FUNWAVE 1.0
Fully Nonlinear Boussinesq Wave Model
Documentation and User's Manual

James T. Kirby, Ge Wei, Qin Chen,
Andrew B. Kennedy, and Robert A. Dalrymple

Center for Applied Coastal Research
Department of Civil Engineering
University of Delaware, Newark, DE 19716

Research Report NO. CACR-98-06
September, 1998

Acknowledgements

The development and application of the numerical model FUNWAVE were supported by the U.S. Army Research Office through University Research Initiative Grant No. DAAL03-92-G-0116, and by the U.S. Office of Naval Research, Base Enhancement Program, through Grant N00014-97-0283. Many thanks are due to H. Mase, A. Chawla, and M.J. Briggs who provided the experimental data for comparisons, and R. Fusina for his comments on the manuscript.

Abstract

This report documents the computer program *FUNWAVE* based on the fully nonlinear Boussinesq model of Wei et al. (1995). The documentation provides a description of the governing equations and the numerical scheme used to solve it. A user's manual is provided and gives instructions on how to use various preprocessors and postprocessors to set up and read data files needed for model runs. Fortran code is provided for one and two-dimensional versions of the model, as well as for the additional data processing programs. Finally, several example calculations are documented.

Contents

List of Figures

1 THEORETICAL BACKGROUND

This section provides an overview of the theory incorporated in FUNWAVE. A more extensive review of the topic of shallow water wave propagation may be found in Kirby (1997).

Boussinesq-type equations provide a general basis for studying wave propagation in two horizontal dimensions. At their core, the equations are the shallow water equations for nondispersive linear wave propagation. This basic foundation is extended by the addition of terms which include the lowest order effects of nonlinearity and frequency dispersion. This formulation provides a sound and increasingly well-tested basis for the simulation of wave propagation in coastal regions.

The standard Boussinesq equations for variable water depth were first derived by Peregrine (1967), who used the depth-averaged velocity as a dependent variable. Numerical models based on Peregrine's equations or equivalent formulations have been shown to give predictions which compare quite well with field data (Elgar and Guza 1985) and laboratory data (Goring 1978; Liu, Yoon and Kirby 1985; Rygg 1988), when applied within their range of validity.

The assumption of weak frequency dispersion effects makes the standard Boussinesq equations invalid in intermediate and deep water. The corresponding dispersion relation of the standard Boussinesq equations is a polynomial approximation to the exact solution based on a hyperbolic tangent function, which only matches well in shallow water areas. Recently, extended forms of Boussinesq equations have been derived by Madsen *et al.* (1991) and Nwogu (1993), among others. In the derivation of Madsen *et al.* (1991), additional terms which are formally equivalent to zero within the accuracy of the model are introduced to the momentum equations. The form of these terms was governed by the constraint of obtaining the best possible linear dispersion relation and the optimal linear shoaling property (Madsen and Sorensen, 1992). Nwogu (1993) used the velocity at a certain depth as a dependent variable and pursued a consistent derivation of the governing equations using this non-standard dependent variable. In the end, the choice of the representative depth was again constrained by the goal of obtaining the most accurate possible dispersion relation. Although the methods of derivation are different, the resulting dispersion relations of these extended Boussinesq equations are similar, and may be thought of as a slight modification of the (2,2) Padé approximant of the full dispersion relation (Witting, 1984). The relation may all be written in the form

$$\omega^2 = ghk^2 \frac{1-(\alpha+\frac{1}{3})(kh)^2}{1-\alpha(kh)^2} \tag{1}$$

Figure 1 shows the comparison of exact dispersion relation with those from Nwogu's equations for several values of α, which corresponding to different models. In shallow water limit as $kh \to 0$, all curves approach together to $\sqrt{gh}$. However, as kh values increase, discrepancies from the exact solution become large. The dispersion relation from Nwogu's Boussinesq equations for the optimal value $\alpha = -0.390$ is much closer to the exact solution in intermediate water depths than that from standard Boussinesq equations. Madsen and Sorensen (1992), and Nwogu (1993) have shown by example that the extended equations are able to simulate wave propagation from relatively deep to shallow water. Wei and Kirby (1995) developed a high order numerical model based on Nwogu's equations, and provided additional validation tests of the model.

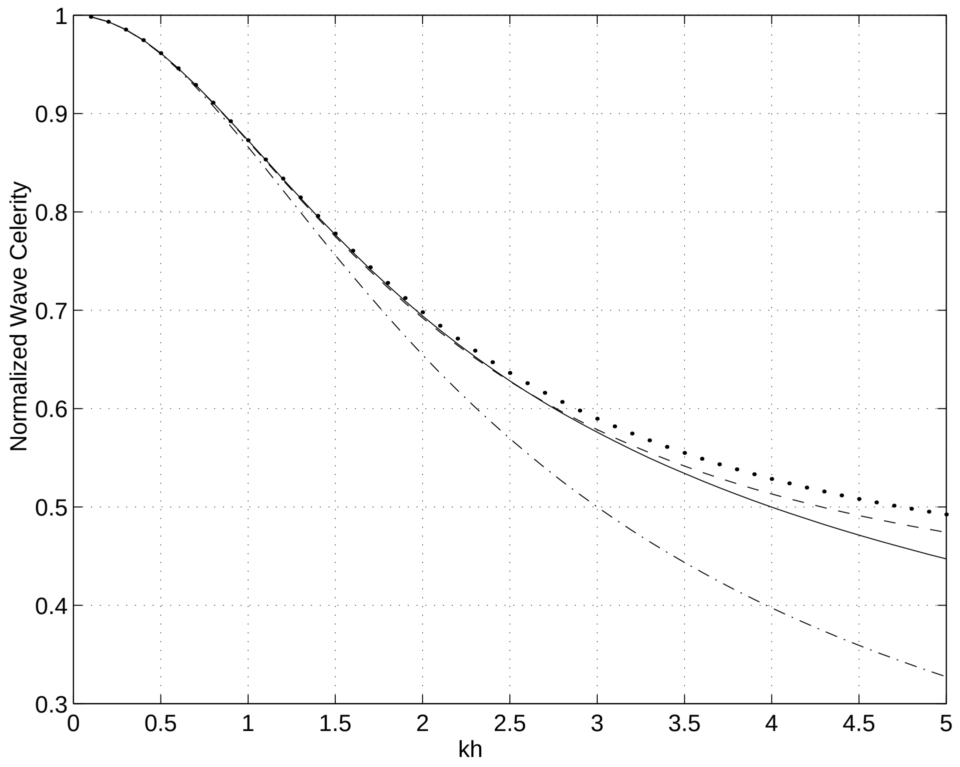


Figure 1: Comparison of linear dispersion relations: solid line – exact solution; dash line – $\alpha = -0.390$, optimal value ; dotted line – $\alpha = -2/5$, (2,2) Padé approximant; dash-dot line – $\alpha = -1/3$, standard Boussinesq equations.

Despite their improved dispersion relation, the extended Boussinesq equations are still restricted to situations with weakly nonlinear interactions. In many practical cases, however, the effects of nonlinearity are too large to be treated as a weak perturbation to a primarily linear problem. As waves approach shore, wave height increases due to the effect of shoaling, and wave breaking occurs on most gentle natural slopes. The wave height to water depth ratios accompanying this physical process are inappropriate for weakly-nonlinear Boussinesq models, and thus extensions to the model are required in order to obtain a computational tool which is locally valid in the vicinity of a steep, almost breaking or breaking wave crest. Moreover, as shown by Chen *et al.* (1998), both sets of equations introduced by Madsen and Sorensen (1992) and Nwogu (1993) are not applicable to combined wave/current motions which are often encountered in nearshore regions.

Adapting the approach of Nwogu (1993) but making no assumption of small nonlinear effect, Wei *et al.* (1995) derived a new set of Boussinesq equations which include additional nonlinear dispersive terms. Not only can the equations be applied to intermediate water depth as the extended Boussinesq equations with improved dispersion relation, but also be capable for simulating wave propagation with strong nonlinear interaction. The inclusion of full nonlinearity also leads to a correct form of Doppler Shift in the equations when an ambient current is present. In other words, the new equations are suitable for modelling wave/current interaction as shown by Kirby (1997). Numerical results obtained with this model have been compared to boundary integral solutions of the full 2-D vertical problem, and have been shown to be quite accurate in predicting the form of solitary waves either propagating in constant depth or shoaling up a planar beach. Additional validation tests are described in the user's manual below.

To enable the Boussinesq model to simulate surf zone hydrodynamics, energy dissipation due to wave breaking is modelled by introducing an eddy viscosity term into the equations, with the viscosity strongly localized on the front face of the breaking waves. Wave runup on the beach is simulated using a permeable-seabed techniques. Both breaking and runup schemes are detailed in Kennedy *et al.* (1999) and Chen *et al.* (1999b). A brief description and verifications of the surf zone model are also given in this manual.

1.1 Model Equations: Two Spatial Dimensions

We now provide the mathematical basis for the model FUNWAVE. The numerical scheme used to obtain solutions of these equations is described in the following section. The fully nonlinear Boussinesq equations derived by Wei *et al.* (1995) are given by

$$\begin{aligned}
\eta_t \quad &+ \quad \nabla\cdot\Big\{(h+\eta)\Big[\,\mathbf{u}_\alpha+\Big(z_\alpha+\tfrac{1}{2}(h-\eta)\Big)\nabla\Big(\nabla\cdot(h\mathbf{u}_\alpha)\Big)\\
&\qquad\qquad +\Big(\tfrac{1}{2}z_\alpha^2-\tfrac{1}{6}(h^2-h\eta+\eta^2)\Big)\nabla(\nabla\cdot\mathbf{u}_\alpha)\Big]\Big\}=0 \qquad (2)\\
\mathbf{u}_{\alpha t} \quad &+ \quad (\mathbf{u}_\alpha\cdot\nabla)\mathbf{u}_\alpha+g\nabla\eta+z_\alpha\Big\{\tfrac{1}{2}z_\alpha\nabla(\nabla\cdot\mathbf{u}_{\alpha t})+\nabla\Big(\nabla\cdot(h\mathbf{u}_{\alpha t})\Big)\Big\}
\end{aligned}$$

$$
\begin{aligned}
&+ \ \nabla\left\{\tfrac{1}{2}(z_\alpha^2-\eta^2)(\mathbf{u}_\alpha\cdot\nabla)(\nabla\cdot\mathbf{u}_\alpha)+\tfrac{1}{2}\left[\nabla\cdot(h\mathbf{u}_\alpha)+\eta\nabla\cdot\mathbf{u}_\alpha\right]^2\right\} \\
&+ \ \nabla\left\{(z_\alpha-\eta)(\mathbf{u}_\alpha\cdot\nabla)\big(\nabla\cdot(h\mathbf{u}_\alpha)\big)-\eta\left[\tfrac{1}{2}\eta\nabla\cdot\mathbf{u}_{\alpha t}+\nabla\cdot(h\mathbf{u}_{\alpha t})\right]\right\}=0 \qquad (3)
\end{aligned}
$$

where η is the surface elevation, h is the still water depth, $\mathbf{u}_\alpha$ is the horizontal velocity vector at the water depth $z = z_\alpha = -0.531h$, $\nabla = (\partial/\partial x, \partial/\partial y)$ is the horizontal gradient operator, g is the gravitational acceleration, and subscript t is the partial derivative with respect to time. Equations (2) and (3) are statements of conservation of mass and momentum, respectively. As detailed below, equations (2) and (3) may be transformed into different equations governing wave propagation by dropping certain terms and/or changing certain coefficients.

Equations (2) and (3) describe the frictionless evolution of nonbreaking waves over a smooth, impermeable bottom. In order to develop a model for practical application, several effects have to be incorporated into the model scheme, including physical effects of frictional damping and wave breaking, as well as extensions needed to perform purely numerical tasks including wave generation, boundary absorption, and moving shoreline. We rewrite equations (2) and (3) including these extensions as

$$
\begin{aligned}
\eta_t &= E(\eta,u,v)+\gamma E_2(\eta,u,v)+f(x,y,t) && (4)\\
[U(u)]_t &= F(\eta,u,v)+[F_1(v)]_t+\gamma[F_2(\eta,u,v)+F^t(\eta,u_t,v_t)] \\
&\quad +F_b+F_{br}+F_{bs}+F_{sp} && (5)\\
[V(v)]_t &= G(\eta,u,v)+[G_1(u)]_t+\gamma[G_2(\eta,u,v)+G^t(\eta,u_t,v_t)] \\
&\quad +G_b+G_{br}+G_{bs}+G_{sp} && (6)
\end{aligned}
$$

Here u and v are the horizontal velocities in horizontal directions x and y at depth $z = z_\alpha = -0.531h$, *i.e.*, $(u,v) = \mathbf{u}_\alpha$, and γ is a control parameter allowing us to choose between fully ($\gamma = 1$) or weakly ($\gamma = 0$) nonlinear cases. The quantities $U, V, E, E_2, F, F_1, F_2, G, G_1, G_2, F^t$ and G^t are functions of η, u, v, u_t or v_t which are defined as

$$
\begin{aligned}
U &= u+h\left[b_1hu_{xx}+b_2(hu)_{xx}\right] && (7)\\
V &= v+h\left[b_1hv_{yy}+b_2(hv)_{yy}\right] && (8)\\
E &= -\frac{1}{\kappa}[(\Lambda u)_x+(\Lambda v)_y] \\
&\quad -\left\{a_1h^3(u_{xx}+v_{xy})+a_2h^2[(hu)_{xx}+(hv)_{xy}]\right\}_x \\
&\quad -\left\{a_1h^3(u_{xy}+v_{yy})+a_2h^2[(hu)_{xy}+(hv)_{yy}]\right\}_y && (9)\\
F &= -g\eta_x-(uu_x+vu_y) && (10)\\
G &= -g\eta_y-(uv_x+vv_y) && (11)\\
F_1 &= -h\left[b_1hv_{xy}+b_2(hv)_{xy}\right] && (12)\\
G_1 &= -h\left[b_1hu_{xy}+b_2(hu)_{xy}\right] && (13)
\end{aligned}
$$

$$
\begin{aligned}
E_2 &= -\left\{\left[a_1 h^2\eta + \tfrac{1}{6}\eta(h^2-\eta^2)\right](u_{xx}+v_{xy})\right\}_x \\
&\quad -\left\{\left[a_2 h\eta - \tfrac{1}{2}\eta(h+\eta)\right][(hu)_{xx}+(hv)_{xy}]\right\}_x \\
&\quad -\left\{\left[a_1 h^2\eta + \tfrac{1}{6}\eta(h^2-\eta^2)\right](u_{xy}+v_{yy})\right\}_y \\
&\quad -\left\{\left[a_2 h\eta - \tfrac{1}{2}\eta(h+\eta)\right][(hu)_{xy}+(hv)_{yy}]\right\}_y && (14) \\
F_2 &= -\left\{\tfrac{1}{2}({z_\alpha}^2-\eta^2)[u(u_x+v_y)_x+v(u_x+v_y)_y]\right\}_x \\
&\quad -\left\{(z_\alpha-\eta)\left[u[(hu)_x+(hv)_y]_x+v[(hu)_x+(hv)_y]_y\right]\right\}_x \\
&\quad -\tfrac{1}{2}\left\{[(hu)_x+(hv)_y+\eta(u_x+v_y)]^2\right\}_x && (15) \\
G_2 &= -\left\{\tfrac{1}{2}({z_\alpha}^2-\eta^2)[u(u_x+v_y)_x+v(u_x+v_y)_y]\right\}_y \\
&\quad -\left\{(z_\alpha-\eta)\left[u[(hu)_x+(hv)_y]_x+v[(hu)_x+(hv)_y]_y\right]\right\}_y \\
&\quad -\tfrac{1}{2}\left\{[(hu)_x+(hv)_y+\eta(u_x+v_y)]^2\right\}_y && (16) \\
F^t &= \left\{\tfrac{1}{2}\eta^2[(u_t)_x+(v_t)_y]+\eta\left[\,[h(u_t)]_x+[h(v_t)]_y\right]\right\}_x && (17) \\
G^t &= \left\{\tfrac{1}{2}\eta^2[(u_t)_x+(v_t)_y]+\eta\left[\,[h(u_t)]_x+[h(v_t)]_y\right]\right\}_y && (18)
\end{aligned}
$$

where a_1, a_2, b_1, b_2 are constants which are related to the dimensionless reference water depth $\beta = z_\alpha/h = -0.531$ by

$$a_1 = \frac{1}{2}\beta^2 - \frac{1}{6}, \qquad a_2 = \beta + \frac{1}{2}, \qquad b_1 = \frac{1}{2}\beta^2, \qquad b_2 = \beta \tag{19}$$

The extended Boussinesq equations of Nwogu (1993) can be recovered by setting $\gamma = 0$. For standard Boussinesq equations of Peregrine (1967), we replace velocity at a certain depth $\mathbf{u}_\alpha$ by depth-averaged velocity $\bar{\mathbf{u}}$ in the governing equation and define the control parameters as:

$$\gamma = 0, \qquad a_1 = 0, \qquad a_2 = 0, \qquad b_1 = \frac{1}{6}, \qquad b_2 = -\frac{1}{2} \tag{20}$$

Nonlinear shallow water equations can be obtained by replacing $\mathbf{u}_\alpha$ as $\bar{\mathbf{u}}$ and setting

$$\gamma = a_1 = a_2 = b_1 = b_2 = 0 \tag{21}$$

Notice that Λ and κ in (9) result from the use of a permeable seabed or slot technique for simulation of shoreline runup. They are described in Section 1.5. In the absence of the slot scheme, $\Lambda = h + \eta$ and $\kappa = 1$.

The remaining terms are added to the model to perform specific tasks. First, the term $f(x, y, t)$ in (4) is the source function for wave generation, described in section 1.3 below and in more detail in Wei et al. (1999).

Secondly, the vector (F_b, G_b) in (5-6) is the bottom friction vector, given by

$$(F_b, G_b) = \frac{K}{h+\eta}\mathbf{u}_\alpha|\mathbf{u}_\alpha| \tag{22}$$

K is the friction coefficient and has been set to as $K = 1\times 10^{-5}$, pending more careful studies of mean flow generation in the model.

Thirdly, the vector (F_{br}, G_{br}) appearing in (5)-(6) is the wave breaking model, described in Section 1.4. Smogorinsky-type subgrid model is introduced by (F_{bs}, G_{bs}) to account for the effects of unresolved turbulence on the computed flow field. They are discussed in Section 1.6.

Finally, the vector (F_{sp}, G_{sp}) provides for wave absorption by damping at model boundaries, and is described in section 2.2.2.

1.2 Model Equations: One Spatial Dimension

The model detailed above is considerably faster to run as a purely one-dimensional code if one-dimensional results are sought, as the two-dimensional version must be a minimum of five grid spaces wide. The governing equations for the one-dimensional version are given here.

$$\begin{aligned}
\eta_t &= E(\eta,u) + \gamma E_2(\eta,u) + f(x,t) && (23)\\
[U(u)]_t &= F(\eta,u) + \gamma[F_2(\eta,u) + F^t(\eta,u_t)] + F_{br} + F_b + F_{sp} && (24)
\end{aligned}$$

The quantities U, E, E_2, F, F_2 and F^t involve spatial derivatives of η, u, and u_t and are defined by

$$\begin{aligned}
U &= u + h\left[b_1 h u_{xx} + b_2 (hu)_{xx}\right] && (25)\\
E &= -\frac{1}{\kappa}(\Lambda u)_x - \left\{a_1 h^3 u_{xx} + a_2 h^2 (hu)_{xx}\right\}_x && (26)\\
F &= -g\eta_x - u u_x && (27)\\
E_2 &= -\left\{\left[a_1 h^2 \eta + \tfrac{1}{6}\eta(h^2-\eta^2)\right] u_{xx}\right\}_x - \left\{\left[a_2 h\eta - \tfrac{1}{2}\eta(h+\eta)\right](hu)_{xx}\right\}_x && (28)\\
F_2 &= -\left\{\frac{1}{2}({z_\alpha}^2 - \eta^2) u u_{xx}\right\}_x - \left\{(z_\alpha - \eta)u(hu)_{xx}\right\}_x - \frac{1}{2}\left\{[(hu)_x + \eta(u_x)]^2\right\}_x && (29)\\
F^t &= \left\{\frac{1}{2}\eta^2 u_{xt} + \eta(hu_t)_x\right\}_x && (30)
\end{aligned}$$

where a_1, a_2, b_1, b_2 and γ are defined in (19)- (21), and Λ and κ are defined in 1.5.

1.3 Wave Generation by Source Function

In keeping with the idea that we are solving an initial boundary value problem, it would be desirable to develop boundary conditions for Boussinesq models which can provide a combination of wave generation, wave absorption, and wave reflection effects. However, the problem of providing a well-posed boundary value problem for these equations is essentially an unsolved problem. In particular, the problem of providing absorbing conditions which will work for the entire range of modeled frequencies is complicated immensely by the wide range of phase speeds seen at the boundary.

In recognition of the difficulties imposed by this approach to model operation, we follow the lead of a number of previous Boussinesq model developers and, instead, generate waves using an internal source mechanism. Such an approach has been documented previously by Larsen and Dancy (1983) who used a somewhat ad-hoc source mechanism where water mass is added and subtracted along a straight source/sink line inside the computing domain. Sponge layers are placed in the ends of the domain to effectively damp the energy of outgoing waves with different frequencies and directions. This approach works well in a staggered-grid differencing scheme, where water is essentially being added to or drained from a single grid block. In applying this technique to the Boussinesq model on an unstaggered grid, however, we found that use of a single source line caused high frequency noise, leading to blowup of the model. Instead, we use a spatially distributed mass source $f(x, y, t)$ in equation (4). The corresponding linearized theory for the distributed source function is given in Wei et al. (1999).

