BOUSSINESQ-EQUATION AND RANS HYBRID WAVE MODEL

A Dissertation

by

KHAIRIL IRFAN SITANGGANG

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May 2008

Major Subject: Ocean Engineering

BOUSSINESQ-EQUATION AND RANS HYBRID WAVE MODEL

A Dissertation

by

KHAIRIL IRFAN SITANGGANG

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Approved by:

Chair of Committee,	Patrick J. Lynett
Committee Members,	Billy L. Edge
	Robert D. Hetland
	Vivek Sarin
Head of Department,	David Rosowsky

May 2008

Major Subject: Ocean Engineering

ABSTRACT

Boussinesq-Equation and RANS

Hybrid Wave Model. (May 2008)

Khairil Irfan Sitanggang, B.S., Institut Teknologi Bandung;

M.S., Institut Teknologi Bandung;

M.S., Texas A&M University

Chair of Advisory Committee: Dr. Patrick J. Lynett

This dissertation presents the development of a novel hybrid wave model, comprised of the irrotational, 1-D horizontal Boussinesq and 2-D vertical turbulence-closed Reynolds Averaged Navier-Stokes (RANS) wave models. The two constituents are two-way coupled with the interface placed at a location where turbulence is relatively small. Boundary conditions on the interfacing side of each model is provided by its counterpart model through data exchange, requiring certain transformation due to the difference in physical variables employed in both models. The model is intended for large-scale wave simulation, accurate in both the nonbreaking and breaking zones with relatively coarser grid in the former and finer in latter, and yet efficient.

Hybrid model tests against idealized solitary and standing wave motions and wave-overtopping on structure exhibit satisfactory to very good agreement. Compared with pure RANS simulations, the hybrid model saves computational time by a factor proportional to the reduction in the size of the RANS domain. Also, a large-scale tsunami simulation is provided for a numerical setup that is practically unapproachable using RANS alone; not only does the hybrid model offer more rapid simulation of relatively small-scale problems, it provides an opportunity to examine very large total domains with the fine resolution typical of RANS simulations.

To allow for implementation on even larger domain with affordable CPU time,

the hybrid model is parallelized to run on distributed memory machine. This is done by parallelizing the RANS model while leaving the Boussinesq model serial. One of the processors is responsible for both the sub-RANS and Boussinesq calculations. ICCG(0) for solving the pressure equation is parallelized using the nonoverlapping-decomposition technique, requiring more iterations than the serial one. Standing wave and hypothetical tsunami simulations with 960×66 and 1000×100 grids, and using 8 processors confirm model validity and computational efficiency of 82% and 65%.

Finally, the 2-D Boussinesq model is parallelized using domain decomposition technique. The solution to the tridiagonal system arising in the model is calculated as the sum of the homogeneous and particular solutions. Parallel model tests using up to 32 processors exhibit model accuracy and efficiency of 80% for simulation with 500×500–2000×2000 grids.

ACKNOWLEDGMENTS

In the Name of Allah, the Most Beneficent, the Most Merciful. Many people have contributed to this research in various ways. First, I would like to express my sincere gratitude to my advisor, Dr. Patrick J. Lynett, whose constant support, advice, intellect, and enthusiasm helped me a lot through the difficult years in doing this research. I also thank my committee members, Dr. Billy L. Edge, Dr. James Kaihatu, Dr. Robert Hetland, and Dr. Vivek Sarin, particularly for his feedback on the parallel and iterative method. I appreciate Dr. Philip L.-F. Liu of Cornell University for providing the COBRAS code and Dr. Tom Hsu of the University of Florida for his help with the problem I had with COBRAS. I am grateful to the people in the TAMU Super Computing for their help in using Cosmos and Hydra and to Thomas Mather in the Zachry Civil Engineering Computer Lab.

I have been joyful through the years working in the Hydromechanics Laboratory with nice and friendly people around me: Adeniyi Ade-onojobi, Ali Nakhaee, Baran Aydin, Chandan Lakothia, Chen Xie, Dae Hong Kim, Dong Guan Seol, Duncan Bryant, Fares Aljeeran, Gregery Belsmeier, Ho Joon Lim, Hoda El Safty, John Christopher Henriksen , Kusalika Ariyarathne, Lauren Augustin, Mir Emad Mousavi, Oscar Cruz Castro, Seung Jae Lee, Youn Kyun Song, Zhi Zhang, and others I could not mention one by one here.

Finally, I would like to thank my wife, Dine Rachmawati, for her love, support, and patience, my parents and family, and my parents-in-law for their continuous support and encouragement.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

FIGURE Page

FIGURE Page

FIGURE Page

FIGURE Page

CHAPTER I
INTRODUCTION

Problem Statement

Most of the currently available numerical ocean wave models were developed based on a single set of governing equations applied on the entire computational domain. While model tests have shown agreement with various target scenarios, the applicability of such models for more general purpose applications such as the simulation of offshore-to-shoreline wave propagation, which includes both the nonbreaking and breaking wave processes, is still limited due to the physical assumptions in the models. For instance, depth integrated equations using potential flow assumptions are one of the more commonly employed equations/assumptions in existing wave models. The implementation of such models on a certain domain is valid only if the flow regime is far from the high intensity turbulence area such as the nearshore breaking zone.

Although the application of such models with the help of the artificial turbulence, Kennedy et al. (2000) and Chen et al. (2000), may be pushed further nearshore to include the wave-breaking processes, only the mean flow is calculable with this type of model, leaving the more detail, small scale flow unpredictable.

Another set of the governing equations which nowadays are widely used for developing wave models is the turbulence-closed Reynolds Averaged Navier-Stokes (RANS) equations. The models which were developed based on these equations are well suited for breaking wave simulation, Lin and Liu (1998). The implementation of this model, however, is usually confined to the nearshore zone where a relatively large number of grids and fine mesh are needed to accurately capture the turbulence. This incurs

---

The journal model is *Journal of Waterway, Port, Coastal, and Ocean Engineering*.

expensive computational effort. Hence, extending the implementation of the model to larger domain is often not practically feasible.

In summary, neither one of the two types of wave models is suitable for employment in the simulation of large scale wave propagation in the ocean which includes both the nonbreaking and breaking processes. A wave model should meet two important criteria for such a simulation: first, both the nonbreaking and breaking processes must be adequately simulated, and second, the simulation must be efficiently done. The aforementioned models fulfill only one of the two criteria.

For such large scale simulation, either a new model which has both capabilities should be developed or a method should be developed to couple the two types of wave models so that they can be applied simultaneously for the large scale nonbreaking/breaking simulation. The last option seems more reasonable since both constituents are already available. The big challenge for coupling the models is that both models are different in physics: i.e. nonviscous and viscous, and nonturbulence and turbulence, and additionally, in numerical solution. Coupling two such different models may not always work since, due to the said differences, the variables from the two models may, in some cases, disagree on the interface of the two models, leading to numerical instability.

Objective

The main objective of this research is to develop a hybrid model which couples two types of wave models which consist of nonviscous, potential Boussinesq-equation type and viscous, turbulence-closed RANS-equation based wave models. In this study, the scope is limited to coupling the 1-D Boussinesq and 2-D RANS wave models. To allow for the application on a wider range of wave nonlinearity, we use the fully nonlinear

Boussinesq-equation model, Wei et al. (1995) and Lynett and Liu (2002). For the second model, we use the turbulence-closed RANS-based Cornell Breaking Wave and Structure (COBRAS) model, Lin and Liu (1998).

The two models are two-way coupled, and so act as if they are a single model working on a continuous domain. In the coupling implementation, the Boussinesq is applied on the nonbreaking zone and the RANS on the breaking/high-turbulence zone. The two models share a common domain interface for exchanging data, used as boundary conditions in the models. By coupling the two models, accurate large scale wave simulation using a coarse grid and "simple" physics in the deep-to-intermediate water and fine grid and detailed physics in the nearshore area is computationally feasible. In summary, the coupled hybrid model bridges the two widely used wave models.

Large scale simulation using huge number of computational grids on both the Boussinesq and RANS models requires extensive CPU time. In certain simulation with a huge number of grids involved, the computation cannot be done on a single machine with limited resources. For extensive simulation, people usually have recourse to distributed computation, i.e. the problem is divided into several smaller subproblems and run in parallel in several computers with certain communication happening among the computers. In so doing, not only can the originally huge problem now be calculated, the computation itself becomes faster. The secondary objective of this research is to parallelize 2-D RANS/hybrid and the 2-D Boussinesq models. The former supports the main objective in that it increases the speed of the hybrid computation. The latter is independent from the main objective of this research. This is particularly useful for large scale 2-D Boussinesq simulation and future 2-D Boussinesq and 3-D RANS coupling, where both constituent models should ideally be parallelized. Since the computational time of the Boussinesq model is in general not

significant, i.e. 2 or 3 orders in magnitude less than the RANS model, in parallelizing the hybrid wave model, it is sufficient to parallelize the RANS model only and leave the Boussinesq model serial.

Overview of the Boussinesq, RANS, and Hybrid Model Development

The Boussinesq-type equation wave model has been successfully used in the past decade for simulating wave propagation in the ocean of relatively great depth to breaking zone. The pioneering work on the Boussinesq wave model is that of Peregrine (1967) who introduced the weakly nonlinear Boussinesq equation model over variable bathymetry; its implementation was limited to the shallow water regime, $kh \approx 0.3$. Enhancement of the model which pushed its accuracy further off-shore and improved the dispersive properties followed thereafter. Madsen and Sørensen (1992), for instance, through manipulation of the dispersive term of the depth-averaged Boussinesq equation, managed to push the model accuracy to intermediate depth. Nwogu (1993) used the velocity at arbitrary depth, instead of the depth-averaged velocity, in his derivation to get an excellent linear dispersive properties and good accuracy up to $kh \approx 3$.

In deriving the weakly nonlinear Boussinesq equation, higher order nonlinear terms had been truncated. Hence its implementation to simulate highly nonlinear wave, i.e. ratio of wave height to depth is large, is not accurate. To better capture the high nonlinearity properties in the model, Wei et al. (1995) retained higher order derivative terms and derived a more accurate highly nonlinear Boussinesq equation model with excellent dispersive properties up to $kh \approx 3$. Similar attempts were made by a number of researchers, e.g. Gobbi et al. (2000), Agnon et al. (1999), and Lynett and Liu (2004b). As a result, the current version of the Boussinesq model can be

implemented to an extremely deep water with a very good accuracy.

Attempts were made to extend the model applicability to the nearshore zone. Kennedy et al. (2001) and Chen et al. (2000), using the artificial turbulence, were able to calculate the breaking wave with very good agreement with the laboratory data. Chen et al. (2003) introduced the vertical vorticity into the irrotational Boussinesq equations to model the surface waves and longshore currents under the laboratory and field conditions. The simulation agreed very well with measurements. Veeramony and Svendsen (2000) described the breaking wave by accounting for the effect of vorticity generated by the breaking process. The vorticity was obtained by solving the vorticity transport, which is based on the Reynold equations. Model test showed that calculated wave height, set-up, and velocity profiles were in good agreement with laboratory data. Although the results look promising, such depth-integrated models with artificial turbulence are only accurate for mean flow calculation. With this type of model, small scale velocity prediction might not be accurate or even not doable. To achieve this purpose, people usually have recourse to more sophisticated turbulence-closed RANS-equation based model.

The RANS wave model with the VOF (volume of fluid) technique for tracking the fluid particle has been widely used to simulate breaking wave in the nearshore zone and wave breaking due to wave interaction with structure. Lin and Liu (1998), for instance, employing the nonlinear eddy viscosity model, performed a numerical study of breaking wave in the surf zone. Comparison with the experimental data showed that the numerical calculation of the breaking wave was in good agreement with the data. The model was also tested for nonbreaking wave simulation. Perfect match with the analytic solution was found. Employing Lin and Liu's model, Lara et al. (2006) studied the wave interaction with breakwater. This study showed that the model could correctly calculate the wave reflection, shoaling, and breaking near the structure

and hence could be used as a tool for design purposes. Using the same type of model, Liu and Cheng (2001) investigated the behavior of a solitary wave propagating over a step. Both the nonbreaking and breaking cases were considered. Here the authors also found that the model was able to compute the turbulence generated by the breaking and flow separation at the corner of the step. The application of the model extended to the study of waves generated by nearshore submarine mass-movement, Yuk et al. (2006). The numerical simulation result was compared with laboratory experiment and analytic solution. Very good agreement was obtained for the submarine and aerial mass movements. Other researchers, Zhao et al. (2004), also found success in studying the breaking wave using RANS model with a multi-scale turbulence model.

Typical simulation of wave breaking using RANS model is usually computationally expensive. This is due in part to the iterative solver employed in the model to solve the pressure equation. This expensive computation hinders users from performing large scale simulation. Hence, the implementation of the RANS model is limited to a small domain size with relatively small number of computational grids.

As to the coupling of wave models, we found very few references on this topic. Here, we presented two such references. First, Fujima et al. (2002) developed the 2-D/3-D hybrid model intended for numerical tsunami simulation around structures. The constituent models were the 2-D depth-averaged model and 3-D turbulence-closed model. The two models were two-way coupled. In its implementation, the 3-D domain should be sufficiently large so that the velocity distribution is vertically uniform. In cases where vertical velocity distribution is not uniform, the zero velocity gradient is imposed on the interface, which has no rational justification in the physics. Comparison with the laboratory experiment showed that the hybrid model was able to reproduce the 3-D characteristics of flow around structure, and comparison with similar simulation using full 3-D model proved that the hybrid model could reduce the

computational time significantly. Second, a series of papers, Lachaume et al. (2003), Grilli et al. (2004), and Biausser et al. (2004), discussed the coupling of the fully nonlinear potential flow model based on a Boundary Element Method (BEM) and VOF/N-S (Navier Stokes) model. Two methods of coupling were proposed: weak and strong coupling. In the first method, the solution of the BEM model is used as an initialization of the VOF/N-S solver, with no feedback from the second to the first. In the second method, both models are exchanging information. Up to the publication date of these papers, only the first method was well developed, leaving the second for future development. Although the constituents of the two hybrid models are different from what we employ in this research, the ideas are still useful for the hybrid development herein.

