1/2} = \phi_i - \frac{1}{4}\left[(1+\kappa_1)\chi(r)\Delta^*\phi_{i-1/2} + (1-\kappa_1)\chi(1/r)\Delta^*\phi_{i+1/2}\right] \tag{30}$$

where $\phi^L_{i+1/2}$ is the constructed value at the left-hand side of the interface $i + \frac{1}{2}$ and $\phi^R_{i-1/2}$ is the value at the right-hand side of the interface $i - \frac{1}{2}$. The values of $\Delta^*\phi$ are evaluated as follows:

$$\begin{aligned}
&\Delta^*\phi_{i+1/2} = \Delta\phi_{i+1/2} - \kappa_2\Delta^3\bar{\phi}_{i+1/2}/6,\\
&\Delta\phi_{i+1/2} = \phi_{i+1} - \phi_i,\\
&\Delta^3\bar{\phi}_{i+1/2} = \Delta\bar{\phi}_{i+3/2} - 2\Delta\bar{\phi}_{i+1/2} + \Delta\bar{\phi}_{i-1/2},\\
&\Delta\bar{\phi}_{i-1/2} = \text{minmod}(\Delta\phi_{i-1/2}, \Delta\phi_{i+1/2}, \Delta\phi_{i+3/2}),\\
&\Delta\bar{\phi}_{i+1/2} = \text{minmod}(\Delta\phi_{i+1/2}, \Delta\phi_{i+3/2}, \Delta\phi_{i-1/2}),\\
&\Delta\bar{\phi}_{i+3/2} = \text{minmod}(\Delta\phi_{i+3/2}, \Delta\phi_{i-1/2}, \Delta\phi_{i+1/2})
\end{aligned} \tag{31}$$

In (31), minmod represents the Minmod limiter and is given by

$$\text{minmod}(j,k,l) = \text{sign}(j)\max\{0, \min[|j|, 2\text{sign}(j)k, 2\text{sign}(j)l]\}. \tag{32}$$

κ_1 and κ_2 in (29) and (30) are control parameters for orders of the scheme in the compact form. The complete form with $(\kappa_1, \kappa_2) = (1/3, 1)$ is the fourth-order scheme given by Yamamoto et al. (1995). $(\kappa_1, \kappa_2) = (1/3, 0)$ yields a third-order scheme, while the second-order scheme can be retrieved using $(\kappa_1, \kappa_2) = (-1, 0)$.

$\chi(r)$ in (29) and (30) is the limiter function. The original scheme introduced by Yamamoto et al. (1998) uses the Minmod limiter as used in (31). Erduran et al. (2005) found that the use of the van-Leer limiter for the third-order scheme gives more accurate results. Their finding was confirmed by using the present model in the benchmark tests for wave runup conducted by Tehranirad et al. (2011). The van-Leer limiter can be expressed as

$$\chi(r) = \frac{r + |r|}{1 + r} \tag{33}$$

where

$$r = \frac{\Delta^* \phi_{i+1/2}}{\Delta^* \phi_{i-1/2}}. \tag{34}$$

The numerical fluxes are computed using a HLL approximate Riemann solver

$$\mathbf{\Theta}(\mathbf{\Psi}^L, \mathbf{\Psi}^R) = \begin{cases} \mathbf{\Theta}(\mathbf{\Psi}^L) & \text{if} \quad s_L \geq 0 \\ \mathbf{\Theta}^*(\mathbf{\Psi}^L, \mathbf{\Psi}^R) & \text{if} \quad s_L < 0 < s_R \\ \mathbf{\Theta}(\mathbf{\Psi}^R) & \text{if} \quad s_R \leq 0, \end{cases} \tag{35}$$

where

$$\mathbf{\Theta}^*(\mathbf{\Psi}^L, \mathbf{\Psi}^R) = \frac{s_R \mathbf{\Theta}(\mathbf{\Psi}^L) - s_L \mathbf{\Theta}(\mathbf{\Psi}^R) + s_L s_R (\mathbf{\Psi}^R - \mathbf{\Psi}^L)}{s_R - s_L} \tag{36}$$

The wave speeds of the Riemann solver are given by

$$s_L = \min(\mathbf{V}^L \cdot \mathbf{n} - \sqrt{g(h+\eta)^L}, u_s - \sqrt{\varphi_s}), \tag{37}$$

$$s_R = \max(\mathbf{V}^R \cdot \mathbf{n} + \sqrt{g(h+\eta)^R}, u_s + \sqrt{\varphi_s}), \tag{38}$$

in which u_s and φ_s are estimated as

$$u_s = \frac{1}{2}(\mathbf{V}^L + \mathbf{V}^R) \cdot \mathbf{n} + \sqrt{g(\eta+h)^L} - \sqrt{g(\eta+h)^R} \tag{39}$$

$$\sqrt{\varphi_s} = \frac{\sqrt{g(\eta+h)^L} + \sqrt{g(\eta+h)^R}}{2} + \frac{(\mathbf{V}^L - \mathbf{V}^R) \cdot \mathbf{n}}{4} \tag{40}$$

and $\mathbf{n}$ is the normalized side vector for a cell face.

Higher derivative terms in ψ_x and ψ_y were discretized using a central difference scheme at the cell centroids, as in Wei et al. (1995). No discretization of dispersion terms at the cell interfaces is needed due to using $\mathbf{M}$ as a flux variable. The Surface Gradient Method (Zhou et al, 2001) was used to eliminate unphysical oscillations. Because the pressure gradient term is re-organized as in section 2.2, there is no imbalance issue for the high-order MUSCL scheme.

3.3 Time stepping

The third-order Strong Stability-Preserving (SSP) Runge-Kutta scheme for nonlinear spatial discretization (Gottlieb et al., 2001) was adopted for time stepping. The scheme is given by

$$\begin{aligned}
\mathbf{\Psi}^{(1)} &= \mathbf{\Psi}^n + \Delta t(-\nabla\cdot\mathbf{\Theta}(\mathbf{\Psi}^n) + \mathbf{S}^{(1)}) \\
\mathbf{\Psi}^{(2)} &= \frac{3}{4}\mathbf{\Psi}^n + \frac{1}{4}\left[\mathbf{\Psi}^{(1)} + \Delta t\left(-\nabla\cdot\mathbf{\Theta}(\mathbf{\Psi}^{(1)}) + \mathbf{S}^{(2)}\right)\right] \\
\mathbf{\Psi}^{n+1} &= \frac{1}{3}\mathbf{\Psi}^n + \frac{2}{3}\left[\mathbf{\Psi}^{(2)} + \Delta t\left(-\nabla\cdot\mathbf{\Theta}(\mathbf{\Psi}^{(2)}) + \mathbf{S}^{n+1}\right)\right]
\end{aligned} \tag{41}$$

in which $\mathbf{\Psi}^n$ denotes $\mathbf{\Psi}$ at time level n. $\mathbf{\Psi}^{(1)}$ and $\mathbf{\Psi}^{(2)}$ are values at intermediate stages in the Runge-Kutta integration. As $\mathbf{\Psi}$ is obtained at each intermediate step, the velocity (u, v) can be solved by a system of tridiagonal matrix equations formed by (23). $\mathbf{S}$ needs to be updated using (u, v, η) at the corresponding time step and an iteration is needed to achieve convergence.

An adaptive time step is chosen, following the Courant-Friedrichs-Lewy (CFL) criterion:

$$\Delta t = C\min\left(\min\frac{\Delta x}{|u_{i,j}| + \sqrt{g(h_{i,j}+\eta_{i,j})}}, \min\frac{\Delta y}{|v_{i,j}| + \sqrt{g(h_{i,j}+\eta_{i,j})}}\right) \tag{42}$$

where C is the Courant number and $C = 0.5$ was used in the following examples.

3.4 Wave breaking and wetting-drying schemes for shallow water

The wave breaking scheme follows the approach of Tonelli and Petti (2009), who successfully used the ability of NSWE with a TVD scheme to model moving hydraulic jumps. Thus, the fully nonlinear Boussinesq equations are switched to NSWE at cells where the Froude number exceeds a certain threshold. Following Tonelli and Petti, the ratio of wave height to total water depth is chosen as the criterion to switch from Boussinesq to NSWE, with threshold value set to 0.8, as suggested by Tonelli and Petti.

The wetting-drying scheme for modeling a moving boundary is straightforward. The normal flux $\mathbf{n}\cdot\mathbf{M}$ at the cell interface of a dry cell is set to zero. A mirror boundary condition is applied to the fourth-order MUSCL-TVD scheme and discretization of dispersive terms in ψ_x, ψ_y at dry cells. It may be noted that the wave speeds of the Riemann solver (37) and (38) for a dry cell are modified as

$$s_L = \mathbf{V}^L\cdot\mathbf{n} - \sqrt{g(h+\eta)^L}, \quad s_R = \mathbf{V}^L\cdot\mathbf{n} + 2\sqrt{g(h+\eta)^L} \quad \text{(right dry cell)} \tag{43}$$

and

$$s_L = \mathbf{V}^R\cdot\mathbf{n} - \sqrt{g(h+\eta)^R}, \quad s_R = \mathbf{V}^R\cdot\mathbf{n} + 2\sqrt{g(h+\eta)^R} \quad \text{(left dry cell)} \tag{44}$$

3.5 Boundary conditions and wavemaker

We implemented various boundary conditions including wall boundary condition, absorbing boundary condition following Kirby et al. (1998) and periodic boundary condition following Chen et al. (2003).

Wavemakers implemented in this study include Wei and Kirby's (1999) internal wavemakers for regular waves and irregular waves. For the irregular wavemaker, an extension was made to incorporate an alongshore periodicity into wave generation, in order to eliminate a boundary effect on wave simulations. The technique exactly follows the strategy in Chen et al. (2003), who adjusted the distribution of wave directions in each frequency bin to obtain alongshore periodicity. This approach is effective in modeling of breaking wave-induced nearshore circulation such as alongshore currents and rip currents.

3.6 Parallelization

In parallelizing the computational model, we used a domain decomposition technique to subdivide the problem into multiple regions and assign each subdomain to a separate processor core. Each subdomain region contains an overlapping area of ghost cells, three-row deep, as required by the fourth order MUSCL-TVD scheme. The Message Passing Interface (MPI) with non-blocking communication is used to exchange data in the overlapping region between neighboring processors. Velocity components are obtained from Equation (23), by solving tridiagonal matrices using the parallel pipelining tridiagonal solver described in Naik et al. (1993).

To investigate performance of the parallel program, numerical simulations of an idealized case are tested with different numbers of processors on a Linux cluster located at University of Delaware. The test case is set up in a numerical grid of 1800 $\times$ 1800 cells. Figure 1 shows the model speedup versus number of processors. It can be seen that performance scales nearly proportional to the number of processors, with some delay caused by inefficiencies in parallelization, such as inter-processor communication time.

3.7 Implementation of weakly nonlinear Boussinesq equations in spherical coordinates

The weakly nonlinear Boussinesq equations in spherical coordinates are solved in the same model framework. We used the spherical Boussinesq equations derived by Kirby et al. (2004, 2011):

$$
\begin{aligned}
&H_t + \frac{1}{r_0 \cos\theta}\left\{(Hu)_\phi + (Hv\cos\theta)_\theta\right\} = 0 \qquad (45)\\
&u_t - fv + \frac{1}{r_0\cos\theta}uu_\phi + \frac{1}{r_0}vu_\theta \; + \frac{g}{r_0\cos\theta}\eta_\phi\\
&+\frac{1}{r_0^2\cos^2\theta}\left\{\frac{h^2}{6}\left[u_{\phi\phi t} + (v\cos\theta)_{\phi\theta t}\right] - \frac{h}{2}\left[(hu_t)_{\phi\phi} + (h\cos\theta v_t)_{\phi\theta}\right]\right\}
\end{aligned}
$$

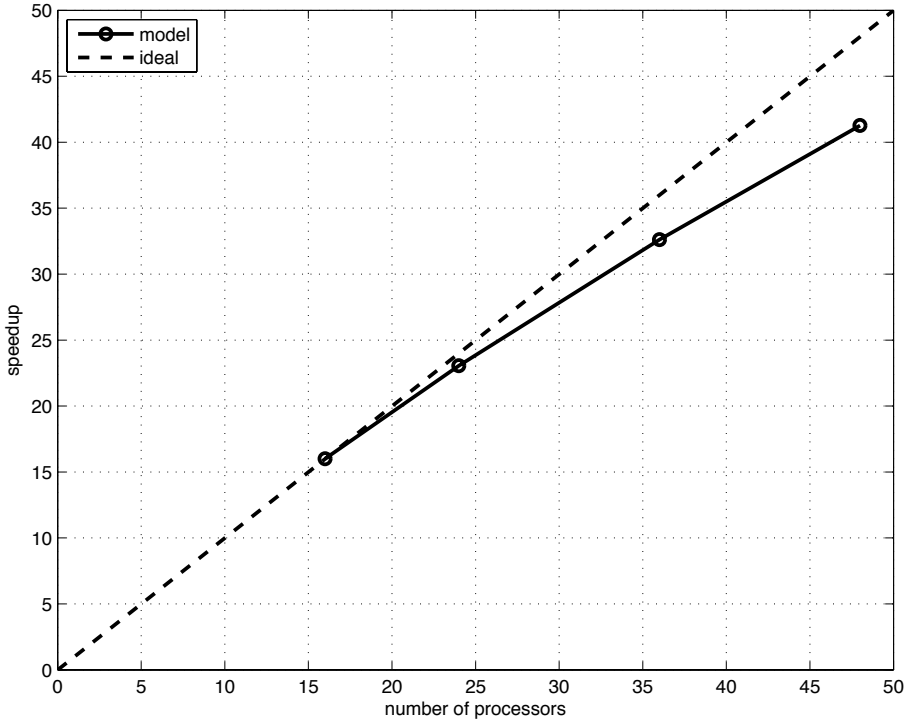

Figure 1: Variation in model performance with number of processors for a 1800 x 1800 domain. Straight line indicates arithmetic speedup. Actual performance is shown in the curved line.

$$-\tau_b^{\phi} + \frac{1}{r_0 \cos\theta}(BFT)_{\phi} = 0 \tag{46}$$

$$\begin{aligned} &v_t + fu + \frac{1}{r_0 \cos\theta} uv_{\phi} + \frac{1}{r_0} vv_{\theta} + \frac{g}{r_0}\eta_{\theta} \\ &+\frac{1}{r_0^2}\left\{ \frac{h^2}{6}\left[\frac{1}{\cos\theta}\{u_{\phi t} + (v\cos\theta)_{\theta t}\}\right]_{\theta} - \frac{h}{2}\left[\frac{1}{\cos\theta}\{(hu_t)_{\phi} + (h\cos\theta v_t)_{\theta}\}\right]_{\theta}\right\} \\ &-\tau_b^{\theta} + \frac{1}{r_0}(BFT)_{\theta} = 0 \end{aligned} \tag{47}$$

where θ and ϕ denote latitude and longitude, respectively, r_0 is the earth radius, f is the Coriolis parameter, $H = h + \eta$, (u, v) represent the depth-averaged velocity. BFT denotes forcing terms resulting from motion of the ocean bottom and it is not taken into account in the present program.