Assume that in a constant water depth of h we want to generate a plane wave with amplitude a_0 and angular frequency ω. The angle between the propagation direction of the wave and the x-axis is θ, as shown in Figure 2. Without losing generality, we assume that the center of the source region is parallel to the y-axis. Then we split the source function $f(x, y, t)$ into two parts as

$$f(x, y, t) = g(x)s(y, t) \tag{31}$$

where $g(x)$ is a Gaussian shape function and $s(y, t)$ the input time series of the magnitude of source function. It is convenient to make this separation, since the dimensionality of the input signal required to run the model, $s(y, t)$, is reduced by one relative to $f(x, y, t)$. The functions $g(x)$ and $s(y, t)$ are defined as

$$\begin{aligned} g(x) &= \exp[-\beta(x - x_s)^2] && (32) \\ s(y, t) &= D \sin(\lambda y - \omega t) && (33) \end{aligned}$$

where β is the shape coefficient for the source function, and x_s is the central location of the source in the x direction, for a source oriented parallel to the y axis, as shown in Figure 2. The model also provides for the presence of sources along the lateral boundaries parallel to the x axis. D is the magnitude of the source function, $\lambda = k\sin(\theta)$ the wavenumber in the y direction, and k is the linear wavenumber.

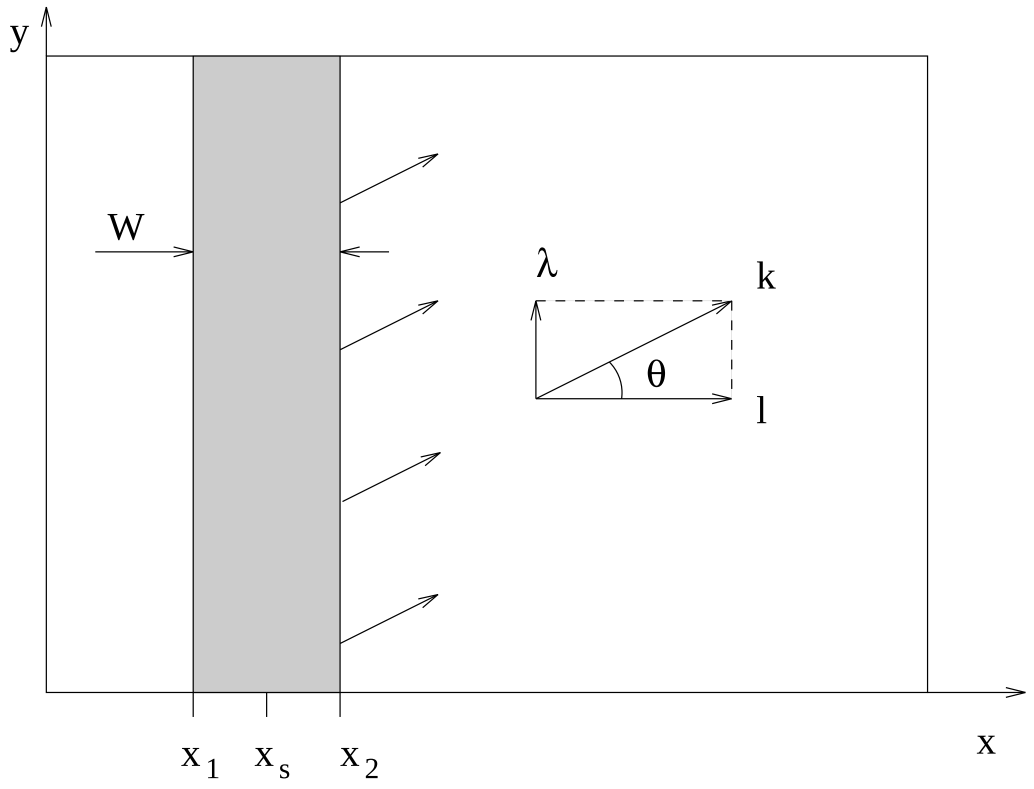


Figure 2: Source function definition in the computing domain.

For a monochromatic wave or a single wave component of a random wave train, the magnitude D of source function can be determined by

$$D = \frac{2a_0\cos(\theta)(\omega^2 - \alpha_1 g k^4 h^3)}{\omega k I[1-\alpha(kh)^2]} \tag{34}$$

where $\alpha = -0.390$, $\alpha_1 = \alpha + 1/3$, and I is the integral given by

$$I = \int_{-\infty}^{\infty} \exp(-\beta x^2)\exp(-ilx)dx = \sqrt{\frac{\pi}{\beta}}\exp(-l^2/4\beta) \tag{35}$$

where $l = k\cos(\theta)$ is the wavenumber in x direction. In theory, the shape coefficient β can be any number. The larger the value β is, the narrower the source function becomes. However, in our numerical model, good results were obtained when the width of the source function W equals about half of the wavelength for monochromatic wave. The definition of W is not unique, and here we define W to be the distance between two coordinates x_1 and x_2 where the corresponding source function heights are equal to $\exp(-5) = 0.0067$ times the maximum height D. Then x_1 and x_2 must satisfy the quadratic equation

$$\beta(x - x_s)^2 = 5 \tag{36}$$

from which the width of source function is given by

$$W = |x_2 - x_1| = 2\sqrt{\frac{5}{\beta}} \tag{37}$$

If L is the wavelength, the requirement of $W = \delta L/2$ (where δ is of order 1) results in

$$\beta = \frac{5}{(\delta L/4)^2} = \frac{80}{\delta^2 L^2} \tag{38}$$

For random waves, the value of β is determined according to the peak frequency component and then used for all components in the wave train.

1.4 Wave Breaking

Wave breaking in Boussinesq models has been modeled using a range of techniques. It appears that a technique which is capable of preserving the shape of the breaking wave as well as modeling wave height decay requires a dissipation mechanism which is spatially and temporally localized and tied to the front face of the wave. Available techniques of this type range (chronologically and in complexity) from the eddy viscosity formulation of Zelt (1991), through the momentum correction - surface roller model of Schäffer et al (1993), to the similar model Svendsen et al. (1996). At present, there is very little evidence showing that any formulation significantly outperforms any other, and so we utilize the eddy viscosity formulation similar to that of Zelt (1991), but with extension to provide a more realistic description of the initiation and cessation of wave breaking.

Following Kennedy et al. (1999), we model the energy dissipation due to wave breaking in shallow water by introducing the momentum mixing terms:

$$F_{br} = \frac{1}{h+\eta}\left[(\nu((h+\eta)u_\alpha)_x)_x + \frac{1}{2}\left(\nu(((h+\eta)u_\alpha)_y + ((h+\eta)v_\alpha)_x)\right)_y\right] \tag{39}$$

$$G_{br} = \frac{1}{h+\eta}\left[\frac{1}{2}\left(\nu(((h+\eta)v_\alpha)_x + ((h+\eta)u_\alpha)_y)\right)_x + (\nu((h+\eta)v_\alpha)_y)_y\right] \tag{40}$$

where superscripts x and y represent the directions in the horizontal plane, subscripts x and y denote spatial derivatives, and ν is the eddy viscosity localized on the front face of the breaking wave. It should be emphasised that the localization of the eddy viscosity is of importance for modelling nonlinear waves. In contrast, a global eddy viscosity would smear the asymmetry and skewness of the breaking waves in a non-physical manner.

We define eddy viscosity as

$$\nu = B\delta^2 |(h + \eta)\nabla\cdot\mathbf{M}| \tag{41}$$

in which δ is a mixing length coefficient with an empirical value of $\delta = 1.2$. The quantity B that controls the occurrence of energy dissipation is given by

$$B = \begin{cases} 1, & \eta_t \geq 2\eta_t^* \\ \frac{\eta_t}{\eta_t^*} - 1, & \eta_t^* < \eta_t \leq 2\eta_t^* \\ 0, & \eta_t \leq \eta_t^* \end{cases} \tag{42}$$

In analogy to the "roller" model by Schäffer et al. (1993), we determine the onset and cessation of wave breaking using the parameter, η_t^*, which is defined as

$$\eta_t^* = \begin{cases} \eta_t^{(F)}, & t \geq T^* \\ \eta_t^{(I)} + \frac{t-t_0}{T^*}(\eta_t^{(F)} - \eta_t^{(I)}), & 0 \leq t - t_0 < T^* \end{cases} \tag{43}$$

where T^* is the transition time, t_0 is the time when wave breaking occurs, and $t - t_0$ is the age of the breaking event. The value of $\eta_t^{(I)}$ varies between $0.35\sqrt{gh}$ and $0.65\sqrt{gh}$, while the values of $\eta_t^{(F)}$, and T^* are $0.15\sqrt{gh}$, and $5\sqrt{h/g}$, respectively. The construction and verification of the breaking model was detailed by Kennedy et al. (1999). The lower limit of $\eta_t^{(I)}$ is found to be more suitable to bar/trough beaches while the upper limit gives optimal agreement for waves breaking on monotone sloping beaches. Chen et al. (1999b) described the implementation and verification of the breaking model in two horizontal dimensions.

1.5 A Treatment of Moving Shorelines

Instead of tracking the moving boundary during wave runup/run-down on the beach, we treat the entire computational domain as an active fluid domain by employing an improved version of the slot or permeable-seabed technique proposed by Tao (1983, 1984) for simulation of runup. The original slot technique has been used by Madsen et al. (1997) in a Boussinesq model formulated in terms of mass flux and free surface elevation. The basic idea behind this technique is to replace the solid bottom where there is very little or no water covering the land by a porous seabed, or to assume that the solid bottom contains narrow slots. This allows the water level to be below the beach elevation.

The replacement of the solid bottom by narrow slots results in a modification of the mass equation (i.e. (4) and (9)), where

$$\kappa = \begin{cases} \delta + (1-\delta)e^{\lambda\frac{(\eta - z^*)}{h_0}}, & \eta \leq z^* \\ 1, & \eta > z^* \end{cases} \tag{44}$$

and

$$\Lambda = \begin{cases} \delta(\eta + h_0) + \frac{(1-\delta)h_0}{\lambda}\left(e^{\lambda\frac{(\eta - z^*)}{h_0}} - e^{-\lambda\frac{(h_0+z^*)}{h_0}}\right), & \eta \leq z^* \\ (\eta - z^*) + \delta(z^* + h_0) + \frac{(1-\delta)h_0}{\lambda}\left(1 - e^{-\lambda\frac{(h_0+z^*)}{h_0}}\right), & \eta > z^* \end{cases} \tag{45}$$

Here δ is the relative width of a slot with respect to a unit width of beach, λ is the parameter for the smooth transition from unity to δ, and h_0 is the offshore water depth where a slot begins.

Madsen et al. (1997) showed that, even though a very narrow slot width is used, there is still about a ten percent error in the computed maximum runup in comparison with the analytical solution by Carrier and Greenspan (1958). This is attributed to the additional cross-sectional area introduced by the narrow slot because the maximum runup is very sensitive to the total volume of mass at the runup tip. In contrast to Tao's (1984) formulation, which does not conserve mass in the presence of a slot, we retain an equivalent cross-sectional area of a unit width of beach, leading to improvement in the simulation of runup as shown Kennedy et al. (1999). The resulting z^* may be expressed as

$$z^* = \frac{z^s}{1-\delta} + h_0\left(\frac{\delta}{1-\delta} + \frac{1}{\lambda}\right) \tag{46}$$

in which z^s is the elevation of the solid seabed.

The optimal values of δ and λ are found to be 0.002 and 80 (Kennedy et al., 1999), respectively, which give the best agreement with the analytical solution by Carrier and Greenspan (1958). For simulations of wave runup on steep slopes, however, a slightly larger slot width and a localized filter may be needed to avoid numerical instability. Chen et al. (1999b) verified the Boussinesq model with the improved permeable-seabed technique against the laboratory experiment on solitary wave runup on a circular island described by Liu et al. (1995). Good agreement between the computed and measured maximum runup was found.

1.6 Subgrid Turbulent Mixing

Boussinesq models are based on vertically-integrated mass and momentum equations. However, the grid size involved with the simulation of surface waves is usually smaller than the typical water depth. The horizontally-distributed eddy viscosity resulting from subgrid turbulent processes may therefore become an important factor influencing the flow pattern of the wave-generated current field. In the absence of the subgrid model in the governing equations, the underlying current field generated by wave breaking may become so chaotic that no realistic flow pattern can be recognized. Thus, we utilize the Smagorinsky-type subgrid model (Smagorinsky 1963) to account for the effect of the resultant eddy viscosity on the underlying flow.

$$F_{bs} = \frac{1}{h+\eta}\left[(\nu_s((h+\eta)u_\alpha)_x)_x + \frac{1}{2}\left(\nu_s(((h+\eta)u_\alpha)_y + ((h+\eta)v_\alpha)_x)\right)_y\right] \tag{47}$$

$$G_{bs} = \frac{1}{h+\eta}\left[\frac{1}{2}\left(\nu_s(((h+\eta)v_\alpha)_x + ((h+\eta)u_\alpha)_y)\right)_x + (\nu_s((h+\eta)v_\alpha)_y)_y\right] \tag{48}$$

where ν_s is the eddy viscosity due to the subgrid turbulence.

$$\nu_s = c_m \triangle x \triangle y \left[(U_x)^2 + (V_y)^2 + \frac{1}{2}(U_y + V_x)^2\right]^{\frac{1}{2}} \tag{49}$$

in which U and V are the velocity components of the time-averaged underlying current field, $\triangle x$ and $\triangle y$ are the grid spacing in the x and y directions, respectively, and c_m is the mixing coefficient with a default value of 0.2. In the course of simulation, we obtain the underlying current field by averaging the instantaneous velocity over two peak wave periods and update ν_s accordingly.

2 NUMERICAL MODEL

This section details the numerical solution techniques used to obtain solutions of the system of model equations described in the previous section. This numerical approach provides the foundation for the FUNWAVE model. The programs themselves are described in the following section.

2.1 Finite Difference Scheme

Numerical solutions of Boussinesq equations can be significantly corrupted if truncation errors, arising from differencing of the leading order wave equation terms, are allowed to grow in size and become comparable to the terms describing the weak dispersion effects. Many schemes developed to solve Boussinesq equations use the artifice of explicitly subtracting out terms comparable to the leading order truncation errors, in order to move the unbalanced remaining errors to a higher order than the nonlinear and dispersive effects (Madsen & Warren, 1984; Nwogu, 1993; Rygg, 1988).

In the present model, we take the somewhat different approach of using a higher-order scheme in order to perform computations. A composite 4th-order Adams-Bashforth-Moulton scheme (utilizing a 3d-order Adams-Bashforth predictor step and a 4th-order Adams-Moulton corrector step) is used to step the model forward in time. Terms involving first-order spatial derivatives are differenced to $O(\Delta x^4)$ accuracy utilizing a five-point formula. All errors involved in solving the underlying nonlinear shallow water equations are thus reduced to 4th order in grid spacing and time step size. Spatial and temporal differencing of the higher-order dispersion terms is done to second-order accuracy, which again reduces the truncation errors to a size smaller than those terms themselves. No further back-substitution of apparent truncation error terms is performed.

The resulting model scheme, as detailed below, was initially described in Wei and Kirby (1995) and was subsequently extended to include fully nonlinear effects by Wei et al (1995).

2.1.1 Time-differencing

The arrangement of cross-differentiated and nonlinear time derivative terms on the right hand side of equations (5)-(6) makes the resulting set of left-hand sides purely tridiagonal. The governing equations are finite-differenced on a centered grid in $x = i\Delta x, y = j\Delta y, t = n\Delta t$. Level n refers to information at the present, known time level. The predictor step is the third-order explicit Adams-Bashforth scheme, given by

$$\eta_{i,j}^{n+1} = \eta_{i,j}^{n} + \frac{\Delta t}{12}\left[23(E')_{i,j}^{n} - 16(E')_{i,j}^{n-1} + 5(E')_{i,j}^{n-2}\right] \tag{50}$$

$$\begin{aligned} U_{i,j}^{n+1} = U_{i,j}^{n} &+ \frac{\Delta t}{12}\left[23(F')_{i,j}^{n} - 16(F')_{i,j}^{n-1} + 5(F')_{i,j}^{n-2}\right] \\ &+ \frac{\Delta t}{12}\left[23((F_1)_t)_{i,j}^{n} - 16((F_1)_t)_{i,j}^{n-1} + 5((F_1)_t)_{i,j}^{n-2}\right] \end{aligned} \tag{51}$$

$$\begin{aligned} V_{i,j}^{n+1} = V_{i,j}^{n} &+ \frac{\Delta t}{12}\left[23(G')_{i,j}^{n} - 16(G')_{i,j}^{n-1} + 5(G')_{i,j}^{n-2}\right] \\ &+ \frac{\Delta t}{12}\left[23((G_1)_t)_{i,j}^{n} - 16((G_1)_t)_{i,j}^{n-1} + 5((G_1)_t)_{i,j}^{n-2}\right] \end{aligned} \tag{52}$$

where

$$E' = E + \gamma E_2 + f(x,y,t) \tag{53}$$

$$F' = F + \gamma(F_2 + F^t) + F_{br} + F_b + F_{sp} \tag{54}$$

$$G' = G + \gamma(G_2 + G^t) + G_{br} + G_b + G_{sp} \tag{55}$$

All information on the right hand sides of (50) - (51) is known from previous calculations. The values of $\eta_{i,j}^{n+1}$ are thus straightforward to obtain. The evaluation of horizontal velocities at the new time level, however, requires simultaneous solution of tridiagonal matrix systems which are linear in the unknowns at level $n+1$. Specifically, for a given j, $u_{i,j}^{n+1}(i = 1, 2, ..., M)$ are obtained through tridiagonal matrix solution. Similarly, $v_{i,j}^{n+1}(j = 1, 2, ..., N)$ are solved by a system of tridiagonal matrix equation for given i. The matrices involved are constant in time and may be pre-factored, inverted and stored for use at each time step.

After the predicted values of $\{\eta, u, v\}_{i,j}^{n+1}$ are evaluated, we obtain the corresponding quantities of $\{E', F', G'\}_{i,j}$ at time levels $(n+1), (n), (n-1), (n-2)$, and apply the fourth-order Adams-Moulton corrector method

$$\eta_{i,j}^{n+1} = \eta_{i,j}^{n} + \frac{\Delta t}{24}\left[9(E')_{i,j}^{n+1} + 19(E')_{i,j}^{n} - 5(E')_{i,j}^{n-1} + (E')_{i,j}^{n-2}\right] \tag{56}$$

$$\begin{aligned} U_{i,j}^{n+1} = U_{i,j}^{n} &+ \frac{\Delta t}{24}\left[9(F')_{i,j}^{n+1} + 19(F')_{i,j}^{n} - 5(F')_{i,j}^{n-1} + (F')_{i,j}^{n-2}\right] \\ &+ \frac{\Delta t}{24}\left[9((F_1)_t)_{i,j}^{n+1} + 19((F_1)_t)_{i,j}^{n} - 5((F_1)_t)_{i,j}^{n-1} + ((F_1)_t)_{i,j}^{n-2}\right] \end{aligned} \tag{57}$$

$$\begin{aligned} V_{i,j}^{n+1} = V_{i,j}^{n} &+ \frac{\Delta t}{24}\left[9(G')_{i,j}^{n+1} + 19(G')_{i,j}^{n} - 5(G')_{i,j}^{n-1} + (G')_{i,j}^{n-2}\right] \\ &+ \frac{\Delta t}{24}\left[9((G_1)_t)_{i,j}^{n+1} + 19((G_1)_t)_{i,j}^{n} - 5((G_1)_t)_{i,j}^{n-1} + ((G_1)_t)_{i,j}^{n-2}\right] \end{aligned} \tag{58}$$

From the definition, we see that the calculation of F^t and G^t at certain time level requires the corresponding values of u_t and v_t. Also, the terms $(F_1)_t$ and $(G_1)_t$ involves time derivatives. Defining quantity w as

$$w = \{u, v, F_1, G_1\} \tag{59}$$

then its time derivatives for predictor stage are

$$\begin{aligned}
(w_t)^n_{i,j} &= \frac{1}{2\Delta t}\left[3w^n_{i,j} - 4w^{n-1}_{i,j} + w^{n-2}_{i,j}\right] + O(\Delta t^2) && (60)\\
(w_t)^{n-1}_{i,j} &= \frac{1}{2\Delta t}\left[w^n_{i,j} - w^{n-2}_{i,j}\right] + O(\Delta t^2) && (61)\\
(w_t)^{n-2}_{i,j} &= \frac{-1}{2\Delta t}\left[3w^{n-2}_{i,j} - 4w^{n-1}_{i,j} + w^n_{i,j}\right] + O(\Delta t^2) && (62)
\end{aligned}$$

For the corrector stage, we evaluate w_t according to

$$\begin{aligned}
(w_t)^{n+1}_{i,j} &= \frac{1}{6\Delta t}\left[11w^{n+1}_{i,j} - 18w^n_{i,j} + 9w^{n-1}_{i,j} - 2w^{n-2}_{i,j}\right] + O(\Delta t^3) && (63)\\
(w_t)^n_{i,j} &= \frac{1}{6\Delta t}\left[2w^{n+1}_{i,j} + 3w^n_{i,j} - 6w^{n-1}_{i,j} + w^{n-2}_{i,j}\right] + O(\Delta t^3) && (64)\\
(w_t)^{n-1}_{i,j} &= \frac{-1}{6\Delta t}\left[2w^{n-2}_{i,j} + 3w^{n-1}_{i,j} - 6w^n_{i,j} + w^{n+1}_{i,j}\right] + O(\Delta t^3) && (65)\\
(w_t)^{n-2}_{i,j} &= \frac{-1}{6\Delta t}\left[11w^{n-2}_{i,j} - 18w^{n-1}_{i,j} + 9w^n_{i,j} - 2w^{n+1}_{i,j}\right] + O(\Delta t^3) && (66)
\end{aligned}$$

By substituting the $(F_1)_t$ and $(G_1)_t$ into the equations (51), (52), (57) and (58), the last terms in these equations reduce to

$$\begin{aligned}
\frac{\Delta t}{12}\left[23(w_t)^n_{i,j} - 16(w_t)^{n-1}_{i,j} + 5(w_t)^{n-2}_{i,j}\right] &= 2w^n_{i,j} - 3w^{n-1}_{i,j} + w^{n-2}_{i,j} && (67)\\
\frac{\Delta t}{24}\left[9(w_t)^{n+1}_{i,j} + 19(w_t)^n_{i,j} - 5(w_t)^{n-1}_{i,j} + (w_t)^{n-2}_{i,j}\right] &= w^{n+1}_{i,j} - w^n_{i,j} && (68)
\end{aligned}$$

where $w = \{F_1, G_1\}$.