Organization of the Dissertation

In Chapter II, the governing equation and numerical solution techniques of the Boussinesq and RANS models are explained. In the first model, the higher order Adams-Bashforth predictor and corrector method is used to solve the Boussinesq governing equation, and in the second model, the two-step projection method is used to solve the turbulence-closed RANS equation.

In Chapter III, the coupling method of the two models is explained. The conversion of the variables from one model to the other to be used as boundary conditions is explained in detail. Also described is the numerical filter to remove two-grid wave component occurring on the interface due to the velocity discrepancy.

In Chapter IV, a series of model tests are presented. These include the solitary wave simulation, standing wave simulation, flux and free surface of wave overtopping on hard, impermeable structure, wave propagation over porous structure, and

hypothetical tsunami wave generation and propagation.

In Chapter V, the parallelization of the hybrid model is described. In parallelizing this model, only the RANS model is parallelized. The Boussinesq model is still in its sequential form and running in one of the processors involved in the calculation. This is a preliminary work which still needs various model tests and optimization.

In Chapter VI, the domain decomposition method to parallelize the 2-D Boussinesq model is described. The domain is decomposed into several equal-area subdomains and distributed into several computers to run in parallel. The tridiagonal system of linear equations is parallelized by decomposing it into the homogeneous and particular solutions, resembling the solution of the inhomogeneous differential equation.

In Chapter VII, the works in this dissertation are summarized. The summary consists of the three parts: the hybrid model, the parallel hybrid model, and the parallel 2-D Boussinesq model. Also included are suggestions for future study and development.

CHAPTER II

GOVERNING EQUATION AND NUMERICAL SOLUTION

Boussinesq Wave Model

Different approaches have been used to derive the Boussinesq equations. Nwogu (1993) and Wei et al. (1995), for instance, employed the perturbation method in their derivation and chose the velocity at a reference level ($z_\alpha = -0.531h$) instead of the depth-averaged velocity for the velocity variable. Lynett and Liu (2002) also followed the same procedure to derive the Boussinesq equation with the inclusion of the time dependent bathymetric-change to study the submarine-landslide-generated wave. Of significant relevance to this study is the fact that the Boussinesq model is fundamentally inviscid and does not permit any rotationality. While these restrictions do permit the creation of an extremely efficient wave propagation model, they govern the limits of application of the model and thus will play an important role in the development of the coupled system.

Following Lynett and Liu (2002), the Boussinesq equations consist of the depth-integrated continuity and momentum equations:

$$\frac{\partial H}{\partial t} + \frac{\partial \left(H u_\alpha\right)}{\partial x} - \frac{\partial}{\partial x}\left\{H\left[\left(\frac{1}{6}\left(\eta^2 - \eta h + h^2\right) - \frac{1}{2}z_\alpha^2\right)\frac{\partial^2 S}{\partial x^2} + \left(\frac{1}{2}\left(\eta - h\right) - z_\alpha\frac{\partial T}{\partial x}\right)\right]\right\} = 0 \tag{2.1}$$

$$\frac{\partial u_\alpha}{\partial t} + \frac{1}{2}\frac{\partial^2 u_\alpha}{\partial x^2} + g\frac{\partial \eta}{\partial x} + \frac{\partial}{\partial t}\left\{\frac{1}{2}z_\alpha^2\frac{\partial S}{\partial x} + z_\alpha\frac{\partial T}{\partial x} - \frac{\partial}{\partial x}\left(\frac{1}{2}\eta^2 S + \eta T\right)\right\} + \frac{\partial}{\partial x}\left\{\frac{\partial \eta}{\partial t}\left(T + \eta S\right) + \left(z_\alpha - \eta\right)\left(u_\alpha\frac{\partial T}{\partial x}\right) + \frac{1}{2}\left(z_\alpha^2 - \eta^2\right)\left(u_\alpha\frac{\partial S}{\partial x}\right) + \frac{1}{2}\left(T + \eta S\right)^2\right\} = 0, \tag{2.2}$$

where x, t, g, η, z_α, u_α, and h are the horizontal axis, time, gravity, free surface elevation, reference level, velocity at the reference level, and depth, respectively. The other parameters are defined as $H = h+\eta$, $S = \partial u_\alpha/\partial x$, and $T = \partial\left(hu_\alpha\right)/\partial x+\partial h/\partial t$.

Equations (2.1) and (2.2) are solved using the higher-order Adams-Bashforth predictor-corrector finite difference method. This scheme consists of the explicit predictor:

$$\eta_i^{n+1} = \eta_i^n + \tfrac{1}{12}\Delta t\left(23E_i^n - 16E_i^{n-1} + 5E_i^{n-2}\right) \tag{2.3}$$

$$U_i^{n+1} = U_i^n + \tfrac{1}{12}\Delta t\left(23F_i^n - 16F_i^{n-1} + 5F_i^{n-2}\right) \tag{2.4}$$

and implicit corrector:

$$\eta_i^{n+1} = \eta_i^n + \tfrac{1}{24}\Delta t\left(9E_i^{n+1} + 19E_i^n - 5E_i^{n-1} + E_i^{n-2}\right) \tag{2.5}$$

$$U_i^{n+1} = U_i^n + \tfrac{1}{24}\Delta t\left(9F_i^{n+1} + 19F_i^n - 5F_i^{n-1} + F_i^{n-2}\right). \tag{2.6}$$

The terms E, F, and U are:

$$\begin{aligned} E = &-h_t - \left[\left(\eta+h\right)u_\alpha\right]_x + \\ &\left\{\left(\eta+h\right)\left[\left(\tfrac{1}{6}\left(\eta^2-\eta h+h^2\right)-\tfrac{1}{2}z_\alpha^2\right)S_x + \left(\tfrac{1}{2}\left(\eta-h\right)-z_\alpha\right)T_x\right]\right\}_x, \end{aligned} \tag{2.7}$$

$$\begin{aligned} F = &-\tfrac{1}{2}\left(u_\alpha^2\right)_x - g\eta_x - z_\alpha h_{xtt} - z_{\alpha t}h_{xt} + \left(\eta h_{tt}\right)_x - \left[E\left(\eta S+T\right)\right]_x - \\ &\left[\tfrac{1}{2}\left(z_\alpha^2-\eta^2\right)u_\alpha S_x\right]_x - \left[\left(z_\alpha-\eta\right)u_\alpha T_x\right]_x - \tfrac{1}{2}\left[\left(T+\eta S\right)^2\right]_x, \end{aligned} \tag{2.8}$$

$$U = u_\alpha + \tfrac{1}{2}\left(z_\alpha^2-\eta^2\right)u_{\alpha,xx} + \left(z_\alpha-\eta\right)\left(hu_\alpha\right)_{xx} - \eta_x\left[\eta u_{\alpha,x} + \left(hu_\alpha\right)_x\right]. \tag{2.9}$$

The solution procedure starts with the calculation of η_i^{n+1} and U_i^{n+1} in (2.3) and (2.4) and then (2.9) is solved for $u_{\alpha i}^{n+1}$. The explicitly calculated free surface elevation and velocity are not highly accurate, in general, due to the explicit scheme used in the predictor. To improve the solutions, the implicit corrector (2.5) and (2.6) are employed. Since both sides of the nonlinear equations contain the variables at the

current time level, $(n+1)$, the computations are iterated until the variables converge with the convergence criteria:

$$\left| \frac{w^{n+1} - w_*^{n+1}}{w^{n+1}} \right| < 10^{-4}, \tag{2.10}$$

where w represents u and η and w_* is the values from the previous iteration.

Figure 1. Finite difference stencil of the Boussinesq model.

The finite difference stencil of the scheme is depicted in Figure 1. To perform the computation at grid i, five neighboring variables from the left and right sides of the grid are required. The large stencil is required due to the high-order derivatives in the model equations and the associated necessity of a high-order numerical solution. Note that in this finite different solution, the nonstaggered grid, i.e. both the free surface elevation and the velocity are evaluated at the same grids, is employed.

RANS Wave Model

The derivation of the RANS equations starts with splitting the variables into the time-mean and turbulent fluctuation components. Introducing the split-variables into the N-S equations and taking the necessary algebraic manipulations result in the following equations:

$$\frac{\partial \langle u_i \rangle}{\partial x_i} = 0 \tag{2.11}$$

$$\frac{\partial \langle u_i \rangle}{\partial t} + \langle u_j \rangle \frac{\partial \langle u_i \rangle}{\partial x_j} = -\frac{1}{\langle \rho \rangle} \frac{\partial \langle p \rangle}{\partial x_i} + g_i + \frac{1}{\langle \rho \rangle} \frac{\partial \langle \tau_{ij} \rangle}{\partial x_j} - \frac{\partial \langle u_i' u_j' \rangle}{\partial x_j}, \tag{2.12}$$

where u_i and g_i are the velocity and gravity in the x_i direction, p is the pressure, ρ is the density, and τ_{ij} is the stress tensor in the ij direction. The bracket $\langle\,\rangle$ is the mean value sign. To close these equations, Lin and Liu (1998) employed the nonlinear eddy viscosity model which expresses the correlation of the velocity fluctuations, the last term of (2.12), in terms of a nonlinear function of the mean variables:

$$\left\langle u_i' u_j' \right\rangle = \frac{2}{3}k\delta_{ij} - C_d\frac{k^2}{\epsilon}\left(\frac{\partial\langle u_i\rangle}{\partial x_j} + \frac{\partial\langle u_j\rangle}{\partial x_i}\right) - \frac{k^3}{\epsilon^2}\left\{C_1\left(\frac{\partial\langle u_i\rangle}{\partial x_l}\frac{\partial\langle u_l\rangle}{\partial x_j} + \frac{\partial\langle u_j\rangle}{\partial x_l}\frac{\partial\langle u_l\rangle}{\partial x_i} - \frac{2}{3}\frac{\partial\langle u_l\rangle}{\partial x_k}\frac{\partial\langle u_k\rangle}{\partial x_l}\delta_{ij}\right) + C_2\left(\frac{\partial\langle u_i\rangle}{\partial x_k}\frac{\partial\langle u_j\rangle}{\partial x_k} - \frac{1}{3}\frac{\partial\langle u_l\rangle}{\partial x_k}\frac{\partial\langle u_l\rangle}{\partial x_k}\delta_{ij}\right) + C_3\left(\frac{\partial\langle u_k\rangle}{\partial x_i}\frac{\partial\langle u_k\rangle}{\partial x_j} - \frac{1}{3}\frac{\partial\langle u_l\rangle}{\partial x_k}\frac{\partial\langle u_l\rangle}{\partial x_k}\delta_{ij}\right)\right\}, \quad (2.13)$$

where the turbulence kinetic energy k and the turbulence energy dissipation rate ϵ are obtained by solving the k–ϵ equation:

$$\frac{\partial k}{\partial t} + \langle u_j\rangle\frac{\partial k}{\partial x_j} = \frac{\partial}{\partial x_j}\left\{\left(\frac{\upsilon_t}{\sigma_k} + \nu\right)\frac{\partial k}{\partial x_j}\right\} - \left\langle u_i' u_j'\right\rangle\frac{\partial\langle u_i\rangle}{\partial x_j} - \epsilon \quad (2.14)$$

$$\frac{\partial\epsilon}{\partial t} + \langle u_j\rangle\frac{\partial\epsilon}{\partial x_j} = \frac{\partial}{\partial x_j}\left\{\left(\frac{\upsilon_t}{\sigma_k} + \nu\right)\frac{\partial\epsilon}{\partial x_j}\right\} + C_{1\epsilon}\frac{\epsilon}{k}\nu_t\left(\frac{\partial\langle u_i\rangle}{\partial x_j} + \frac{\partial\langle u_j\rangle}{\partial x_i}\right)\frac{\partial\langle u_i\rangle}{\partial x_j} - C_{2\epsilon}\frac{\epsilon^2}{k}. \quad (2.15)$$

The coefficients in (2.13) are $C_d = \frac{1}{3}\left(\frac{1}{3.7+S_{\max}}\right)$, $C_1 = \frac{1}{185.2+\gamma D_{\max}^2}$, $C_2 = -\frac{1}{58.5+\gamma D_{\max}^2}$, and $C_3 = \frac{1}{370.4+\gamma D_{\max}^2}$. In (2.14) and (2.15), $\nu_t = C_d k^2/\epsilon$, $C_{1\epsilon} = 1.44$, $C_{2\epsilon} = 1.92$, $\sigma_k = 1.3$, and $\gamma = 3.0$.

To model the flow in the porous structure, Lin and Liu (1998) employed the spatially-averaged N-S equations:

$$\frac{\partial\overline{U_i}}{\partial x_i} = 0 \quad (2.16)$$

$$\frac{1+c_A}{n}\frac{\partial\overline{U_i}}{\partial t} + \frac{1}{n^2}\overline{U_j}\frac{\partial\overline{U_i}}{\partial x_j} = -\frac{1}{\rho}\frac{\partial\overline{p_0}}{\partial x_i} + \frac{\nu}{n}\frac{\partial^2\overline{U_i}}{\partial x_j\partial x_j} - \frac{1}{n^2}\frac{\overline{\partial U_i'' U_j''}}{\partial x_j}. \quad (2.17)$$

The spatial correlation in (2.17) is modeled as a combination of linear and nonlinear friction:

$$-\frac{1}{n^2}\overline{\frac{\partial U_i'' U_j''}{\partial x_j}} = -ga\overline{U_i} - gb\overline{VU_i}, \tag{2.18}$$

where $\overline{U_i}$ is the i-th component of the mean velocity and $\overline{V}$ is the magnitude of the local velocity. The coefficients in (2.17) and (2.18) are

$$c_A = \gamma_p \frac{1-n}{n} \tag{2.19}$$

$$a = \alpha \frac{(1-n)^2}{n^3} \frac{\nu}{gD_{50}^2} \tag{2.20}$$

$$b = \beta \left(1 + \frac{7.5}{KC}\right) \frac{1-n}{n^3} \frac{1}{gD_{50}} \tag{2.21}$$

where $\gamma_p = 0.34$, $KC = \frac{\overline{V}T}{nD_{50}}$, and α and β are the calibration coefficients, Garcia et al. (2004).