To facilitate solving the spherical equations in the same model framework as in the Cartesian coordinates, we solve the spherical governing equations by transforming the equations into an equivalent set of equations in Cartesian coordinates using a standard cylindrical projection. We define

$$\begin{cases} \xi_1 = & r_0 \cos\theta_0 (\phi - \phi_0) \\ \xi_2 = & r_0(\theta - \theta_0) \end{cases} \tag{48}$$

where (ϕ_0, θ_0) are the reference longitude and latitude, respectively. The differentials of ξ_1 and ξ_2 are

$$\begin{cases} d\xi_1 = & r_0 \cos\theta_0 d\phi \\ d\xi_2 = & r_0 d\theta \end{cases} \tag{49}$$

$d\phi$ and $d\theta$ in (45) - (47) are replaced by $d\xi_1$ and $d\xi_2$. Detailed derivations for each term are described below.

$$\frac{1}{r_0 \cos\theta}(Hu)\phi = S_p (Hu)_{\xi_1} \tag{50}$$

$$\frac{1}{r_0 \cos\theta}(Hv\cos\theta)_{\theta} = \frac{1}{r_0 \cos\theta}\left[\cos\theta (Hv)_{\theta} - Hv\sin\theta\right] = (Hv)_{\xi_2} - \frac{1}{r_0}\tan\theta Hv \tag{51}$$

$$\frac{1}{r_0 \cos\theta} uu_{\phi} = S_p uu_{\xi_1} \tag{52}$$

$$\frac{1}{r_0} vu_{\theta} = vu_{\xi_2} \tag{53}$$

$$\frac{1}{r_0 \cos\theta} g\eta_{\phi} = S_p g\eta_{\xi_1} \tag{54}$$

$$\frac{h^2}{6}\frac{1}{r_0^2 \cos^2\theta} u_{\phi\phi t} = S_p^2 \frac{h^2}{6} u_{\xi_1 \xi_1 t} \tag{55}$$

$$\frac{h^2}{6}\frac{1}{r_0^2\cos^2\theta}(v\cos\theta)_{\phi\theta t} = \frac{h^2 S_p}{6}\left(v_{\xi_1\xi_2} - \frac{1}{r_0}\tan\theta v_{\xi_1}\right)_t \tag{56}$$

$$-\frac{h}{2}\frac{1}{r_0^2\cos^2\theta}(hu)_{\phi\phi t} = -\frac{hS_p^2}{2}(hu)_{\xi_1\xi_1 t} \tag{57}$$

$$-\frac{h}{2}\frac{1}{r_0^2\cos^2\theta}(hv\cos\theta)_{\phi\theta t} = -\frac{hS_p}{2}\left[(hv)_{\xi_1\xi_2} - \frac{1}{r_0}\tan\theta(hv)_{\xi_1}\right]_t \tag{58}$$

$$\frac{1}{r_0\cos\theta}uv_\phi = S_p uv_{\xi_1} \tag{59}$$

$$\frac{1}{r_0}vv_\theta = vv_{\xi_2} \tag{60}$$

$$\frac{1}{r_0}g\eta_\theta = g\eta_{\xi_2} \tag{61}$$

$$\frac{h^2}{6}\frac{1}{r_0^2}\left(\frac{u_{\phi t}}{\cos\theta}\right)_\theta = \frac{h^2 S_p}{6}\left(u_{\xi_1\xi_2} + \frac{1}{r_0}\tan\theta u_{\xi_1}\right)_t \tag{62}$$

$$-\frac{h}{2}\frac{1}{r_0^2}\left[\frac{(hu)_{\phi t}}{\cos\theta}\right]_\theta = -\frac{hS_p}{2}\left[(hu)_{\xi_1\xi_2} + \frac{1}{r_0}\tan\theta(hu)_{\xi_1}\right]_t \tag{63}$$

$$\frac{h^2}{6}\frac{1}{r_0^2}\left[\frac{(v\cos\theta)_{\theta t}}{\cos\theta}\right]_\theta = \frac{h^2}{6}\left[v_{\xi_2\xi_2} - \frac{1}{r_0}\tan\theta v_{\xi_2} - \frac{1}{r_0^2\cos^2\theta}v\right]_t \tag{64}$$

$$-\frac{h}{2}\frac{1}{r_0^2}\left[\frac{(hv\cos\theta)_{\theta t}}{\cos\theta}\right]_\theta = -\frac{h}{2}\left[(hv)_{\xi_2\xi_2} - \frac{1}{r_0}\tan\theta(hv)_{\xi_2} - \frac{1}{r_0^2\cos^2\theta}hv\right]_t \tag{65}$$

where S_p is a spherical coordinate correction factor expressed by

$$S_p = \frac{\cos\theta_0}{\cos\theta}. \tag{66}$$

The equations in (ξ_1, ξ_2) coordinates are

$$H_t + S_p(Hu)_{\xi_1} + (Hv)_{\xi_2} = \frac{1}{r_0}\tan\theta Hv \tag{67}$$

$$u_t + S_p uu_{\xi_1} + vu_{\xi_2} - fv + S_p g\eta_{\xi_1} + F_t - \tau_b^{\xi_1} + S_p(BFT)_{\xi_1} = 0 \tag{68}$$

$$v_t + S_p uv_{\xi_1} + vv_{\xi_2} + fu + g\eta_{\xi_2} + G_t - \tau_b^{\xi_2} + (BFT)_{\xi_2} \tag{69}$$

where

$$F = \frac{h^2 S_p^2}{6}u_{\xi_1\xi_1} + \frac{h^2 S_p}{6}(v_{\xi_1\xi_2} - \frac{1}{r_0}\tan\theta v_{\xi_1}) - \frac{hS_p^2}{2}(hu)_{\xi_1\xi_1} - \frac{hS_p}{2}\left[(hv)_{\xi_1\xi_2} - \frac{1}{r_0}\tan\theta(hv)_{\xi_1}\right] \tag{70}$$

$$G = \frac{h^2 S_p}{6}\left(u_{\xi_1\xi_2} + \frac{1}{r_0}\tan\theta u_{\xi_1}\right) + \frac{h^2}{6}\left(v_{\xi_2\xi_2} - \frac{1}{r_0}\tan\theta v_{\xi_2} - \frac{1}{r_0^2\cos^2\theta}v\right)$$
$$-\frac{hS_p}{2}\left[(hu)_{\xi_1\xi_2} + \frac{1}{r_0}\tan\theta(hu)_{\xi_1}\right] - \frac{h}{2}\left[(hv)_{\xi_2\xi_2} - \frac{1}{r_0}\tan\theta(hv)_{\xi_2} - \frac{1}{r_0^2\cos^2\theta}(hv)\right] \quad (71)$$

The conservation forms of the spherical equations can be rewritten in the same form of the Cartesian equations (24) and $\mathbf{\Psi}$,$\mathbf{\Theta}$ and $\mathbf{S}$ are defined as

$$\mathbf{\Psi} = \begin{pmatrix} \eta \\ U_s \\ V_s \end{pmatrix}, \qquad \mathbf{\Theta} = \begin{pmatrix} S_p P\mathbf{i} + Q\mathbf{j} \\ \left[\frac{S_p P^2}{h+\eta} + \frac{1}{2}S_p g(\eta^2 + 2\eta h)\right]\mathbf{i} + \frac{PQ}{h+\eta}\mathbf{j} \\ \frac{S_p PQ}{h+\eta}\mathbf{i} + \left[\frac{Q^2}{h+\eta} + \frac{1}{2}g(\eta^2 + 2\eta h)\right]\mathbf{j} \end{pmatrix}. \quad (72)$$

$$\mathbf{S} = \begin{pmatrix} \frac{1}{r_0}\tan\theta(h+\eta)v \\ S_p g\eta\frac{\partial h}{\partial \xi_1} + f(h+\eta)v + \tau_b^{\xi_1} + \psi_1 \\ g\eta\frac{\partial h}{\partial \xi_2} - f(h+\eta)u + \tau_b^{\xi_2} + \psi_2 \end{pmatrix}, \quad (73)$$

where $P = (h+\eta)u$, $Q = (h+\eta)v$,

$$U_s = (h+\eta)(u+F) \quad (74)$$

$$V_s = (h+\eta)(v+G) \quad (75)$$

$$\psi_1 = \eta_t F \quad (76)$$

$$\psi_2 = \eta_t G \quad (77)$$

4 Users' Manual

4.1 Program outline and flow chart

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 FUNWAVE-TVD includes a number of options including (1) choice of serial or parallel code (2) Cartesian or spherical coordinate (Tsunami propagation mode), (3) samples, (4) one-way nesting mode, and (5) wave breaking index and aging (bubble and foam mode).

The flow chart is shown in Figure 2.

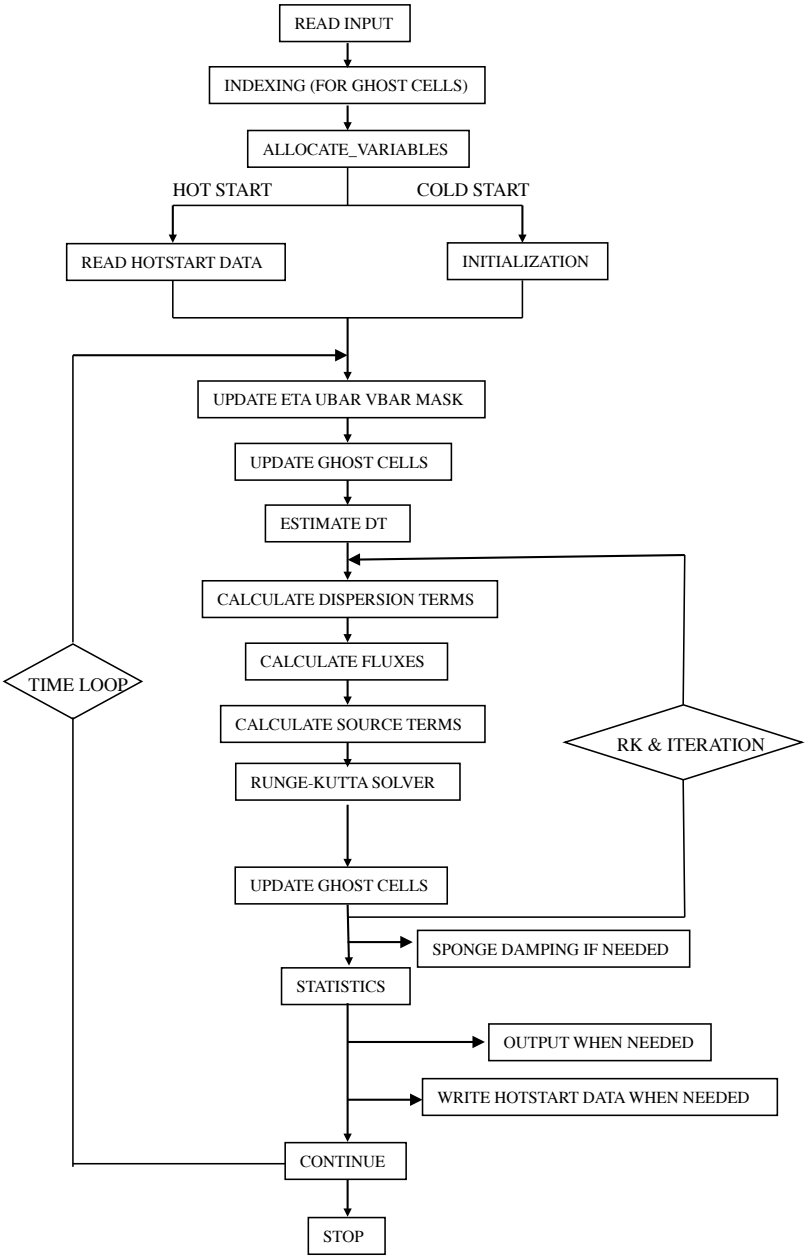

Figure 2: Flow chart of the main program.

4.2 Subroutine and function descriptions

ALLOCATE_VARIABLES: allocate variables. It is called by MAIN.

BREAKING: print breaking index and breaking age calculated based on Kennedy et al. (2000). This option is defined in input.txt.

BOUNDARY_CONDITION: provide boundary conditions at four side boudaries. It is called by FLUXES.

CAL_DISPERSION: calculate dispersion terms. The first and second derivatives with respect to x, y are also calculated in this subroutine. In addition, the dispersion values in ghost cells are updated in this subroutine. It is called by the main program.

CALCULATE_Cm_Sm: calculate C_m and S_m used in Wei and Kirby's internal wave maker for irregular waves (TMA). Detailed formulation can be found in Shi et al. (2003). The subroutine is called by INITIALIZATION.

CALCULATE_MEAN: calculate mean u v required by the smagorinsky mixing. Mean η is also calculated. Called by MAIN.

CALCULATE_SPONGE: setup sponge layer. It is called by INITIALIZATION.

CONSTRUCTION: second- and third-order interface construction. It is called by FLUXES. It calls CONSTRUCT_X and CONSTRUCT_Y.

CONSTRUCTION_HO: high order interface construction. It is called by FLUXES. It calls CONSTRUCT_HO_X and CONSTRUCT_HO_Y.

CONSTRUCT_HO_X: high order interface construction of specific variables in x direction. It is called by CONSTRUCTION_HO.

CONSTRUCT_HO_Y: high order interface construction of specific variables in y direction. It is called by CONSTRUCTION_HO.

CONSTRUCT_X: construct variables in x direction. It is called by CONSTRUCTION.

CONSTRUCT_Y: construct variables in y direction. It is called by CONSTRUCTION.

CORRECTOR: corrector time scheme. It was used for a comparison between Predictor-corrector scheme and Runge-Kutta. It is not suggested using it in the present program.

DelxFun: calculate derivatives with respect to x. It is called by DelxFun.

DelxyFun: calculate derivatives with respect to x and y. It is called by FLUXES when the lower order construction is applied. It calls DelxFun and DelyFun.

DelyFun: calculate derivatives with respect to y. It is called by DelxyFun.

DERIVATIVE_X: calculate first-derivative respect to x in second order. The subroutine is called by CAL_DISPERSION.

DERIVATIVE_X_High: calculate first-derivative respect to x in higher-order. The subroutine is called by CAL_DISPERSION.

DERIVATIVE_XX: calculate second-derivative respect to x. The subroutine is called by CAL_DISPERSION.

DERIVATIVE_XY: calculate cross-derivative. The subroutine is called by CAL_DISPERSION.

DERIVATIVE_Y: calculate first-derivative respect to y in second order. The subroutine is called by CAL_DISPERSION.

DERIVATIVE_Y_High: calculate first-derivative respect to y in higher-order. The subroutine is called by CAL_DISPERSION.

DERIVATIVE_YY: calculate second-derivative respect to y. The subroutine is called by CAL_DISPERSION.

ESTIMATE_DT: evaluate dt based on CFL criteria. It is called by MAIN.

ESTIMATE_HUV: Runge-Kutta solver. It is called by MAIN. It calls GET_Eta_U_V_HU_HV.

EXCHANGE: update η, u, v, Hu, Hv and MASK in the ghost cells. It is called by MAIN. It calls PHI_COLL.

EXCHANGE_DISPERSION: update dispersion variables in the ghost cells. It is called by MAIN. It calls PHI_COLL.

FLUX_AT_INTERFACE: calculate numerical fluxes at four cell interfaces using the averaging method. It is called by FLUXES.

FLUX_AT_INTERFACE_HLLC: calculate numerical fluxes at four cell interfaces using HLLC scheme. It is called by FLUXES. It calls HLLC.