The corrector step is iterated until the error between two successive results reaches a required limit. The error is computed for each of the three dependent variables η, u and v and is defined as

$$\Delta f = \frac{\sum\limits_{i=1,j=1}^{i=M,j=N} |f^{n+1}_{i,j} - f^*_{i,j}|}{\sum\limits_{i=1,j=1}^{i=M,j=N} |f^{n+1}_{i,j}|} \tag{69}$$

where $f = \{\eta, u, v\}$, f^{n+1} and f^* denote the current and previous results, respectively. The corrector step is iterated if any of the Δf's exceeds 10^{-4} or 10^{-3}. For "cold start" running of the model, the denominator in (69) is zero initially, which will result in infinite value of Δf. To eliminate this problem, we first compute the corresponding denominator. If value of the denominator is smaller than a small value (say, 10^{-3}), then only numerator from (69) is used for iteration errors.

For weakly nonlinear case, the scheme typically requires no iteration unless problems arise from boundaries, or inappropriate values for Δx, Δy and Δt are used. For strong nonlinearity, however, the model tends to take more iterations. Further analysis shows that the iterated results oscillate around the desired solution. To increase the convergence rate, we apply an over-relaxation technique to the iteration stage. Writing the previous and current iterated values as $f^*_{i,j}$ and $f_{i,j}$, then the adjusted value $f^r_{i,j}$ for over-relaxation is given by

$$f^r_{i,j} = (1-R)f^*_{i,j} + Rf_{i,j} \tag{70}$$

where R is a coefficient which is in the range of $(0,1)$. In all computations, we found that $R = 0.2$ gave quite satisfactory results.

2.1.2 Spatial differencing

For first order spatial derivatives, the following five-point difference schemes are used

$$\begin{aligned}
(w_x)_{1,j} &= \frac{1}{12\Delta x}(-25w_{1,j} + 48w_{2,j} - 36w_{3,j} + 16w_{4,j} - 3w_{5,j}) && (71)\\
(w_x)_{2,j} &= \frac{1}{12\Delta x}(-3w_{1,j} - 10w_{2,j} + 18w_{3,j} - 6w_{4,j} + w_{5,j}) && (72)\\
(w_x)_{i,j} &= \frac{1}{12\Delta x}[8(w_{i+1,j} - w_{i-1,j}) - (w_{i+2,j} - w_{i-2,j})] && (73)\\
&\quad (i = 3, 4, \cdots, M-2) &&\\
(w_x)_{M-1,j} &= \frac{1}{12\Delta x}(3w_{M,j} + 10w_{M_1,j} - 18w_{M_2,j} + 6w_{M_3,j} - w_{M_4,j}) && (74)\\
(w_x)_{M,j} &= -\frac{1}{12\Delta x}(25w_{M,j} - 48w_{M_1,j} + 36w_{M_2,j} - 16w_{M_3,j} + 3w_{M_4,j}) && (75)
\end{aligned}$$

where $w = \{\eta, u, v\}$, $M_k = M - k(k = 1, 2, 3, 4)$, and M is the total number of grid point in x direction.

For second order derivatives, we use three-point difference schemes

$$\begin{aligned}
(w_{xx})_{i,j} &= \frac{w_{i+1,j} - 2w_{i,j} + w_{i-1,j}}{(\Delta x)^2} && (76)\\
&\quad (i = 2, 3, \cdots, M-1) &&
\end{aligned}$$

Similar expressions can be obtained for derivatives with respect to y. For mixed derivatives, we use

$$\begin{aligned}
(w_{xy})_{i,j} &= \frac{w_{i+1,j+1} + w_{i-1,j-1} - w_{i-1,j+1} - w_{i+1,j-1}}{4\Delta x \Delta y} && (77)\\
&\quad (i = 2, 3, \cdots, M-1; j = 2, 3, \cdots, N-1) &&
\end{aligned}$$

2.2 Boundary Conditions

To obtain solutions to wave propagation over a finite domain, appropriate boundary conditions have to be specified in the numerical model. Two kinds of boundary conditions are used in the model, *i.e.* total reflected wall and sponge layer. Waves are generated inside the domain by a source function technique which was described in section 1.3.

2.2.1 Wall Boundary

For a perfectly reflecting vertical wall, the horizontal velocity normal to the wall is always zero. The corresponding values of surface elevation and tangential velocity, in theory, could be obtained from the governing equations. However, numerical implementation for the latter is cumbersome and instability always occurs for our testing cases. Here we adapt the conditions of specifying the normal derivatives of surface elevation and tangential velocity as zero (Wei and Kirby, 1995), which is quite simple yet accurate to the order of standard and extended Boussinesq equations. Figure 3 shows the definition of computational domain. Grid numbers in x and y directions are represented by integer $i = 1, 2, 3, \cdots, M$ and $j = 1, 2, 3, \cdots, N$. Assuming a wall is located in the right end of the domain (*i.e.* at $i = M$), then the corresponding boundary conditions are given by

$$\begin{aligned} u_{M,j} &= 0 && (78)\\ (\eta_x)_{M,j} &= 0 && (79)\\ (v_x)_{M,j} &= 0 && (80)\\ &(j = 1, 2, 3, \cdots, N) \end{aligned}$$

Applying five-point off-center finite difference to equations (79) and (80), we have

$$w_{M,j} = \frac{1}{25}(48w_{M_1,j} - 36w_{M_2,j} + 16w_{M_3,j} - 3w_{M_4,j}) \tag{81}$$

where $w = \{\eta, v\}$. Similar expressions can be obtained for walls at other locations.

2.2.2 Absorbing Boundary

There are several types of absorbing boundary condition which allows waves to propagate out of the domain with minimum reflection. A sponge layer boundary condition is used here since it is able to damp wave energy for a wide range of frequencies and

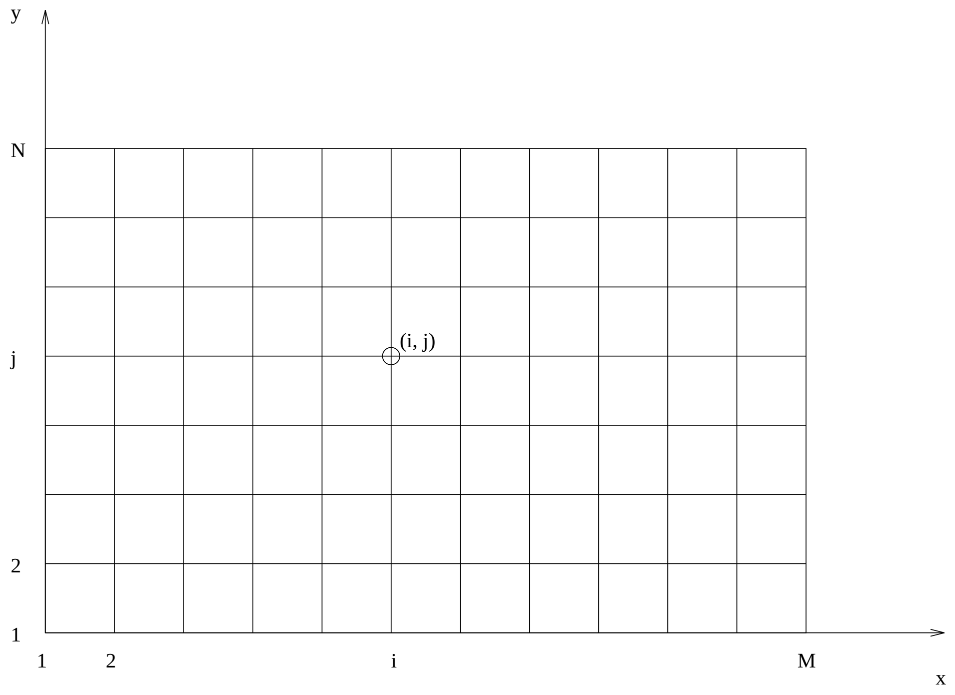


Figure 3: Definition for computational domain

directions. Although extra grid points are needed, it is justified to apply sponge layer due to the decreasing cost of computer storage and the stability of the numerical model.

To absorb wave energy, artificial damping terms F_{sp} and G_{sp} are added to the right hand side of momentum equations (5) and (6), respectively. The damping terms are defined as

$$F_{sp} = -w_1(x,y)u + w_2(x,y)(u_{xx} + u_{yy}) + w_3(x,y)\sqrt{\frac{g}{h}}\eta \tag{82}$$

$$G_{sp} = -w_1(x,y)v + w_2(x,y)(v_{xx} + v_{yy}) + w_3(x,y)\sqrt{\frac{g}{h}}\eta \tag{83}$$

where w_1, w_2 and w_3 are functions for three different kinds of damping mechanism, which were referred to as Newtonian cooling, viscous damping, and sponge filter, respectively (Israeli and Orszag, 1981).

Assuming that there is only one sponge layer on the right end of the domain (see Figure 3), *i.e.*, from $x = x_s = (i_s - 1)\Delta x$ to $x = x_l = (M-1)\Delta x$. Then at the range of $x_s < x < x_0$, $w_i(x,y)$ $(i = 1,2,3)$ are zero. At the range of $x_s < x < x_l$, $w_i(x,y)$ $(i = 1,2,3)$ are defined as

$$w_i(x,y) = c_i \omega f(x) \tag{84}$$

where $c_i(i = 1,2,3)$ are constant coefficients corresponding to the three damping functions. Israeli and Orszag (1981) claimed that sponge filter is the best among the three damping terms for an open boundary condition. However, from our numerical experiment for closed boundary, we found that Newtonian cooling work the best. The notation ω in equation (84) is the frequency of waves to be damped, and $f(x)$ is a smooth monotonically increasing function varying from 0 to 1 when x varies from x_s to x_l. Function $f(x)$ is defined as

$$f(x) = \frac{\exp[(x - x_s)/(x_l - x_s)]^2 - 1}{\exp(1) - 1}, \qquad x_s < x < x_l \tag{85}$$

The width of the damping layer (*i.e.* $x_l - x_s$) is usually taken to be two or three wave lengths. Similar expressions can be obtained for sponge layers on three other ends of the domain. The final representation of functions $w_i(x,y)$ $(i = 1,2,3)$ are the summation of all sponge layers.

2.3 Numerical Filtering

Due to nonlinear interaction in the model, higher harmonic waves will be generated as the program runs. These super-harmonic waves could have very short wavelength (the minimum wavelength for given grid resolution is twice of the grid size). For such short wave components, the present Boussinesq model (even with improved dispersion relation in intermediate water depth) is not valid to apply because of the large depth to wavelength ratio. In addition, the magnitude of these short waves usually grows rapidly once they are generated and eventually causes the blowup of the model itself.

One effective way to eliminate such undesired short waves is to apply a numerical filter in the model. Shapiro (1970) described in detail the method of weighted average and derived expressions for several orders of filters. Here we adopt the expression of Shapiro for a 4th order filter which determines a new value at each grid point by using the original values at 9 adjacent points.

$$\begin{aligned} Z_i^* = \quad & \frac{1}{256}[186 Z_i + 56(Z_{i+1} + Z_{i-1}) - 28(Z_{i+2} + Z_{i-2}) \\ & 8(Z_{i+3} + Z_{i-3}) - (Z_{i+4} + Z_{i-4})] \end{aligned} \tag{86}$$

where $Z = \{\eta, u\}$ represents the original values which consist of long and short wave components, Z^* represent the new values with short wave being filtered out.

The response function for the above 9-point filter is represented by the ratio of the smoothed to unsmoothed amplitudes

$$R = 1 - \sin^8(\frac{k\Delta x}{2}) = 1 - \sin^8(\frac{\pi L}{\Delta x}) \tag{87}$$

where k is the wavenumber, Δx the grid size, and L the corresponding wave length. Figure 4 shows the response function of the filter with respect to the ratio of wavelength to grid size. Waves with wavelength twice of the grid size are filtered out completely since the corresponding response function R is zero. As the wave length increases, however, the value of R increases (with asymptotic value of 1), indicating that the effect of filtering decreases for longer waves.

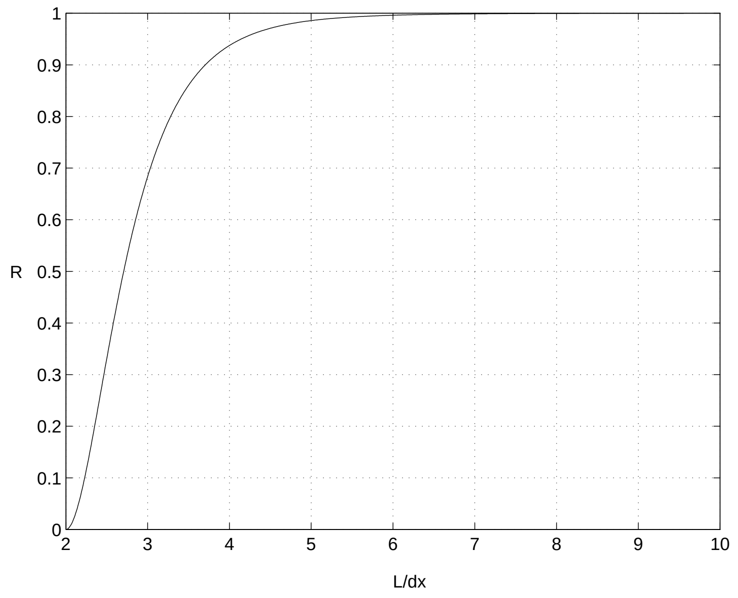


Figure 4: Response function of the 1-D 9 point filter.

It is straight forward to obtain an expression for a 2-D filter by applying equation (86) to first the x and then the y direction. However, it will be cumbersome to write the corresponding 2-D formula and code it in the program since the formula contains $9 \times 9 = 81$ neighboring points for one filtered point. Instead, the 1-D formula is employed twice, with the first in the x direction for all y values and the second in the y direction for all x values. The corresponding response function for the 2-D case is

$$R(l, \lambda) = \left[1 - \sin^8(l\Delta x/2)\right] \left[1 - \sin^8(\lambda\Delta y/2)\right] \tag{88}$$

where l and λ are the wavenumbers in x and y directions, respectively, Δx and Δy are the grid sizes. Denoting L_x and L_y to be the wavelengths in x and y directions, then the above response function can be rewritten as

$$R = \left\{1 - \sin^8[\pi/(L_x/\Delta x)]\right\}\left\{1 - \sin^8[\pi/(L_y/\Delta y)]\right\} \tag{89}$$

Figure 5 shows the response function of the filter with respect to the ratios of wavelengths to grid sizes. Again, waves with wavelengths twice the grid size are filtered out completely since the corresponding response function R is zero for these waves. As wavelength increases, the filtering effect is reduced.

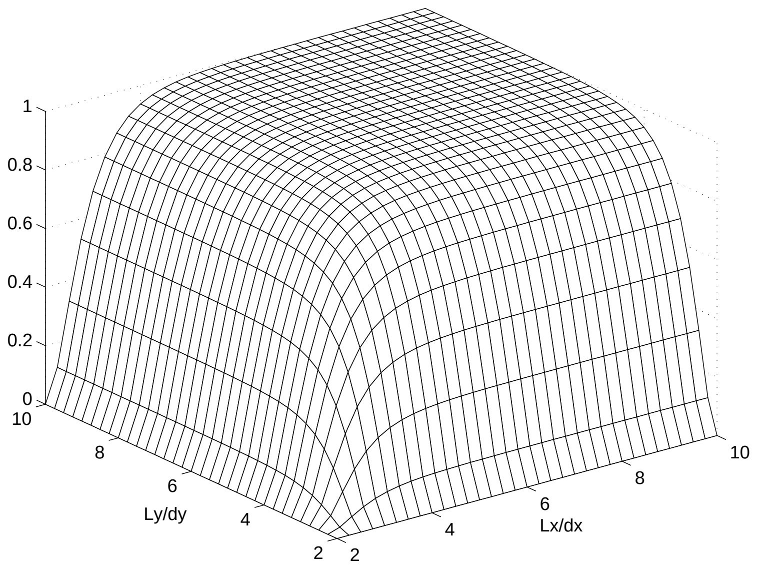


Figure 5: Response function of the 2-D 9 × 9 point filter

The use of the above numerical filter should be kept to minimum, since a fraction of long wave energy is also filtered out every time the above formula is applied. In the testing cases 2 and 3, the numerical filter is applied every 200 time steps (4 wave periods) and good results are obtained.

3 USER'S MANUAL

The Boussinesq model developed here is written in the Fortran 77 language and in theory should be able to run on all type of computers which have Fortran 77 compilers. We have been running the program on three types of computers (Sun workstations, SGI workstations and a Cray J90) and have not found any machine related problems, except for different record length for binary data on the SGI machines. To take advantage of the graphic capability for SGI machines, special subroutines for animation are written to plot wave field as computation goes on. Graphic Library (GL) routines for SGI machines are used in this case. Detailed description can be found in section 3.6.

In general, two-dimensional programs can be used to simulate one-dimensional cases of wave propagation. However, purely 1-D programs are much simpler to code and much faster to run due to the fact that 1-D programs include less terms and do not require a minimum of five grid points in the y direction as does the 2-D program. Therefore, we develop both 1-D and 2-D programs for simulating corresponding cases of wave propagation.

Although the 1-D and 2-D programs are separate, the basic structure of these two programs is the same. In the following, we will describe the model for the 2-D case in general and point out the difference for the 1-D program.

3.1 Revision History

Version 1.0 represents the initial release of the model.

3.2 Program Outline and Flow Chart

The numerical model for the 2-D case consists of a main program and 20 subroutines. The main program consists of a do loop for time stepping. Inside the loop, various subroutines are called to compute the corresponding values of $\{\eta, u, v\}$ at each time step, using predictor or corrector formulas given in section 1.2. Figure 6 shows the flow chart for the main program.

A short description of each routine follows:

1. *init*: Called by main program, this routine reads input data files and computes constants and arrays which will be used by the main program and other

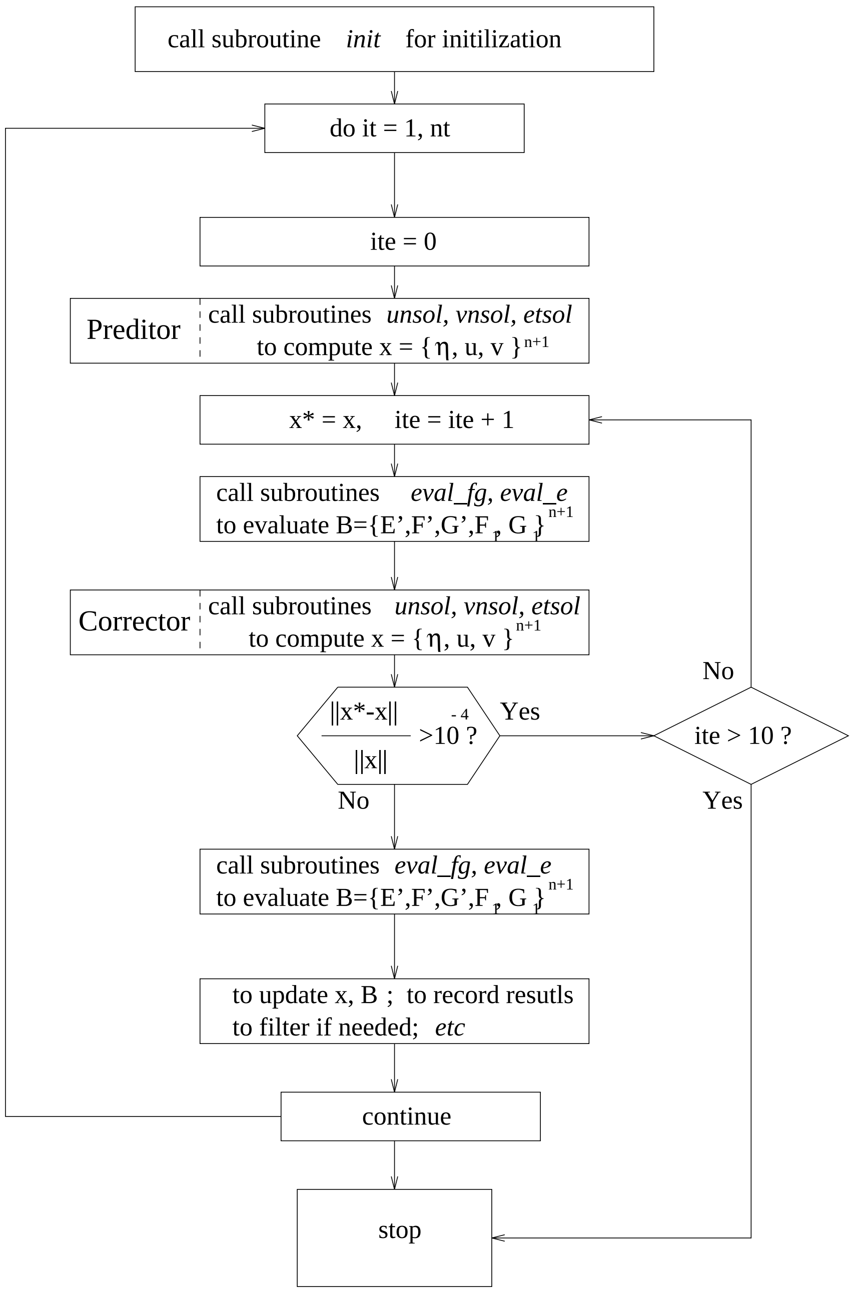

call subroutine *init* for initilization
do it = 1, nt
ite = 0
Preditor
call subroutines *unsol*, *vnsol*, *etsol* to compute x = {η, u, v }$^{n+1}$
x* = x, ite = ite + 1
call subroutines *eval_fg*, *eval_e* to evaluate B={E',F',G',F$_1$, G$_1$}$^{n+1}$
Corrector
call subroutines *unsol*, *vnsol*, *etsol* to compute x = {η, u, v }$^{n+1}$
$\frac{\|x^*-x\|}{\|x\|} > 10^{-4}$?
Yes
No
ite > 10 ?
No
Yes
call subroutines *eval_fg*, *eval_e* to evaluate B={E',F',G',F$_1$, G$_1$}$^{n+1}$
to update x, B ; to record resutls
to filter if needed; *etc*
continue
stop

Figure 6: Flow chart of the main program

subroutines.