The two-step projection method, Kothe et al. (1994), is used to solve (2.11) and (2.12). In this procedure, the explicit finite difference solution to (2.12):

$$\frac{u_i^{n+1} - u_i^n}{\delta t} + u_j^n \frac{\partial u_i^n}{\delta x_j} = -\frac{1}{\rho^n}\frac{\partial p^{n+1}}{\partial x_i} + g_i + \frac{\partial \tau_{ij}^n}{\partial x_j} \tag{2.22}$$

is split into step 1:

$$\frac{\tilde{u}_i^{n+1} - u_i^n}{\delta t} = -u_j^n \frac{\partial u_i^n}{\partial x_j} + g_i + \frac{\partial \tau_{ij}^n}{\partial x_j} \tag{2.23}$$

and step 2:

$$\frac{u_i^{n+1} - \tilde{u}_i^{n+1}}{\delta t} = -\frac{1}{\rho^n}\frac{\partial p^{n+1}}{\partial x_i}. \tag{2.24}$$

Here the $\langle\,\rangle$ sign is dropped for convenience. In (2.23), the provisional velocity $\tilde{u}_i^{n+1}$ is first calculated without taking the pressure p^{n+1} into account. The provisional velocity is then used in (2.24) to determine the true velocity u_i^{n+1} which satisfies both (2.11) and (2.12). The pressure p^{n+1} in (2.24) is obtained from the solution of the

pressure Poisson equation (PPE):

$$\frac{\partial}{\partial x_i}\left(\frac{1}{\rho^n}\frac{\partial p^{n+1}}{\partial x_i}\right) = \frac{1}{\delta t}\frac{\partial \tilde{u}_i^{n+1}}{\partial x_i}, \tag{2.25}$$

which is a combination of (2.11) and (2.24). The spatial derivatives appearing in the equations are evaluated using second order centered finite difference. After gettting the correct velocities, the k–ϵ equations are solved. The volume of fluid, F, is then advected using the equation:

$$\frac{\partial F}{\partial t} + u_i\frac{\partial F_i}{\partial x_i} = 0. \tag{2.26}$$

In the RANS mesh, both the pressure and the volume of fluid are located at the center of the mesh and the horizontal and vertical velocities are located on the right and top faces of the mesh. Hence, unlike the Boussinesq grid system, in the RANS model the staggered grid system is employed.

Note that in solving (2.25) for p^{n+1}, we encounter a positive definite penta-diagonal system of linear equations. This system is solved using the iterative conjugate gradient solver, which is usually the heaviest computational load in all of the solution procedure. An additional note of significance regarding the PPE solver is that when increasing the matrix size (grid dimensions), the number of iterations required to reach convergence also increases. For example, while a 100 by 100 grid may take 10 iterations to converge, a larger domain using the same grid resolution, with a mesh of say 800 by 100 points, can take upwards of 50 iterations to converge.

CHAPTER III
BOUSSINESQ-RANS MODEL COUPLING

Coupling Method

Consider a typical problem of simulating the wave propagation from offshore to the shoreline as depicted in Figure 2. Depending on the wave condition along the domain, the offshore-shoreline domain is divided into two subdomains for the coupled-simulation purpose. The subdomain where the wave does not break is termed the prebreaking zone and the one where the wave breaks is termed the breaking/turbulence zone. In the first zone, we employ the Boussinesq-equation model and in the second zone we employ the RANS wave model.

To accommodate the data exchange between the two models, the computational grids of both models must overlap properly. Since the two models use different grid systems, not all the computational points in the RANS mesh are aligned with the Boussinesq grids on the interface. In our implementation, we align the Boussinesq velocity and free surface elevation grids with the RANS horizontal velocity grid. Hence, the RANS vertical velocity is centered between two consecutive Boussinesq grids (Figure 2). For physical consistency, the interface of the two models should be located seaward of the breaking point where the turbulence intensity is small.

In the Boussinesq model, the boundary conditions are imposed on the left and right sides of the domain. Of particular interest for the model coupling is the right boundary condition which imposes the free surface elevation and velocity on the Boussinesq ghost grids $N+1$, $N+2$, $N+3$, $N+4$, and $N+5$ on the interface area. These variables are obtained from the RANS grids 2, 3, 4, 5, and 6 which overlap with the aforementioned five Boussinesq ghost points. Here five points are involved

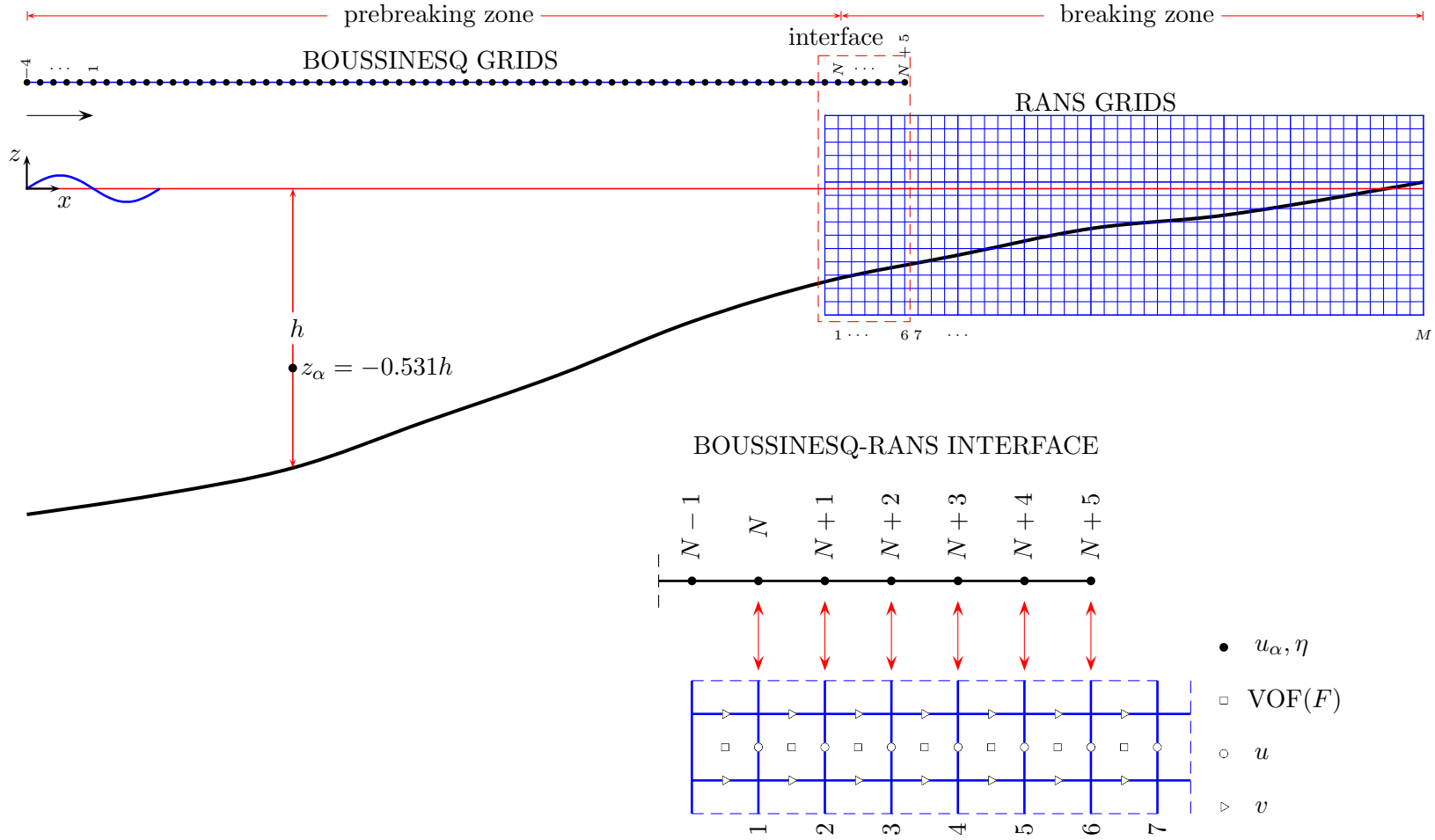


Figure 2. Boussinesq-RANS hybrid grid system.

because the calculation at a point in the Boussinesq model requires five neighboring points as shown in the stencil in Figure 1.

In the RANS model, the left boundary condition is associated with the coupling. This boundary condition imposes the velocity, volume of fluid, pressure, and the turbulence on the first column of the RANS mesh. The first two variables are obtained from the Boussinesq grid N. Pressure is determined from a single time level with the provisional velocity, and thus the pressure boundary condition is implicitly specified along with velocity. Turbulence intensity and dissipation are set to zero along the first column, consistent with the fact that the Boussinesq model is inviscid. This, of course, requires that the interface is located in an area of very low turbulence intensity.

Having explained the concept of coupling the two models and the corresponding interface area, we propose the following algorithm for coupling the Boussinesq-equation and RANS wave models. Let us assume that at the start of the algorithm we have all the dependent variables (i.e. u^n, v^n, etc.) at time level t^n.

1: **while** $t^n < t_{\text{end}}$ **do**

2: Calculate the RANS provisional velocities, $\tilde{u}^{n+1}$ and $\tilde{v}^{n+1}$, from (2.23), using Boussinesq boundary values from time level n.

3: Calculate, iteratively, the RANS fluid pressure, p^{n+1}, from (2.25).

4: Calculate the RANS final velocities, u^{n+1} and v^{n+1}, from (2.24).

5: Calculate the RANS turbulence intensity and dissipation, k^{n+1} and ϵ^{n+1}, using Boussinesq boundary values from time level n.

6: Calculate the RANS VOF function, F^{n+1}, using Boussinesq boundary values from time level n.

7: Calculate η^{n+1} and u_α^{n+1} from the Boussinesq predictor (2.3) and (2.4), using RANS boundary values from time level n.

8: Calculate, iteratively, η^{n+1} and u_α^{n+1} from the Boussinesq corrector (2.5) and (2.6), using RANS boundary values at time level n+1. At this point, all fluid variables in both models have completed calculations for time level n+1.

9: Change δt if flow in RANS exceeds Courant stability constraints; dynamic time-stepping.

10: **if** $\delta t_{\text{new}} \neq \delta t_{\text{old}}$ **then**

11: Interpolate the new η^{n-1}, η^{n-2}, u_α^{n-1}, and u_α^{n-2} at all nodes on the Boussinesq grid.

12: **end if**

13: $t^{n+1} = t^n + \delta t_{\text{new}}$

14: $n = n + 1$

15: **end while**

The algorithm consists of two main parts. The first part is the RANS model calculation in lines 2 through 6 and the second part is the Boussinesq model calculation in lines 7 and 8.

After completing its calculation, the RANS model requires the variables from time level $(n + 1)$ to update the boundary conditions and proceed to the next time level. For the left boundary condition, these variables are still not available from the Boussinesq model since the corresponding calculation has not started yet. However, since the RANS numerical solution scheme is explicit, the imposition of the left boundary condition can be delayed until the Boussinesq model calculation is finished.

Depending on the advection of the fluid in each grid of the RANS mesh, the time step, δt, may be adjusted to allow for stable and accurate calculation. In consequence, variables in the Boussinesq model should be calculated at the new time levels t_{new}^{n-2} and t_{new}^{n-1} which should be equally spaced in the Boussinesq model in order to use the high-order time integrators (2.3)–(2.6). Therefore, the new time levels in the

Boussinesq models, after the adjustment, are $t_{\text{new}}^{n-2} = t^n - 2\delta t_{\text{new}}$, $t_{\text{new}}^{n-1} = t^n - \delta t_{\text{new}}$, t^n, and $t_{\text{new}}^{n+1} = t^n + \delta t_{\text{new}}$. The new variable at t_{new}^{n-2} and t_{new}^{n-1} are determined by interpolation or extrapolation (whichever is appropriate) based on the values of the corresponding variable at $t = t^{n-2}$, t^{n-1}, and t^n. Here, the second order polynomial is employed to do the calculation:

$$\eta_{\text{new}} = \frac{(t_{\text{new}} - t^{n-1})(t_{\text{new}} - t^{n-2})}{(t^n - t^{n-1})(t^n - t^{n-2})}\eta^n + \frac{(t_{\text{new}} - t^{n-2})(t_{\text{new}} - t^{n-3})}{(t^{n-1} - t^{n-2})(t^{n-1} - t^{n-3})}\eta^{n-1} + \frac{(t_{\text{new}} - t^n)(t_{\text{new}} - t^{n-1})}{(t^{n-2} - t^n)(t^{n-2} - t^{n-1})}\eta^{n-2}. \quad (3.1)$$

For updating the boundary condition on the interface, all the necessary variables at the time level $(n+1)$ are available from the RANS model. The whole procedure from line 2 to 11 is repeated until the end of the simulation, $t = t_{\text{end}}$, is attained.

RANS Boundary Condition

As mentioned earlier, the vertical profiles of the velocities and volume of fluid on the first column of the RANS mesh are obtained from the Boussinesq model. These variables, however, are not immediately available from the Boussinesq. To get these variables from the Boussinesq model, certain transformation should be done on η and u_α. Figure 3 depicts the hybrid grids on the interface. Shown in the figure are the first column of the RANS mesh with 18 computational cells[1] and the grids $N-2$, $N-1$, N, and $N+1$ of the Boussinesq model (compare Figure 2). Note that grid N of the Boussinesq model is aligned with the right face of the first column in the RANS mesh.

To find the fluid distribution, F, in this column, we first determine the free

[1] In a typical simulation, the columns of the RANS mesh may have tens or hundreds of computational cells.