FLUXES: calculate numerical fluxes. It is called by MAIN. It calls CONSTRUCTION lower order or CONSTRUCTION_HO for higher order, WAVE_SPEED, DelxyFun (for lower order), FLUX_AT_INTERFACE_HLLC for HLL scheme or FLUX_AT_INTERFACE for averaging scheme, and BOUNDARY_CONDITION.

GET_Eta_U_V_HU_HV: calculate η, u, v, Hu and Hv. It is called by ESTIMATE_HUV. The tridiagonal solver is used to get u and v. Froude number cap is applied in this subroutine. It calls TRIG (wall boundary) or TRIG_PERIODIC (periodic boundary). In the parallel mode, it calls TRIDx and TRIDy.

GetFile: read data in the global dimension and distribute to all processors in the parallel mode. Use flag '-DPARALLEL' in Makefile to include this subroutine.

GatherVariable: gather variables from all processors to processor 0 in the parallel mode but with serial tridiagonal solver. Use flag '-DTRID_NO_PARALLEL' to include this subroutine. It is called by TRIDxyNoParallel.

HLLC: HLLC scheme. It is called by FLUX_AT_INTERFACE_HLLC.

INDEX: indexing for ghost cells and MPI subdomains. It is called by MAIN.

INITIALIZATION: initialization subroutine. It is called MAIN. It may call WK_WAVEMAKER_REGULAR_WAVE, WK_WAVEMAKER_IRREGULAR_WAVE, CALCULATE_Cm_Sm, CALCULATE_SPONGE if applied.

INITIAL_GAUSIAN: sample to provide initial Gausian hump. Use flag '-DSAMPLE' in Makefile to include this subroutine. Parameters are defined in input.txt. The subroutine is called by INITIALIZATION.

INITIAL_HOTSTART: initialization when hotstart option is applied. It is called by MAIN. It may call INITIAL_UVZ, INITIAL_SOLITARY_WAVE, INITIAL_N_WAVE, INITIAL_RECTANGULAR, INITIAL_WAVE, WK_WAVEMAKER_REGULAR_WAVE, WK_WAVEMAKER_IRREGULAR_WAVE, CALCULATE_Cm_Sm, CALCULATE_SPONGE if applied.

INITIAL_N_WAVE: sample to provide initial N wave solution. Use flag '-DSAMPLE' in Makefile to include this subroutine. Parameters are defined in input.txt. The subroutine is called by INITIALIZATION.

INITIAL_SOLITARY_WAVE: sample to provide initial solitary wave solution. Use flag '-DSAMPLE' in Makefile to include this subroutine. Parameters are defined in input.txt. The subroutine is called by INITIALIZATION. It calls SUB_SLTRY to get parameters.

INITIAL_RECTANGULAR: sample of given an initial rectangular hump. Use flag '-DSAMPLE' in Makefile to include this subroutine. Parameters are defined in input.txt. The subroutine is called by INITIALIZATION.

INITIAL_UVZ: read initial u, v, and η data from files defined in input.txt. It is called by INITIALIZATION. It calls GetFile.

MINMOD_LIMITER: function of minmod limiter. It is used in CONSTRUCT_HO_X, and CONSTRUCT_HO_Y.

MPI_INITIAL: initialize MPI environment.

OneWayCoupling: one-way nesting subroutine. The nesting data read from INITIALIZATION.

PHI_EXCH: handle float type data exchange between processors in the parallel mode. It is called by PHI_COLL. Use flag '-DPARALLEL' to include this subroutine.

PHI_INT_EXCH: handle integer type data exchange between processors in the parallel mode. It is called by UPDATE_MASK. Use flag '-DPARALLEL' to include this subroutine.

PREDICTOR: predictor scheme. It was used for a comparison between Predictor-corrector scheme and Runge-Kutta. It is not suggested using it in the present program.

PREVIEW: output subroutine. It is called by MAIN. It calls PutFile.

PutFile: print out output in files. In the parallel mode, it gathers data from all processors into processor 0 and prints out in the global dimension. Use flag '-DPARALLEL' in Makefile for the parallel mode. It is called by PREVIEW.

PHI_COLL: update data in ghost cells. It is called by EXCHANGE. in the parallel mode, it calls PHI_EXCH which is a major subroutine to handle data exchange between processors.

READ_INPUT: read input.txt. It is called by MAIN. It calls GET_STRING_VAL, GET_LOGICAL_VAL, GET_INTEGER_VAL, GET_Float_VAL. Input data are written out in LOG.txt.

READ_HOTSTART_DATA: read saved data for hot start and initialize other variables.

ScatterVariable: scatter variables from processor 0 to all processors in the parallel mode but with serial tridiagonal solver. Use flag '-DTRID_NO_PARALLEL' to include this subroutine. It is called by TRIDxyNoParallel.

SOLITARY_WAVE_LEFT_BOUNDARY: nudging boundary condition of solitary wave at left boundary. Use flag '-DSAMPLE' in Makefile to include this subroutine. Parameters are defined in input.txt.

SourceTerms: calculate all source terms including slope term and dispersion terms. It is called by MAIN.

STATISTICS: calculate statistics of total mass volume, energy, maximum and minimum η, u, v, Froude number etc.

SPONGE_DAMPING: use sponge layers to damp waves Parameters are defined in input.txt.

SUB_SLTRY: provide solitary wave solution of Nogu's equations. It is called by INITIAL_SOLITARY_WAVE. Use flag '-DSAMPLE' in Makefile to include this subroutine.

TRI_GE: tridiagonal solver. It allows diagonal variables not equal to unit. It is called by TRIG_PERIODIC.

TRIG: tridiagonal solver. It is called by GET_Eta_U_V_HU_HV. It calls TRI_GE.

TRIDxyNoParallel: no parallel tridiagonal solver but in the parallel mode. It is only for testing. Use flag '-DTRID_NO_PARALLEL' to include this subroutine. It calls GatherVariable, TRIG and ScatterVariable.

TRIG_PERIODIC: cyclic tridiagonal solver. It is called by GET_Eta_U_V_HU_HV. It calls TRI_GE.

UPDATE_MASK: update mask according to wetting and drying. Also update mask and mask9 in ghost cells.

VANLEER_LIMITER: function of Vanleer limiter. It is used in DelxFun and DelyFun.

WAVE_SPEED: calculate numerical wave speed needed by the TVD scheme. It is called by MAIN.

WK_WAVEMAKER_IRREGULAR_WAVE: calculate source function for Wei and Kirby's internal wave maker for irregular waves (TMA). Periodic boundary conditions are included. Parameters are defined in input.txt. The subroutine is called by INITIALIZATION.

WK_WAVEMAKER_REGULAR_WAVE: calculate source function for Wei and Kirby's internal wave maker for regular waves. Periodic boundary conditions are included but for certain wave angles (will ask during a run). Parameters are defined in input.txt. The subroutine is called by **INITIALIZATION**.

WRITE_HOTSTART_DATA: write out data used for hot start.

4.3 Permanent variables

Depth(): still water depth h at element point

DepthNode(): still water depth h at node

DepthX(): still water depth h at x-interface

DepthY(): still water depth h at y-interface

Eta(): surface elevation, for dry point, Eta() = MinDepth - Depth(), MinDepth is specified in input.txt.

Eta0(): η at previous time level

MASK(): 1 - wet, 0 - dry

MASK_STRUC(): 0 - permanent dry point

MASK9: mask to switch from Boussinesq equation to SWE, 1 - Boussinesq, 0 - SWE

U(): depth-averaged u or u at the reference level (u_α) at element

V(): depth-averaged v or v at the reference level (v_α) at element

HU(): $(h+\eta)u$ at element

HV(): $(h+\eta)v$ at element

P(): $(h+\eta)(u+U_4)$ at x-interface

Q(): $(h+\eta)(v+V_4)$ at y-interface

Fx(): numerical flux F at x-interface

Fy(): numerical flux F at y-interface

Gx(): numerical flux G at x-interface

Gy(): numerical flux G at y-interface

Ubar(): U

Vbar(): V

U4(): x-component of U_4

V4(): y-component of V_4

U1p(): x-component of U_1'

V1p(): y-component of V_1'

EtaRxL(): η Left value at x-interface

EtaRxR(): η Right value at x-interface

EtaRyL(): η Left value at y-interface

EtaRyR(): η Right value at y-interface

HxL(): total depth Left value at x-interface

HxR(): total depth Right value at x-interface

HyL(): total depth Left value at y-interface

HyR(): total depth Right value at y-interface

HUxL(): $(h+\eta)u$ Left value at x-interface

HUxR(): $(h+\eta)u$ Right value at x-interface

HVyL(): $(h+\eta)v$ Left value at y-interface

HVyR(): $(h+\eta)v$ Right value at y-interface

PL(): $(h+\eta)(u+U_4)$, Left value at x-interface

PR(): $(h+\eta)(u+U_4)$, Right value at x-interface

QL(): $(h+\eta)(v+V_4)$, Left value at y-interface

QR(): $(h+\eta)(v+V_4)$, Right value at y-interface

FxL = HUxL*UxL + 1/2*g*(EtaRxL^2 + 2*EtaRxL*Depthx)

FxR = HUxR*UxR + 1/2*g*(EtaRxR^2 + 2*EtaRxR*Depthx)

FyL = HyL*UyL*VyL

FyR = HyR*UyR*VyR

GxL = HxL*UxL*VxL

GxR = HxR*UxR*VxR

GyL = HVyL*VyL + 1/2*g*(EtaRyL^2 + 2*EtaRyL*Depthy)

GyR = HVyR*VyR + 1/2*g*(EtaRyR^2 + 2*EtaRyR*Depthy)

4.4 Installation and compilation

FUNWAVE-TVD is distributed in a compressed fie. To install the programs, first, uncompress the package. Then use

$>$ tar xvf *.tar

to extract files from the uncompressed package. The exacted files will be distributed in two new directories: /src and /work.

To compile the program, go to /src and modify Makefile if needed. There are several necessary flags in Makefile needed to specify below.

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

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

-DSAMPLES: include all samples, default is no sample included.

-DCARTESIAN: Cartesian version, otherwise Spherical version

-DINTEL: if INTEL compiler is used, this option can make use of FPORT for the RAND() function

-DMIXING: include Smagorinsky mixing.

-DCOUPLING: nesting mode.

CPP: path to CPP directory.

FC: Fortran compiler.

Then execute

> make

The executable file ''mytvd' will be generated and copied from /src to /work/. Note: use 'make clean' after modifying Makefile.

To run the model, go to /work. Modify input.txt if needed and run.

4.5 Input

Following are descriptions of parameters in input.txt (**NOTE:** all parameter names are capital sensitive).

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

SPECIFICATION OF HOT START

HOT_START: logical parameter, T for hot start, F for cold start.

FileNumber_HOTSTART: number of hotstart file used for a hot start, e.g., 1,2, ...

SPECIFICATION OF MULTI-PROCESSORS

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)

SPECIFICATION OF WATER DEPTH

DEPTH_TYPE: depth input type.

DEPTH_TYPE=DATA: from a depth file.

The program includes several simple bathymetry configurations such as

DEPTH_TYPE=FLAT: flat bottom, need DEPTH_FLAT

DEPTH_TYPE=SLOPE: plane beach along x direction. It needs three parameters: slope,SLP, slope starting point, Xslp and flat part of depth, DEPTH_FLAT

DEPTH_FILE: bathymetry file if DEPTH_TYPE=DATA, file dimension should be Mglob x Nglob with the first point as the south-west corner. The read format in the code is shown below.

DO J=1,Nglob

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

ENDDO

DEPTH_FLAT: water depth of flat bottom if DEPTH_TYPE=FLAT or DEPTH_TYPE=SLOPE (flat part of a plane beach).

SLP: slope if DEPTH_TYPE=SLOPE

Xslp: starting x (m) of a slope, if DEPTH_TYPE=SLOPE

SPECIFICATION OF RESULT FOLDER

RESULT_FOLDER: result folder name, e.g., RESULT_FOLDER = /Users/fengyanshi/tmp/

SPECIFICATION OF DIMENSION

Mglob: global dimension in x direction.

Nglob: global dimension in y direction.

SPECIFICATION OF TIME

TOTAL_TIME: simulation time in seconds

PLOT_INTV: output interval in seconds (Note, output time is not exact because adaptive dt is used.)

SCREEN_INTV: time interval (s) of screen print.

PLOT_INTV_STATION: time interval (s) of gauge output

HOTSTART_INTV: time interval (s) to save hot start data.

SPECIFICATION OF GRID SIZE

DX: grid size(m) in x direction.

DY: grid size(m) in y direction.

SPECIFICATION OF INITIAL CONDITION

INT_UVZ : logical parameter for initial condition, default is FALSE

ETA_FILE: name of file for initial η, e.g., ETA_FILE= /Users/fengyanshi/work/input/CVV_H.grd, data format is the same as depth data.

U_FILE: name of file for initial u, e.g.,U_FILE= /Users/fengyanshi/work/input/CVV_U.grd, data format is the same as depth data.

V_FILE: name of file for initial v, e.g., V_FILE= /Users/fengyanshi/work/input/CVV_V.grd, data format is the same as depth data.

SPECIFICATION OF WAVEMAKER

WAVEMAKER: wavemaker type.

WAVEMAKER = INI_REC: initial rectangular hump, need Xc,Yc and WID

WAVEMAKER = LEF_SOL: left boundary solitary, need AMP,DEP, and LAGTIME

WAVEMAKER = INI_SOL: initial solitary wave, WKN B solution, need AMP, DEP, and XWAVEMAKER

WAVEMAKER = INI_OTH: other initial distribution specified by users

WAVEMAKER = WK_REG: Wei and Kirby 1999 internal wave maker, need Xc_WK, Tperiod, AMP_WK, DEP_WK, Theta_WK, and Time_ramp (factor of period)

WAVEMAKER = WK_IRR: Wei and Kirby 1999 TMA spectrum wavemaker, need Xc_WK, DEP_WK, Time_ramp, Delta_WK, FreqPeak, FreqMin,FreqMax, Hmo, GammaTMA, ThetaPeak

WAVEMAKER = WK_TIME_SERIES: *fft* a time series to get each wave component and then use Wei and Kirby's (1999) wavemaker. Need input WaveCompFile (including 3 columns: per,amp,pha) and NumWaveComp,PeakPeriod,DEP_WK, Xc_WK,Ywidth_WK

WAVEMAKER = GAUSIAN: initial Gausian hump, need AMP, Xc, Yc, and WID.

AMP: amplitude (m) of initial η, if WAVEMAKER = INI_REC, WAVEMAKER = INI_SOL, WAVEMAKER = LEF_SOL.

DEP: water depth at wavemaker location, if WAVEMAKER = INI_SOL, WAVEMAKER = LEF_SOL.

LAGTIME, time lag (s) for the solitary wave generated on the left boundary, e.g., WAVEMAKER = LEF_SOL.

XWAVEMAKER: x (m) coordinate for WAVEMAKER = INI_SOL.

Xc: x (m) coordinate of the center of a rectangular hump if WAVEMAKER = INI_REC.

Yc: y (m) coordinate of the center of a rectangular hump if WAVEMAKER = INI_REC.

WID: width (m) of a rectangular hump if WAVEMAKER = INI_REC, or INI_GAU.

Time_ramp: time ramp (s) for Wei and Kirby (1999) wavemaker.

Delta_WK: width parameter δ for Wei and Kirby (1999) wavemaker. $\delta = 0.3 \sim 0.6$

DEP_WK: water depth (m) for Wei and Kirby (1999) wavemaker.