2. *tridag*: Called by *init*, this routine obtains the tridiagonal matrix for solving velocities u and v.

3. *ludec*: Called by *init*, this routine decomposes the tridiagonal matrix for obtaining velocities u and v.

4. *wavenb*: Called by *init*, this routine computes wavenumbers for given water depth and angular frequency.

5. *etsol*: Called by main program, this routine computes surface elevation η at new time step. It calls subroutine *bcet* to obtain boundary values of η.

6. *unsol*: Called by main program, this routine computes x-direction velocity u at new time step. It calls subroutine *bcu* to obtain boundary values of u.

7. *vnsol*: Called by main program, this routine computes y-direction velocity v at new time step. It calls subroutine *bcv* to obtain boundary values of v.

8. *bcet*: Called by *etsol*, this routine computes boundary values of η.

9. *bcu*: Called by *unsol*, this routine computes boundary values of u.

10. *bcv*: Called by *vnsol*, this routine computes boundary values of v.

11. *error*: Called by main program, this routine computes errors of η, u, v for successive iteration.

12. *printing*: Called by main program, this routine writes out computational results into output files.

13. *fltr*: Called by main program, this routine filters out short waves which may cause instability.

14. *eval_fg*: Called by *init* and main program, this routine computes the quantities defined in the right hand side of momentum equations, *i.e.*, $F + \gamma F_2$, F_1, F^t, $G + \gamma G_2$, G_1, G^t.

15. *eval_e*: Called by *init* and main program, this routine computes $E + \gamma E_2$.

16. *pre_eval*: Called by *eval_fg* and *eval_e*, this routine computes common terms in E, F, G, F_1, F_2, G_2, F^t, G^t. These common terms are: u_{xy}, v_{xy}, $(hu)_{xy}$, $(hv)_{xy}$, $(u_x + v_y)_x$, $(u_x + v_y)_y$, $[(hu)_x + (hv)_y]_x$, $[(hu)_x + (hv)_y]_y$.

17. *eval_e2*: Called by *eval_e*, this routine computes E_2.

18. *eval_f2g2*: Called by *eval_fg*, this routine computes F_2 and G_2.

19. *eval_utvt*: Called by *eval_fg*, this routine computes u_t and v_t and calls subroutine *eval_ftgt* to compute F^t and G^t.

20. *eval_ftgt*: Called by *eval_utvt*, this routine computes F^t and G^t.

For 1-D programs, subroutines *vnsol* and *bcv* will not exist. In addition, variables with y dependent such as G, G_1, G_2, G^t and F_1 will be zero. Computation in every subroutine will be reduced more than half due to the absence of variation in y direction.

3.3 Program Input

There are four input data files to be read by subroutine *init*. The first file consists of control parameters and is named *funwave2d.data* for 2-D programs and *funwave1d.data* for 1-D programs. With the use of intrinsic function NAMELIST in the program, variable name and its corresponding data can be put together. The logical device number for this file is chosen as 1 and the form of the file is ASCII.

The other three input files are water depth data, initial wave field data, and time series of source function amplitude, respectively. Their names are represented by *f1n*, *f2n* and *f3n* which are specified in *funwave2d.data* or *funwave1d.data*. Binary format is used for these three files to increase I/O speed for 2-D programs while ASCII format is used for 1-D programs. Since the record length of data for binary format in SGI computer is different from other machines, a control parameter *imch* is used in *funwave2d.data* or *funwave1d.data* to adjust for different computers. In the following, more detailed descriptions of each input file such as definition of parameters and how to generate such files are given.

3.3.1 Input file 1: *funwave2d.data* or *funwave1d.data*

The file consists of control parameters to be used through the programs. These control parameters are defined as:

- *ibe*

  Control parameter for different types of Boussinesq equations: (1) *ibe* = 1 is for Nwogu's (1993) extended Boussinesq equations; (2) *ibe* = 2 for the fully nonlinear Boussinesq equations of Wei *et al.* (1995); (3) *ibe* = 3 for Peregrine's (1967) Boussinesq equations; (4) *ibe* = 4 for nonlinear shallow water equations; and (5) *ibe* = 0 is for linearized Nwogu's equations.

- *imch*

  Identification number for different types of computer due to difference in record length of data for binary format: (1) $imch = 1$ is for SGI workstations; (2) $imch = 8$ is for Sun and Cray J90.

- *ianm*

  Identification number for the use of online animation for the 1-D code. $ianm = 1$ is for animation on SGI machines; $ianm = 0$ is no animation.

- *a0*

  Input wave amplitude in meters. For Gaussian hump testing case, it is the initial wave height in the center of the hump. For regular waves over submerged shoal testing cases, it is the incident wave amplitude. For random wave generated by specified spectrum, it is the root-mean-square wave amplitude. No need in the case of a given time series of free surface elevation.

- *h0*

  Constant water depth in meters over the wave generation region.

- *tpd*

  Wave period for monochromatic waves, dominant wave period for random waves.

- *dx, dy*

  Grid size in meters for x and y directions, respectively. There is no need for dy in 1-D programs.

- *dt*

  Time step size in seconds, $dt < 0.5dx/\sqrt{gh_max}$.

- *mx, ny*

  Numbers of grid points in x and y directions, respectively. There is no need for ny in 1-D programs.

- *nt*

  Number of time steps for the program to run.

- *itbgn, itend, itdel*

  The beginning, ending and interval numbers of time step to store spatial maps of η or time-averaged velocity.

- *itscr*

  Number of time steps between the output of a partial list of computed results on the screen.

- *itftr*

  Number of time steps between application of the numerical smoothing filter.

- *theta*

  Input angle (in degrees) between wave direction and x direction. No need in 1-D program.

- *cbkv*

  Coefficient allowing the variation of parameter for the breaking scheme. *cbkv*$=\eta_t^{(I)}$. (see Section 1.4 for the description of the parameter).

- *delta*

  Slot width relative to a unit width of beach. $delta = 0.002 - 0.02$.

- *slmda*

  Parameter controlling the slot shape. $slmda = 20 - 80$.

- *isltb*

  Inception of a slot in the x direction.

- *islte*

  The end of a slot in the x direction.

- *isrc*

  Index number for the center line of source function in x direction. Source function is not valid if $isrc \leq 1$.

- *jsrc*

  Index number for the center line of source function in y direction. No need for 1-D programs.

- *swidth*

  Ratio of source function width W to half wavelength $L/2$. *swidth* is a constant of $O(1)$.

- *cspg, cspg2, cspg3*

  Coefficients for three different kinds of sponge layers. The usual values for all the testing cases is $cspg = 10, cspg2 = 0, cspg3 = 0$.

- *ispg(4)*

  Control parameters for sponge layer widths in four boundaries. The sponge layers are in the grids from $i = 1$ to $i = ispg(1)$, from $i = mx - ispg(2) + 1$ to $i = mx$, from $j = 1$ to $j = ispg(3)$, and from $j = ny - ispg(4) + 1$ to $j = ny$. Only the first two elements in $ispg$ are used in 1-D programs.

- *ngage*

  Total number of wave gages in computing domain where time series of η is recorded. The coordinates of these points are defined by ixg and iyg. The maximum value of $ngage$ is 20 in the program and could be increased by changing the predefined size for arrays ixg and iyg.

- *ixg(25), iyg(25)*

  A pair of grid index which define the wave gages where time series of η are recorded. The corresponding coordinates for these points are $(x_k, y_k)(k = 1, 2, ..., ngage)$ with $x_k = [ixg(k) - 1] * dx$ and $y_k = [iyg(k) - 1] * dy$.

- *itg(6)*

  Time steps where spatial profiles of η are stored.

- *f1n, f2n, f3n*

  Names of input data files: (1) $f1n$ is for water depth; (2) $f2n$ is for initial wave field; and (3) $f3n$ is for time series of source function amplitude for random wave input.

- *f4n, f5n, f6n, f7n*

  Names of output files: (1) $f4n$ is for time series of η at gages specified by grid index ixg and iyg; (2) $f5n$ is for time series of other quantities such as water volume or u, v; (3) $f6n$ is reserved for spatial profiles specified by time step itg; (4) $f7n$ is for spatial profiles of η or time-averaged velocity components at the time steps controlled by $itbgn, itend, itdel$.

- *cbrk*

  Coefficient for wave breaking as in equation (41). The typical value is $cbrk = 1.2$.

- *ck_bt*

  Coefficient for bottom friction formulated by the quadratic law, which is appeared in equation (22). The typical value is $ck\_bt = 1.0 \times 10^{-3}$ to 5.0×10^{-3}. It is equal to zero for smooth bottom.

- *c_dm*

  Coefficient for the subgrid mixing model. The typical value is $c\_dm = 0.1$. For non-breaking waves, set $c\_dm = 0.0$.

- *isld*

  $isld = 1$ is for the simulation of runup on a conical island in Liu et al. (1995). A sponge layer is placed inside the island and a subroutine is called to compute the maximum runup height on the island when $isld = 1$. In the case of an open coast, set $isld = 0$.

- *idout*

  $idout = 1$ leads to the output of time-averaged velocity (over two peak wave periods) in $f7n$; $idout = 0$ leads to the output of free surface elevation in $f7n$.

- *idft*

  idft=1 imposes a local filter along lateral boundaries if blow-up occurs there; idft=0 indicates that no local filter is used.

- *itide*

  Control parameter for tidal effect. For $itide = 1$ tidal effect is turned on as specific by *tideco* defined bellow. For $itide = 0$, no tidal effect is turned on. Currently this parameter is only implemented in 1-D programs.

- *tideco(3)*

  Coefficients of parabolic curve fitting for tidal effect in 1-D model.

3.3.2 Input file 2: *f1n*

This file consists of water depth data which is needed for the Boussinesq model to run. The depth data could generated by Fortran program *depth.f*, if the bottom geometry could be represented by some mathematical formula. The Fortran program *depth.f* reads the some of the control parameter file *funwave2d.data* or *funwave1d.data* and computes the corresponding water depth matrix. In the current version of *depth.f* program, four examples of depth grid can be generated. These four cases of depth grid correspond to the four example calculations which will be described in Sections 4.1-4. The first case is combination of a constant depth and a constant slope of 1-D. The other three cases are all 2-D, *i.e.* constant depth, topography of Berkhoff *et al.* (1982) and topography of shoal experiment of Chawla and Kirby (1996).

Data in *f1n* is written in the following format:

```
open (unit, file = f1n, access = 'direct', recl = imch*ny)
do i = 1, mx
write (unit, rec = i) (h(i,j),j=1,ny)
enddo
```

3.3.3 Input file 3: *f2n*

This file consists of initial values of surface elevation η and horizontal velocity components u and v for each point in the computational domain. The Fortran program to generate the initial wave data file is called *initw.f.* For the test case of evolution of initial Gaussian shape water column, the surface elevation η is obtained from the given formula, and the velocity components of u and v are zero. For the other testing cases of 1-D random wave propagation and 2-D regular wave shoaling, all variables of η, u and v are set zero.

Data in *f2n* is written in the following format:

```
open (unit, file = f2n, access = 'direct', recl = 3*imch*ny)
do i = 1, mx
write (unit, rec=i) (ui(i,j),j=1,ny), (vi(i,j),j=1,ny), (eti(i,j),j=1,ny)
enddo
```

3.3.4 Input file 4: *f3n*

This file consists of time series of source function amplitude $s(t)$ which is used to generate desired waves for the model. The Fortran file to generate $s(t)$ is called *source.f.* In order to efficiently generate random waves with different direction components, at least two crossing source lines are needed. The 2-D version of *source.f* is still under development. The 1-D version is named *1dsource.f* and has been tested to generate desired random waves successfully. In program *1dsource.f*, two methods can be chosen to generate time series of source function amplitude. The first is by specified power spectrum of η and the second is by measured time series of η at the corresponding location. Due to the special requirement of the program, a separate input data file named $1dsource.data$ is required to run *1dsource.f.* The parameters in $1dsource.data$ are defined as:

- *imeth*

  Control parameter for different methods to generate time series of source function magnitude. For $imeth = 1$, input spectrum of η is required. For $imeth = 2$, input time series of measured η is used.

- *f1, f2*

  The lowest and the highest frequency components to be used in the spectrum of η.

- *nf*

  Number of frequency components between $f1$ and $f2$.

- *fnin*

  Name of the input time series of η.

- *ntd*

  Number of time step in the input time series of η.

- *dtd*

  Time step in seconds for the input time series of η.

- *nttrans*

  Number of time step to perform FFT transform for each segment of data. The number should be power of 2.

- *hscale*

  Coefficient to convert the input η into meters.

- *itide, tideco(3)*

  Tidal effect parameters which are the same definition as in $funwave1d.data$.

3.4 Program Output

There are total nine output files from the model, eight of which are for surface elevations η. The first file is time series of η at locations specified by ixg and iyg, with the total number of points specified by $ngage$. The data dimension of the file is $ngage \times nt$. The second output file is time series of other quantities, such as velocity components u and v. Binary format are used to store these files to increase the I/O speed, especially for post-processing Matlab programming.

The next six files are spatial profile of η at time steps specified by $itg(6)$. All these six files have the same prefix $f6n$ for their names. Their postfix names are string conversion of the corresponding time steps. Data in these six files have the dimension of $mx \times ny$. Again, binary format is used to store these six files so that these data can be read directly by Matlab low order Input/Output command.

The last data file is the spatial profiles for time steps controlled by $itbgn, itend, itdel$. Data in this file have the dimension of $mx \times ny \times ntn$ with $ntn = (itend - itbgn)/itdel$. Binary format is used for this file that contains either free surface elevation or velocity components averaged over two peak wave periods as specified by the user through *idout*.

As the program runs, several outputs are shown on the command line screen. For every *itscr* time steps, surface elevation at 4 corner points, total mass, number of interation, etc., are printed out, which serves as a running check for the model.

3.5 Programs to Analyze Output Files

All the first 8 output files are 2-D data and can be analyzed directly by the plotting software Matlab. Time series of η at the specified locations and spatial profiles of η at time steps can be obtained directly (both ASCII and binary forms) from Matlab. Following standard procedures, it is straightforward to plot the results or to perform data analysis such as spectrum computation and other statistics.

The last output data file $f7n$ is in a 3-D array and in binary format. Depending on the domain dimension and the number of time steps at which data are stored, the size of the file can be quite large. We write Fortran programs to decompose the 3-D data to 2-D according to specific time steps. Program $b2dp1.f$ reads the output data file $f7n$ and write out a number of files, each of which corresponding to the spatial profile of η at a specified time step. For regular wave propagation, wave height distribution along the domain is desired to compare with experiment data. And program $b2dp2.f$ reads the file $f7n$ and computes the averaged wave height by zero crossing method for the entire grid points.

For the output of time-averaged velocity components in $f7n$ (i.e. $idout = 1$), a program $b2dp3.f$ may be used to extract the information of the wave-induced unsteady current at a given time or averaged over a long period of time.

3.6 Implementation of Animation Graphics

For SGI machines and other computer systems with animation software (*i.e.*, Graphic Library for SGI, OpenGL for other computers), it is quite useful to implement animation to visualize computing results. Real time animation as the program is running provides an efficient way to detect possible cause of model blowup from coding error or boundary problem.

In the beginning of developing the model, we wrote subroutines utilizing Graphic Library (GL) provided by SGI machines. To extend the animation to other computer systems, these subroutines are being transferred into OpenGL. In the present version, only GL is used to described animation as follows.

Since it is relatively faster for the 1-D programs to run and the effect of slowing down the computation by adding animation to the Boussinesq model is not apparent, we provide the option of turning on animation as the 1-D Boussinesq model is running. There are three subroutines for animation which are all written in C. The first subroutine is called *plotinit.c* which initializes the SGI Graphic Library such as window size and camera position. The second is called *plot.c* which is called by the main program to plot the surface elevation η at the specified time step. The third is called $plotfn.c$ which closes down the graphic windows. The option of turning on animation is provided by using the makefile *Makefile.* To turn off animation for non-SGI computers, we should use the makefile *Makefilebg.* The function of animation can be switched off on SGI machines by setting $ianm = 0$ in the input file $funwave1d.data$.

In the 2-D case, the computation of the Boussinesq model takes quite a long time for SGI machine, even with smaller grid numbers (*e.g.* 50×50). Therefore, adding animation to the 2-D model would be less effective. Instead, we write out data of η as much as possible (limited by machine capacity) to the file $f7n$. A separate program which consists of SGI Graphics Library routines is used to read the data $f7n$ and to plot animation image of the computing results.

4 EXAMPLE CALCULATIONS

The numerical model has been applied to compute wave fields for several cases of wave propagation. In the following, seven examples of running the model are described. The first two are for 1-D programs to simulate wave propagation over a planar beach. The third case is the evolution of an initially Gausaian elevation in a rectangular basin. The last four examples are regular wave propagation over submerged shoals and adound a conical island. Agreement between model results and available experimental data is found to be quite reasonable.

4.1 Wave Propagation over a Planar Beach: Mase & Kirby (1992)

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. Figure 7 shows the experiment layout, where a constant depth on the left connects to a constant slope on the right. Two sets of random waves with peak frequencies $0.6Hz$ ($run1$) and $1.0Hz$ ($run2$) are generated by wavemakers on left end and propagate through the flat bottom and then over the slope. Starting at the toe, 12 wave gages are deployed along the slope at locations whose water depths are h = 47, 35, 30, 25, 20, 17.5, 15, 12.5, 10, 7.5, 5, 2.5 cm. Time series of surface elevation η at these locations are collected simultaneously for about 14 minutes for $run1$ and 12 minutes for $run2$.

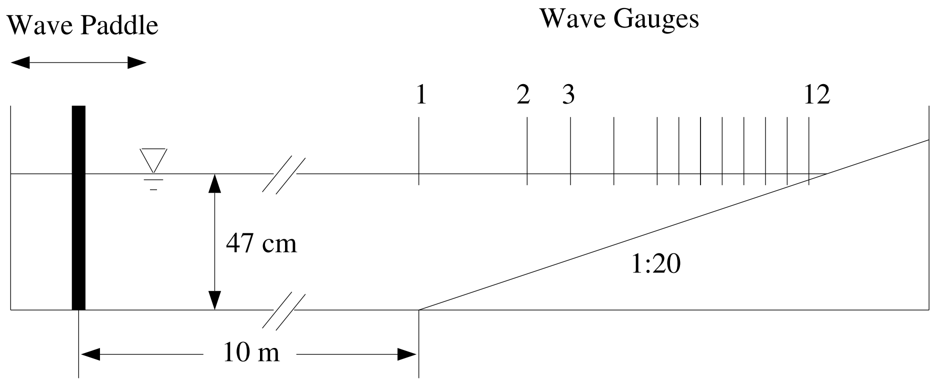


Figure 7: Experiment layout of Mase & Kirby (1992).

The 1-D model is used to simulate the wave propagation. Here we present the model results for $run2$ and compare them with the experimental data. The dispersion coefficient kh for the peak frequency wave in $run2$ is about 2, which is out of the validity range for the standard Boussinesq equations. However, as shown in the following, the extended Boussinesq model is applicable for this case as good agreement between the numerical results and experimental data is observed.

The closest gage to the wavemaker is at the toe with $h = 47cm$. The measured data of η at that location is used as an input to the Fortran program $1dsource.f$ to generate time series of source function amplitude. The input file for $1dsource.f$ is:

```
$data_0
imeth   =  2
$end

$data_1
f1    = 0.4           f2    = 2.0          nf = 161
$end

$data_2
fnin   = 'r2d470.dat'
ntd    = 15000        dtd   = 0.05    nttrans = 1024
hscale = 0.01
$end
```

Since the time series of $\eta(t)$ is used as an input to generate time series of source function amplitude $s(t)$, we chose $imeth = 2$. The parameters in $\$data1$ define the interested frequency range to be modelled. In order to use FFT to perform transform between time domain and frequency domain data, the value of $nttrans$ is required to be power of 2. The experimental data of $\eta(t)$ is divided into 14 segments, each of which has 1024 data. The total number of $\eta(t)$ used to generate $s(t)$ is $14 \times 1024 = 14336$. The time length of the output time series of $s(t)$ is the same as the input $\eta(t)$. However, due to smaller time step $dt = 0.01$ (see file $funwave1d.data$ below) required to run the wave model, the total number of data for $s(t)$ is $14336 * dtd/dt = 71680$.