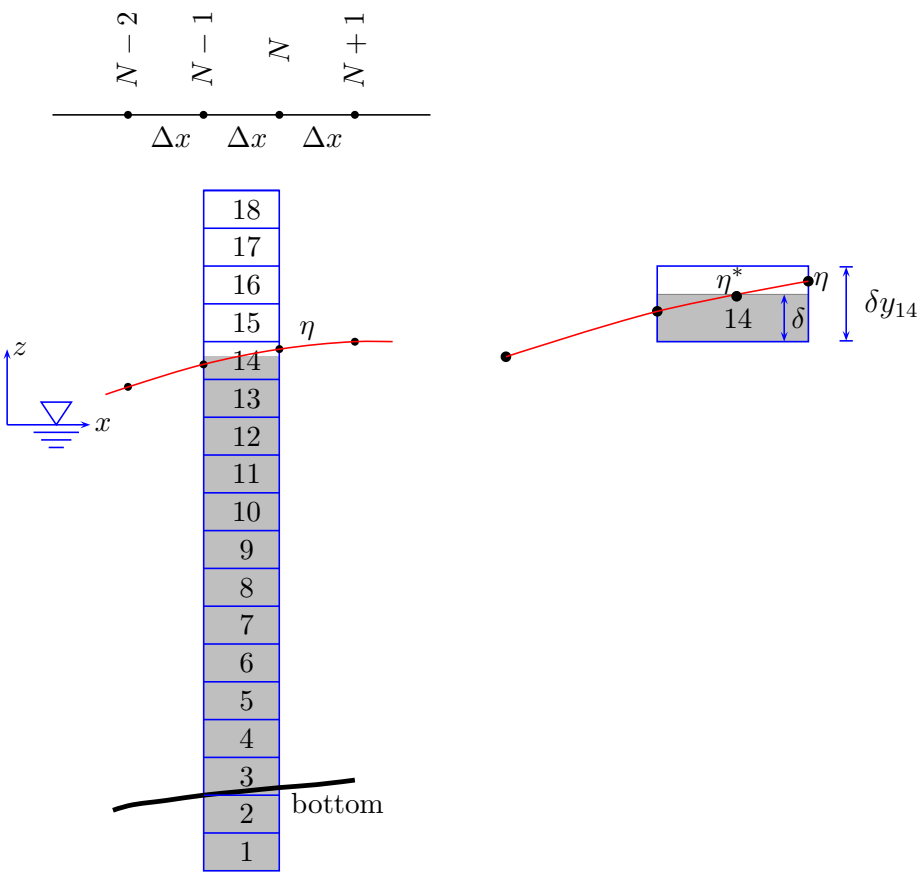


Figure 3. Calculation of the fluid distribution on column 1 of RANS mesh based on the Boussinesq free surface elevation.

surface elevation, $\eta_{N-1/2}$, in the middle of the column by quadratically interpolating η_{N-2}, η_{N-1}, and η_N. Then we search for the cell at which $\eta_{N-1/2}$ is located in the column. In the exemplified figure this interpolated free surface elevation is located at cell 14. All the cells below the cell are full cells with $F = 1$ and the cells above it are empty cells with $F = 0$. For the free surface cell itself, F is calculated as the ratio of the height of the interpolated free surface elevation, δ, in the cell to the cell height, δy_{14}:

$$F_{14} = \frac{\delta}{\delta y_{14}}. \tag{3.2}$$

The vertical structures of the horizontal and vertical velocities in the Boussinesq model are given as in Nwogu (1993):

$$u = u_\alpha - \tfrac{1}{2}\left(z^2 - z_\alpha^2\right)\left(u_\alpha\right)_{xx} - \left(z - z_\alpha\right)\left(hu_\alpha\right)_{xx} \tag{3.3}$$

and

$$v = -z\left(u_\alpha\right)_x - \left(hu_\alpha\right)_x. \tag{3.4}$$

Equations (3.3) and (3.4) are employed to calculate the vertical structures of u along the right face and v along the center of the first column of the RANS mesh. Employing the second order centered finite difference formula on (3.3) at x_N gives:

$$u_N\left(z\right) = u_{\alpha N} - \tfrac{1}{2}\left(z^2 - z_\alpha^2\right)_N \frac{u_{\alpha N+1} - 2u_{\alpha N} + u_{\alpha N-1}}{\Delta x^2} - \left(z - z_\alpha\right)_N \frac{\left(hu_\alpha\right)_{N+1} - 2\left(hu_\alpha\right)_N + \left(hu_\alpha\right)_{N-1}}{\Delta x^2}. \tag{3.5}$$

Similar scheme is also employed to discretize (3.4). Since the reference velocity grid is not aligned with the center of the RANS mesh where the vertical velocity is located, prior to the calculation, we first determine u_α at $x_{N-3/2}$, $x_{N-1/2}$, and $x_{N+1/2}$ by quadratic interpolation. Then the vertical structure of the velocity is calculated at

$x_{N-1/2}$:

$$v_{N-1/2}(z) = -z\frac{u_{\alpha\,N+1/2} - u_{\alpha\,N-3/2}}{2\Delta x} - \frac{(hu_\alpha)_{N+1/2} - (hu_\alpha)_{N-1/2}}{2\Delta x}. \tag{3.6}$$

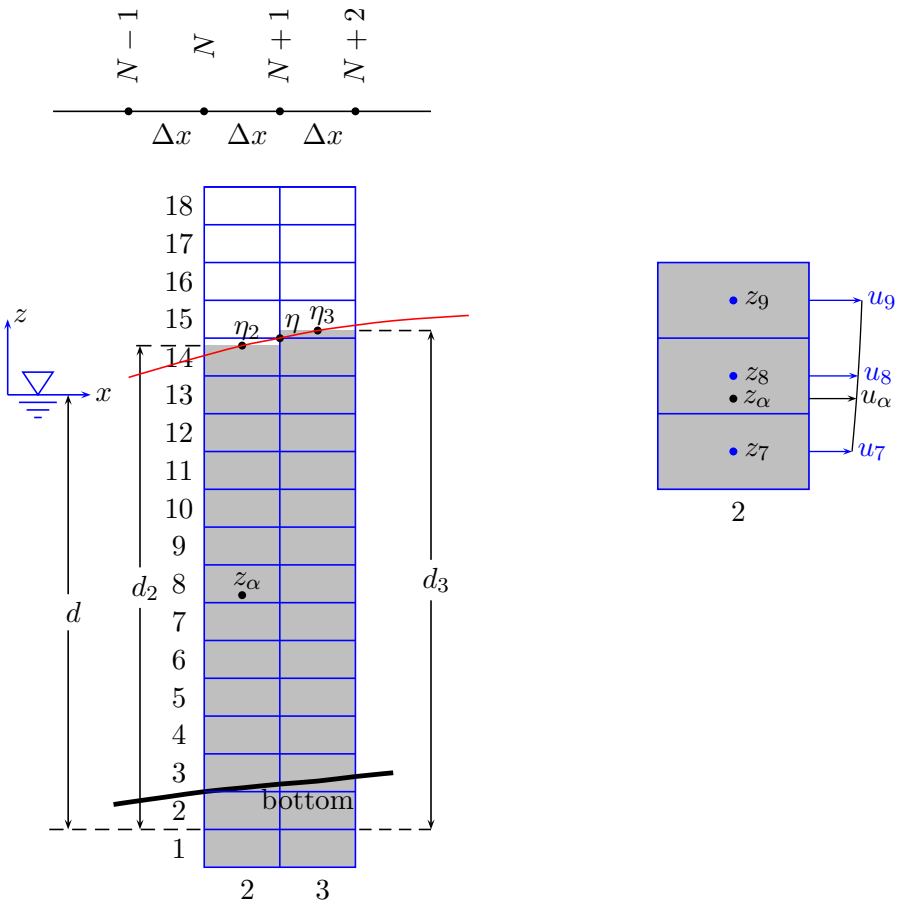


Figure 4. Calculation of the Boussinesq free surface elevation and reference velocity based on the RANS velocity and fluid distribution.

Boussinesq Boundary Condition

The specification of the Boussinesq boundary condition is the inverse procedure of the previous section. In this procedure the reference velocity and the free surface

elevation are calculated based on the velocity, u, and the fluid distribution, F, in the RANS model. This calculation is conducted on the five ghost grids $N+1$ to $N+5$. In this section we describe the calculation of the boundary condition on the grid $N+1$. Similar procedure applies to the rest of the ghost grids.

Figure 4, a slight modification of Figure 3, depicts the RANS-Boussinesq grid system on the interface area. Shown in the Figure are columns 2 and 3 of the RANS model and the grid $N+1$ of the Boussinesq model. Since the Boussinesq free surface elevation grid is not aligned with the center of the RANS column (where F is located), the Boussinesq free surface elevation is calculated as the average (linear interpolation) of two consecutive RANS free surface elevations. In Figure 4, for instance, the free surface elevation at grid $N+1$ is

$$\eta_{N+1} = \tfrac{1}{2}\left(\eta_2 + \eta_3\right), \tag{3.7}$$

where $\eta_2 = d_2 - d$ and $\eta_3 = d_3 - d$. The calculation of the other free surface elevations follows the same procedure.

The reference velocity calculation starts with identifying the cell in the column of the RANS mesh where the reference level is located. In Figure 4 the reference level is located at cell 8. To determine the reference velocity, a quadratic polynomial of the form:

$$u = \beta + \theta\left(z - z_\alpha\right) + \gamma\left(z^2 - z_\alpha^2\right) \tag{3.8}$$

is fitted to the three neighboring velocities u_7, u_8, and u_9. Comparing (3.8) and (3.3), it is apparent that the reference velocity is the zeroth-order coefficient of the polynomial $u_\alpha = \beta$.

Two-Grid, Near-Interface Wave Smoothing

The difference in physics and numerical scheme between the Boussinesq and RANS models gives rise to the discrepancy in velocity profiles along the interface. To demonstrate it, consider a one-way coupling of a Boussinesq and RANS models to simulate a 0.1 m solitary wave propagation in a 0.5 m deep channel, as depicted in Figure 5. Note that in this one-way coupling, only the Boussinesq model passes the information to the RANS model. Figure 5 shows that u and v along the indicated vertical line are different in the Boussinesq and RANS model calculations. Since in two-way coupling both models exchange variables, this discrepancy, although small, gives rise to spurious high-frequency, two-grid wave component, particularly noticeable in the Boussinesq model surface elevation. For longer simulation, this high frequency component will grow and cause instability in the simulation.

Smoothing a signal with such noise may be done by chopping off the unwanted higher components from certain cut-off frequency as done in Phillips (1956). This method is effective but computationally expensive because the spectrum should be first calculated. Another approach is to filter the signal in the spatial domain. This method is straightforward and cheap. To remove this spurious two-grid wave component, we employ the nine-point spatial filter suggested in Shapiro (1970):

$$\eta_i = \tfrac{1}{256}\left\{186\eta_i + 56\left(\eta_{i-1} + \eta_{i+1}\right) - 28\left(\eta_{i-2} + \eta_{i+2}\right) + 8\left(\eta_{i-3} + \eta_{i+3}\right) - \left(\eta_{i-4} + \eta_{i+4}\right)\right\}. \quad (3.9)$$

Filter (3.9) completely removes the spurious two-grid component, attenuates the three- and four-grid components by 32% and 6% respectively, and does not influence components with lower frequencies. Figure 6 shows an example of smoothing the free surface elevation using the nine-point filter. The free surface elevation is

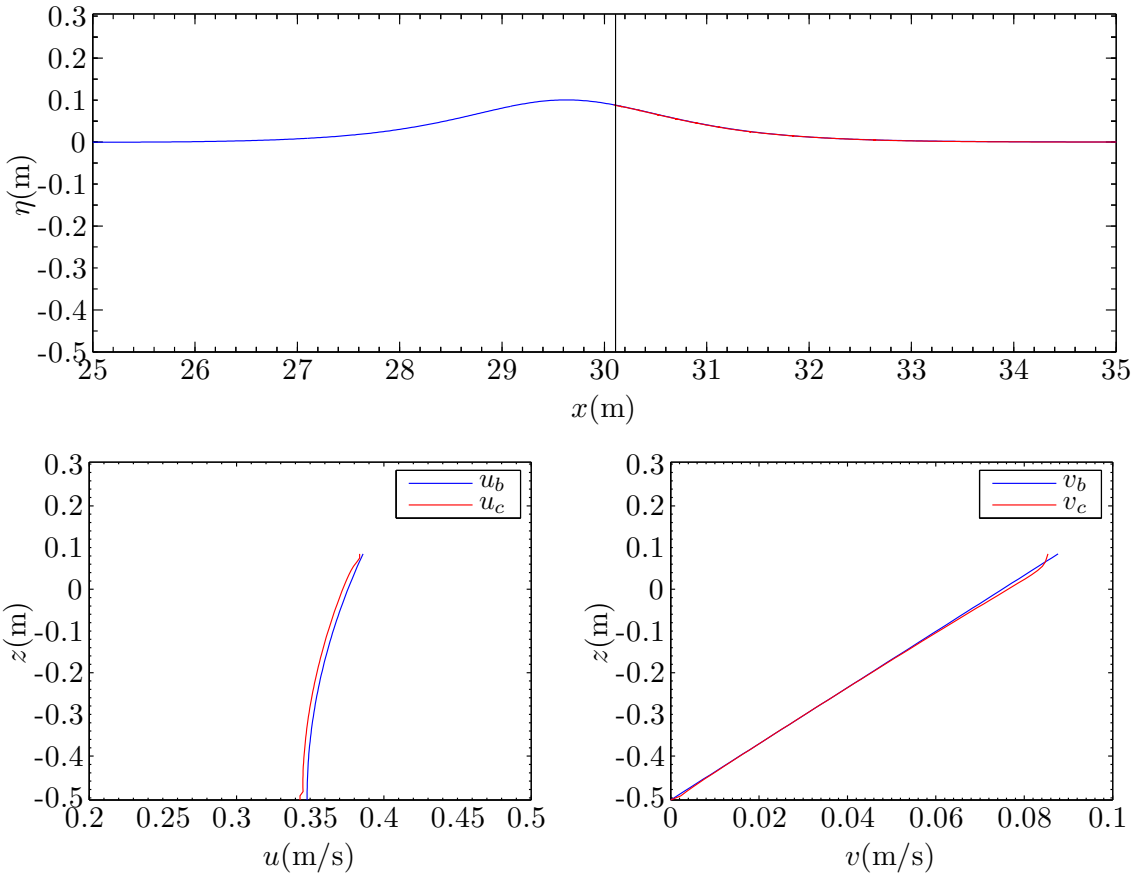


Figure 5. Velocity profile under 0.1 m solitary wave calculated using one-way coupled Boussinesq-RANS model.

produced from a hybrid simulation without filtering, and due to the aforementioned velocity discrepancy, contaminated with two-grid components. Applying the filter once on the data results in the smooth elevation. Most importantly, the higher harmonics are not affected by the application of the filter. Application of 10,000 times successive (equivalent to 100 s simulation with $\delta t = 0.01\,\mathrm{s}$) smoothing also preserves all the higher harmonics, with little change on the boundary. In the hybrid model, however, the filter is applied only once every, say 100 time steps. This, as suggested by the one-time filter application, will not affect the surface elevation near the boundary. Therefore, the implementation of such filter for the whole simulation is, in our opinion, justifiable.