Xc_WK: x coordinate (m) for Wei and Kirby (1999) wavemaker.

Ywidth_WK: width (m) in y direction for Wei and Kirby (1999) wavemaker.

Tperiod: period (s) of regular wave for Wei and Kirby (1999) wavemaker.

AMP_WK: amplitude (m) of regular wave for Wei and Kirby (1999) wavemaker.

Theta_WK: direction (degrees) of regular wave for Wei and Kirby (1999) wavemaker. Note: it may be adjusted for a periodic boundary case by the program. A warning will be given if adjustment is made.

FreqPeak: peak frequency (1/s) for Wei and Kirby (1999) irregular wavemaker.

FreqMin: low frequency cutoff (1/s) for Wei and Kirby (1999) irregular wavemaker.

FreqMax: high frequency cutoff (1/s) for Wei and Kirby (1999) irregular wavemaker.

Hmo: Hmo (m) for Wei and Kirby (1999) irregular wavemaker.

GammaTMA, TMA parameter γ for Wei and Kirby (1999) irregular wavemaker.

ThetaPeak: peak direction (degrees) for Wei and Kirby (1999) irregular wavemaker.

Sigma_Theta: parameter of directional spectrum for Wei and Kirby (1999) irregular wavemaker.

SPECIFICATION OF PERIODIC BOUNDARY CONDITION

(Note: only south-north periodic condition was implemented)

PERIODIC: logical parameter for periodic boundary condition, T - periodic, F - wall boundary condition.

SPECIFICATION OF SPONGE LAYER

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

R_sponge: decay rate in sponge layer. Its values are between 0.85 $\sim$ 0.95.

A_sponge: maximum damping magnitude. The value is $\sim$ 5.0.

SPECIFICATION OF OBSTACLES

OBSTACLE_FILE: name of obstacle file. 1 - water point, 0 - permanent dry point. Data dimension is (Mglob $\times$ Nglob). Data format is the same as the depth data.

SPECIFICATION OF PHYSICS

DISPERSION: logical parameter for inclusion of dispersion terms. T - calculate dispersion, F - no dispersion terms

Gamma1: parameter for linear dispersive terms. 1.0 - inclusion of linear dispersive terms, 0.0 - no linear dispersive terms.

Gamma2: parameter for nonlinear dispersive terms. 1.0 - inclusion of nonlinear dispersive terms, 0.0 - no nonlinear dispersive terms.

Gamma1=1.0, Gamma2=0.0 for NG's equations. Gamma1=1.0, Gamma2=1.0 for the fully nonlinear Boussinesq equations.

Gamma3: parameter for linear shallow water equations (Gamma3 = 1.0). When Gamma3 = 0.0, Gamma1 and Gamma2 automatically become zero.

Beta_ref: parameter β defined for the reference level. β = -0.531 for NG's and FUNWAVE equations.

SWE_ETA_DEP: ratio of height/depth for switching from Boussinesq to NSWE. The value is $\sim$ 0.80.

Cd: bottom friction coefficient.

SPECIFICATION OF NUMERICS

Time_Scheme: stepping option, Runge_Kutta or Predictor_Corrector (not suggested for this version).

HIGH_ORDER: spatial scheme option, FOURTH for the fourth-order, THIRD for the third-order, and SECOND for the second-order (not suggested for Boussinesq modeling).

CONSTRUCTION: construction method, HLL for HLL scheme, otherwise for averaging scheme.

CFL: CFL number, CFL $\sim$ 0.5.

FroudeCap: cap for Froude number in velocity calculation for efficiency. The value could be 5 $\sim$ 10.0.

MinDepth: minimum water depth (m) for wetting and drying scheme. Suggestion: MinDepth = 0.001 for lab scale and 0.01 for field scale.

MinDepthFrc: minimum water depth (m) to limit bottom friction value. Suggestion: MinDepthFrc = 0.01 for lab scale and 0.1 for field scale.

SHOW_BREAKING: logical parameter to calculate breaking index. Note that breaking is calculated using shock wave capturing scheme. The index calculated here is based on Kennedy et al. (2000).

Cbrk1: parameter C1 in Kennedy et al. (2000).

Cbrk2: parameter C2 in Kennedy et al. (2000).

SPECIFICATION OF OUTPUT VARIABLES

NumberStations: number of station for output. If NumberStations > 0, need input i,j in STATION_FILE

DEPTH_OUT: logical parameter for output depth. T or F.

U: logical parameter for output u. T or F.

V: logical parameter for output v. T or F.

ETA: logical parameter for output η. T or F.

MASK: logical parameter for output wetting-drying MASK. T or F.

MASK9: logical parameter for output MASK9 (switch for Boussinesq/NSWE). T or F.

SourceX: logical parameter for output source terms in x direction. T or F.

SourceY: logical parameter for output source terms in y direction. T or F.

P: logical parameter for output of momentum flux in x direction. T or F.

Q: logical parameter for output of momentum flux in y direction. T or F.

Fx: logical parameter for output of numerical flux F in x direction. T or F.

Fy: logical parameter for output of numerical flux F in y direction. T or F.

Gx: logical parameter for output of numerical flux G in x direction. T or F.

Gy: logical parameter for output of numerical flux G in y direction. T or F.

AGE: logical parameter for output of breaking age. T or F.

4.6 Input for the spherical code

All input parameters, except the following grid information, are the same as for the Cartesican code.

Lon_West: longitude (degrees) of west boundary.

Lat_South: latitude (degrees) of south boundary.

Dphi: $d\phi$ (degrees)

Dtheta: $d\theta$ (degrees)

In addition, it is not necessary to specify Gamma2 (for nonlinear dispersive terms) in the spherical code.

Another feature of the spherical code is that a computational grid can be a stretched grid. For a stretched grid, a user should set StretchGrid = T and provide grid files for DX and DY and a file for Coriolis parameters at each grid point. For example,

DX_FILE = dx_str.txt
DY_FILE = dy_str.txt
CORIOLIS_FILE = cori_str.txt

4.7 Model nesting

The present version has a capability for one-way nesting. The nesting scheme passes surface elevation and velocity components calculated from a large domain to a nested small domain through ghost cells at nesting boundaries. To run a nested model, the following procedures should be performed.

1. The coupling option in Makefile should be defined as '-Dcoupling' and the program should be re-compiled.

2. Prepare nesting data using the output of a large-domain model. The following is an example of the data format.

   nesting data
   Number of data:
   100 Time series (s):
   0.0000000
   1.0000000
   2.0000000
   3.0000000
   4.0000000
   ...

99.000000
EAST boundary
Num of points,Start number J
0 0
WEST boundary
Num of points,Start number J
160 20
U,V,Z data ...
0.000 0.000 0.000 ...
...
SOUTH boundary
Num of points,Start number I
0 0
NORTH boundary
Num of points,Start number I
0 0

The example above is a case that a model nesting takes place at the WEST (left) boundary of a small domain. Boundaries are defined with the order: EAST, WEST, SOUTH, and NORTH. If the num of points of a boundary is larger than zero, the program will read a time series of (u, v, η) below 'U,V,Z data ...'. The read format is

```
DO K=1,N_COUPLING_DATA
READ(1,*)(U_COUPLING_WEST(I,K),I=1,N_COUPLING_WEST)
READ(1,*)(V_COUPLING_WEST(I,K),I=1,N_COUPLING_WEST)
READ(1,*)(Z_COUPLING_WEST(I,K),I=1,N_COUPLING_WEST)
ENDDO
```

where N_COUPLING_DATA is 'Number of data' and N_COUPLING_WEST is Num of points at the WEST boundary.

3. Specify the file of coupling data in input.txt

   ! ———————— COUPLING ————————————-
   ! if do coupling, have to set -DCOUPLING in Makefile
   COUPLING_FILE = coupling.txt
   where 'coupling.txt' is the file saved in procedure 2.

4.8 Output

The output files are saved in the result 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 such

eta_0001, eta_0002, ... The format and read/write algorithm are consistent with a depth file. Output for stations is a series of numbered files such as sta_0001, sta_0002 ....

The NetCDF output format is under development.

5 Examples

The model has been validated extensively using laboratory experiments for wave shoaling and breaking as in the FUNWAVE manual by Kirby et al. (1998). In addition, Tehranirad et al. (2011) used FUNWAVE-TVD Version 1.0 to carry out tsunami benchmark testing in conjunction with the National Tsunami Hazard Mitigation Program. Tests of mass conservation and convergence are included in Tehranirad et al. (2011).

5.1 Breaking waves on a beach (Example 1 in the example directory)

Hansen and Svendsen (1979) carried out laboratory experiments of wave shoaling and breaking on a beach. Waves were generated on a flat bottom a 0.36 m depth, and the beach slope was 1:34.26. The experiments included several cases including plunging breakers, plunging-spilling breakers and spilling breakers. In this paper, we simulate two typical cases: a plunging breaker and a spilling breaker, respectively. The wave height and wave period are 4.3 cm and 3.33 s, respectively, for the plunging case, and 6.7 cm and 1.67 s for the spilling case.

Although the shock-capturing breaking algorithm used in Boussinesq wave models has been examined by previous researchers (e.g., Tonelli and Petti, 2009, Shiach and Mingham, 2009 and others), there is a concern about its sensitivity to grid spacing. In this study, we adopted three grid sizes, dx = 0.05 m, 0.025 m and 0.0125 m, respectively, for each cases. Figure 3 shows comparisons of wave height and wave setup between measured data and numerical results from model runs with different grid sizes. The wave breaking location of wave setup/setdown predicted by the three runs are in agreement with the data, however, the predicted maximum wave heights are slightly different. Results from the dx=0.25 m and 0.0125 m grids are very close, indicating a convergence with grid refinement. All three models underpredict the peak wave height at breaking and overpredict wave height inside of the surfzone. This prediction trend was also found in Kennedy et al. (2000, Figure 2). About 10% underprediction of peak wave height can be found in our tests with $\mathrm{d}x = 0.025$ m and 0.0125 m, which is similar to Kennedy et al. (2000). The model with a coarser grid ($\mathrm{d}x$ = 0.05 m) underpredicted the maximum wave height by 20%.

To find the cause of the large underprediction of peak wave height in the coarser grid model, in Figure 4, we show snapshots of surface elevation from model results with $\mathrm{d}x$ = 0.025 m and 0.050 m at different times. The model with the finer grid resolution switched from the Boussinesq equations to NSWE around $t = 19.9s$ (the model with the coarser grid switched slightly later) at the point where the ratio of surface elevation to water depth reached the threshold value of 0.8. Then, a wave is damped at the sharp front and generates trailing high frequency oscillations. The comparison of wave profiles at an early time (i.e. t= 18.6 s) shows that the coarser grid model

underpredicts wave height before the Boussinesq-NSWE switching, indicating that the underprediction is not caused by the shock-capturing scheme, but by the numerical dissipation resulting from the coarse grid resolution.

For the spilling breaker case, the models with three different grid sizes basically predicted slightly different wave peaks as in the plunging wave case. Figure 5 shows results with $\mathrm{d}x$ = 0.25 m, where the model provides very good predictions of wave shoaling to near the breaking limit. Once again, the model overpredicts wave height inside the surf zone.

Some necessary definitions in the input file, input.txt, for the plunging breaker case are

```
    ! ———————DEPTH———————————————-
DEPTH _TYPE = SLOPE
DEPTH _FLAT = 0.36
SLP = 0.0292
Xslp = 55.0
! ———————DIMENSION———————————
Mglob = 3001
Nglob = 3
! ———————— TIME———————————————-
DX = 0.025
DY = 0.2 ! give any larger value than DX for 1-D case
! ———————-WAVEMAKER———————————
WAVEMAKER = WK _REG
Xc _WK = 45.0
Tperiod = 3.33
AMP _WK = 0.0185
DEP _WK = 0.36
Theta _WK = 0.0
Time _ramp = 1.0
Delta _WK = 0.3
! ———————- SPONGE LAYER ——————————
SPONGE _ON = T
Sponge _west _width = 10.0
Sponge _east _width = 0.0
Sponge _south _width = 0.0
Sponge _north _width = 0.0
R _sponge = 0.95
A _sponge = 5.0
! ———————-PHYSICS———————————
DISPERSION = T
Gamma1 = 1.0
Gamma2 = 1.0
```

Gamma3=1.0
Beta _ref=-0.531
SWE _ETA _DEP = 0.8
Cd = 0.0 ! Cd is not sensitive for this case
! ————————-NUMERICS————————————-
Time _Scheme = Runge _Kutta
HIGH _ORDER = FOURTH
CONSTRUCTION = HLLC
CFL = 0.5
! ——————WET-DRY————————————-
MinDepth=0.001
MinDepthFrc = 0.001
! ————————OUTPUT————————————
DEPTH _OUT = T
ETA = T
MASK = T

A MATLAB script, *plot_surface_ele.m*, is used for plots.

5.2 Random wave shoaling and breaking on a slope (Example 2 in the example directory)

To study random-wave properties of shoaling and breaking, Mase and Kirby (1992) conducted a laboratory experiment of random wave propagation over a planar beach. The experiment layout is shown in Figure 6, where a constant depth of 0.47 m on the left connects to a constant slope of 1:20 on the right. Two sets of random waves with peak frequencies of 0.6 Hz (run 1) and 1.0 Hz (run 2) were generated by the wavemaker on the left. The target incident spectrum was a Pierson-Moskowitz spectrum. Wave gauges at depths h = 47, 35, 30, 25, 20, 17.5, 15, 12.5, 10, 7.5, 5, and 2.5 cm collected time series of surface elevation.

Wei and Kirby (1995) carried out a simulation of run 2 without wave breaking. Later, Kirby et al. (1998) and Kennedy et al. (2000) carried out the same simulation with wave breaking. The present model was set up following Kirby et al. (1998), who used an internal wavemaker located at the toe of the slope where surface elevation is measured (gauge 1).

A *FFT* was used to transform between the time domain and frequency domain data required by the wavemaker. A MATLAB script, (*fft4wavemaker.m*) to perform this transform is included in the example. The script reads the measured data from the file called *r2d470.dat*, and saves calculated wave amplitude, period and phase information for each component in the file named as *wavemk_per_amp_pha.txt*. The low and high-frequency cutoffs are 0.2 and 10.0 Hz, respectively.