Owing to the absence of data near the wavemaker, the computing domain is somewhat different from the physical domain. The source location for the model is at the toe (instead of at the wavemaker) where measured data is available. The shoreline boundary is handled by the permeable-seabed method. Sponge layers are added at both ends of the domain to absorb wave energy. The depth data file is generated by the program *depth.f*. The initial wave field of η and u are set to zero everywhere by using the program *initw.f*.

Figure 8 shows the comparison of time series of η between the model and the measurement for the 11 gages at the depths h=35, 30, 25, 20, 17.5, 15, 12.5, 10, 7.5, 5, 2.5 *cm*. Except for small discrepancies for wave phases and wave heights, the computed wave shoaling and breaking agree quite well with the experimental data (most waves start breaking at the depth $h = 15cm$).

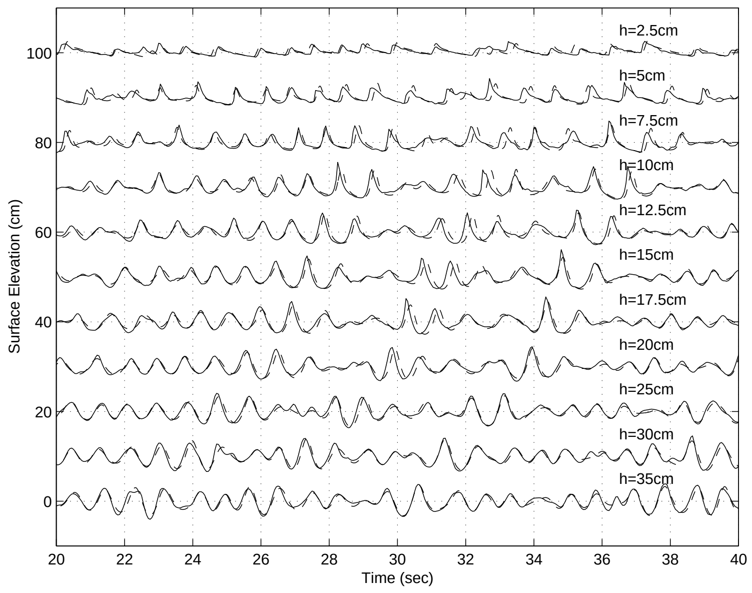


Figure 8: Time series comparison of η between model (dashed lines) and data (solid lines) at 11 wave gage locations.

The specification file *funwave1d.data* for this case is

```
$data1$
ibe   =   1          imch  = 1           ianm  = 0
a0    = 0.05         h0    = 0.47        tpd   = 1.0
dx    = 0.025        dt    = 0.01
mx    = 521          nt    = 35840
itbgn = 1            itend = 35840       itdel = 100
itscr = 100          itftr = 200
cbkv  = 0.35         ck_bt = 0.0
```

```
delta = 0.01         slmda = 50.0
isltb = 101          islte = 521
$end

$data2
isrc  =   81       swidth =  1.0
cspg  =  10.0      cspg2  =  0.0        cspg3  = 0.0
ispg  =  51       61
ngage =  13
ixg   =  101 177   217   257   297   317   337   357   377   397
         417   437   0     0     0     0     0     0     0     0
itg   =  21    51    101   4001  5001  6001
$end

$data3
f1n   = 'depth.mskb'
f2n   = 'initw.data'
f3n   = 'mskb_r2.data'
f4n   = 'eta_ts.out'
f5n   = 'tmp.out'
f6n   = 'eta_sp.out'
f7n   = 'eta_xt.out'
$end

$data4
itide = 0
tideco = 7.89  9.107e-05  -1.3737e-09
$end
```

As defined in the above data file, a numerical filter is applied every 400 time steps if there is no wave breaking. In the event of wave breaking, extra filtering is required in order to stabilize the model. We define the time step at which wave breaking occurs as breaking time step, then the extra numerical filter is applied for every 50 breaking time steps. We run the model for the entire input time series of $s(t)$ and no problem occurred to stop the program. Time series comparisons for other data segments between the model results and the experiment data are as good as that of Figure 8. To further demonstrate the applicability of the model, we performed third moment computation for the resulting time series of $\eta(t)$. Figure 9 shows the comparisons of skewness and asymmetry between the model results and experiment data. The close agreement proves that the model is capable of simulating wave shoaling and breaking.

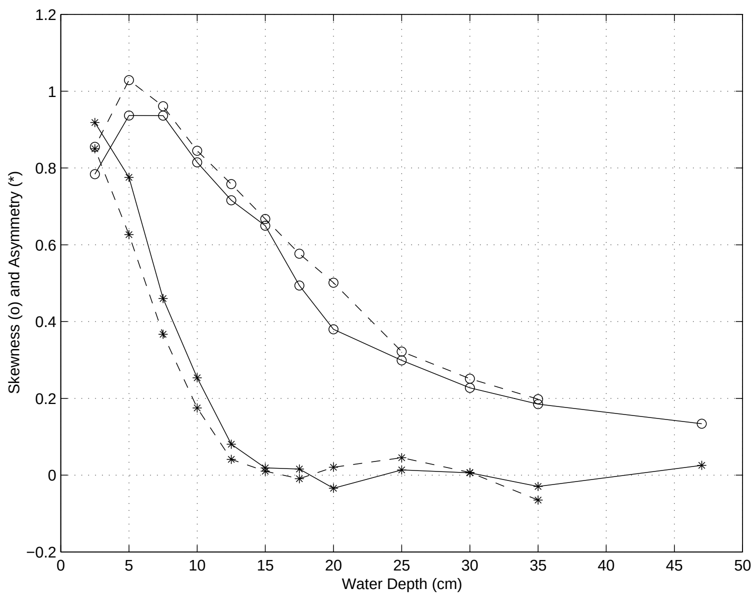


Figure 9: Comparison of skewness (o) and asymmetry (*) at different water depths. Solid lines is experimental data (Mase & Kirby, 1992). Dashed line is numerical result.

4.2 Bichromatic-Wave Runup: Mase (1995)

In a study of frequency downshift in the swash motion, Mase (1995) presented experimental results of bichromatic- wave train runup on a slope. The experiments were conducted using the same wave flume and experimental setup as those in Mase and Kirby (1992) described in preceding section. Mase's laboratory measurements, including shoaling, breaking and swash motion, provide good test cases for the verification of the runup scheme in combination with the wave breaking model.

We chose two typical test cases from Mase's (1995) series of experiments. Each of them represents a different kind of wave pattern, or group pattern. The first one, named WP1, contains fives waves in each wave group with a mean frequency $f = 0.6$ Hz, while in the second case WP2, each wave group consists of ten waves with $f= 1.2$ Hz. In both cases, the nonlinear interactions of wave components and the variation of breaking point in the wave train, among others cause considerable low frequency swash oscillations. We use the Boussinesq model incorporated with the improved slot scheme and the breaking model to simulate these two test cases.

Incident waves are generated using the source function technique. The measured time series of free surface at Gauge 1 near the toe of the slope is used as an input to the model. The same procedure as described in the preceding section is employed to prepare the input data files. Similar to the physical experiment, no special treatment is taken to include the bound second harmonics in the incident waves. Due to the presence of the slot inside the dry beach, the whole channel is an active computational domain with two closed boundaries at both ends of the wave flume. To be able to resolve superharmonics in the wave train, the grid size and the time step is chosen to be 0.02 m and 0.01 s, respectively. With respect to the wave breaking model, the parameter ,$\eta_t^{(I)}$, is chosen to be the lower limit as described in Section 1.4. For the slot scheme, we use $\delta = 0.005$, and $\lambda = 80$.

The specification file for the case of WP1 is as follows

```
$data1$
ibe   =   2         imch  = 1           ianm  = 0
a0    = 0.025       h0    = 0.47        tpd   = 1.6667
dx    = 0.02        dt    = 0.01
mx    = 801         nt    = 5121
itbgn = 101         itend = 5121        itdel = 1
itscr = 100         itftr = 100
cbkv  = 0.65        ck_bt = 0.0
delta = 0.005      slmda = 70.0
isltb = 251         islte = 801
$end

$data2
isrc   =  251       swidth =  0.5
cspg   =  10.0      cspg2  =  0.0       cspg3  = 0.0
ispg   =  180       80
ngage  =  13
ixg    =  251  371   421   471   521   546   571   596   621   646
          671   691   721     0     0     0     0     0     0     0
itg    =  501   1001  2001  3001  4001  5001
$end

$data3
f1n   = 'depth.mase'
f2n   = 'initw.data'
f3n   = 'wp106.data'
f4n   = 'eta_106.out'
f5n   = 'tmp.out'
```

```
f6n   = 'eta_106.out'
f7n   = 'etaxt.out'
$end

$data4
itide = 0
tideco = 7.89  9.107e-05  -1.3737e-09
$end
```

The specification file for Case WP2 is similar to the case of WP1, thus it is omitted here. Comparisons of computed and measured surface elevation for the two test cases are presented in Figs. 10 and 11, including 11 wave gauges along the slope and a runup gauge. The dashed lines represent the computed results while the full lines are the measurements. Generally good agreement is found in both test cases.

First, we notice that the nonlinear shoaling of the bichromatic wave trains is well predicted by the Boussinesq model. Near the shoreline where wave breaking occurs, although a slight discrepancy is observed, the overall agreement is satisfactory between the computed surface elevation and Mase's (1995) data. Moreover, the modelled swash motions are in good agreement with the measurements. The good agreement demonstrates that the present Boussinesq model with the incorporation of the wave breaking model and the improved slot technique works reasonably well for the simulation of wave shoaling, breaking, and swash oscillation. The true value of the schemes is their feasible extension to two horizontal dimensions as shown in the following sections.

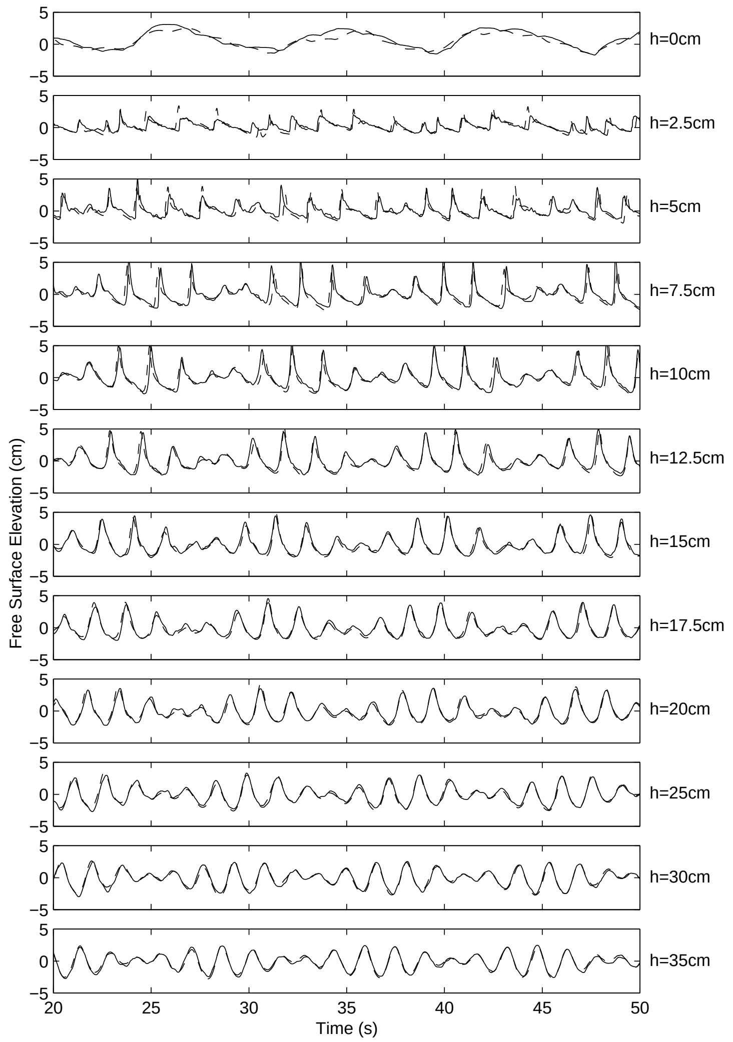


Figure 10: Comparisons of computed (dashed lines) and measured (full lines) free surface elevation including runup in the case of WP1

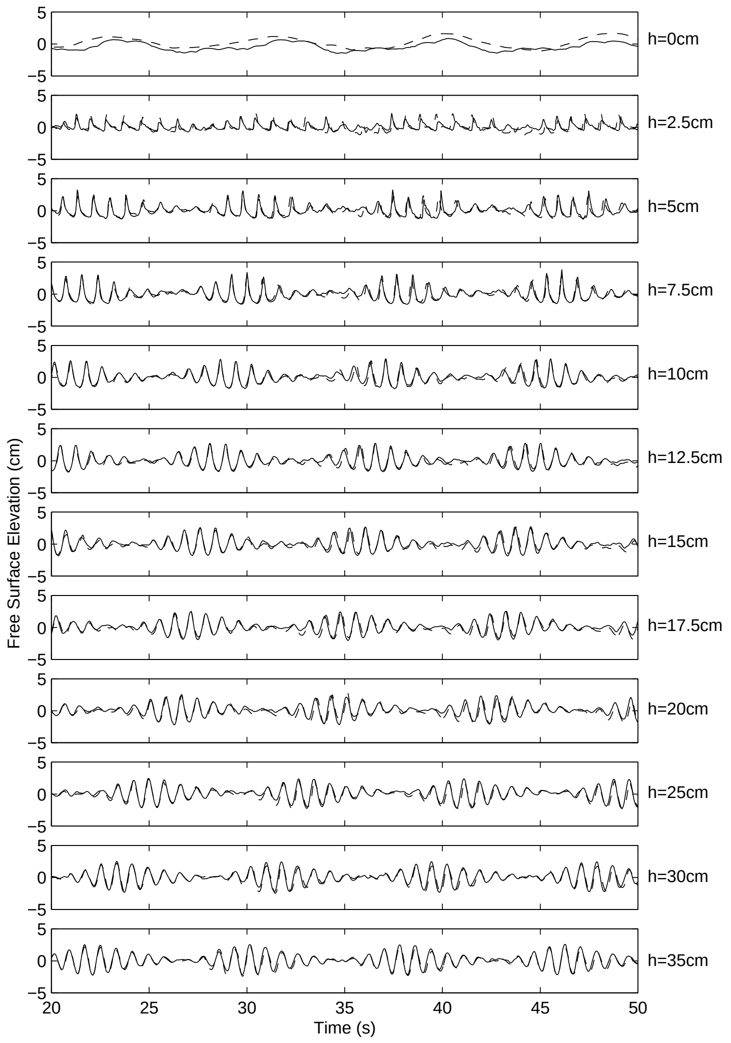


Figure 11: Comparisons of computed (dashed lines) and measured (full lines) free surface elevation including runup in the case of WP2

4.3 Wave Evolution in a Rectangular Basin

The complexity of the 2-D model requires careful programming and objective testing. One simple yet efficient testing case is wave evolution in a closed basin. By providing initial values of η, u and v for the first three time steps ($i.e.$,$it = -1, 0, 1$) to the model, we can obtain the subsequent variations of η, u and v. Analyzing these data has enabled us to correct coding errors and to discover the property of the model.

First, we use program *depth.f* to generate the bathymetric data. Secondly, program *inite.f* is invoked for the generation of initial condition. Unlike other cases for running the model with a 'cold start' condition, the initial values of η are not zero and are defined as

$$\begin{aligned} \eta_{i,j}^k &= a_0 * \exp[-\beta[(i-i_c)^2(\Delta x)^2 + (j-j_c)^2(\Delta y)^2]] \\ & i=1,2,...,M; \quad j=1,2,...,N; \quad k=-1,0,1 \end{aligned} \tag{90}$$

where a_0 is the initial height of the Gaussian hump as specified in *funwave2d.data* (set $a_0 = 0$ after running *initw.f*), β is the shape coefficient ($\beta = 0.4$ in this case), i_c and j_c are the grid numbers for the center of the basin in x and y directions, respectively. The Gaussian hump water is released in a rectangular basin with dimensions $10m \times 10m$ and with constant water depth $h0 = 0.5m$. Due to gravitational forcing, waves are generated and propagate out of the center and then are reflected back in the domain by four side walls. Since no sponge layers are used in this case and there is no wave breaking, there should be no energy loss. Though there exist no analytical solutions or experiment data for this case, the symmetric characteristics of the basin and initial conditions should result in symmetric spatial profiles of η, which are shown in Figure 12. The symmetric property had been proved to be an efficient way for checking coding errors in 2-D models.

The corresponding control parameter file *funwave2d.data* is:

```
$data1$
ibe   = 1          imch  = 1
a0    = 0.1        h0    = 0.50       tpd   = 1.0
dx    = 0.1        dy    = 0.1        dt    = 0.02
mx    = 101        ny    = 101        nt    = 2501
itbgn = 2301       itend = 2501       itdel = 1
itscr = 100        itftr = 100        theta = 0.
cbkv  = 0.0        delta = 0.0        slmda = 1.0
isltb = 101        islte = 101
$end

$data2
isrc   =   1         jsrc  =  1
cspg   =   10.0      cspg2 = 0.0     cspg3 = 0.0
ispg   =   1         1       1       1
ngage  =   6
ixg    =   1    26    51    1     101  101   1     1     1     1
           1    1     1     1     1    1     1     1     1     1
iyg    =   1    26    51    101   101  1     1     1     1     1
           1    1     1     1     1    1     1     1     1     1
itg    =   1  501  1001  1501 2001  2501
cbrk   = 0.0         ck_bt = 0.0           c_dm = 0.0
isld   = 0           idout = 0             idft = 0
```

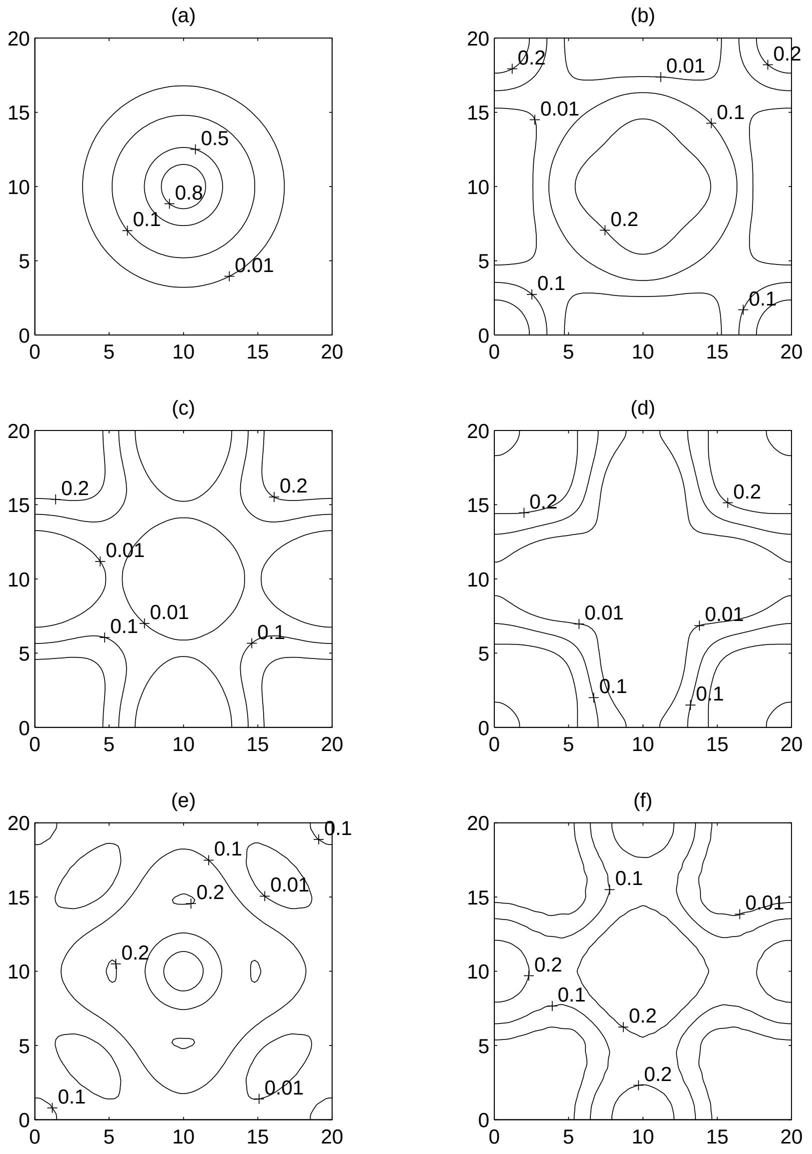


Figure 12: Contour plots of surface elevation at time (a) t=0; (b) t=10; (c) t=20; (d) t=30; (e) t=40; (f) t=50 (s).

```
$end

$data3
f1n = 'dpdata.gau'
f2n = 'inwdata.gau'
f3n = 'srcdata'
f4n = 'eta_ts.out'
f5n = 'muv_ts.out'
f6n = 'etaxy'
f7n = '/tmp/etts_w.out'
$end
```

Notice that the parameter *tpd* is meaningless in this case, but it should be larger than zero. The ratio of initial wave height to water depth is $\delta = a_0/h_0 = 0.2$, which is quite large. However, the subsequent wave height to water depth is only about 0.1, indicating that nonlinear effect is small. The numerical filter is applied for every $itftr = 100$ time steps.

By keeping all parameters in *funwave2d.data* the same except changing *ibe* from 1 to 4 ($ibe = 1$ corresponds to Nwogu's equations, $ibe = 2$ to fully nonlinear Boussinesq model, $ibe = 3$ to standard Boussinesq model and $ibe = 4$ to nonlinear shallow water model), we obtained and compared results from these four models. Figure 13 shows the time series comparisons of surface elevation η at the corner point and the center point of the rectangular domain. The resulting η from nonlinear shallow water model does not converge and causes the program to stop running at about $t = 21.7\ s$ (or $it = 1085$) due to excessive iteration times. Therefore, results from nonlinear shallow water model are not shown in Figure 13. Results of $\eta(t)$ from other three Boussinesq models are quite close, indicating that both effects of dispersion and nonlinearity are weak.