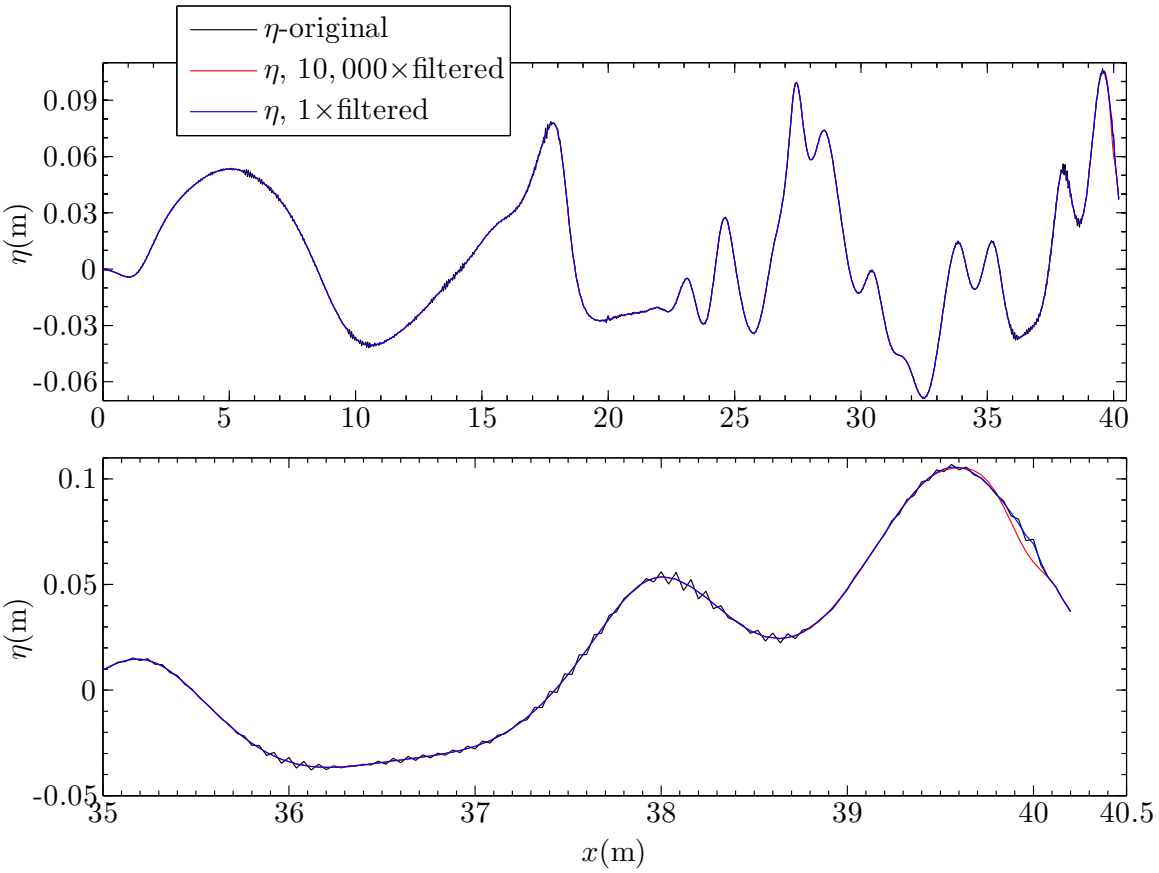


Figure 6. Smoothing free surface elevation from two-grid component using the nine–point filter.

CHAPTER IV
HYBRID MODEL TEST

To test the model for both the validity and relative speedup, we used the hybrid model to simulate the following target scenarios:

1. Solitary wave propagation.
2. Standing wave motion.
3. Sinusoidal wave overtopping of a seawall.
4. Solitary wave overtopping of a levee.
5. Wave propagation over porous structure.
6. Hypothetical large-scale tsunami simulation.

In the first scenario, we simulate the propagation of a solitary wave along a channel of constant depth. In this simulation, we want to observe how the soliton is transported from the RANS to the Boussinesq domains. An important property of a solitary wave propagating in a channel of constant depth is that the wave height is constant. This simulation will be an ideal test for observing this property. In the second test, the hybrid model was used to generate a standing wave in a channel of constant depth. The wave was driven from one end of the channel toward a vertical, reflecting wall. This wave was reflected back toward the origin and thus two waves opposite in directions met on the interface to create a standing wave. In the third simulation, the model was run to simulate the wave overtopping of a seawall. The numerical data collected in this simulation is the mass flux of the wave overtopping across the top of the structure. In the fourth simulation, we also conducted the

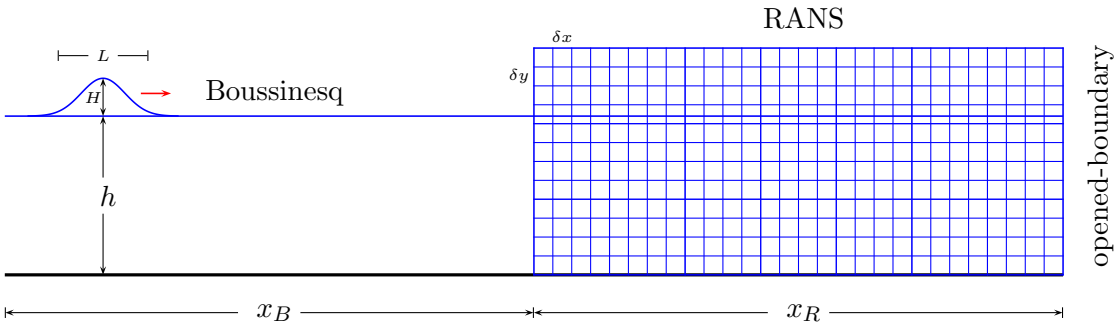
simulation of the wave overtopping of a coastal structure. The free surface elevation of solitary wave, before and after interaction with a levee, is recorded and compared with the experimental data. And in the last simulation a hypothetical tsunami in the deep water is generated and the hybrid model predicts its evolution on the coast, including interaction with a detached breakwater.

To measure the relative speedup of the hybrid computation, the first five scenarios were also run using the RANS model in the whole domain, a "pure RANS" simulation. The CPU times used by the pure RANS to compute those scenarios are compared with the CPU times used in the hybrid model. This comparison gives the relative speedup of the hybrid model.

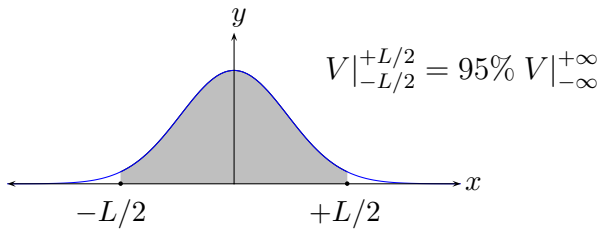
Solitary Wave Propagation

The first test simulates the propagation of a solitary wave in a 0.5 m deep and 100 m long channel. The domain was divided into two subdomains of equal lengths. The first 50-m subdomain, x_B, was occupied by the Boussinesq model and the second 50 m, x_R, by the RANS model. The wave is generated in the Boussinesq domain and propagates to the RANS domain (Figure 7). To observe the behavior of the model with respect to the wave height variations, two waves of different nonlinearities, $\epsilon = H/h$, and dispersiveness, $\delta = h/L$, are considered in the test. While a solitary wave is uniquely defined with only ϵ, the dispersive parameter is included here to both provide additional characterization of the wave and for use as a length scale. The solitary wave length, L, appearing in δ, is defined as the length of the symmetrical-region in the solitary wave (gray-colored area in part (b) of Figure 7) which contains 95% of the total volume of water, Dean and Dalrymple (1991).

The initial free surface elevation and velocity of the solitary wave which was used



(a)



(b)

Figure 7. (a) Hybrid simulation setup for solitary wave propagation. (b) Solitary wave length.

to drive the model has form:

$$\eta = A_1 \text{sech}^2 \left[B \left(x - Ct \right) \right] + A_2 \text{sech}^4 \left[B \left(x - Ct \right) \right] \tag{4.1}$$

and

$$u = A \text{sech}^2 \left[B \left(x - Ct \right) \right]. \tag{4.2}$$

The two equations are the analytic solution to the weakly nonlinear Boussinesq equation, Nwogu (1993). The derivation of the the analytic solutions and the determination of the coefficients A, A_1, A_2, B, and C may be found in Wei and Kirby (1995).

In all the solitary wave simulations the dissipation in both the Boussinesq and RANS models were zero and the fluid viscosity and the turbulence in the RANS model were also set to zero. Hence both models were running an inviscid flow condition, with no means in the RANS model to generate vorticity.

In the first solitary wave simulation, a small amplitude wave of 0.05 m high and initially centered at $x = 10\,\text{m}$ was used. The length of this wave in the 0.5 m deep channel was 6.64 m. The corresponding nonlinearity and dispersiveness were $\epsilon = 0.1$ and $\delta = 0.075$ respectively. Both the Boussinesq and RANS models used uniform spatial grids. The Boussinesq grid size was $\Delta x = 0.125$ m and the RANS grid sizes were $\delta x = 0.0625$ m and $\delta z = 0.0175$ m. In this simulation the constant time step, $\delta t = 0.01$ s, which corresponded to the Courant number $Cr = 0.18$ in the Boussinesq domain and $Cr = 0.35$ in the RANS domain was employed. Here, the characteristic velocity used in the Courant number is the constant, linear long wave speed. To save some computational time, the RANS model starts its calculation when the free surface elevation of the Boussinesq model (which acted as the boundary condition in RANS model) on the interface area exceeds a threshold, $\eta_{\text{threshold}} = 10^{-5}$ m. For this particular simulation, there are no calculations in the RANS domain for the first 13 s

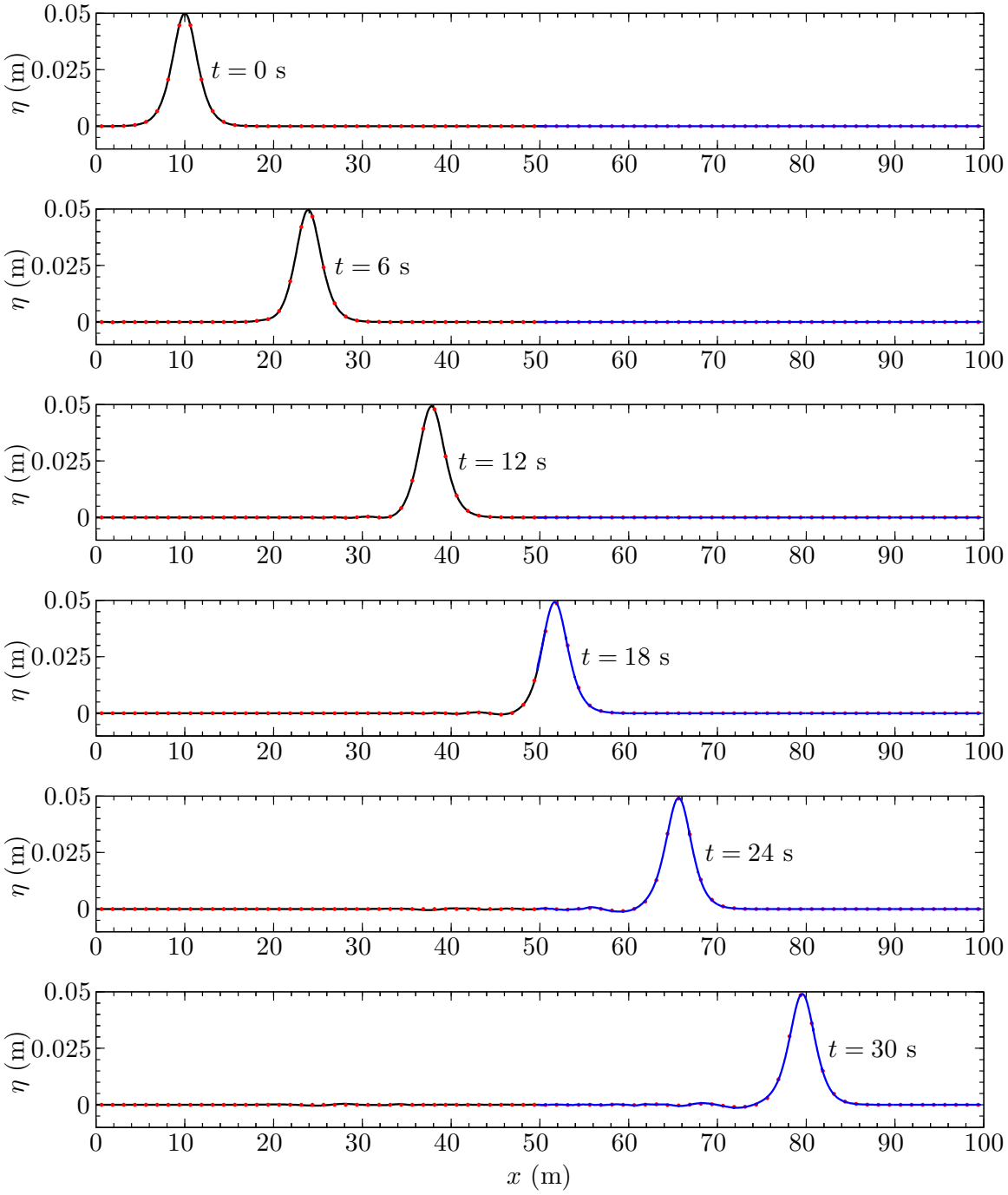


Figure 8. Solitary wave ($\epsilon = 0.1$, $\delta = 0.075$) propagation in a 0.5 m deep channel simulated using the hybrid wave model. Black line is Boussinesq model, blue line is RANS model, and red dots are full-Boussinesq model.

of physical time, or about 44% of the total time. In a huge computational domain, particularly for transient wave studies, this procedure can save a significant amount of computational time. This procedure was implemented for all the simulations in this study.