The simulation time is the same as the time length of data collection. The computational domain is from x = 0 m to 20 m with a grid size of 0.04 m. The toe of the slope starts at x = 10 m. A sponge layer is specified at the left side boundary, to absorb reflected waves, but no sponge layer

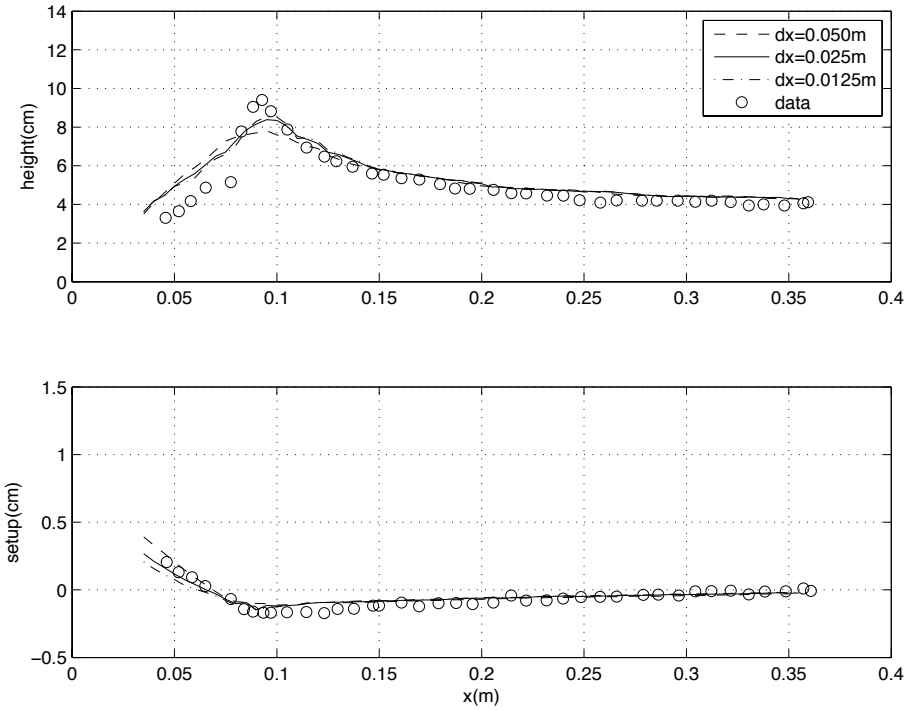

Figure 3: Comparisons of wave height (upper panel) and wave setup (lower panel) between measured data and model results from grid resolutions of dx = 0.0125 m, 0.025 m and 0.050 m, respectively. Case: plunging breaker.

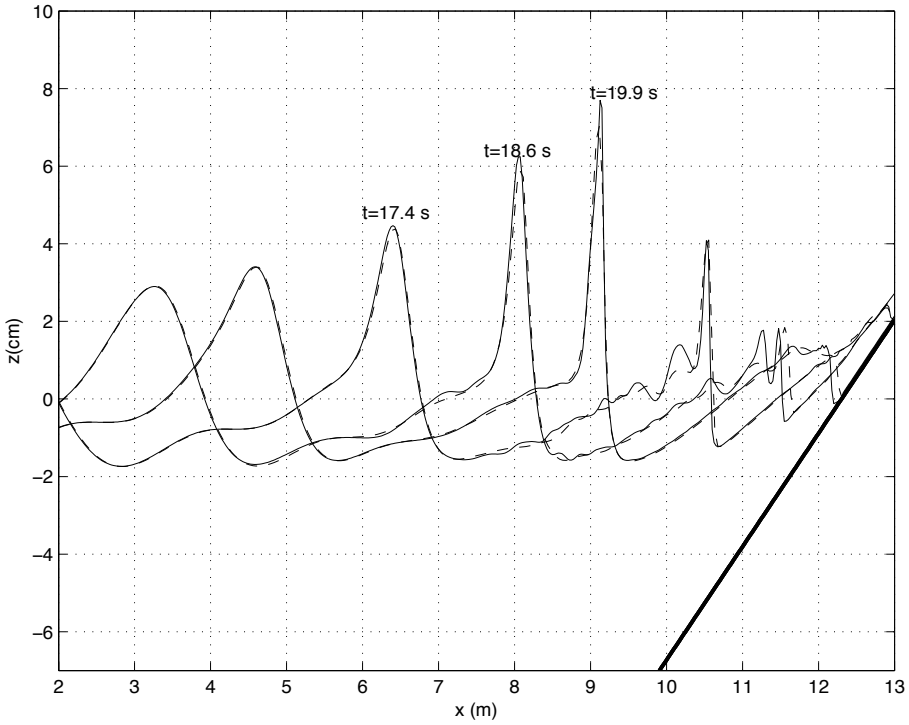

Figure 4: Snapshots of surface elevation at t = 17.4, 18.6 and 19.9 s from models with grid resolutions of dx = 0.025 (solid lines) and 0.050 m (dashed lines).

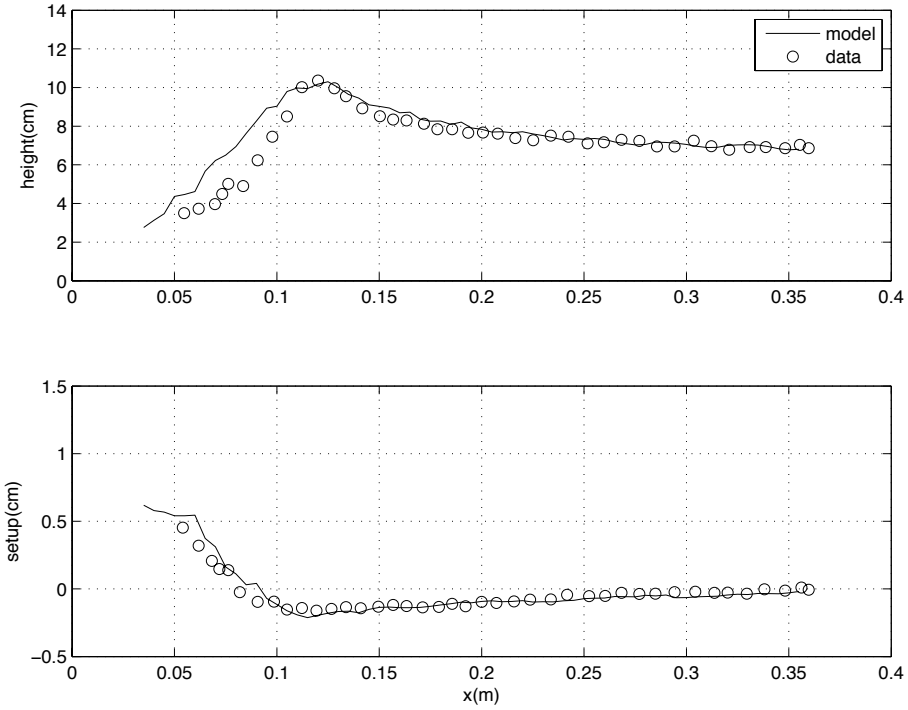

Figure 5: Model/data comparisons of wave height (upper panel) and wave setup (dx = 0.025 m). Case: spilling breaker.

is needed on the right boundary, which differs from Kirby et al. (1998) who used the slot method combined with a sponge layer at the end of the domain.

We present the model results for run 2 and compare with the experimental data measured at the other 11 gauges shown in Figure 6. Figure 7 shows model results (dashed lines) and measured data (solid lines) from t = 20 s to t = 40 s at those gauges. Both model and data show that most waves start breaking at the depth h= 15 cm. Except for small discrepancies for wave phases, the model reproduces the measured waveform quite well.

To further demonstrate the applicability of the model, we performed third moment computations of the resulting time series of surface elevation. Normalized wave skewness and asymmetry were calculated for both measured and modeled time series of surface elevation according to the following formulations,

$$\begin{aligned} \text{skew} &= \frac{<\eta^3>}{<\eta^2>^{3/2}} \\ \text{asym} &= \frac{<H(\eta)^3>}{<\eta^2>^{3/2}} \end{aligned} \tag{78}$$

where H denotes the Hilbert transform, $<>$ is the mean operator, and the mean has been removed from the time series of surface elevation.

Figure 8 shows the comparisons of skewness and asymmetry between the model results and experiment data. The model predicted skewness and asymmetry reasonably well with a slightly overprediction of wave skewness inside the surf zone.

It is worth mentioning that Kirby et al. (1998) employed more frequent use of numerical filtering, especially after wave breaking, so that the model run was stable over the entire data time series. The present model did not encounter any stability problem without filtering.

Some necessary definitions in the input file, input.txt, are

```
    ! ———————DEPTH————————————————-
DEPTH_TYPE = SLOPE
DEPTH_FLAT = 0.47
SLP = 0.05
Xslp = 10.0

    ! ———————DIMENSION————————————
Mglob = 500
Nglob = 3

    ! ——————— TIME————————————————-
TOTAL_TIME = 716.0
PLOT_INTV = 10.0
PLOT_INTV_STATION = 0.05
```

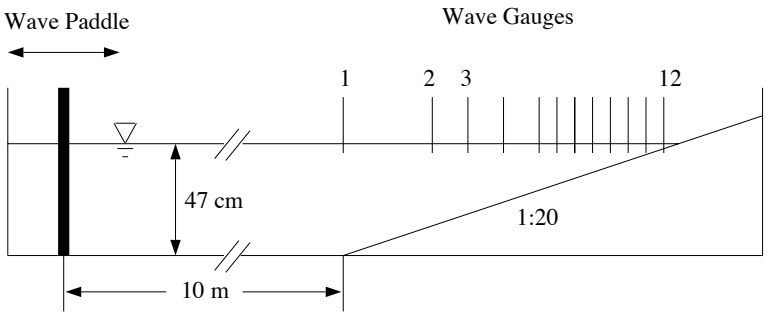

Figure 6: Experiment layout of Mase and Kirby (1992).

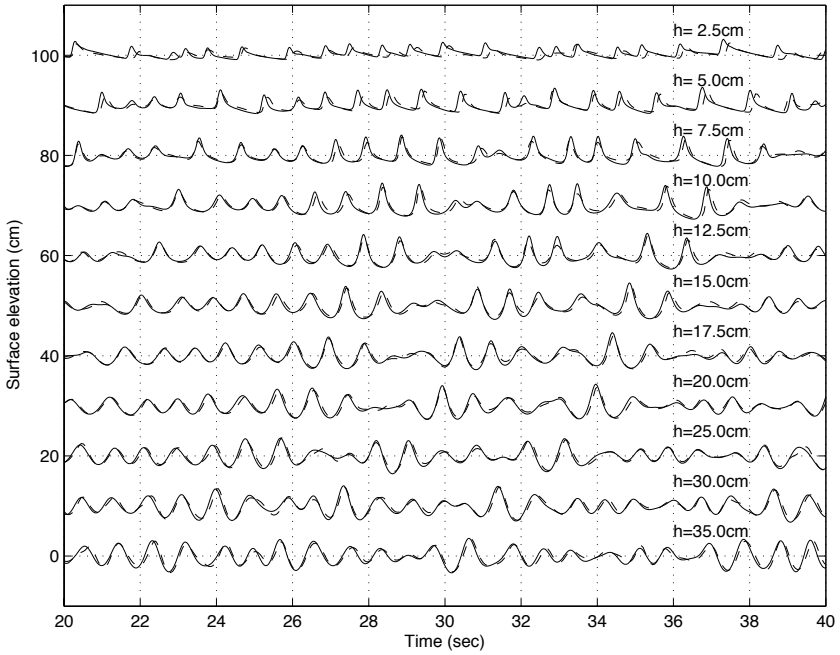

Figure 7: Time series comparison of η between model (dashed lines) and data (solid lines) at 11 wave gauges in Mase and Kirby (1992).

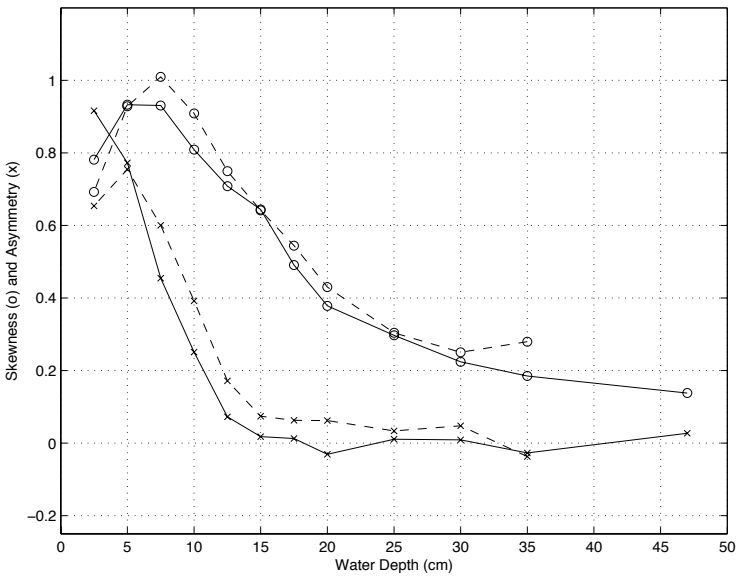

Figure 8: Comparison of skewness ($\circ$) and asymmetry ($\times$) at different water depths. Solid lines are experiment data (Mase and Kirby, 1992). Dashed lines are numerical results

```
SCREEN_INTV = 1.0

    ! ————————GRID————————————————-
DX = 0.04
DY = 0.10

    ! ————————-WAVEMAKER———————————
WAVEMAKER = WK_TIME_SERIES
NumWaveComp = 1505
PeakPeriod = 1.0
WaveCompFile = ../fft/wavemk_per_amp_pha.txt
! Wei and Kirby 1999
Time_ramp = 1.0
Delta_WK = 0.4 ! width parameter 0.3-0.6
DEP_WK = 0.47
Xc_WK = 10.0
Ywidth_WK = 10000.0 ! give any bigger value than the width of tank

    ! ————————- SPONGE LAYER ——————————
SPONGE_ON = T
Sponge_west_width = 2.0
Sponge_east_width = 0.0
Sponge_south_width = 0.0
Sponge_north_width = 0.0
R_sponge = 0.90
A_sponge = 5.0

    ! ————————-PHYSICS———————————————
DISPERSION = T
Gamma1 = 1.0
Gamma2 = 1.0
Gamma3 = 1.0
Beta_ref=-0.531
SWE_ETA_DEP = 0.80
Cd = 0.001 ! not sensitive for this case

    ! ————————-NUMERICS————————————————-
Time_Scheme = Runge_Kutta
HIGH_ORDER = FOURTH
CONSTRUCTION = HLLC
CFL = 0.5
```

```
! ————————WET-DRY————————————————-
MinDepth=0.001
MinDepthFrc = 0.001
! ————————————OUTPUT————————————————
NumberStations = 12
STATIONS_FILE = gauges_004.txt
DEPTH_OUT = T
ETA = T
```

Model output at 12 gauges are saved in *sta_0001, sta_0002, ..., sta_0012*. Use MATLAB scripts, *comp_mkskew.m* and *cmop_series.m* for plots.

5.3 Wave propagation over a shoal: Berkhoff et al. (1982) (Example 3 in the example directory)

The laboratory experiment of wave propagation over a shoal conducted by Berkhoff et al. (1982) has served as a standard test for examining numerical model performances in predicting wave shoaling, refraction, diffraction and nonlinear dispersion. Kirby et al. (1998) showed that the previous version of FUNWAVE accurately reproduces measured wave heights in the experiments. Here, in this manual, we repeat this test exactly following Kirby et al. (1998).

The bottom topography is shown in Figure 9, which is generated using the same program in Kirby et al., (1998). The topography consists of an elliptic shoal resting on a plane beach with a constant slope 1/50. Bottom contours on the slope are oriented at an angle of 20° to the y axis. Regular waves with period of $1s$ and amplitude of $2.32cm$ are generated by a wavemaker at $x = -10m$ and propagate across the domain. Experiment data are collected along 8 transects as shown in the figure. Two vertical side walls are located at $y = -10m$ and $y = 10m$. Detailed information on the geometry may be obtained in Berkhoff et al. (1982) or Kirby and Dalrymple (1984).

The computational domain used in the model is the same as in Figure 9 except for two sponge layers with a width of $2m$ sitting behind wavemaker and on the end of the beach. The source function for generating the corresponding monochromatic wave is located at the wavemaker.