In addition to symmetry of η, another important property for this case is the conservation of water volume inside the basin. Since a source function is not applied here to generate waves and the four wall boundaries around the basin prevent the exchange of water in and out of the basin, the water volume should remain constant if correct boundary conditions are used in the numerical model. Figure 14 shows time series comparison of relative error of water volume E, which is defined as as

$$E(t) = \frac{V(t) - V(0)}{V(0)} \tag{91}$$

where $V(t)$ is the total water volume of the Gaussian hump and is obtained at each

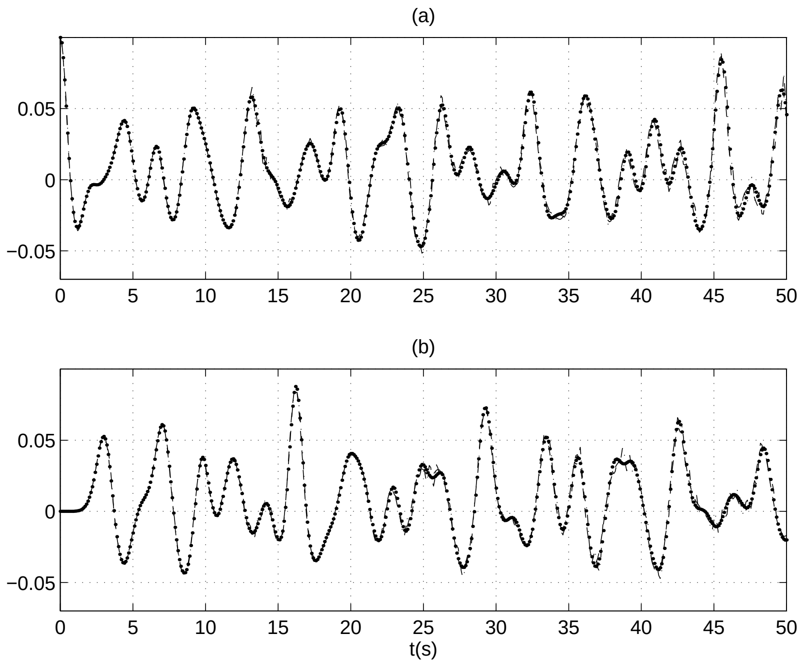


Figure 13: Comparisons of time series of surface elevation η for three models: dashed line – Nwogu's model; dash dot line – fully nonlinear Boussinesq model; dot line – standard Boussinesq model. (a) is at corner point and (b) at center point.

time step by the approximate formula

$$V(t) = \tfrac{1}{4}\Delta x \Delta y \sum_{i=1,j=1}^{M-1,N-1} (\eta_{i,j} + \eta_{i+1,j} + \eta_{i,j+1} + \eta_{i+1,j+1}) \tag{92}$$

As shown from Figure 14, the relative errors E of water volume from three Boussinesq models are all less than 1% of the initial values, indicating that the conservation property of these models works quite well. The smallest values of E correspond to the standard Boussinesq model, where the corresponding wall boundary conditions described in Section 2.2.1 are exact, due to the use of depth-averaged velocity as a dependent variable. For Nwogu's model and the fully nonlinear Boussinesq model which use velocity at the depth $z_\alpha = -0.53h$ as a dependent variable, the wall boundary conditions described in Section 2.2.1 are only accurate to the order of approximation for these models. Therefore, results from these two models are not as good as those from the standard Boussinesq model.

Although no problems have been found related to the mass conservation properties of the model for simulation of nearshore wave propagation without wave breaking, the

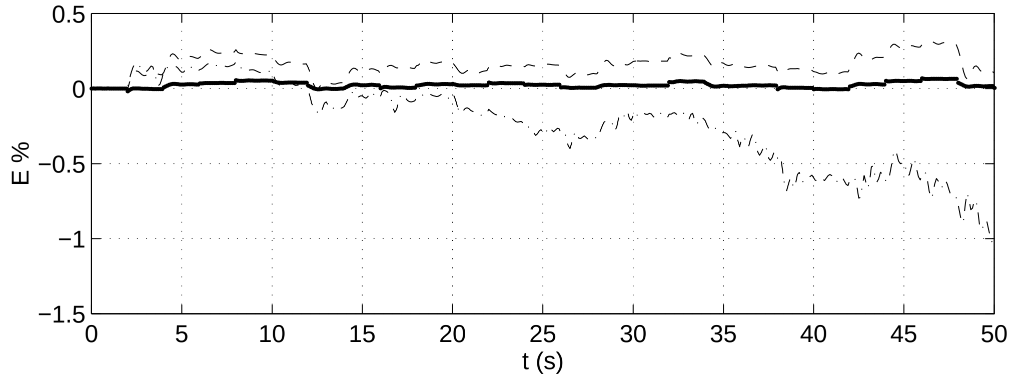


Figure 14: Comparisons of time series of relative error of water volume E for three models: dashed line – Nwogu's model; dash dot line – fully nonlinear Boussinesq model; dot line – standard Boussinesq model.

accumulation of error in the total water volume of the domain may affect the computation of breaking-induced wave setup. This error has been removed by correcting the mass residual in the 2-D code. A more formal correction by strictly imposing the zero-flux condition on closed boundaries will be made in the future release of this model.

4.4 Wave Propagation over a Shoal: Berkhoff *et al.* (1982)

The experiment conducted by Berkhoff *et al.* (1982) has served for a number of years as a standard test for verifying models based on mild slope equation. Correct reproduction of measured wave heights in this experiment depends on a number of factors, including shoaling, refraction, diffraction and nonlinear dispersion. In particular, the central importance of nonlinearity in the experiment was demonstrated by Kirby & Dalrymple (1984).

However, due to large kh values in the experiment ($kh = 1.9$ near wavemaker), models based on nonlinear shallow water equations or standard Boussinesq equations are not appropriate in this situation. However, with improved linear dispersion property in intermediate water depth, the present model based on Nwogu's extended Boussinesq equations or the fully nonlinear Boussinesq equations can be applied and give accurate prediction of the data.

First, we generate the bathymetric data using program *depth.f*. The bottom topography is shown in Figure 15, which 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 $1s$ and amplitude $2.32cm$ 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).

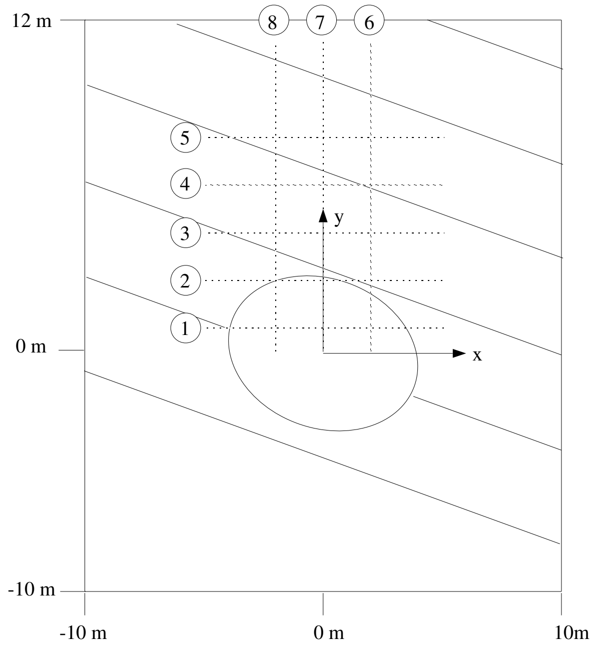


Figure 15: Bottom topography for experiment of Berkhoff *et al.* (1982)

The computing domain used in the model is the same as in the Figure 15 except for two sponge layers with width 2 m and 3 m sitting behind wavemaker and on the end of the beach. Instead of shoreline boundary, a minimum water depth of 0.07 m is used in the model. The source function for generating the corresponding monochromatic wave is located at the wavemaker. The model is run with a cold start. Again, the data file for the inital condition is generated by the use of program *initw.f*. We give the specification file *funwave2d.data* as follows:

```
$data1$
ibe    = 1           imch  = 1
a0     = 0.0232      h0    = 0.45      tpd    = 1.0
dx     = 0.05        dy    = 0.1       dt     = 0.01
mx     = 561         ny    = 201       nt     = 4001
```

```
itbgn = 3601        itend = 4001      itdel = 1
itscr = 100         itftr = 400       theta = 0.0
cbkv  = 0.0         delta = 0.0       slmda = 1.0
isltb = 561         islte = 561
$end

$data2
isrc  =   41       jsrc  =  1
cspg  =   10.0     cspg2 = 0.0    cspg3 = 0.0
ispg  =   41       61       1       1
ngage =   14
ixg   =   141  241  261  301  341  381   421  441  261 301
          341  381  421  441  1    1     1    1    1   1
iyg   =   101  101  101  101  101  101   101  101  81  81
          81   81   81   81   1    1     1    1    1   1
itg   =   401  801  1601 2401 3201 4001
cbrk  = 0.0         ck_bt = 0.0           c_dm = 0.0
isld  = 0           idout = 0             idft = 0
$end

$data3
f1n = 'dpdata.bbr'     f2n = 'inwdata.bbr'  f3n = 'srcdata'
f4n = 'eta_tsx.out'    f5n = 'eta_tsy.out'  f6n = 'etaxy'
f7n = '/tmp/etts.out'
$end
```

From the above input data file, the number of time steps for the model to run for this case is $nt = 4001$, which is equivalents to 40 wave periods. Numerical filter (2-D nine point weighted-average) is used in this example for every 400 time steps (*i.e. itftr = 400*), which eliminates the potential unstable problem caused by short waves generated by the model. Figure 16 shows the spatial profile of surface elevation at the last time step from the model. The dark and light shade regions corresponding positive and negative values of η, respectively. The solid lines in the plot denote the contours of bottom geometry. The surface profiles show strong focus of wave energy behind the shoal, indicating large effect of wave diffraction which prohibits the use of ray tracing method but is valid for models based on mild slope equation. The uniformly grey shade at the two sponge layer regions corresponds to near zero surface elevation, indicating wave energy was absorbed properly at those boundaries.

Figure 17 shows the time series of surface elevation η at various locations, with their corresponding coordinates (x, y) indicated in the plot. Since cold start was used as initial conditions in the model, surface elevation η at $t = 0$ are zero everywhere. As

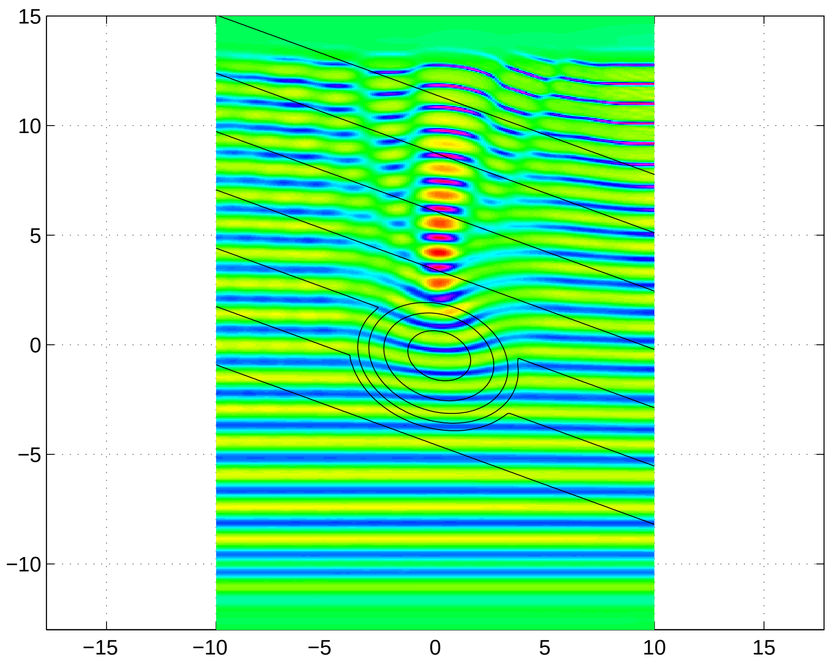


Figure 16: Computed time series of surface elevation at various locations

waves are generated in the source region and propagate, surface elevations at specified points start to increase in magnitude, until reach corresponding stable values, with points close wavemaker taking less time for the transition period. Surface elevations from the points inside strong focus region have narrow crests and broad troughs, indicating that nonlinear interaction has effect on wave transformation. However, there is no experimental data available to compare with these results. Figure 17 shows that the computed wave field reaches a stable state after $t = 30\ s$.

Once the model reaches quasi-stable condition, wave amplitude at various points can be computed from the corresponding time series of η. The output file $f7n$ stores the spatial profiles of η for the last 400 time steps ($i.e.$, from $it = 3601$ to $it = 4001$, or 4 wave periods), which is equivalent to storing the time series of η for the last 400 time series for every grid point. Using a zero-crossing method, the corresponding wave amplitude for every grid point can be computed. This is done by the post-processing program $b2dp2.f$. Figure 18 shows comparisons between model results and experimental data along the eight transects where measurements were made.

The computing results of wave heights from the numerical model agree quite well with experimental data, in both sections parallel or normal to incident wave direction. In the past, models based on mild slope equations have been applied to this case and,

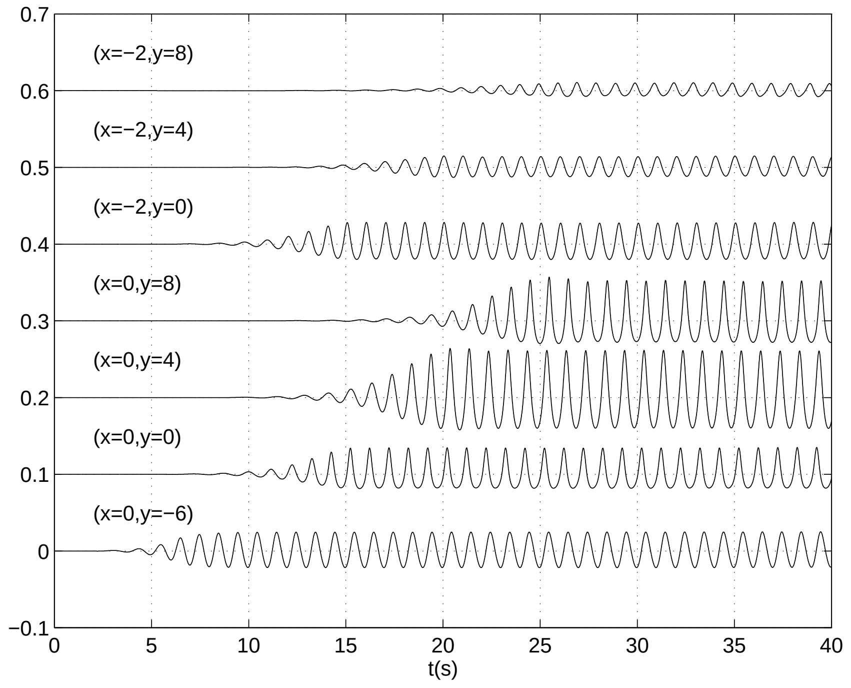


Figure 17: Computed time series of surface elevation at various locations

in general, good agreement has been found, see, for example, Berkhoff *et al.* (1982), Kirby and Dalrymple (1984), and Panchang and Kopriva (1989). Among mild slope models, the one given by Kirby and Dalrymple (1984) used third-order Stokes wave dispersion and showed the best agreement, indicating nonlinearity is important for accurate modeling. This is also verified by the present Boussinesq model which results in better agreement with data compared to mild slope models using linear dispersion relation.

4.5 Wave Propagation over a Shoal: Chawla *et al.* (1996)

4.5.1 Non-breaking Waves

To study wave transformation in coastal regions and to provide measured data for comparison with *REF/DEF* model and other wave models, Chawla et al. (1996) conducted a series of experiments in the wave basin at the Center for Applied Coastal Research in the University of Delaware. The experiments consist of regular and random waves, with low and high wave amplitude (corresponding to nonbreaking and breaking cases, respectively), and with broad and narrow frequency band and

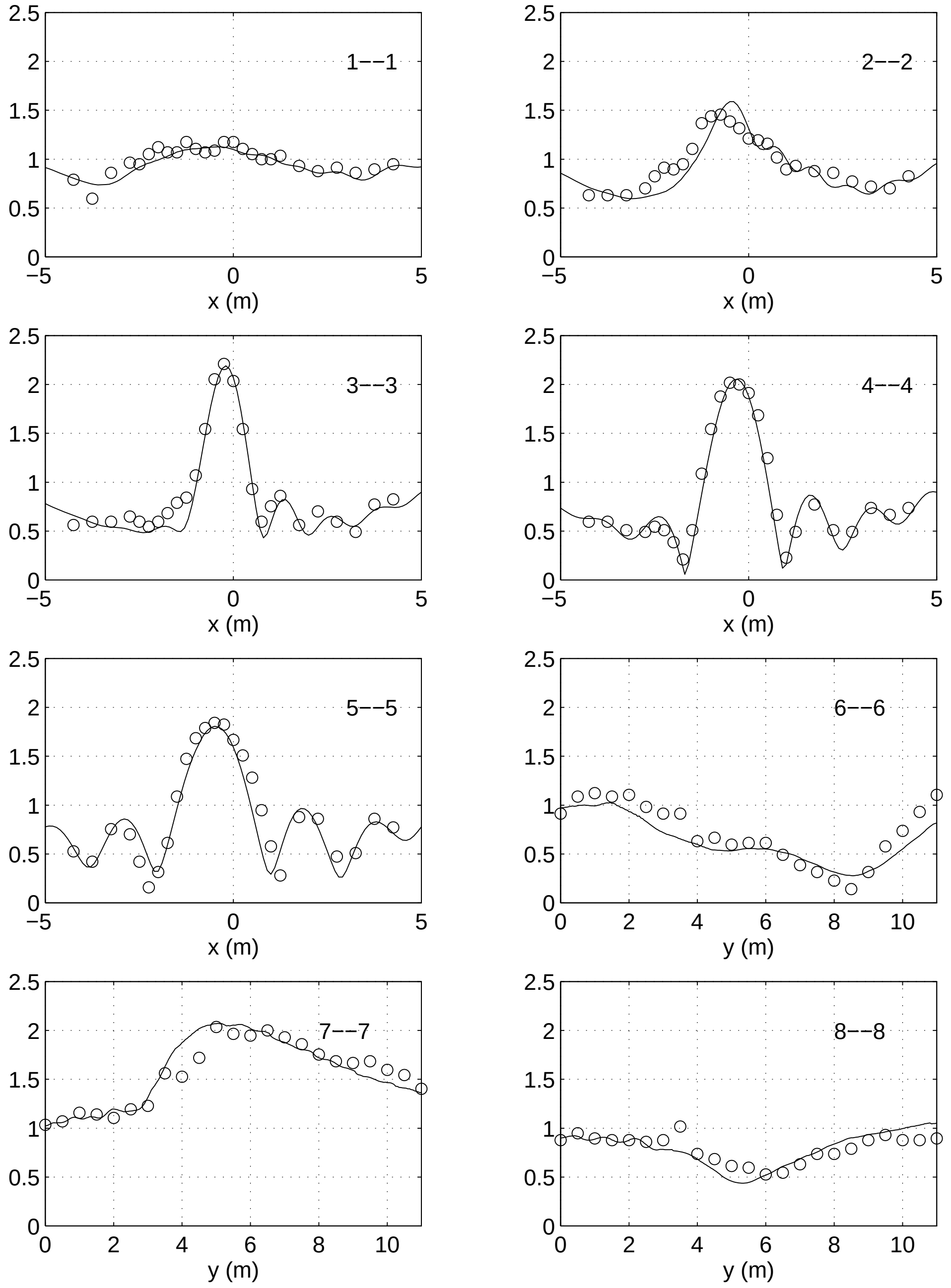


Figure 18: Comparisons of amplitude along specified sections: solid line – model, circle – experiment data.

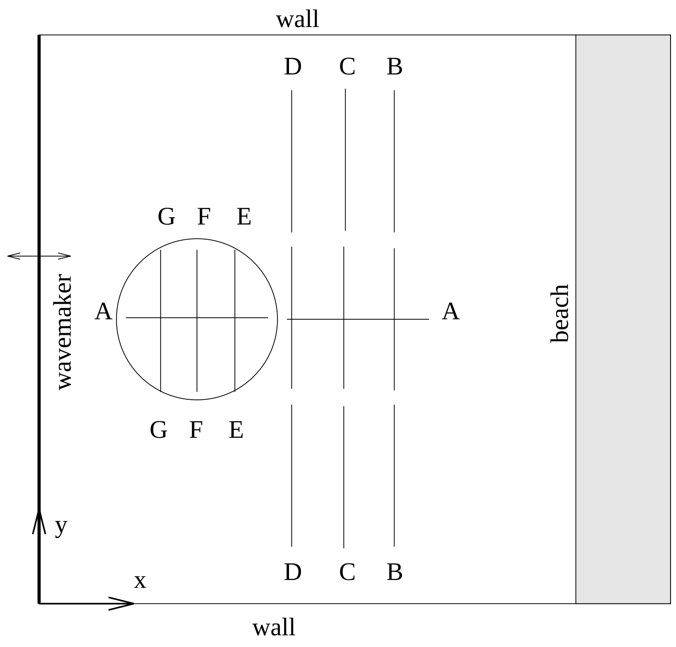


Figure 19: Experiment setup of Chawla *et al.* (1996).

directional spreading. Figure 19 shows the experiment layout, where a circular shoal with radius of 2.57 m is built on a flat bottom basin of dimension $18.2m \times 20m$.