To see the evolution of the soliton as it propagates along the channel, several snapshots of the free surface elevation, η, at different times were captured at 6 s interval (Figure 8). The distance between two consecutive solitons was 13.9 m. The soliton moved at $13.9/6 = 2.32\,\mathrm{m/s}$. As the wave traveled from the initial location to $x = 80$ m, the wave height was invariant. The wave was correctly transmitted from the Boussinesq to the RANS domain. For comparison the simulation was also conducted using the Boussinesq model for the full domain $0 \leq x \leq 100\,\mathrm{m}$. The wave profiles of this simulation were shown in red dots in Figure 8. Both the hybrid and full-Boussinesq wave profiles agree very well. Note the small, oscillatory tail following the soliton at later times in both the hybrid and full-Boussinesq results; this is a common occurrence when using the weakly nonlinear solitary wave solution in the fully nonlinear model, Wei et al. (1995).

In the second simulation, wave height is increased to 0.15 m. The wavelength was 4.3 m, shorter than the previous wave, and thus the wave steepness is considerably larger here. The nonlinearity was $\epsilon = 0.3$ and the dispersiveness was $\delta = 0.12$. For this simulation the spatial and temporal grids are similar to the previous solitary wave simulation.

The snapshots of the soliton at 6 s temporal interval was presented in Figure 9. Here, the wave moves at a constant speed $2.51\,\mathrm{m/s}$, faster than the previous smaller wave. The $t = 18$ s soliton indicates that there was a smooth transition as the wave enters the RANS domain from the Boussinesq domain. As the wave travels in the RANS domain, however, the wave height decays slowly as indicated by snapshots at

$t = 24$ s and $t = 30$ s. At these two instants, the reduction of the wave height is 4.5% and 7.4% respectively. Thus, from $t = 24$ s to $t = 30$ s, there was a 2.9% wave height reduction. Another simulation using the pure RANS model and employing the same wave and temporal/spatial grids confirmed this decay (Figure 10). This figure obviously shows that the wave underwent damping as it propagated from its initial position to $t = 15$ s position. The wave at $t = 9$ s, for instance, decayed from 0.1366 to 0.1318 m at $t = 15$ s, a 3.2% wave height reduction. As in the previous case, we also performed the full-Boussinesq simulation and the result was also presented in red-dots in Figure 10. The figure shows that the wave height of the full-Boussinesq run was invariant in the course of the simulation. In the last three snapshots, the soliton in the hybrid model, besides being smaller, lagged slightly behind the one in the full-Boussinesq model. This is due to smaller wave has smaller celerity. The wave height decay is due to numerical dissipation in the RANS model.

Similar simulations using the RANS model on the whole domain were conducted to measure the relative speedups of the hybrid simulations. The temporal and spatial grids for the pure RANS simulations are similar to RANS grids in the hybrid simulations. Hence, the total number of the RANS horizontal computational grids in these simulations is twice as many as in the hybrid model, while the number of vertical grids remained unchanged. For the hybrid model, the computational clock time for the 0.05 m wave was 312 s and for the 0.15 m wave was 317 s. The same simulations using the RANS model took 1,260 s and 1,328 s for the 0.05 m and 0.15 m waves respectively. These computational clock times corresponded to 30.4 s simulation and were run in a 3-GHz Pentium PC. The speedups gained in the hybrid model were 4 for the first wave and 4.2 for the second. In both the hybrid simulations the threshold was reached at 13.4 s, 44% of the total) simulation time, and took 25 s (8% of the total) computational clock time. This comparison demonstrates that the threshold

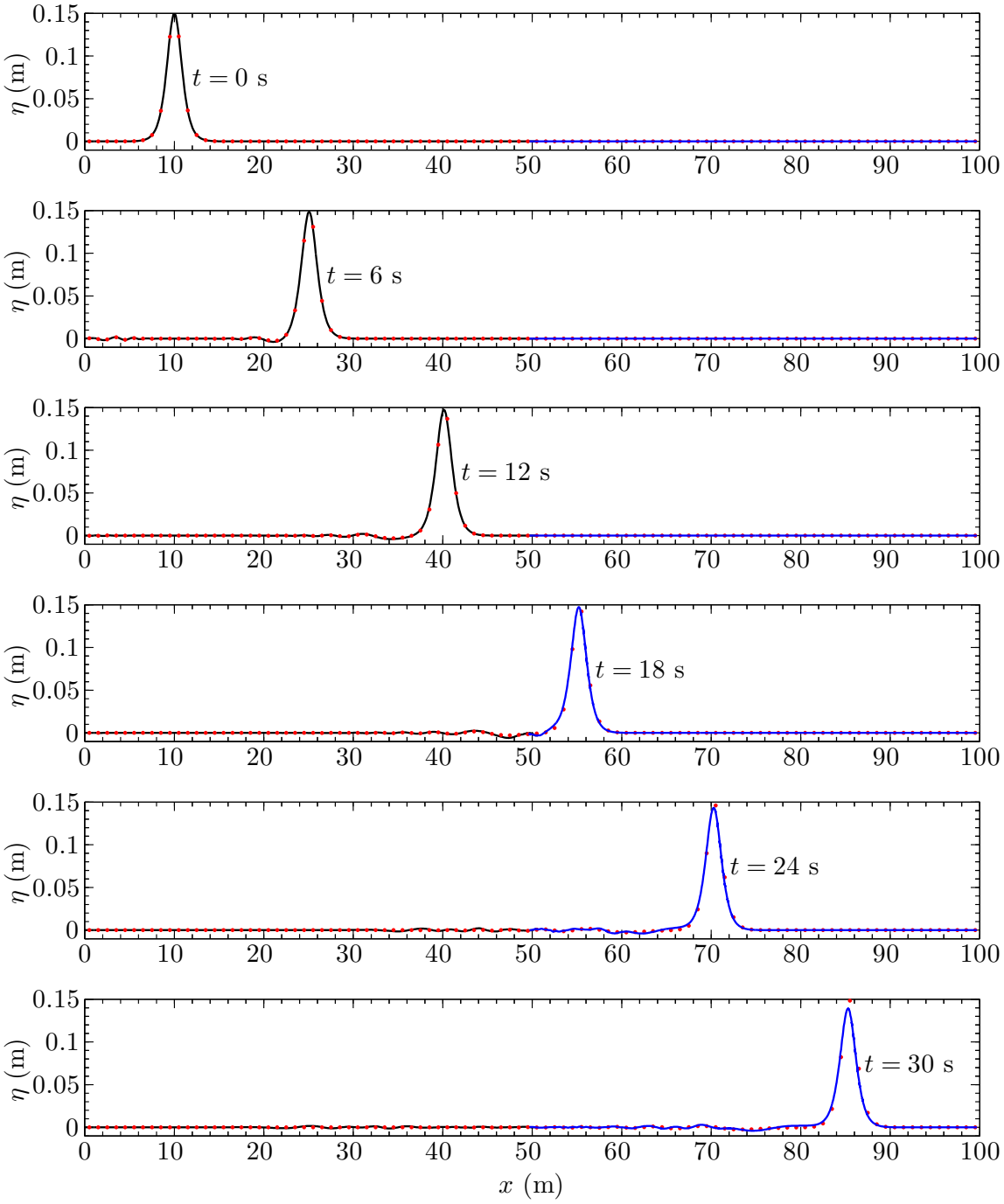


Figure 9. Solitary wave ($\epsilon = 0.3$, $\delta = 0.12$) propagation in a 0.5 m deep channel simulated using the hybrid wave model. Black line is Boussinesq model, blue line is RANS model, and red dots are full Boussinesq model.

procedure can save, for these particular cases, 44% the effort to run RANS in the hybrid model from the beginning.

Standing Wave Motion

In the following test, the hybrid wave model is used to simulate standing wave motion. The standing wave is created by superimposing two sinusoidal waves of the same height, H, moving in the opposite directions in a channel of constant depth, h, as shown in Figure 11. Here, the numerical channel is 72 m long, 0.5 m deep, divided into two subdomains of equal lengths. The left subdomain, x_B, is occupied by the Boussinesq model and the right one, x_R, by the RANS model. Both ends of the channel are impermeable walls and thus act as reflecting boundaries. Adjacent to the left boundary, a sponge layer was installed to damp out all the waves that entered the sponge layer area. For wave generation, we employed the internal source method, Hsiao et al. (2005). Identical sinusoidal waves with height, H, propagated from the source to both the left and right directions. The wave that propagated to the left was damped out by the sponge layer and the one to the right would propagate along the Boussinesq and the RANS domains. As the wave reached the right boundary, it was reflected and propagated back toward the wave source and superimposed with the incoming wave to create the standing wave.

In the first test of this simulation, a relatively small wave, 0.01 m high and 4 s period, was generated 5 m to the right of the left boundary. A 3 m sponge layer was located adjacent to the left boundary as the damping mechanism. From the dispersion relationship, the 4 s wave, in the 0.5 m deep channel, had an 8.7 m wavelength. The nonlinearity and the dispersiveness of this wave are then $\epsilon = 0.02$ and $\delta = 0.06$ respectively. The Boussinesq domain was discretized into uniform

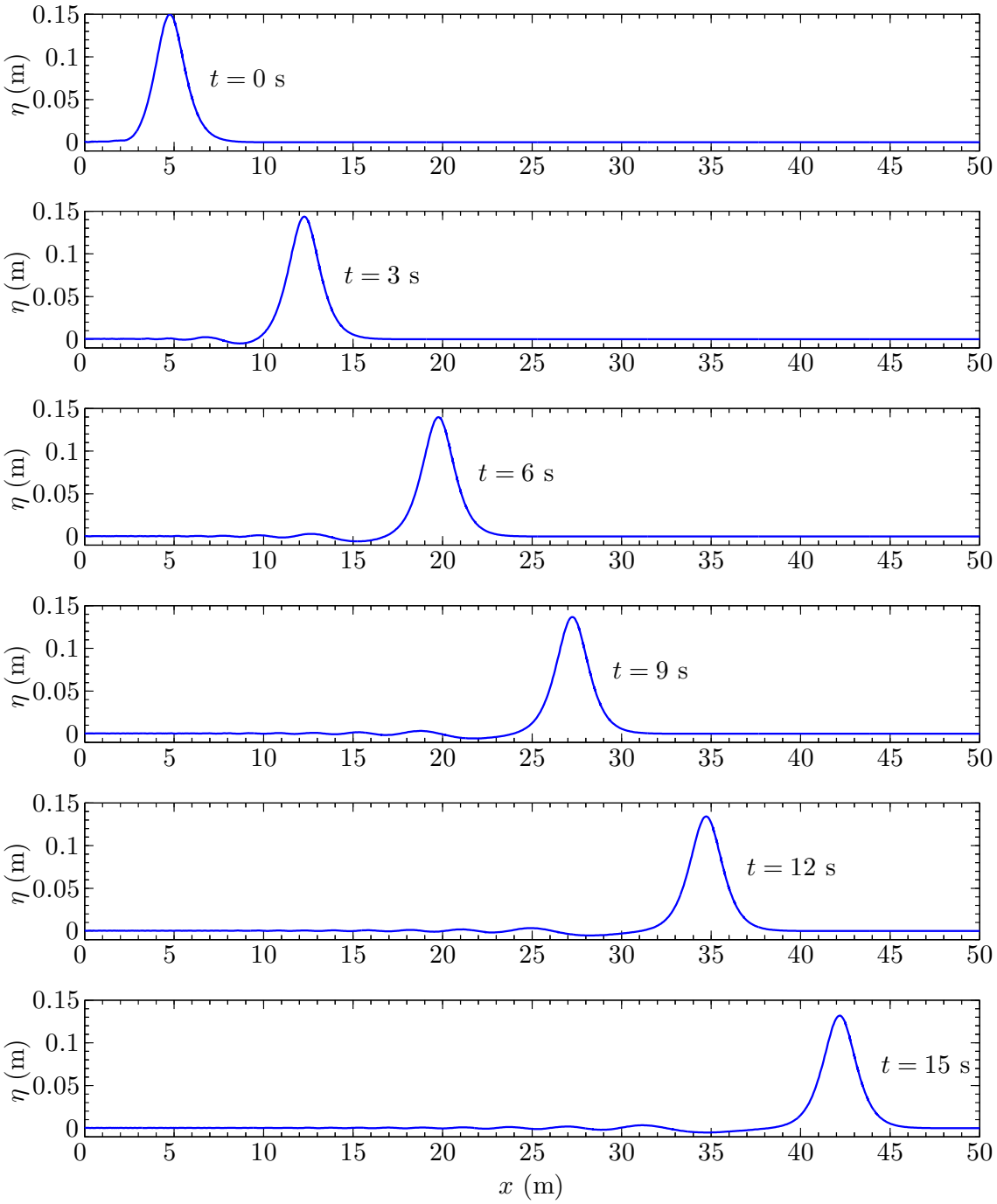


Figure 10. Solitary wave ($\epsilon = 0.3$, $\delta = 0.12$) propagation in a 0.5 m deep channel simulated using the RANS model.

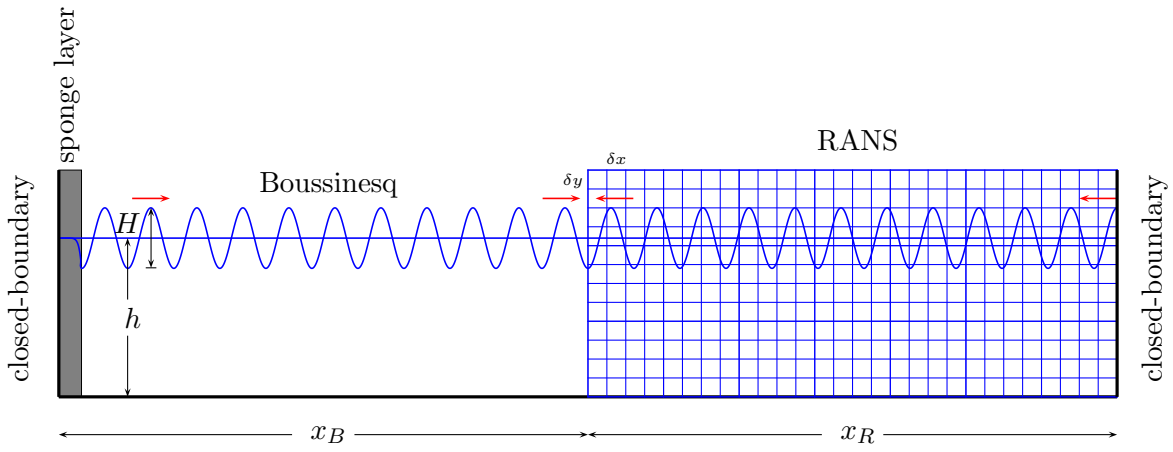


Figure 11. Hybrid model setup for standing wave simulation.

grids, $\Delta x = 0.08$ m. Similarly, the RANS domain was uniformly discretized both horizontally, $\delta x = 0.04$ m, and vertically, δy= 0.001 m. The simulation was run for 100 s with the constant time step $\Delta t = 0.009$ s and zero viscosity and turbulence.