Thirty waves are simulated in order to get a quasi-stable wave condition. The time series of surface elevation in the last 5 seconds (5 wave periods) are used for the wave height estimation at each grid point. Matlab post-processing scripts, *calcheight.m* and *showheight.m* are provided in the example. Figure 10 shows comparisons between model results and experimental data along the eight transects where measurements were made. The model results of wave height agree well with experimental data, in both sections parallel or normal to incident wave direction. The results from the present model are also similar to that from the previous version of FUNWAVE in Kirby et al. (1998).

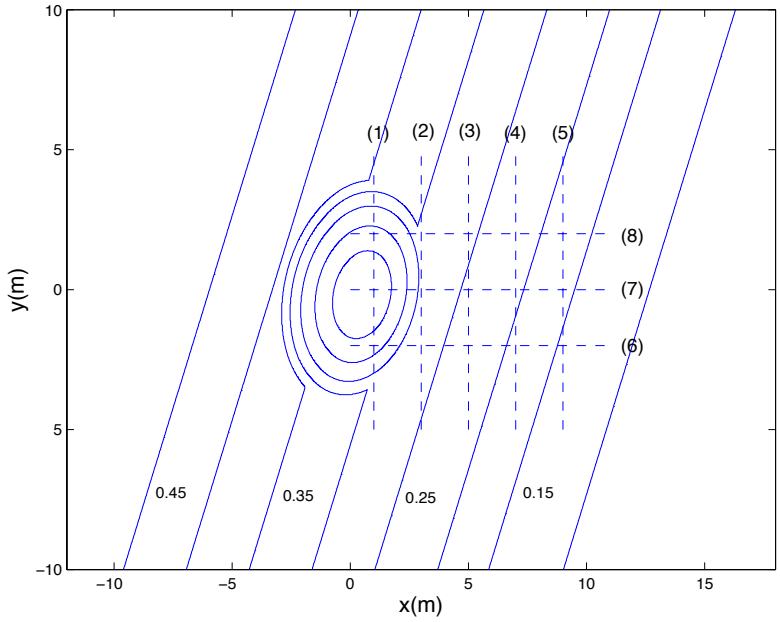

Figure 9: Experiment layout for wave focusing experiment of Berkhoff et al. (1982).

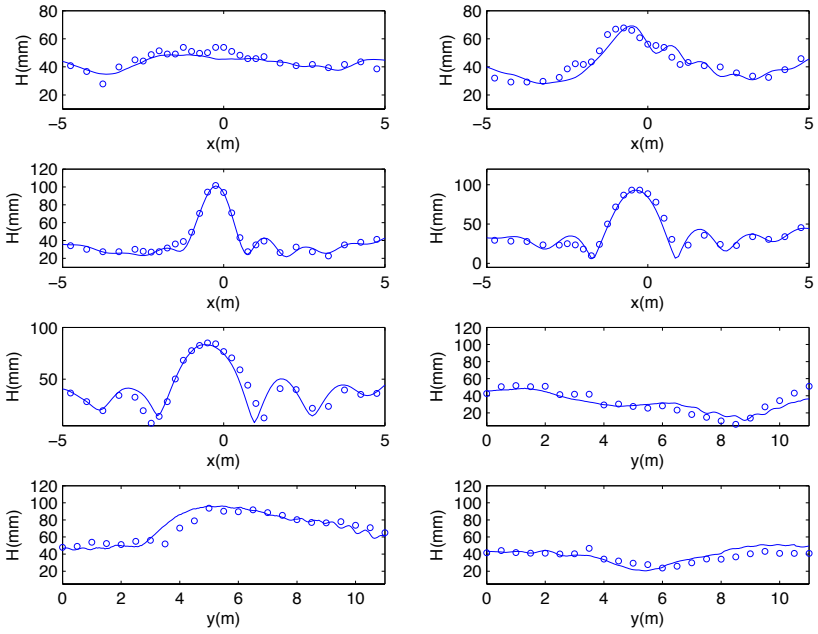

Figure 10: Comparisons of wave height along specified sections between the model (solid lines) and experiment data (circles).

Parameters and other definitions specified in *input.txt* are listed below.

```
    ! ———————DEPTH———————————————-
DEPTH_TYPE = DATA
DEPTH_FILE = ../input/depth.txt

    ! ———————DIMENSION———————————
Mglob = 600
Nglob = 200

    ! ——————— TIME———————————————-
TOTAL_TIME = 30.0
PLOT_INTV = 0.02
PLOT_INTV_STATION = 0.02
SCREEN_INTV = 0.1

    ! ———————GRID———————————————-
DX = 0.05
DY = 0.10

    ! ———————-WAVEMAKER———————————
WAVEMAKER = WK_REG
Time_ramp = 1.0
Delta_WK = 0.5 ! width parameter 0.3-0.6
DEP_WK = 0.45
Xc_WK = 3.0
Ywidth_WK = 10000.0 ! give any larger value than the width of flume
Tperiod = 1.0
AMP_WK = 0.0232
Theta_WK = 0.0

    ! ———————- SPONGE LAYER ————————————
SPONGE_ON = T
Sponge_west_width = 2.0
Sponge_east_width = 2.0
Sponge_south_width = 0.0
Sponge_north_width = 0.0
R_sponge = 0.90
A_sponge = 5.0

    ! ———————-PHYSICS———————————————
```

```
DISPERSION = T
Gamma1 = 1.0
Gamma2 = 1.0
Gamma3 = 1.0
Beta_ref=-0.531
SWE_ETA_DEP = 0.80
Cd = 0.0

    ! ————————-NUMERICS————————————-
Time_Scheme = Runge_Kutta
HIGH_ORDER = FOURTH
CONSTRUCTION = HLLC
CFL = 0.5

    ! ——————WET-DRY————————————-
MinDepth=0.001
MinDepthFrc = 0.001

    ! ————————OUTPUT———————————
NumberStations = 0
DEPTH_OUT = T
ETA = T
```

5.4 Solitary wave on a conical island (Example 4 in the example directory)

Laboratory experiments on the interaction between solitary waves and a conical island were conducted by Briggs et al (1995). The three cases from this test illustrate the important fact that runup and inundation heights on the sheltered back sides of an island can exceed the incident wave height on the exposed front side, due to trapping of wave fronts propagating around the island circumference. These tests have been used in a number of validation studies for a variety of models, including nonlinear shallow water equations (Liu et al 1995) and Boussinesq equations (Chen et al, 2000). The benchmark test is specified in Section 3.3 of Appendix A of Synolakis et al (2007).

Large-scale laboratory experiments were performed at Coastal Engineering Research Center, Vicksburg, Mississippi, in a 30m-wide, 25m-long, and 60cm-deep wave basin (Figure 11). In the physical model, a 62.5cm-high, 7.2m toe-diameter, and 2.2m crest-diameter circular island with a 1:4 slope was located in the basin (Figure 12). Experiments were conducted at depth of 32cm, with three different solitary waves (H/d=0.045, 0.091, 0.181). Water-surface time histories were measured with 27 wave gages located around the perimeter of the island (Figure 13).

For this benchmark test, time histories of the surface elevation around the circular island are given at four locations, i.e., in the front of the island at the toe (Gauge 6) and gauges closest to the

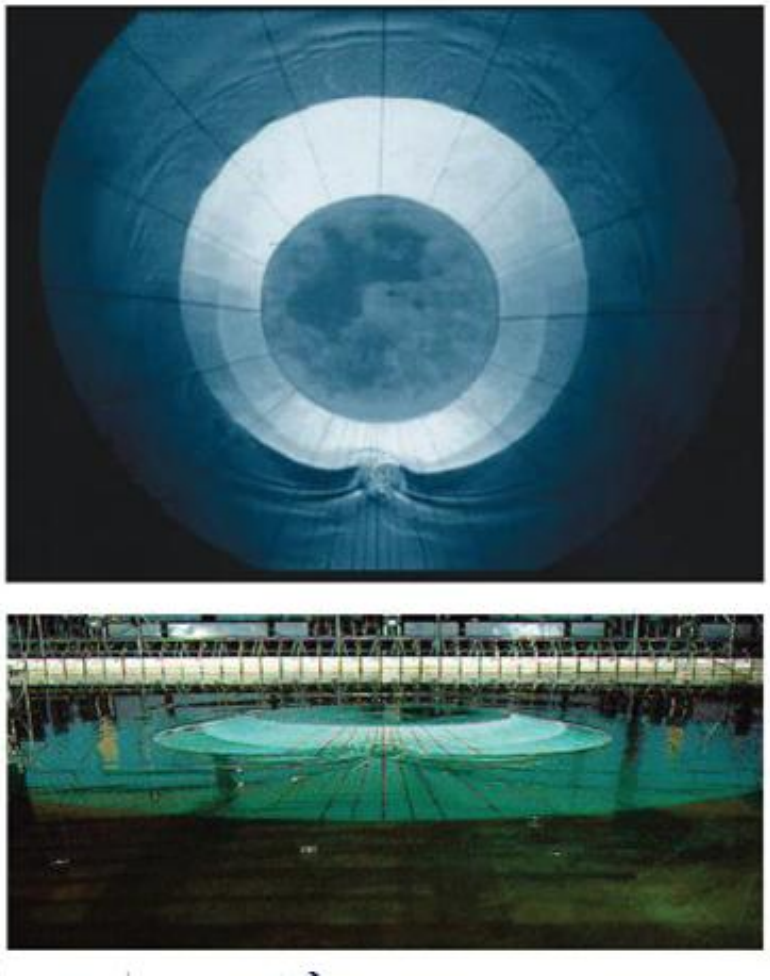

Figure 11: View of conical island(top) and basin(bottom)(from Synolakis et al (2007, Figure A16)).

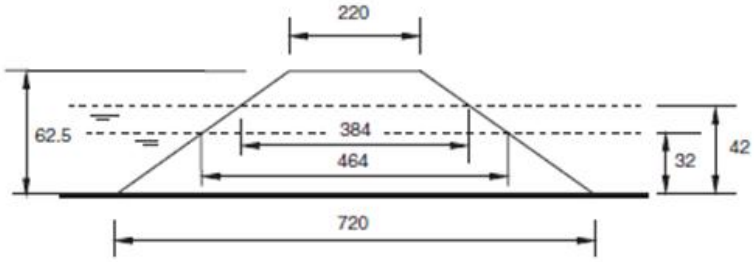

Figure 12: Definition sketch for conical island. All dimensions are in cm (from Synolakis et al (2007, Figure A17)).

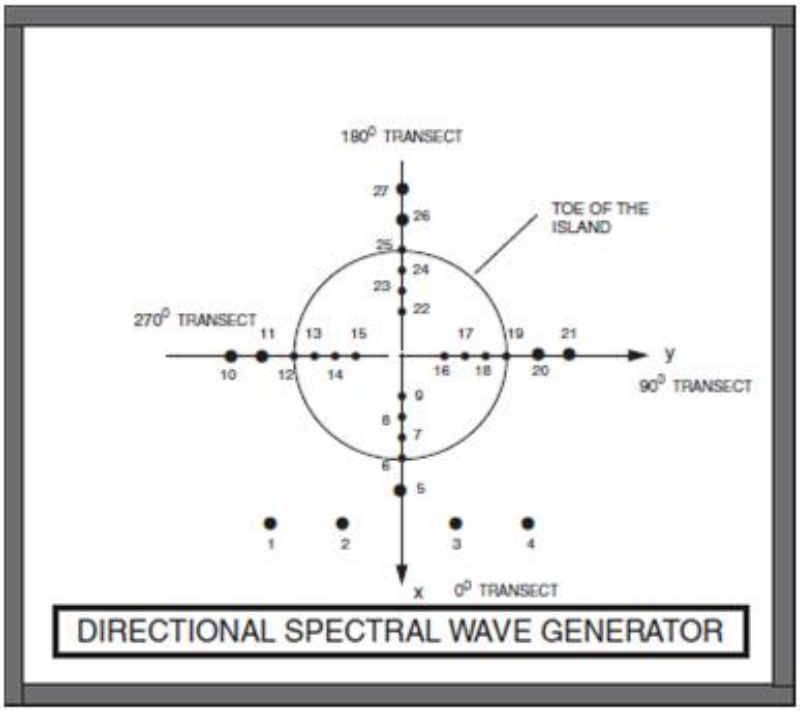

Figure 13: Schematic gauge locations around the conical island(from Synolakis et al (2007, Figure A18)).

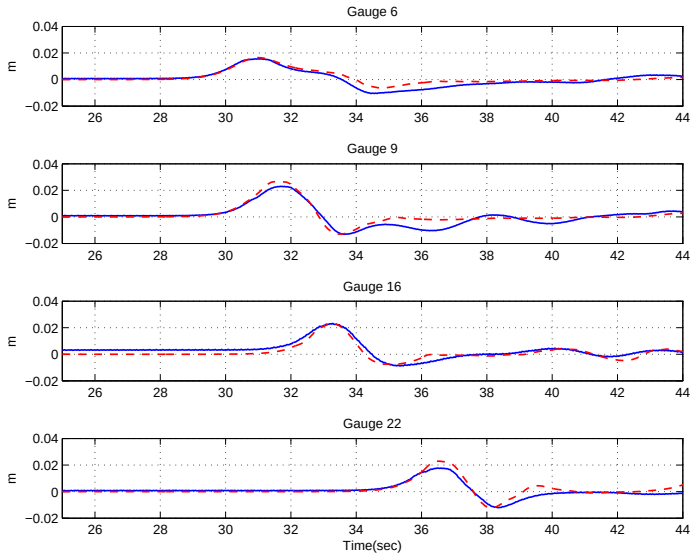

Figure 14: Comparison of computed and measured time series of free surface for $H/d = 0.045$.Solid lines: measured, Dashed lines: Computed.

	Gauge Number			
H/d	6	9	16	22
0.045	6.0	13.2	0.1	18.9
0.091	3.2	16.6	11.6	0.26
0.181	1.6	13.33	13.8	13.3

Table 1: Percent error of predicted maximum runup calculated for each gauge in conical island test.

shoreline with the numbers 9, 16, and 22 located at the $0°$, $90°$, and $180°$ radial lines (Figure 13). A grid size of $\Delta x = 0.10m$ is considered for proper numerical simulation of this benchmark. Figures 14-16 shows the comparison between the laboratory data with numerical calculations. Table 1 represents the error of the maximum runup for each gauge for different wave heights.

Input parameters in the tests are listed as below.

! ——————DEPTH————————————————-
DEPTH_TYPE = DATA
DEPTH_FILE = ../input/depth.txt

! ——————DIMENSION————————————

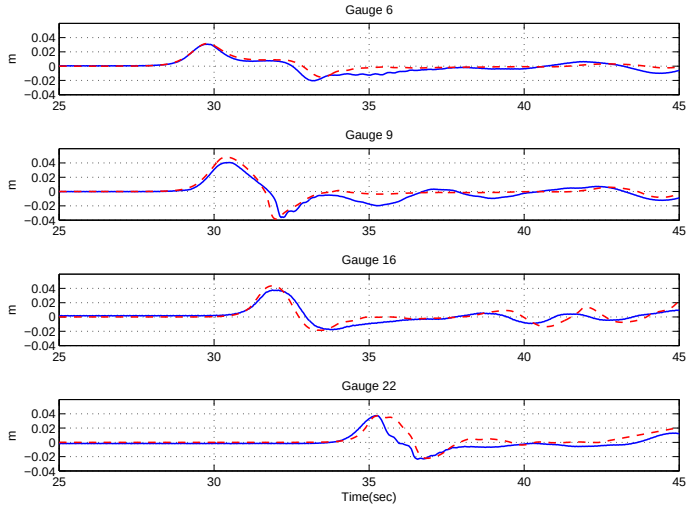

Figure 15: Comparison of computed and measured time series of free surface for $H/d = 0.091$.Solid lines: measured, Dashed lines: Computed.