Incident waves are generated by a flap-type wavemaker on the left end of the domain and propagate across the basin. On the right end there is a beach which consists of small rocks and dissipates wave energy. The top and bottom edges are vertical walls which allow waves to reflect back to the domain. Time series of surface elevation η are collected by the gage array which locates along the transects of $A-A$, $B-B$, $C-C$, $D-D$, $E-E$, $F-F$, and $G-G$. The corresponding wave amplitudes at the wave gage locations are computed from the collected data. The detailed setup of the experiment can be found in Chawla (1995).

Three sets of data (TEST 1, TEST 2 and TEST 4) are available for nonbreaking regular wave experiments. The numerical model was applied to all three test cases and good agreement between data and model was found. In the following, only results from TEST 4 from data and model are compared in details. First, the bathymetric data file is generated by program *depth.f*. The next step is to prepare the initial data file using program *initw.f*. We start the model from still water (i.e. cold start). The specification file *funwave2d.data* for this case is:

```
$data1$
ibe   = 1          imch  = 1
a0    = 0.00583    h0    = 0.45      tpd   = 1.0
dx    = 0.05       dy    = 0.1       dt    = 0.01
mx    = 441        ny    = 183       nt    = 4001
itbgn = 3601       itend = 4001      itdel = 1
itscr = 100        itftr = 200       theta = 0.0
cbkv  = 0.0        delta = 0.0       slmda = 1.0
isltb = 441        islte = 441
$end

$data2
isrc  =   41        jsrc  =  1
cspg  =   10.0      cspg2 = 0.0    cspg3 = 0.0
ispg  =   41        61       1       1
ngage =   20
ixg   =   103  140  165  177  189  201  219  232  244  256
          143  168  168  201  201  201  234  234  234  234
iyg   =   91   91   91   91   91   92   91   91   91   91
          113  101  113  101  113  127  101  113  128  143
itg   =   401  801  1601 2401 3201 4001
cbrk  =   0.0      ck_bt = 0.0    c_dm = 0.0
isld  =   0        idout = 0      idft = 0
$end

$data3
f1n = 'dpdata.cacr'
f2n = 'inwdata.cacr'
f3n = 'srcdata'
f4n = 'eta_tsx.out'
f5n = 'eta_tsy.out'
f6n = 'etaxy'
f7n = '/tmp/etts.out'
$end
```

Except for the number of grids (*i.e.* *mx, ny*) and input data file names for $f1n$ and $f2n$, The parameters in the above input file are the same as those in previous section. Since the experiment data for time series of surface elevation $\eta(t)$ are available, we compare these results of $\eta(t)$ with the numerical models, as shown in Figure 20. Good agreement is found not only in wave heights and wave phases, but also in the asymmetry of wave shapes.

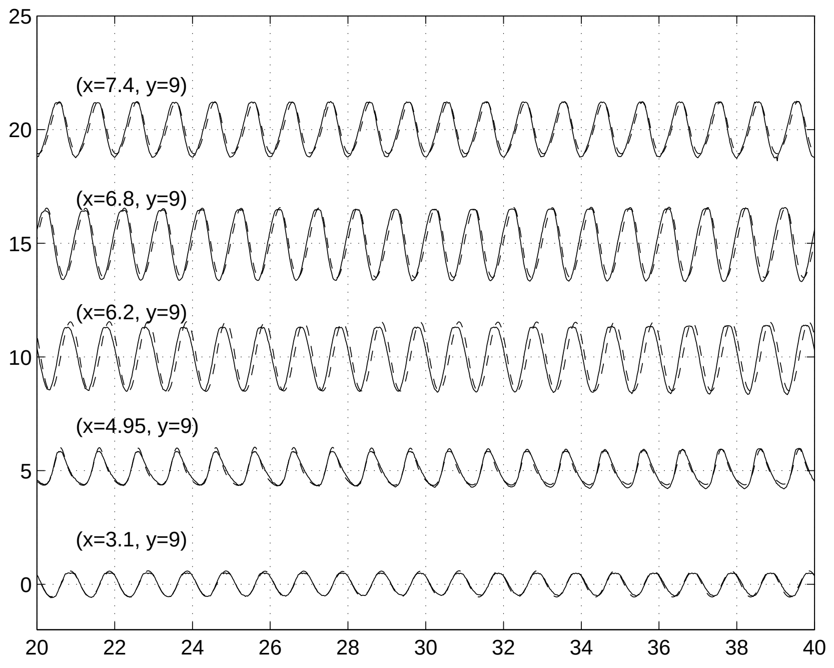


Figure 20: Computed (dashed line) and measured (solid line) time series of surface elevation at various locations for Test 4 of Chawla et al. (1996).

In the experiment, data from nine gages in the gage array were collected simultaneously. Then the gage array was moved to a new position to take measurement by repeating the same experiment. Due to the uncertainty for the exact time to start each experiment, the original time series of data exhibit arbitrary phase shift for different gage array positions. It is necessary to adjust experimental data by shifting phase for meaningful comparison with the numerical results. For each gage array, only one gage is required to do phase shift due to exact synchronization for all gage data.

After wave fields reached stable condition, zero-upcrossing method was applied to the subsequent time series of surface elevation for obtaining wave heights. For experiment data, a time interval of 10 periods was used to compute averaged wave height. Prior to wave height computation, a Butterworth fifth-order bandpass filter was first applied to eliminate noise or higher harmonic components. For time series computed from the model, data segments from $t = 36\ s$ to $t = 40\ s$ were used to compute wave height. Since numerical filtering has been applied spatially for every 400 time steps in the model simulation, it is not necessary to apply extra filter to the resulting time series. Comparisons of wave amplitude between the model and data along all transects from $A - A$ to $G - G$ are shown in Figure 21.

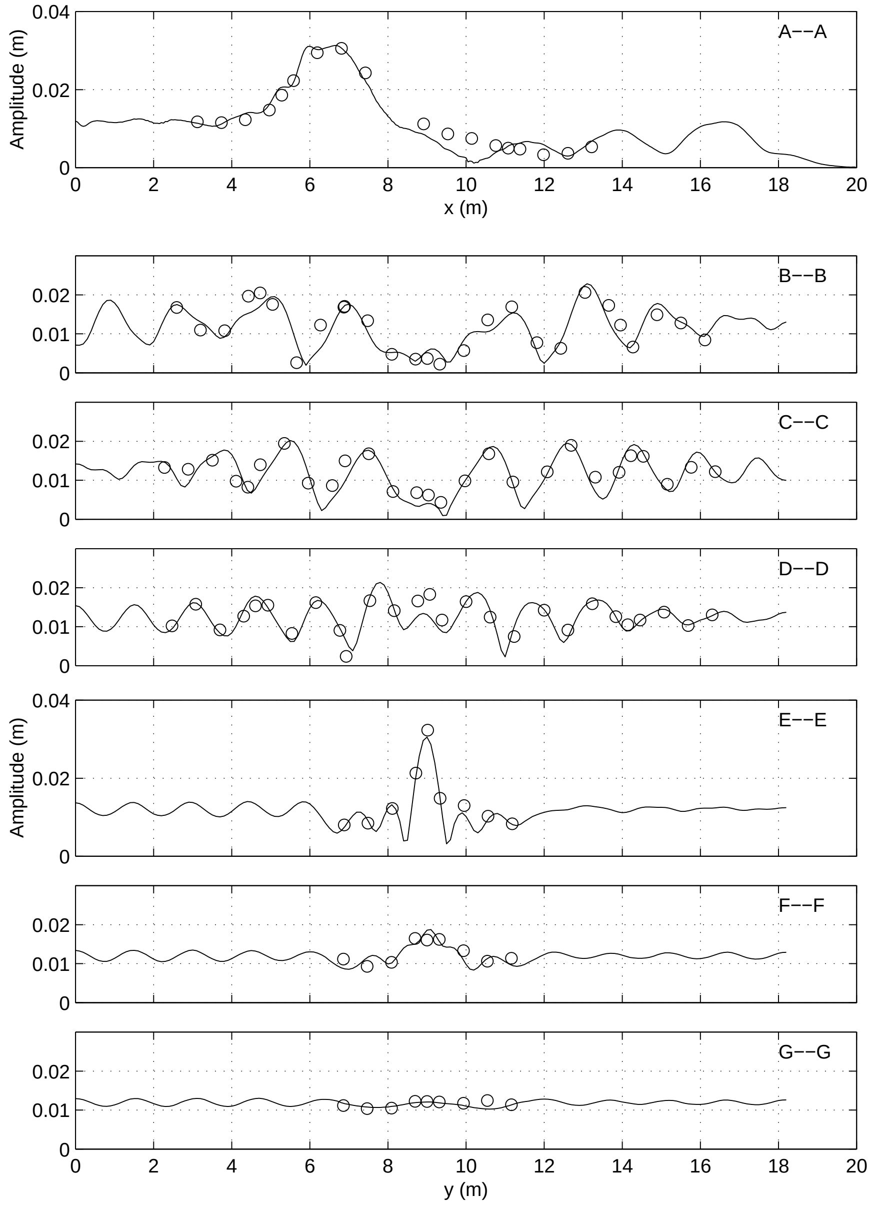


Figure 21: Comparison of wave amplitude along specified transects: solid line – model, circle and star – experiment data.

Due to a slightly off-center position for the shoal location in the basin, the wave height distributions along y axis are not symmetric. The asymmetry becomes more apparent for transects away from the wavemaker, indicating reflecting effect from side walls increases. The effect of asymmetry and the zigzag variation of wave height distribution along y axis were accurately predicted by the numerical model.

4.5.2 Breaking Waves

A more demanding test for the Boussinesq model is wave propagation and breaking over a submerged shoal. We choose another test case from Chawla et al.'s (1996) experiment with wave breaking to verify our model. In this case, the water depth h_o is 39.5 cm. The input monochromatic wave has 2 cm wave height and 1 s period. As the front face of a breaking/broken wave becomes very steep, finer grid size in comparison with that for the case of non-breaking is required in order to resolve the wave. Thus we reduce the grid size in the x direction to 0.025 m, leading to about 20 grids per wave length on top of the submerged shoal, while the grid increment along the y axis and the time step remain identical to those in the case of non-breaking waves.

The specification file is given as follows

```
$data1$
ibe   = 2         imch  = 1
a0    = 0.01      h0    = 0.395     tpd   = 1.0
dx    = 0.025     dy    = 0.1       dt    = 0.01
mx    = 881       ny    = 183       nt    = 5001
itbgn = 3001      itend = 5001      itdel = 1
itscr = 50        itftr = 100       theta = 0.
cbkv  = 0.35      delta = 0.0       slmda = 1.0
isltb = 881       islte = 881
$end

$data2
isrc  =   81        jsrc  =  1
cspg  =   20.0      cspg2 = 0.0    cspg3 = .0
ispg  =   61        81       1       1
ngage =   21
ixg   =   100  113  125  137  143  149  162  174  186  218
          230  242  254  261  267  279  291  304  1    1    1
iyg   =   91   91   91   91   91   91   91   91   91   91
          91   91   91   91   91   91   91   91   1    1    1
```

```
itg   =   1001  2001 3001 4001 4501 5001
cbrk  =   1.2  ck_bt =1.0e-3   c_dm = 0.01
isld  =   0    idout = 0       idft = 0
$end

$data3
f1n = 'dpdata_bk.cacr'
f2n = 'inwdata_bk.cacr'
f3n = 'srcdata'
f4n = 'eta_tsx_bk.out'
f5n = 'eta_tsy_bk.out'
f6n = 'etaxy'
f7n = '/tmp/etts.out'
$end
```

First, we use program *depth.f* to generate the bathymetric data. Then, program *initw.f* is employed to prepare the initial condition. Starting from still water, the Boussinesq model is run for 50 seconds. To remove the effects of transients associated with the cold start of wave field and the wave breaking, we compute the root-mean-square wave height H_{rms} using the last 20 seconds of numerical results and the collected data. The empirical parameters for the wave breaking model are chosen to be the lower limit of the values indicated in Section 1.4. Figure 22 presents the comparisons of the modelled results and measurements along the longitudinal transect (A-A) with respect to the normalized H_{rms}, skewness and asymmetry. The computed H_{rms} and third-moments agree fairly well with the laboratory data.

The bottom topography along the A-A transect is shown by panel d in Figure 22. Several interesting phenomena are observed from Figure 22. First, we notice that the wave height does not reach the largest on top of the shoal but on the downward slope instead. This is attributed to the focusing effect of wave refraction on the shoal. However, the wave skewness and asymmetry appear to be the maximum near the crest of the shoal, as indicated by both the numerical and measured results. It is known that skewness and asymmetry are a measure of wave nonlinearity. Apparently, the degree of wave nonlinearity at the focusing point with the maximum wave height is weaker than that on top of the shoal where the water depth is the minimum. Secondly, both depth-limited wave breaking and wave de-focusing reduce the wave height. The combined decrease of wave height is much faster in comparison with the case of non-breaking waves. Consequently, the large gradient of radiation stresses will drive horizontal circulations around the submerged shoal.

Figure 23 depicts the data/model comparison of normalized H_{rms} along six transverse transects. First, on top of the shoal (i.e. F-F), we notice that the agreement is fairly

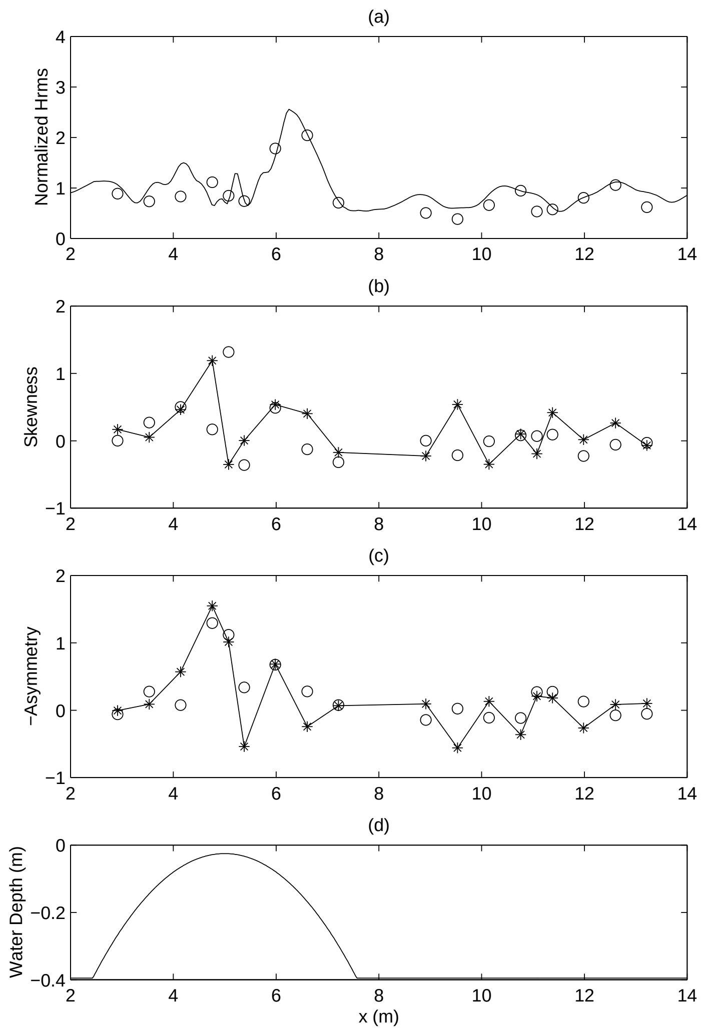


Figure 22: Model/data comparisons along A-A transect: (a) normalized H_{rms}; (b) normalized skewness; (c) normalized asymmetry; (d) bottom topography. Solid Lines: model; circles: data.

good. On the downward slope, the Boussinesq model captures the focusing effects very well as shown by E-E transect. Furthermore, the de-focusing and diffraction of the broken waves behind the shoal (i.e. B-B to D-D) are also predicted reasonably well by the Boussinesq model.

A computed wave field at the end of the simulation is shown in Figure 24 where the gray areas represent the modelled wave crests while the dark areas are the wave troughs. Wave refaction over the shoal is clearly shown by the bending of wave crests on top of the shoal. Wave diffraction is also visible by the variation of wave crests in the transverse direction. It is worth mentioning that secondary wave crests are observed behind the submerged shoal due to the release of superharmonics generated by nonlinear shoaling. However, the magnitude of harmonics higher than the primary wave is somewhat over-estimated and their phases may be inaccurate because of the large wave numbers of the higher harmonics at those locations. Model equations with Pade [4,4] dispersion properties applicable to $kh = 6$ (e.g. Chen et al., 1998a, Madsen and Schäffer, 1998a, Gobbi et al., 1999) are therefore needed in order to accurately model the released, free, higher harmonics in these two test cases.

There is no measurement of breaking-induced circulation in the Chawla et al.'s (1996) experiments. However, a strong jet associated with wave breaking on top of the shoal was visually observed during the experiment. The information of circulation can be extracted from the numerical results. Figure 24b illustrates the underlying current field generated by wave breaking over the shoal after 50 seconds have elapsed in the simulation. The current field is obtained by averaging the instantaneous fluid particle velocity at the reference level z_α over two wave periods. In connection with the simulation of breaking-induced currents, the bottom friction coefficient of $f = 1.0 \times 10^{-4}$ and a subgrid mixing model is used. Notice that the jet-like current tends to be unstable and vortices are likely to appear as shown by the meandering of the computed current and the vortex pair. An account of the instability of jet-like rip currents and the mechanism of vortex generation is given in Chen et al. (1999b).

4.6 Wave Runup on a Conical Island

Briggs et al.'s (1995) laboratory experiment on solitary wave runup on a conical island has served as a benchmark for the verification of tsunami runup models (see e.g. Liu et al., 1995; Titov and Synolakis, 1998). We shall use the measurements from their physical experiment to validate our runup schemes for two horizontal dimensions.

A schematic view of the wave basin for Briggs et al.'s (1995) experiments is shown in Figure 25 where solitary waves are generated on the western boundary and propagate toward the eastern boundary. The wave basin is 25 m long and 30 m wide. A conical

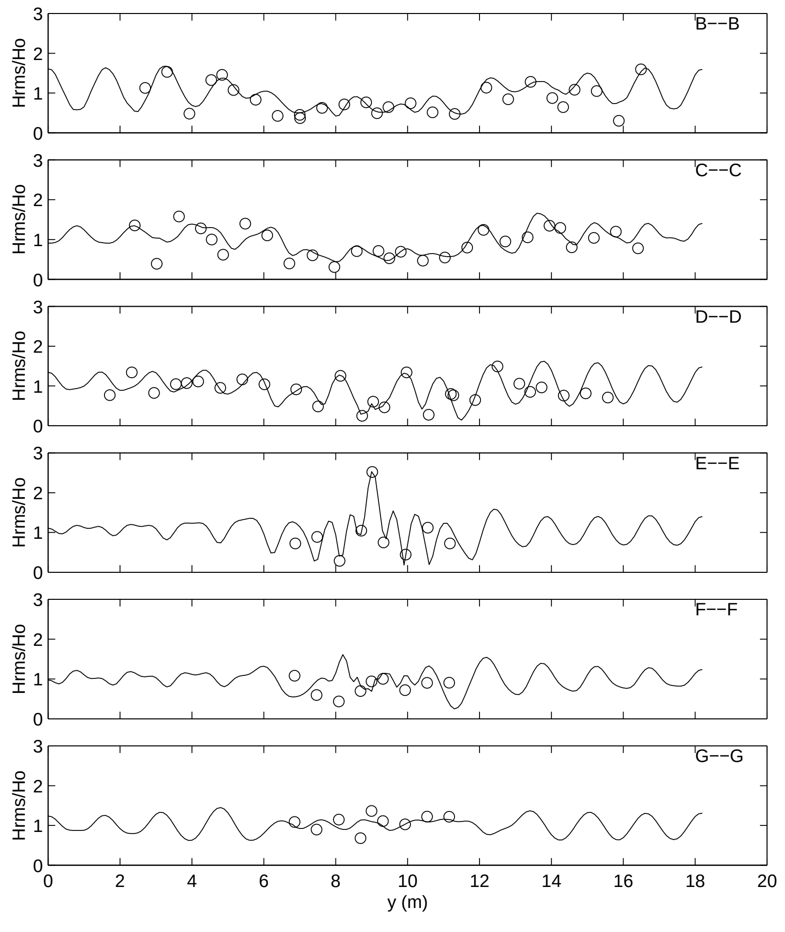
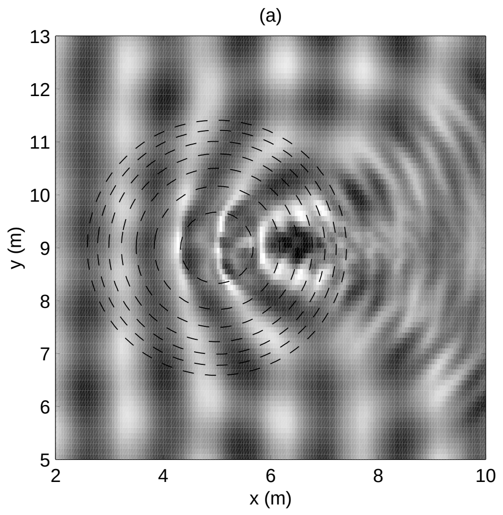


Figure 23: Comparisons of computed H_{rms} and Chawla et al.'s (1996) measurements with wave breaking along transverse transects. Solid Lines: model; circles: data.