Figure 12 depicts the instantaneous wave profiles at six different instants. The first three profiles show the wave propagating to the right and reaching the wall at $t = 31$ s. Within this period the wave in the channel was not contaminated by the reflected wave from the right boundary and the height was still 0.01 m. Afterwards, the reflected wave starts propagating in the channel and is superimposed with the wave from the source. Since the two waves are identical and 180 degrees out of phase, this superposition results in a standing wave in the channel with height 0.02 m, i.e. twice the original wave height. In all the snapshots the wave profile on the interface is smooth. Also, for comparison, the full-Boussinesq simulation was conducted and the corresponding instantaneous profiles are plotted in the figure in red dots. The two simulations are in good agreement.

Although the wave source input is linear (single harmonic), the wave undergoes

some nonlinear evolution due to the nonlinear governing equations employed in both the Boussinesq and RANS models. The higher the nonlinearity, ϵ, the stronger the interaction. In the first case the nonlinearity is relativity small and its effect might not be so obvious in the wave profiles in Figure 12. Looking at the profile from a different perspective, for instance in the frequency domain, the nonlinearity effect becomes apparent. Figure 13 gives the amplitude spectrum of the time series of the free surface elevation at $x = 38$ m. This particular time series was recorded before the reflected wave reached the recording location. This spectrum clearly shows that the initially monochromatic wave, in the course of the propagation, transforms into a polychromatic wave. In the spectrum, there are two distinct spikes corresponding to $f_1 = 0.25$ Hz and $f_2 = 0.5$ Hz. Note that the frequency of the original signal was $f = 0.25$ Hz. Here, the first harmonic, f_1, interacts with itself resulting in the second harmonic, $f_2 = f_1 + f_1$. The interaction of the two harmonics, f_1 and f_2, is not too strong in this case as there is no other distinct spike occurring in the spectrum. In the next test, as the wave height becomes higher, the effect of nonlinearity is stronger.

In the second test, the wave height is increased to 0.05 m, with a nonlinearity of $\epsilon = 0.1$. The model setup remains the same as in the previous test. Figure 14 shows snapshots of the wave profiles as the wave propagates in the channel. Here, a smooth transition is again observed on the interface of the two models. Although there is very slight discrepancy between the hybrid and full-Boussinesq profiles, in general the two simulations show good agreement. This slight discrepancy is probably due to the numerical dissipation in the RANS model. The wave profile differs from the profile in the previous test in two ways: a wider trough and an occurrence of a secondary crest on the trough which are due to the nonlinearity effect. The amplitude spectrum shown in Figure 15 has three distinct harmonics f_1, f_2, and f_3. The first two harmonics are at the same frequency as in the previous case. The interaction between

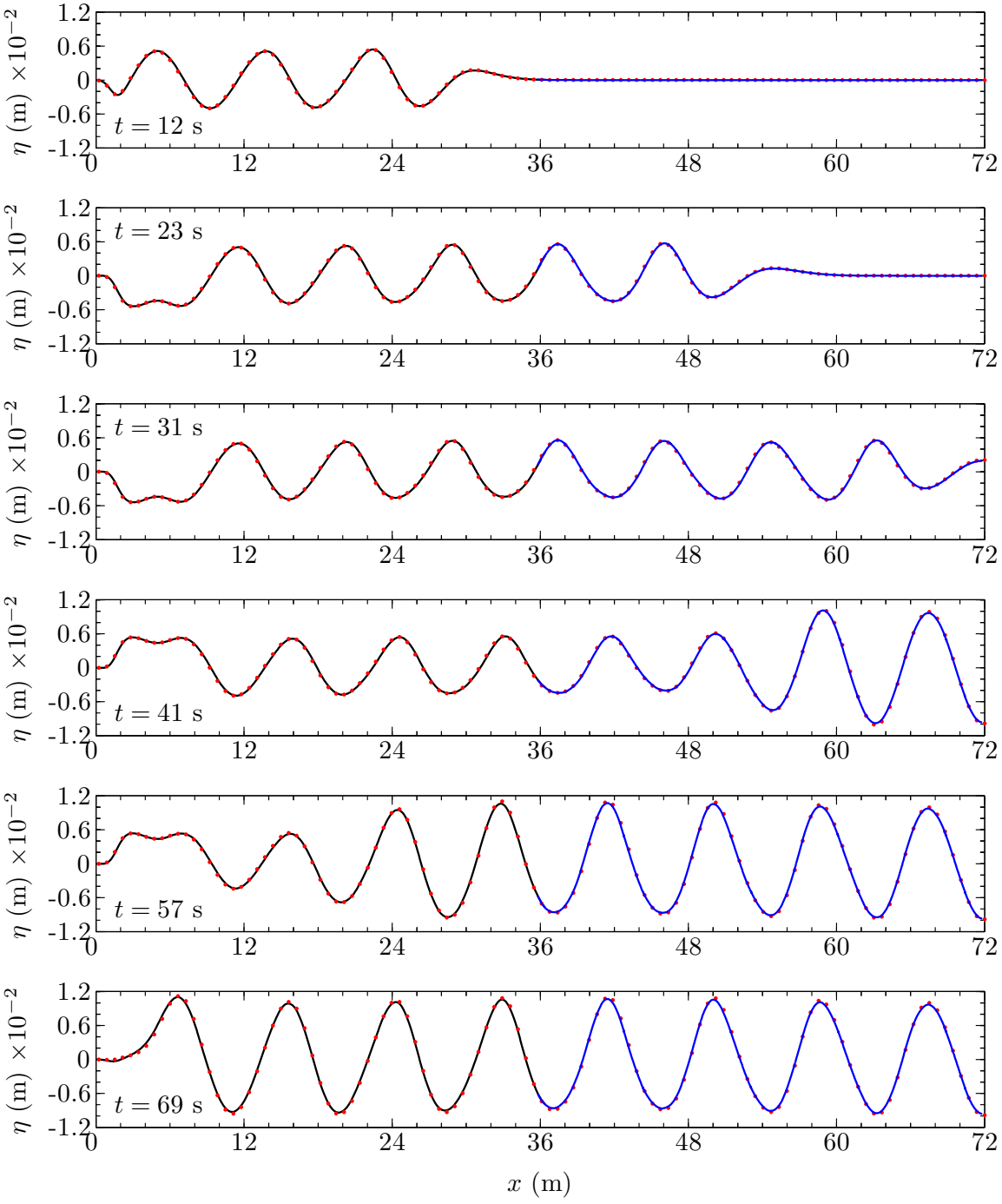


Figure 12. Standing wave ($\epsilon = 0.02$, $\delta = 0.06$) motion in a 0.5 m deep channel simulated using the hybrid model. Black line is Boussinesq model, blue line is RANS model, and red dots are full-Boussinesq model.

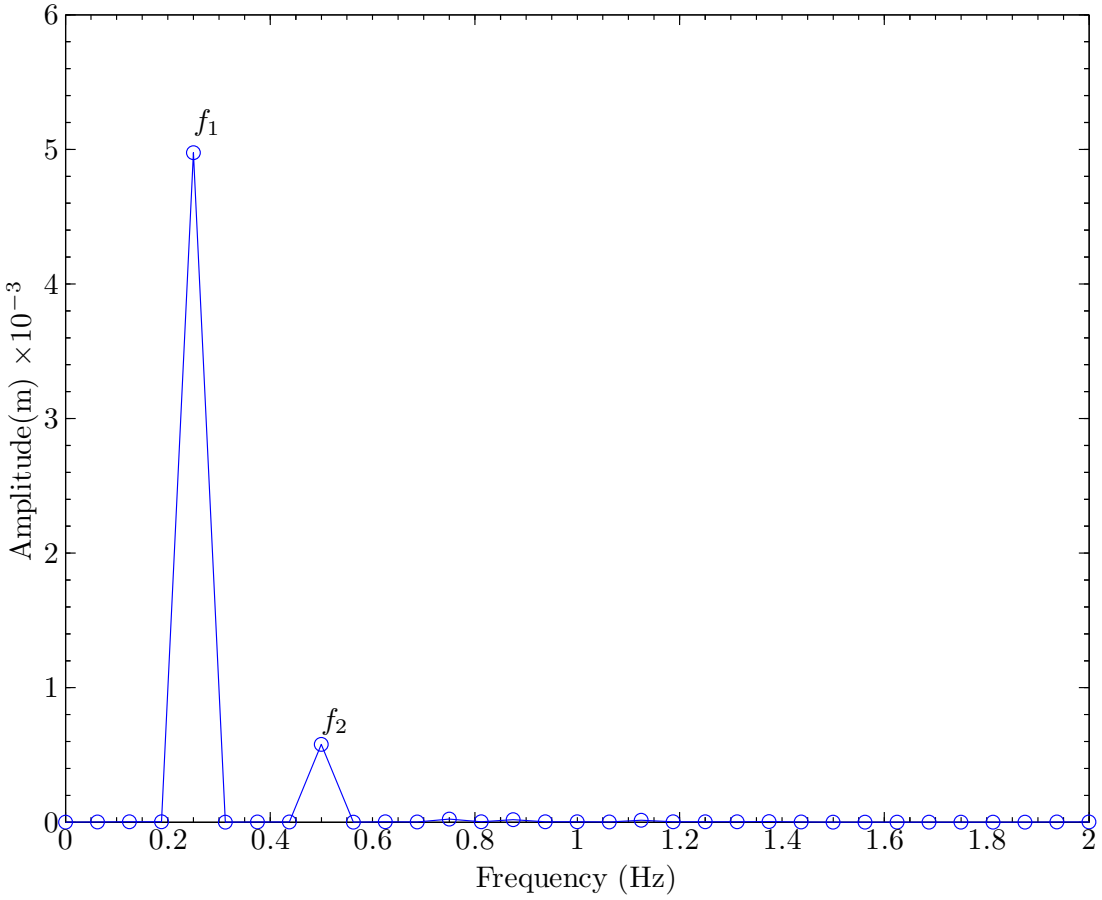


Figure 13. Amplitude spectrum of the free surface elevation time series of the $\epsilon = 0.02$ standing wave simulation. The time series was recorded at $x = 38$ m before the reflected wave reached this location.

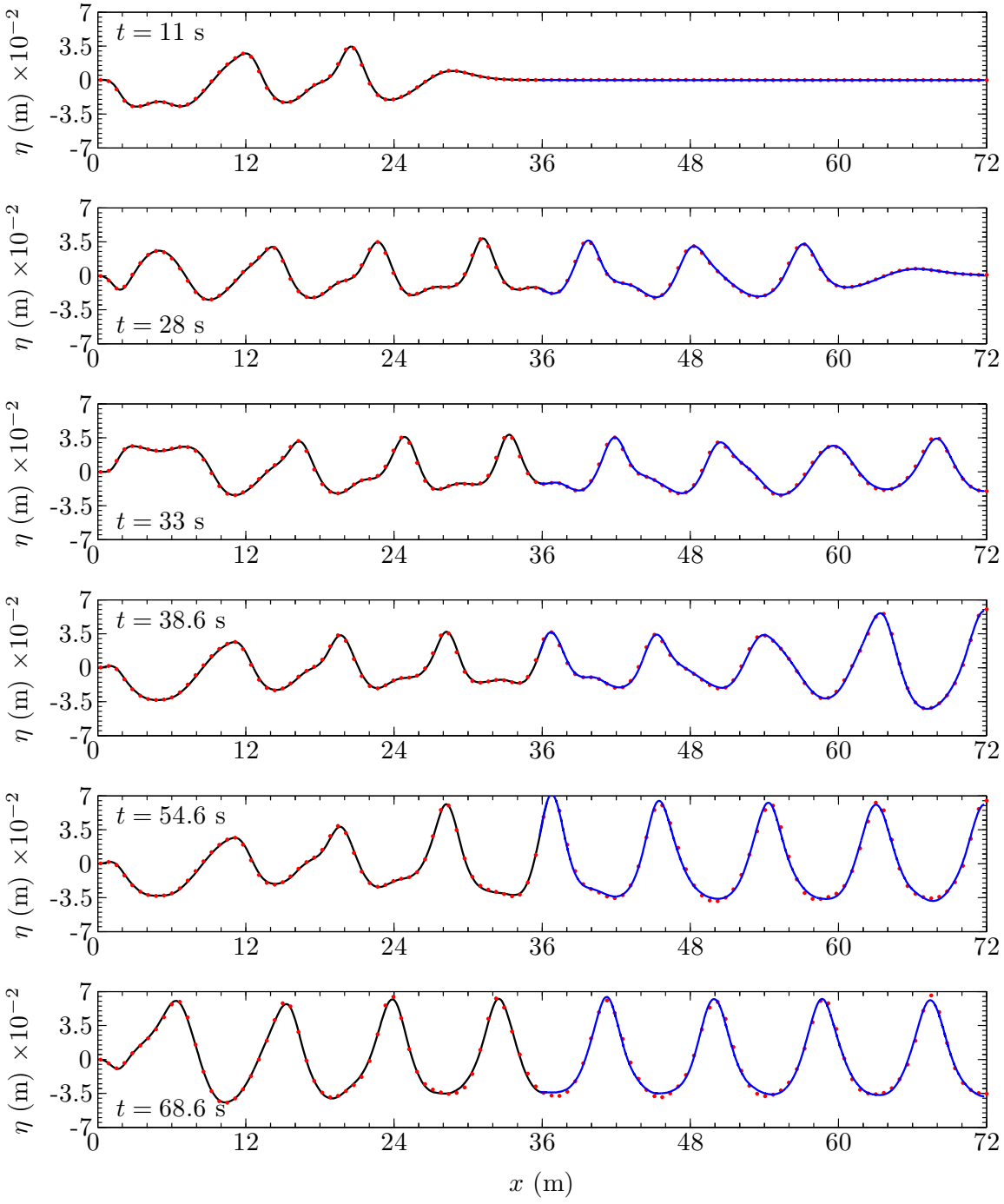


Figure 14. Standing wave ($\epsilon = 0.1$, $\delta = 0.06$) propagation in a 0.5 m deep channel simulated using the hybrid model. Black line is Boussinesq model, blue line is RANS model, and red dots are full-Boussinesq model.