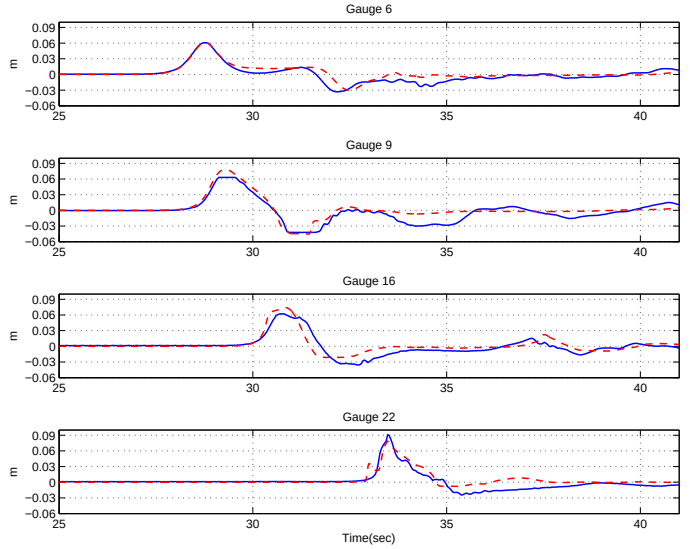

Figure 16: Comparison of computed and measured time series of free surface for $H/d = 0.181$.Solid lines: measured, Dashed lines: Computed.

```
Mglob = 702
Nglob = 602

    ! ———————— TIME————————————————-
TOTAL_TIME = 25.0
PLOT_INTV = 1.0
PLOT_INTV_STATION = 0.05
SCREEN_INTV = 1.0

    ! ———————GRID————————————————-
DX = 0.05
DY = 0.05

    ! ———————-WAVEMAKER———————————————
WAVEMAKER = INI_SOL
AMP = 0.0144 ! 0.02912 for case B and 0.05792 for case C
DEP = 0.32
LAGTIME = 5.0
XWAVEMAKER = 8.0

    ! ———————- SPONGE LAYER ———————————
SPONGE_ON = F

    ! ———————-PHYSICS———————————————
DISPERSION = T
Gamma1 = 1.0
Gamma2 = 1.0
Gamma3 = 1.0
Beta_ref=-0.531
SWE_ETA_DEP = 0.80
Cd = 0.0 ! note: not sensitive, could give a small value like 0.001

    ! ———————-NUMERICS————————————————-
Time_Scheme = Runge_Kutta
HIGH_ORDER = FOURTH
CONSTRUCTION = HLLC
CFL = 0.5

    ! ——————WET-DRY————————————————-
MinDepth=0.001
MinDepthFrc = 0.001
```

```
! ————————OUTPUT————————————
NumberStations = 7
STATIONS_FILE = gauges.txt
DEPTH_OUT = T
ETA = T
Hmax = T
MASK = T
```

The results of surface elevation at seven gauges are saved as sta_0001 ... sta_0007 as specified in input.txt. A post-processing MATLAB script called *BM6_loader_a.m* for case A can be used for model/data comparisons.

5.5 Solitary wave runup on a shelf with an island (Example 5 in the example directory)

In this test, we performed a simulation of the solitary wave runup measured recently in a large wave basin at Oregon State University's O.H. Hinsdale Wave Research Laboratory. The basin is 48.8 m long, 26.5 m wide, and 2.1 m deep. A complex bathymetry has been constructed, which consists of a 1:30 slope planar beach connected to a triangle shaped shelf and a conical island on the shelf as shown in Figure 17. Solitary waves were generated on the left side by a piston-type wavemaker. Surface elevation and velocity were collected at many locations by wave gauges and ADV's in alongshore and cross-shore arrays (See Swigler and Lynett, 2011 for details). Figure 17 shows wave gauges (circles) and ADV's (triangles) used for model/data comparisons in the present study. Gauge 1 - 9 were located at (x, y) = (7.5, 0.0), (13.0, 0.0), (21.0 0.0), (7.5 5.0), (13.0 5.0), (21.0 5.0), (25.0 0.0), (25.0 5.0) and (25.0, 10.0), respectively. ADV 1 - 3 were located at (13.0 0.0), (21.0, 0.0) and (21.0, -5.0), respectively.

The modeled bathymetry was constructed by combining the measured data of the shelf and the analytical equation of the cone, which was used for the design of the island in the experiment. The computational domain was modified by extending the domain from x = 0.0 m to -5.0 m with a constant water depth of 0.78 m in order to use a solitary wave solution as an initial condition. The width of the computational domain in the y direction is the same as OSU's basin. Grid spacing used in the model is 0.1 m in both directions. A solitary wave solution based on Nwogu's extended Boussinesq equations was used with centroid located at x = 5.0 The wave height is 0.39 m as that used in the laboratory experiment.

Figure 18 shows results of computed water surfaces at t= 6.4 s, 8.4 s and 14.4 s, respectively. The wave front becomes very steep as the wave climbs on the shelf, which was well captured by the model. The wave scattering pattern is clearly seen in the bottom panel of Figure 18. Wave breaking on the shelf was observed in the laboratory experiment and was also seen in the model. Figure 19 shows the variation in time stepping during the simulation. The time step dropped to a minimum, at around t = 6.5 s, as the wave collided with the island (top panel of figure 18). The

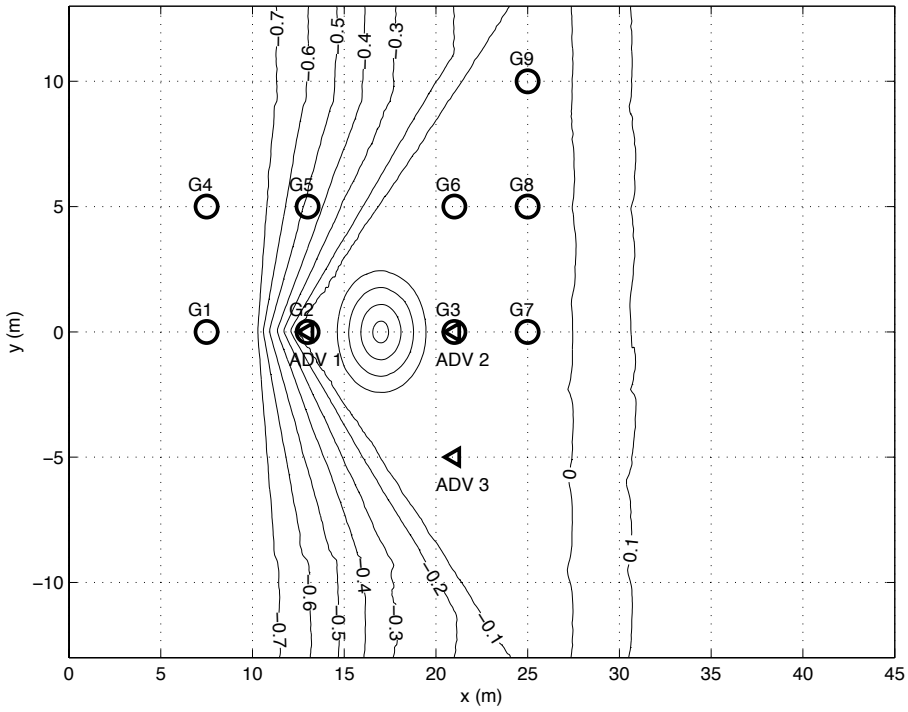

Figure 17: Bathymetry contours (in meters) and measurement locations used in model/data comparisons. Circles: pressure gauges, triangles: ADV.

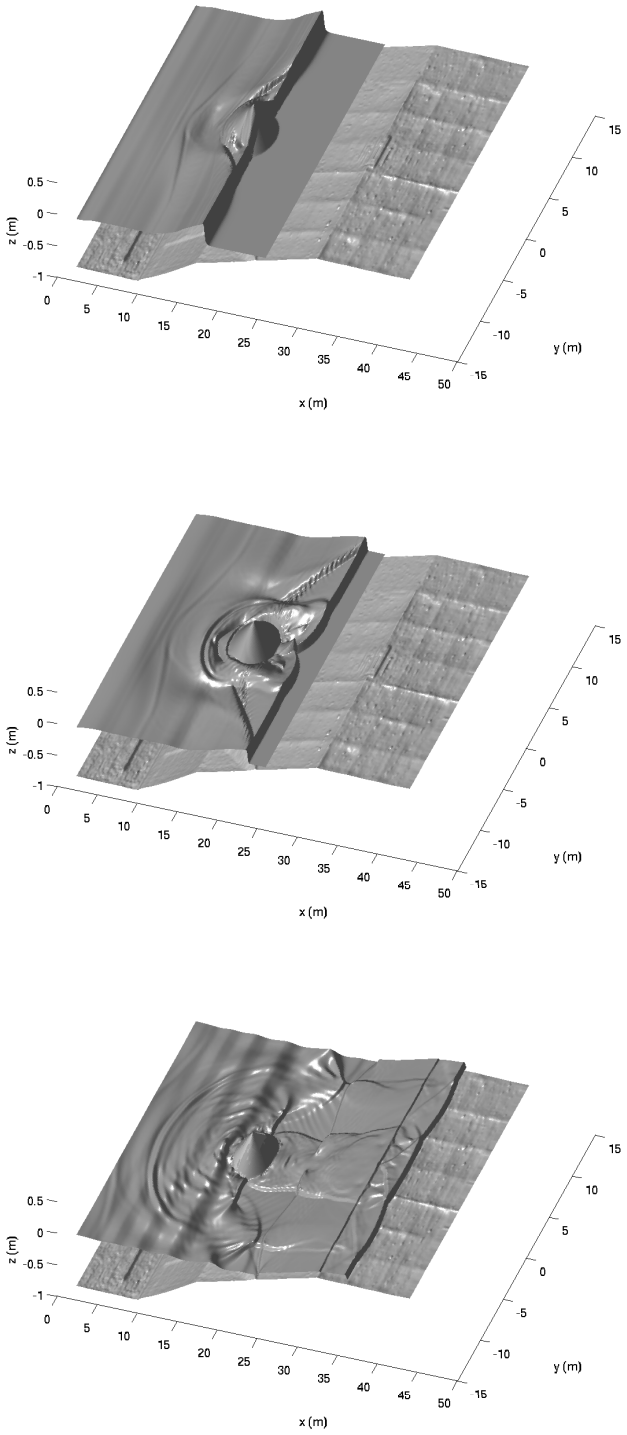

Figure 18: Modeled water surface at (top) t = 6.4 s, (middle) t =8.4 s, (bottom) t =14.4 s.

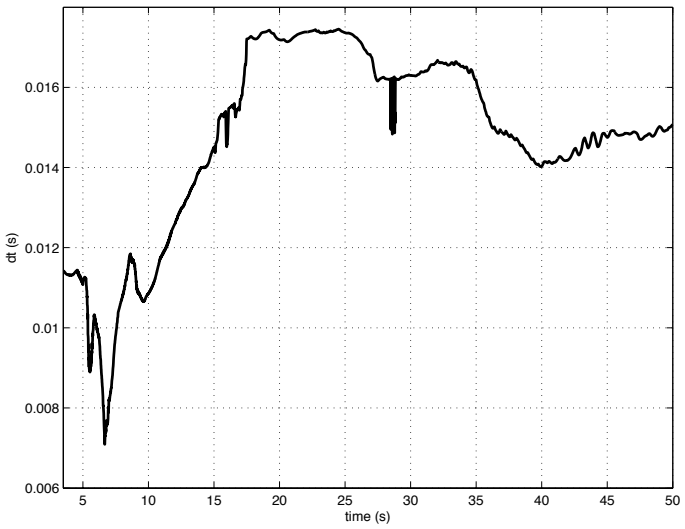

Figure 19: Time step variation.

local Froude number reached a maximum at t = 6.5 s, reducing the value of the time step based on (42).

Figure 20 shows time series of modeled surface elevations and measurements at Gauge 1 - 9 (from top to bottom). Good agreement between model and data is found at the gauge in front of the island (Gauge 1, top panel), as the model successfully predicts the solitary wave propagation and its reflection from the shore. The model also captures the collision of edge waves propagating around the two sides of the island, as indicated at the gauge behind the island (Gauge 3). The model predicts the timing of wave collision well but over-predicts the peak of wave runup. The model/data comparisons at Gauges 5, 6, 8, and 9, which are located at the north-side shelf, indicates that the model predicts wave refraction and breaking on the shelf reasonably well.

Figure 21 shows model/data comparisons of velocity time series in the x-direction at ADV 1 (top) and ADV2 (bottom). The model predicts the peak velocity and the entire trend of velocity variation in time at both locations. An underprediction of the seaward velocity is found at ADV 2. The velocity in the y-direction was not compared at ADV 1 and ADV2 because the measured values were too small. Figure 22 shows the u and v velocity components in the x and y directions at ADV 3, and shows that the model predicts the velocity components in both x and y directions well.

The input parameters for this test are listed below.

```
    ! ————————————DEPTH————————————————————-
DEPTH_TYPE = DATA
DEPTH_FILE = ../input/depth_wkshop.txt
```

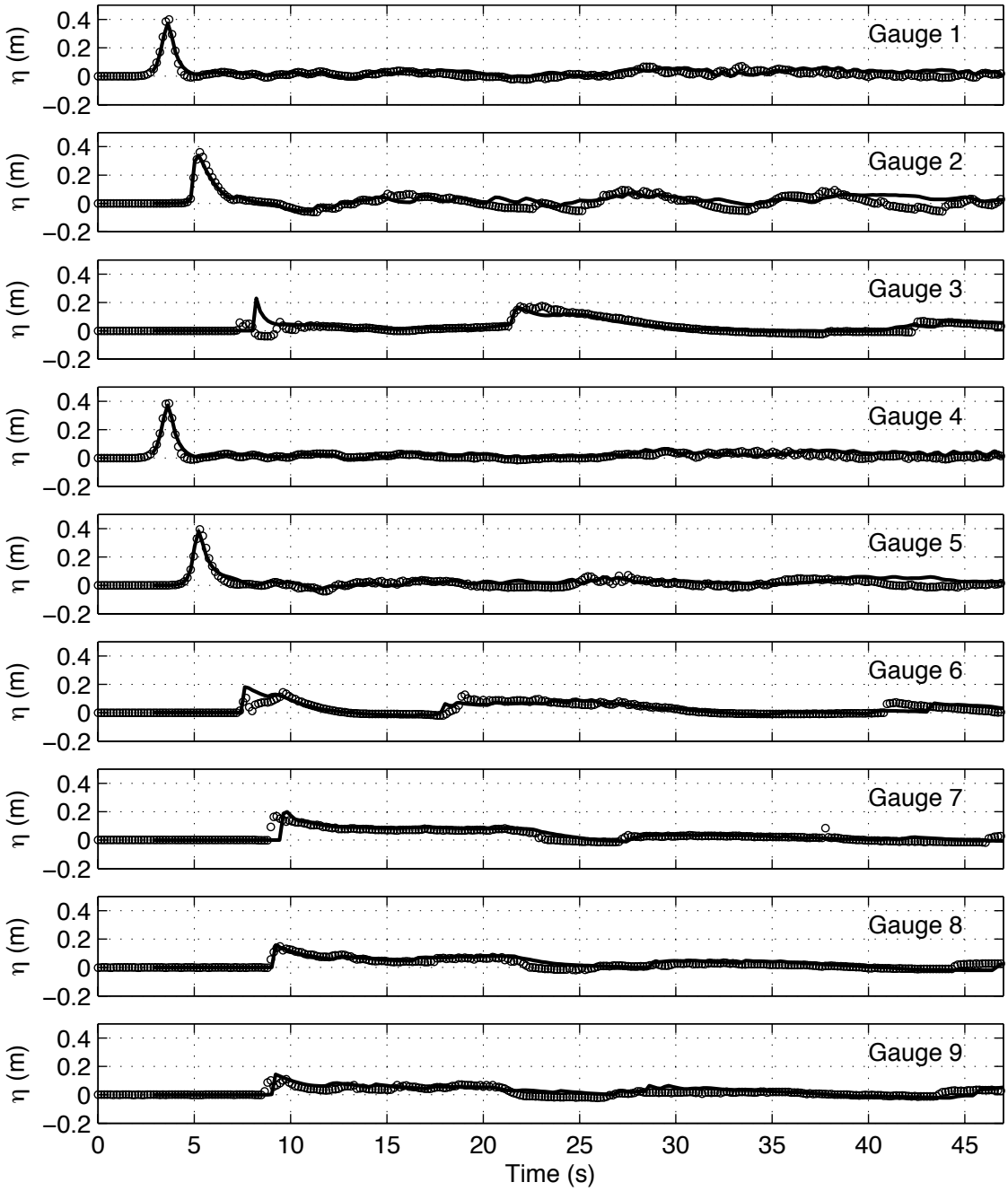

Figure 20: Model/data comparisons of time series of surface elevation at (top) Gauge 1 - Gauge 9. Solid line: model, stars: data.