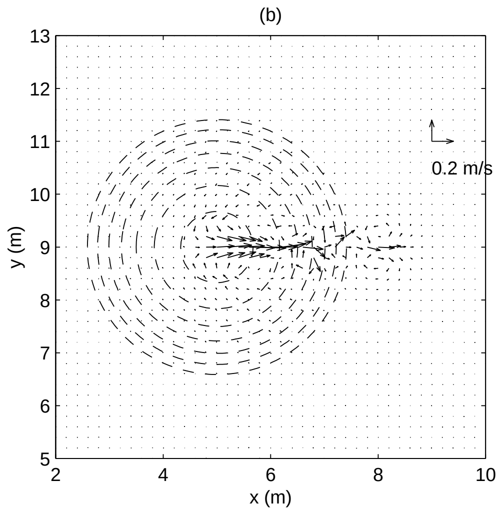


Figure 24: Illustration of (a) computed wave fields and (b) modelled underlying current field. The dashed lines are the contours of water depth.

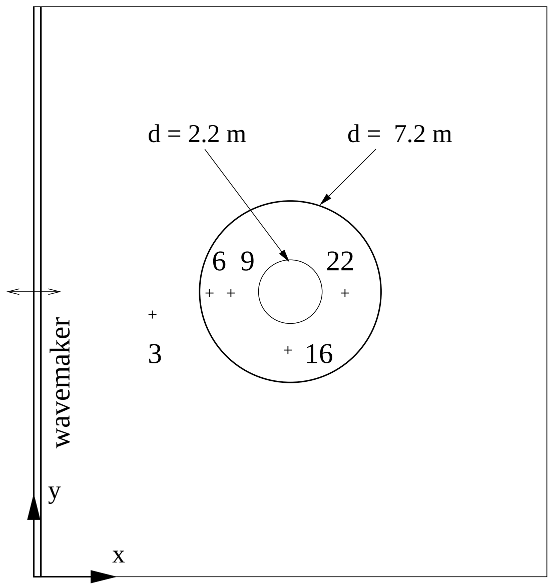


Figure 25: Schematic of the island and wave gauges in Briggs et al.'s (1995) Experiment.

island with a slope of 1:4 is placed on an otherwise flat bottom in the basin. The center of the island is located at $x = 13$ m and $y = 15$ m. The diameters of the island on the bottom, at the still water line, and on the top are respectively 7.2 m, 4.64 m, and 2.2 m. The water depth of the basin is 0.32 m. Further information about the experiment set-up can be found in Briggs et al. (1995) and Liu et al. (1995).

There are three test cases with available data sets including measurements of maximum runup height and free surface elevation around the island. The initial conditions for the three cases have $\epsilon = 0.05$, 0.1 and 0.2, respectively. ϵ denotes the height-to-depth ratio of the incident solitary waves. The water depth remain the same in all three cases. The grid size is chosen to be 0.1 m in both x and y directions. 0.02 second is used as the time step. Solitary waves are generated by defining the initial conditions for the model based on the analytical solutions to the Boussinesq equations with a constant water depth as implemented in *init.f*. The crest of the solitary waves at $t = 0$ is located at $x = 1.0$ m. At the four lateral boundaries, closed boundaries are imposed. For illustration, we give the specification file for Case 2 as follows

```
$data1
ibe   = 2        imch  = 1
a0    = 0.032    h0    = 0.320    tpd   = 1
dx    = 0.1      dy    = 0.1      dt    = 0.02
```

```
mx    = 351       ny    = 301       nt    = 1201
itbgn = 801       itend = 1201      itdel = 1
itscr = 20        itftr = 150       theta = 0.0
cbkv  = 0.35      delta = 0.02      slmda =  25.0
isltb = 101       islte = 351
$end

$data2
isrc  =   51        jsrc  =  1
cspg  =   10.0      cspg2 = 0.0    cspg3 = 0.0
ispg  =   11        51       1       1
ngage =   23
xg    =  119 135 145 149 152 154
         181 181 181 181 181 181
         208 210 213 217 227 242
         158 204 1   1   1
iyg   =  159   151   151   151   151   151
         177   180   183   187   197   212
         151   151   151   151   151   151
         151   151   1     1     1
itg   =  491 501 511 521 531 541
cbrk  =  1.2  ck_bt = 1.0e-6  c_dm = 0.0
isld  =  1     idout = 0       idft = 0
$end

$data3
f1n = 'island.bat'
f2n = 'island.int'
f3n = 'srcdata'
f4n = 'eta_tsisld.out'
f5n = 'eta_tsyrip.out'
f6n = 'etaxy'
f7n = '/tmp/etts_w.out'
$end
```

First, we prepare the bathymetry using *depth.f*. As the second step, program *initw.f* is employed to generate the initial solitary waves. a_0 in *funwave2d.data* should be set to zero after running *initw.f*. Notice that $isld = 1$ in this case. Figure 26 presents model/data comparisons of the maximum runup heights around the island for the three test cases. The full lines represent the numerical results while stars denote the data measured by Briggs et al. (1995). The horizontal axis is the angles between the radius and the center line of the island in the incident wave direction. In other words, a zero degree is the front side of the island while the location with 180 degrees is the lee side. Runup heights are normalized by the height of the incident solitary wave. The model results of runup are stored in *rpild.out*. As the measurements show that the maximum runup heights are not perfectly symmetric about the center line of the island, we compare the numerical results with the averaged data. Good agreement between model predictions and measurements is observed. The runup scheme captures the signature of two dimensional runup as shown by the correct variation of the runup heights around the island. Although a slight discrepancy is found on front side of the island in the comparison of Case 3, the overall agreement is as good as the non-breaking cases.

In addition to the distribution of maximum runup heights, we compare the computed time series of free surface with the measurements at five locations in Case 2. Gauge 3 is located close to the input boundary while the other four gauges are around the island near the still water shoreline. Figure 27 shows that the computed primary waves agree reasonably well with the measurements.

The present model with permeable-seabed technique for shoreline runup, however, appears to under-predict the depression of the free surface, or the reflected waves by the island. This is shown by the model/data comparison of Gauge 9 located in front of the island. The discrepancy is attributed to the slight loss of wave energy and momentum because of the presence of the narrow slot. Owing to the very steep slope of the island (1:4) and the high nonlinearity of the solitary waves in Cases 2 and 3, a slot width ten times larger than the optimal values (see Kennedy et al., 1999) are used here for the concerns of numerical instability. A low pass filter localized in the swash region is also utilized to suppress possible noises due to the use of a slot.

As discussed in Liu et al. (1995), a collision of the trapped solitary waves on the lee side of the island could lead to a runup height which can exceed the maximum runup on front side of the island. Figure 28 illustrates the collision process in Case 2. The time sequence of the computed free surface shows the scenarios of head on and collision of the trapped solitary waves on the lee side of the island. The generally good agreement in this section demonstrates the capability of the model for the simulation of shoreline runup in two horizontal dimensions.

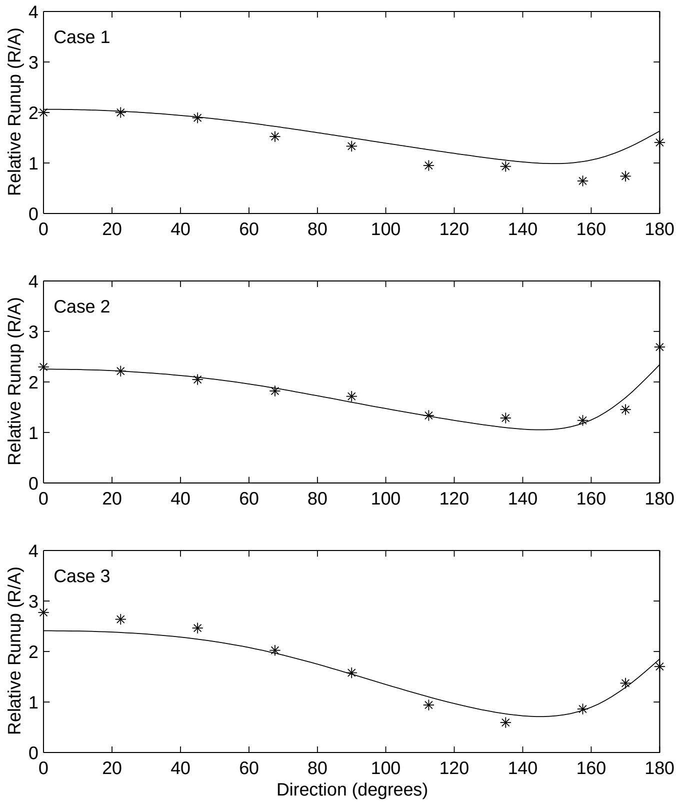


Figure 26: Comparison of computed and measured runup heights. Stars: measured; solid lines: computed.

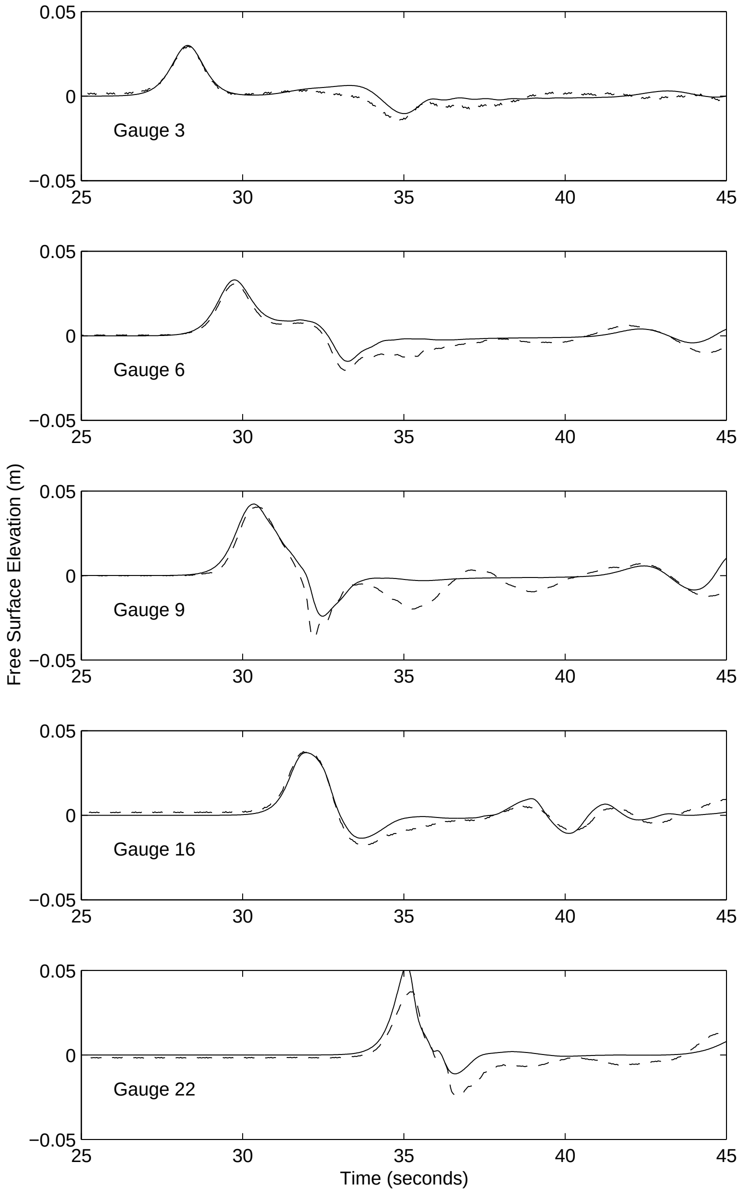
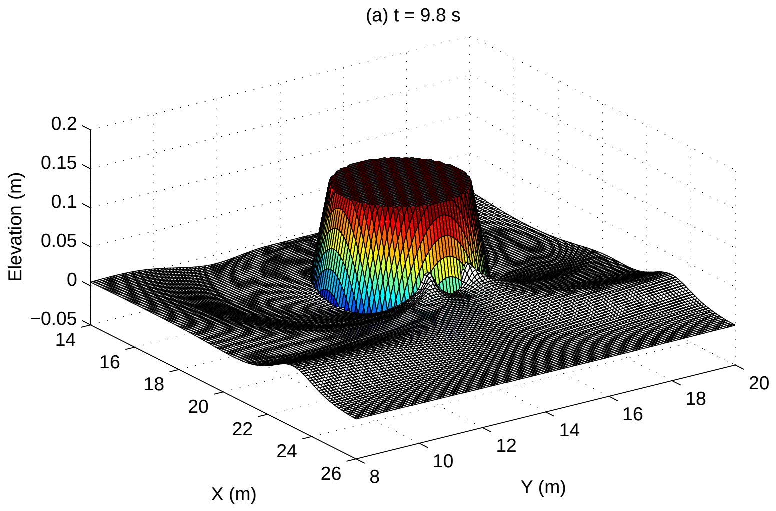


Figure 27: Comparison of computed and measured time series of free surface in Case 2. Dashed lines: measured; solid lines: computed.



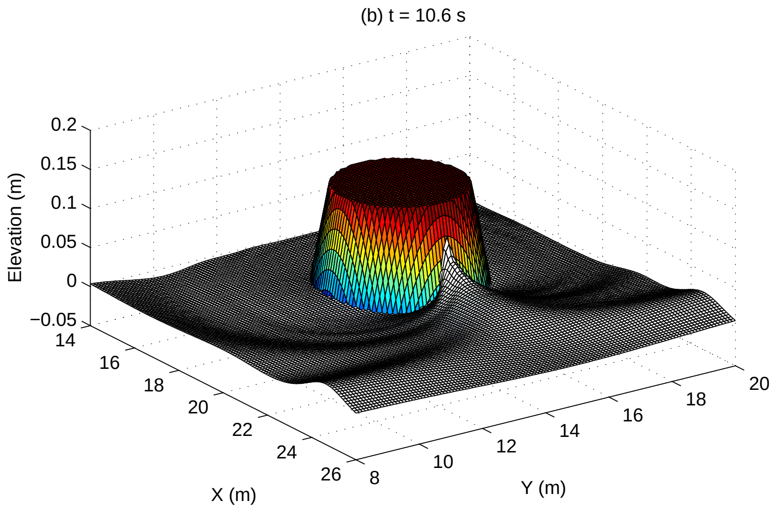


Figure 28: Sequence of solitary wave runup on the lee side of the island in Case 2.

5 REFERENCES

Briggs, M. J., Synolakis, C.E., and Harkins, G. S., 1994, "Tsunami run-up on a conical island." *Proc. Waves - Physical and Numerical Modelling*, Vancouver, Canada, 446-455.

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.

Carrier, G. F. and Greenspan, H. P, 1958, "Water waves of finite amplitude on a sloping beach." *J. Fluid Mech.*, 4, 97-109.

Chawla, A., 1995 "Wave transformation over a submerged shoal", MS thesis, Department of Civil Engineering, University of Delaware.

Chawla, A. , /"Ozkan-Haller, H. T., and Kirby, J. T., 1996, "Experimental study of breaking waves over a shoal". *Proc. Int. Conf. Coast. Engrg.*, Orlando, Florida, 2-15.

Chen, Q., Madsen, P. A., Schäffer, H. A., and Basco, D. R, 1998, "Wave-current interaction based on an enhanced Boussinesq approach." *Coast. Engrg.*, 33, 11-40.

Chen Q., Dalrymple, R. A., Kirby, K. T., Kennedy, A. B., and Haller, M. C, 1999a, "Boussinesq modeling of a rip current system." *J. Geophys. Res.*, in press.

Chen, Q., Kirby, J. T., and Dalrymple, R. A. Kennedy, B. A., Chawla, A. 1999b. "Boussinesq modeling of wave transformation, breaking, and runup. II: 2D". *J. Wtrwy, Port, Coast. and Oc. Engrg.*, in press.

Elgar, S. and Guza, R. T., 1985, "Shoaling gravity waves: comparisons between field observations, linear theory and a nonlinear model", *J. Fluid Mech.*, **158**, 47-70.

Goring, D. G., 1978, "Tsunamis-the propagation of long waves onto a shelf". Ph.D dissertation, California Institute of Technology, Pasadena, California.

Israeli, M. and Orszag, S. A., 1981, "Approximation of radiation boundary conditions", *J. Computational Physics*, **41**, 115-135.

Kennedy, A. B., Chen, Q., Kirby, J. T., and Dalrymple, R. A., 1999, "Boussinesq modelling of wave transformation, breaking, and runup. I: 1D." *J. Wtrwy, Port, Coast. and Oc. Engrg.*, in press.

Kirby, J. T., 1997, "Nonlinear, dispersive long waves in water of variable depth", Chapter 3 in *Gravity Waves in Water of Variable Depth*, J. N. Hunt (ed), Advances in Fluid Mechanics, 55-126. Computational Mechanics Publications.

Larsen, J. & Dancy, H., 1983, "Open boundaries in short wave simulations – a new approach", *Coast. Engrg*, **7**, 285-297.

Liu P. L-F., Yoon, S. B. & Kirby, J. T., 1985, "Nonlinear refraction- diffraction of waves in shallow water", *J. Fluid Mech.*, **153**, 185-201.

Liu, P. L.-F., Cho, Y.-S., Briggs, M. J., Kanoglu, U., and Synolakis, C. E., 1995, "Runup of solitary waves on a circular island." *J. Fluid Mech.*, 302, 259-285.

Madsen, P. A., Murray, R. & Sørensen, O.R., 1991, "A new form of Boussinesq equations with improved linear dispersion characteristics", *Coast Engrg*, **15**, 371-388.

Madsen, P. A., and Sørensen, O. R., 1992, "A new form of the Boussinesq equations with improved linear dispersion characteristics. Part 2. A slowly varying bathymetry", *Coast. Engrg.*, **18**, 183-204.

Madsen, P. A. & Warren, I. R., 1984, "Performance of a numerical short-wave model", *Coast. Engrg*, **8**, 73-93.

Madsen, P. A., Sørensen, O. R. and Schäffer, H. A., 1997b, "Surf zone dynamics simulated by a Boussinesq-type model. Part II. Surf beat and swash oscillations for wave groups and irregular waves", *Coast. Engrg.*, **32**, 289-319.

Mase, H., 1995, "Frequency down drift of swash oscillation compared to incident waves." *J. Hydr. Res.*, 33 (3), 397-411.

Nwogu, O., 1993, "An alternative form of the Boussinesq equations for nearshore wave propagation", *J. Waterway, Port, Coast. Ocean Engrg*, **119**(6), 618-638.

Peregrine, D. H., 1967, "Long waves on a beach", *J. Fluid Mech.*, **27**, 815-82.

Rygg, O. B., 1988, "Nonlinear refraction-diffraction of surface waves in intermediate and shallow water", *Coast. Engrg*, **12**, 191–211.

Schäffer, H. A., Madsen, P. A. & Deigaard, R., 1993, "A Boussinesq model for waves breaking in shallow water", *Coast. Engrg*, **20**, 185-202.

Shapiro, R. , 1970, "Smoothing, filtering, and boundary effects", *Review of geophysics and space physics*, Vol. **8**, No. 2, 359-386.

Smagorinsky, J.S., 1963, "General circulation experiments with the primitive equations." *Mon. Weather Rev.*, 93, 99-.

Svendsen, I. A., Yu, K., and Veeramony, J., 1996, "A Boussinesq breaking wave model with vorticity", *Proc. Int. Conf. Coast. Engrg.*, Orlando, Florida, 1192-1204.

Tao, J., 1983, "Computation of wave run-up and wave breaking", *Internal report, Danish Hydraulic Institute*, 40 pp.

Tao, J., 1984, "Numerical modelling of wave runup and breaking on the beach." *Acta Oceanologica Sinica*, 6(5), 692-700, in Chinese.

Titov, V. V., Synolakis, C. E., 1998, "Numerical modeling of tidal wave runup". *J. Wtrwy, Port, Coast. and Oc. Engrg.*, 124 (4), 157-171.

Wei, G. & Kirby, J. T., 1995, "A time-dependent numerical code for the extended Boussinesq equations", *J. Waterway, Port, Coastal and Ocean Engineering*, Vol. 121, No. 5, 251-261.

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

Wei, G. & Kirby, J. T., and Sinha, A. 1998, "Generation of waves in Boussinesq models using a source function method", *Coast. Engrg*, Submitted.

Witting, J. M., 1984, "A unified model for the evolution of nonlinear water waves", *J. Comp. Phys.*, **56**, 203–236.

Zelt, J. A., 1991, "The run-up of nonbreaking and breaking solitary waves", *Coast. Engrg*, **15**, 205-246.

Appendix A: FUNWAVE 1.0: One-Dimensional Version.

The package of *funwave1d* model contains the following files:

```
   funwave1d.f - main program;
       depth.f - program to generate bathymetry;
       initw.f - program to generate initial conditions;
    1dsource.f - program to generate data for incident waves;
       param.h - parameters for the arrays;

funwave1d.data - specification file for funwave1d.f;
 1dsource.data - specification file for 1dsource.f;

  plotstuff1.f - for online animation;
  plotstuff2.f - no animation;
        plot.c - animation;
    plotinit.c - animation;
      plotfn.c - animation;

 funwave1d.web - source code with documentation for funwave1d.f;
  1dsource.web - source code with documentation for 1dsource.f;
      Makefile - To generate *.f files from *.web files,
                 to compile the funwave1d.f and 1dsource.f, and
                 to generate *.tex files in latex format.
    Makefileng - A counterpart of Makefile for machines without GL.
```

Appendix B: FUNWAVE 1.0: Two-Dimensional Version.

The package of *funwave2d* model contains the following files:

```
   funwave2d.f - main program;
       depth.f - program to generate bathymetry;
       initw.f - program to generate initial conditions;

       b2dp1.f - program to extract free-surface data from f7n;
       b2dp2.f - program to compute zero-crossing wave height;
       b2dp3.f - program to extract mean-current data from f7n;

funwave2d.data - specification file for funwave1d.f;
 funwave2d.web - source code with documentation for funwave1d.f;
      Makefile - To generate *.f file from *.web file,
                 to compile the *.f files, and
                 to generate *.tex file in latex format.
```