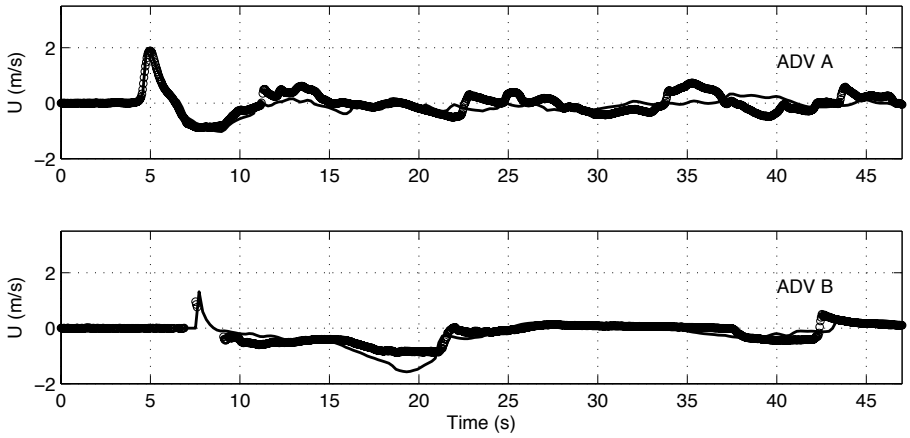

Figure 21: Model/data comparisons of time series of velocity at (top) ADV 1 and (bottom) ADV 2. Solid line: model, stars: data.

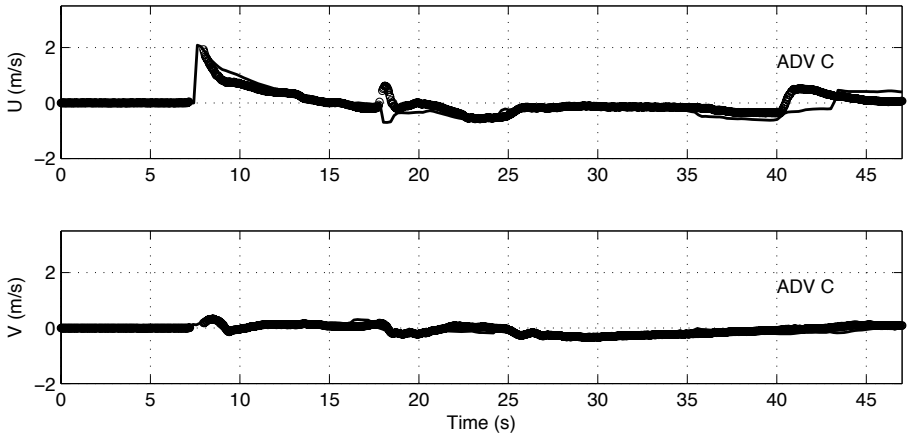

Figure 22: Model/data comparisons of time series of velocity in (top) x direction and (bottom) y direction at ADV 3. Solid line: model, stars: data.

```
    ! ————————DIMENSION————————————
Mglob = 501
Nglob = 261

    ! ———————— TIME————————————————-
TOTAL_TIME = 45.0
PLOT_INTV = 0.2
PLOT_INTV_STATION = 0.2
SCREEN_INTV = 0.2

    ! ————————GRID————————————————-
DX = 0.1
DY = 0.1

    ! ————————-WAVEMAKER————————————
WAVEMAKER = INI_SOL
AMP = 0.39
DEP = 0.78
LAGTIME = 5.0
XWAVEMAKER = 10.0 ! note that means x = 5 m because the domain starts at x = -5

    ! ————————- SPONGE LAYER ———————————
SPONGE_ON = F

    ! ————————-PHYSICS————————————
DISPERSION = T
Gamma1 = 1.0
Gamma2 = 1.0
Gamma3 = 1.0
Beta_ref=-0.531
SWE_ETA_DEP = 0.80
Cd = 0.001

    ! ————————-NUMERICS————————————-
Time_Scheme = Runge_Kutta
HIGH_ORDER = FOURTH
CONSTRUCTION = HLLC
CFL = 0.5

    ! ————————WET-DRY————————————————-
```

MinDepth=0.001
MinDepthFrc = 0.001

! ————————OUTPUT————————————
DEPTH_OUT = T
U = T
V = T
ETA = T
MASK = T

Several post-processing procedures are performed in order to get model/data comparisons. First, *read_result.m* is used to obtained time series of η at 9 pressure gauges , and u and v at 3 ADV locations. Then, *plot_WG.m* is used for comparisons of surface elevation at the 9 gauges, *plot_ADV_AB.m* for comparisons of u at ADV 1 and 2, and *plot_ADV_C.m* for u and v at ADV 3.

References

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

Bermùdez, A. and Vàzquez, M. E., 1994, "Upwind methods for hyperbolic conservation laws with source terms", *Comput. Fluids*, 23 (8), 1049-1071.

Briggs, M. J., Synolakis, C. E., and Harkins, G. S., 1994, "Tsunami run- up on a conical island", *Proc., Waves-Physical and Numerical Modelling*, International Association for Hydraulic Research, Delft, The Netherlands, 446-455.

Chen, Q., Dalrymple, R. A., Kirby, J. T., Kennedy, A. B. and Haller, M. C., 1999, "Boussinesq modelling of a rip current system", em J. Geophys. Res., 104, 20,617-20,637.

Chen, Q., Kirby, J. T., Dalrymple, R. A., Kennedy, A. B. and Chawla, A., 2000, "Boussinesq modeling of wave transformation, breaking and runup. II: 2D", *J. Waterway, Port, Coastal and Ocean Engineering*, **126**, 48-56.

Chen, Q., Kirby, J. T., Dalrymple, R. A., Shi, F. and Thornton, E. B., 2003, "Boussinesq modeling of longshore currents", *Journal of Geophysical Research*, 108(C11), 3362, doi:10.1029/2002JC001308.

Chen, Q., 2006, "Fully nonlinear Boussinesq-type equations for waves and currents over porous beds", *Journal of Engineering Mechanics.* 132 (2): 220-230.

Erduran, K. S., Ilic, S., and Kutija, V., 2005, "Hybrid finite-volume finite-difference scheme for the solution of Boussinesq equations", *Int. J. Numer. Meth. Fluid.*, 49, 1213-1232.

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

Gottlieb, S., Shu C.-W., and Tadmore, E., 2001, "Strong stability-preserving high-order time discretization methods", *SIAM Review*, **43** (1), 89 - 112.

Hansen, J. B., and Svendsen, I. A., 1979, "Regular waves in shoaling water: Experimental data", Tech. Rep. ISVA Ser., 21, Technical Univ. of Denmark, Denmark.

Kennedy, A. B., Chen, Q., Kirby, J. T. and Dalrymple, R. A., 2000, "Boussinesq modeling of wave transformation, breaking and runup. I: 1D", *J. Waterway, Port, Coastal and Ocean Engineering*, **126**, 39-47.

Kennedy, A. B., Kirby, J. T., Chen, Q. and Dalrymple, R. A., 2001, "Boussinesq-type equations with improved nonlinear performance", *Wave Motion*, 33, pp. 225-243.

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", *Journal of Engineering Mechanics*, 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 Modeling*, **27**, 198-214.

Kim, D. H., 2009, "Turbulent flow and transport modeling by long waves and currents", Ph.D. dissertation, Texas A& M University.

Kirby, J. T., Wei, G., Chen, Q., Kennedy, A. B. and Dalrymple, R. A., 1998, "FUNWAVE 1.0, Fully nonlinear Boussinesq wave model. Documentation and users manual". Report CACR-98-06, Center for Applied Coastal Research, Department of Civil and Environmental Engineering, University of Delaware.

Kirby, J.T., Shi, F., Watts, P., Grilli, S.T., 2004, "Propagation of short, dispersive tsunami waves in ocean basins". EOS Transactions of the AGU 85 (47) Abstract OS21E-02.

Kirby, J. T., Shi, F., Harris, J. C., and Grilli, S. T., 2011, "Sensitivity analysis of trans-oceanic tsunami propagation to dispersive and Coriolis effects", in preparation.

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

Long, W. and Kirby, J. T., 2006, "Boussinesq modeling of waves, currents and sediment transport", Research Report No. CACR-06-02, Center for Applied Coastal Research, Dept. of Civil and Environmental Engineering, Univ. of Delaware, Newark.

Lynett, P. J., Wu, T.-R. and Liu, P. L.-F., 2002, "Modeling wave runup with depth-integrated equations", *Coastal Engineering*, 46, 89-107.

Madsen, P.A., Srensen, O.R., 1992, "A new form of the Boussinesq equations with improved linear dispersion characteristics. Part 2. A slowly-varying bathymetry", *Coastal Engineering* 18 (3-4), 183-204.

Mase, H., and Kirby, J. T., 1992, "Hybrid frequency-domain KdV equation for random wave transformation", *Proc., 23rd Int. Conf. Coast. Eng.*, ASCE, New York, 474-487.

Naik, N. H., Naik, V. K., and Nicoules, M., 1993, "Parallelization of a class of implicit finite difference schemes in computational fluid dynamics", *International Journal of High Speed Computing*, 5: 1-50.

Ning, D. Z., Zang, J., Liang, Q., Taylor, P. H., and Borthwick, A. G. L., 2008, "Boussinesq cut-cell model for non-linear wave interaction with coastal structures", *International Journal for Numerical Methods in Fluids*, 57 (10), 1459-1483.

Nwogu, O., 1993, "An alternative form of the Boussinesq equations for nearshore wave propagation", *Journal of Waterway, Port, Coastal, and Ocean Engineering*, 119 (6), pp. 618-638.

Nwogu, O. and Demirbilek, Z., 2001, "BOUSS-2D: A Boussinesq wave model for coastal regions and harbors", ERDC/CHL TR-01-25, Coastal and Hydraulics Laboratory, USACOE Engineer Research and Development Center, Vicksburg, MS.

Roeber, V., Cheung, K. F., and Kobayashi, M. H., 2010, "Shock-capturing Boussinesq-type model for nearshore wave processes", *Coastal Engineering*, 57, 407-423.

Rogers, B. D., Borthwick, A. G. L., and Taylor, P. H., 2003, "Mathematical balancing of flux gradient and source terms prior to using Roe's approximate Riemann solver", *Journal of Computational Physics*, 192, 422-451.

Shi, F., Kirby, J. T., Harris, J. C., Geiman, J. D., and Grilli, S. T., 2011, "A high-order adaptive time-stepping TVD solver for Boussinesq modeling of breaking waves and coastal inundation", to be submitted to *Ocean Modelling*.

Shi, F., Dalrymple, R. A., Kirby, J. T., Chen, Q. and Kennedy, A., 2001, "A fully nonlinear Boussinesq model in generalized curvilinear coordinates". *Coastal Engineering*, Vol. 42, pp. 337-358.

Shiach, J. B. and Mingham, C. G., 2009, "A temporally second-order accurate Godunov-type scheme for solving the extended Boussinesq equations", *Coastal Engineering*, 56, 32-45.

Sitanggang, K. I. and Lynett, P., 2005, "Parallel computation of a highly nonlinear Boussinesq equation model through domain decomposition", *Int. J. Num. Meth. Fluids*, **49**, 57-74.

Smagorinsky, J., 1963, "General circulation experiments with the primitive equations. I. The basic experiment", *Mon. Weather Rev*, 91, 99-165.

Synolakis, C. E., Bernard, E. N., Titov, V. V., Kânoglu, U. and González, F. I., 2007, "Standards, criteria, and procedures for NOAA evaluation of tsunami numerical models", *NOAA Tech. Memo. OAR PMEL-135*, National Oceanic and Atmospheric Administration.

Tehranirad, B., Shi, F., Kirby, J. T., Harris, J. C. and Grilli, S., 2011, "Tsunami benchmark results for fully nonlinear Boussinesq wave model FUNWAVE-TVD, Version 1.0", Research Report No. CACR-11-02, Center for Applied Coastal Research, University of Delaware.

Ting, F.C.K., Kirby, J.T., 1994, "Observation of undertow and turbulence in a laboratory surf zone". *Coast. Eng.* 24, 5180.

Tonelli, M. and Petti, M., 2009, "Hybrid finite volume - finite difference scheme for 2DH improved Boussinesq equations", *Coast. Engrng.*, **56**, 609-620.

Tonelli, M. and Petti, M., 2010, "Finite volume scheme for the solution of 2D extended Boussinesq equations in the surf zone", *Ocean Engrng.*, **37**, 567-582.

Toro, E. F., 2009, *Riemann solvers and numerical methods for fluid dynamics: a practical introduction*, Third edition, Springer, New York.

Wei, G., Kirby, J.T., Grilli, S.T., Subramanya, R., 1995, "A fully nonlinear Boussinesq model for surface waves: Part I. Highly nonlinear unsteady waves", *Journal of Fluid Mechanics* 294, 7192.

Wei, G. and Kirby, J. T., 1995, "A time-dependent numerical code for extended Boussinesq equations", *Journal of Waterway, Port, Coastal and Ocean Engineering*, 120, pp. 251-261.

Yamamoto, S., Daiguji, H., 1993, "Higher-order-accurate upwind schemes for solving the compressible Euler and NavierStokes equations", *Computers and Fluids*, 22 (2/3), 259270.

Yamamoto, S., Kano, S. and Daiguji, H, 1998, "An efficient CFD approach for simulating unsteady hypersonic shockshock interference flows", *Computers and Fluids* 27 (56), pp. 571-580.

Zelt, J. A., 1991, "The runup of nonbreaking and breaking solitary waves", *Coastal Engineering*, 15, pp. 205-246.

Zhou, J. G., Causon, 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", *Journal of Computational Physics*, 168, 1-25.