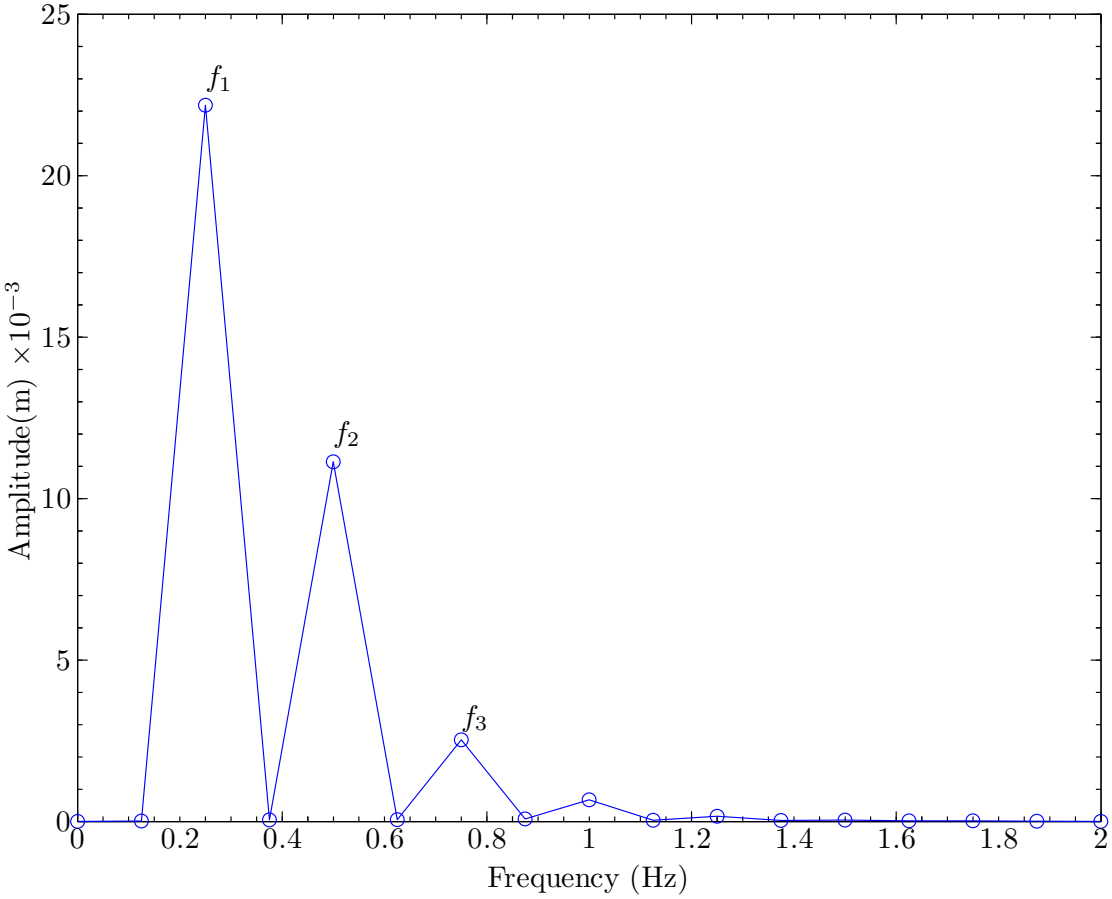


Figure 15. Amplitude spectrum of the free surface elevation time series of the $\epsilon = 0.1$ standing wave simulation. The time series was recorded at $x = 38$ m and before the reflected wave reached this location.

the first and second harmonics resulted in the third harmonic whose frequency was $f_3 = f_1 + f_2 = 0.75$ Hz.

In addition to the hybrid and full-Boussinesq simulations, we also simulated both cases of the standing wave motion using the RANS model for speedup comparison. The RANS model was applied on the whole domain with the same grid size as in the hybrid-RANS model. In the horizontal direction the number of computational grids was twice as many as the grids in the hybrid-RANS, and in the vertical direction both setups employed the same number of grids. Therefore, in total, the number of grids in the pure-RANS model was twice the number of grids in the hybrid-RANS. Both the pure-RANS and hybrid simulations were conducted in a Pentium 4 PC 3.4-GHz for 400 s simulation time. In the first standing wave test the hybrid model spent an average of 321 s for one wave period simulation while the pure-RANS model took about 592 s. Hence, in the first test we gained a 1.8 factor of speedup. In the second test, for one wave period the hybrid and full-RANS models spent 248 and 592 s respectively and the gained speedup was 2.4.

Sinusoidal Wave Overtopping of a Seawall

As reported in Saville (1955) the Beach Erosion Board (BEB) conducted a laboratory experiment in the Waterways Experiment Station of the Corps Engineers, at Vicksburg, Mississippi, to study wave run-up and overtopping of shore structures. The experiment was conducted in a concrete wave flume 36.6 m long, 1.52 m wide, and 1.52 m deep. The model was an undistorted scale model with $1 : 17$ length scale and $1 : 4.1$ time and velocity scales. A wavemaker was used on the upstream side of the flume for wave generation. Downstream the flume, shore structures of various shapes (smooth slope, curve-faced wall, recurved wall, etc.) were built. Behind the structure

a calibrated measuring tank was installed for collecting the overtopping water. The water from the first three or four waves was wasted to allow for the wave to attain a stable condition, after which the water from six or seven waves was collected in the tank for an overtopping flux measurement.

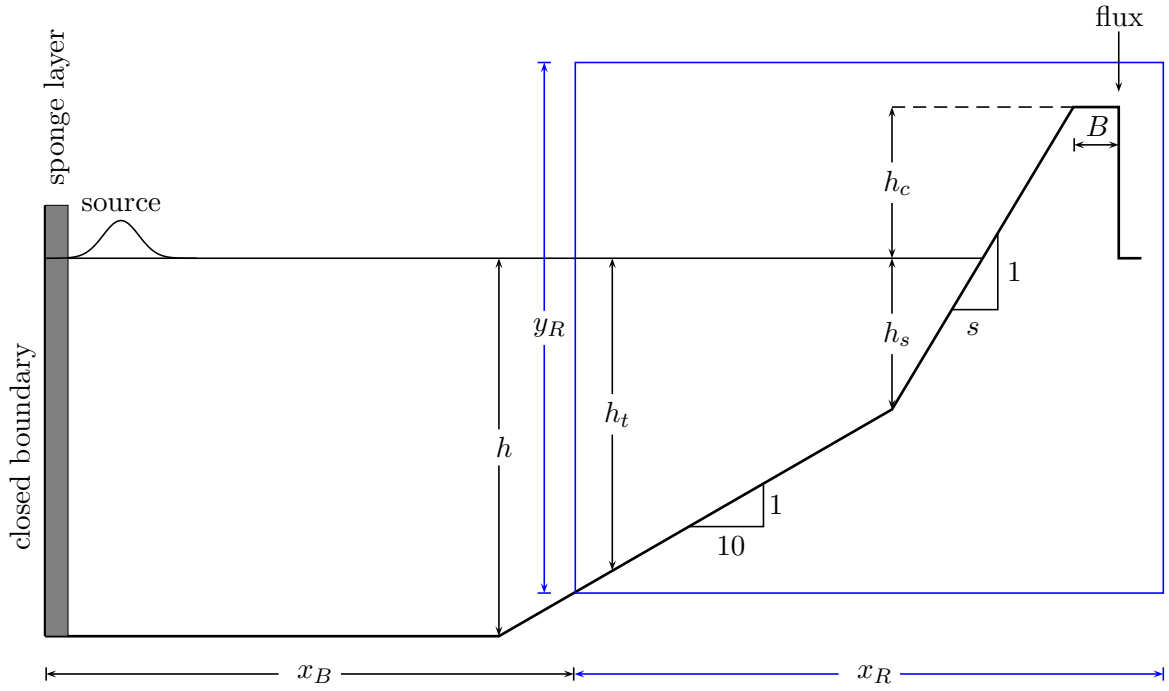


Figure 16. Beach Erosion Board experimental setup of the sinusoidal wave overtopping; figure is not scaled.

In this study, numerical simulations are undertaken of the BEB flux overtopping experiment using the hybrid model whose setup is given in Figure 16. For the simulation comparisons, the smooth structure data, as used by the previous researchers, Kobayashi and Wurjanto (1989) and Dodd (1999), in their overtopping studies, is employed here. In the numerical simulations the wave is generated using the sinusoidal wave source combined with the sponge layer on the left boundary for damping. This approach is different from those of Kobayashi and Wurjanto (1989) and Dodd

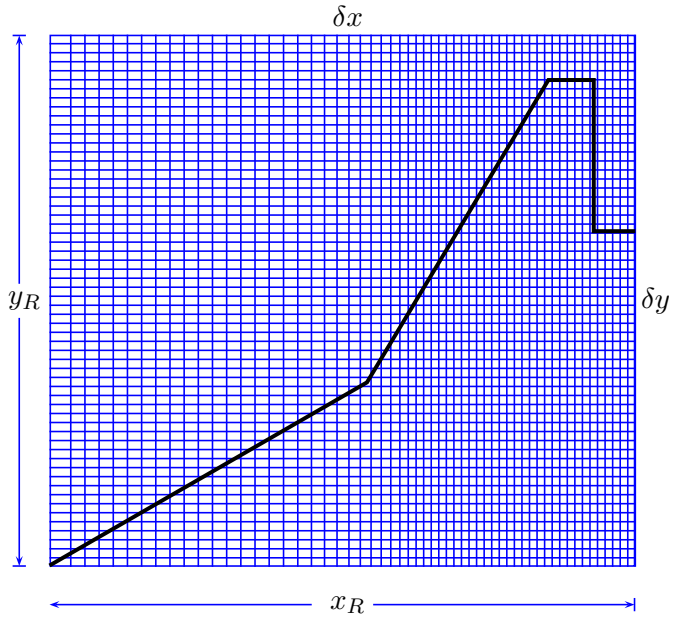


Figure 17. RANS computational mesh for hybrid simulation of BEB sinusoidal wave overtopping.

(1999) where, depending on the Ursell number, U_r, at the model boundary, h_t, Stokes or Cnoidal waves are used to drive the simulations, although the wavemaker in the physical experiments created single harmonic waves only. In all the experiments, the structure with slope $1 : s$ was fronted by a fixed $1 : 10$ inclined floor. The domain is divided into the Boussinesq subdomain, x_B, and RANS subdomain, x_R. The model interface is located near the toe of the $1 : 10$ floor where turbulence was small and the wave did not yet break. In all the simulations, this interface divides the domain into the two subdomains with ratio $x_B/x_R \approx 7$. The height of the RANS domain, y_R, is large enough to prevent the overtopping wave from reaching the RANS top boundary. For this simulation, the Boussinesq domain is discretized into a uniform grid, Δx, and the RANS domain is discretized, for efficiency, into nonuniform horizontal grid, δx, and uniform vertical grid, δy, as depicted in Figure 17. As shown in the figure, the

mesh is relatively coarse near the left boundary and finer around the crest to allow for an accurate flux measurement. Also for efficiency, a dynamic time step, δt, is employed. Similar to Kobayashi and Wurjanto (1989), the flux is computed at the the back edge of the structure and given by:

$$q = \sum_{n=\mathrm{n}t_1}^{\mathrm{n}t_2} \sum_{j=1}^{\mathrm{n}y} F_{ji} u_{ji} \delta y_j \delta t_n. \tag{4.3}$$

The indices i and j correspond to a cell in the column along which the flux is computed. Since some cells might be not full, equation (4.3) includes the corresponding value of the volume of fluid, F_{ji}. $\mathrm{n}t_1$ and $\mathrm{n}t_2$ are the time indices which correspond to the starting/ending times of the water collecting. $\mathrm{n}y$ is the number of vertical grids in the RANS mesh.

As in the BEB experiment, the simulation is run under various geometrical setups with different combinations of variables including the offshore depth, h, the depth at the toe of the structure, h_s, free board height, h_c, slope of the structure, s, wave height, and period. These variables, the experimental data, the computed hybrid model fluxes, and the published results from Kobayashi and Wurjanto (1989) and Dodd (1999) are presented in Table 1. In general the computed fluxes are in good agreement with the experimental data and consistent with the results of the previous two researchers. These data also demonstrate the relatively wide variability that can be found in published overtopping predictions, particularly for low overtopping rates.

As previously explained, the interface of the model should be located such that the turbulence intensity on the interface is small. To provide insight into this, in Figure 18 the instantaneous intensity of the turbulent kinetic energy, k, for run 1, is given. This figure shows that the intensity of the turbulent kinetic energy is high near the structure compared with other location. The kinetic energy near the structure is roughly $10^{-2}\,\mathrm{m}^2$ in contrast to a value less than $10^{-2}\,\mathrm{m}^2$ on the interface. As the

wave approaches the structure, the wave height to depth ratio becomes so large that the wave brakes while impinging on the structure and hence releases kinetic energy. Figure 18 also indicates that in the course of the simulation the location of the hot spot remains close to the crest of the structure while the interface area is always low in kinetic energy. This satisfies the requirement that the turbulence intensity be low on the interface.

To benchmark the simulation time, five of the previous simulations are rerun using the RANS model in the whole domain. Table 2 presents the run times per wave period for both the hybrid and pure-RANS wave models for the five runs. In discretizing the pure-RANS model, the part of the hybrid domain that used the RANS model, the breaking zone, uses an identical mesh as in the hybrid model. Offshore of this point, where the Boussinesq is used in the hybrid, the pure-RANS domain is uniformly discretized with a grid size equal to the grid size at the hybrid interface location.

The computation times of both the hybrid and pure-RANS simulations are summarized in Table 2. From this table it is clear that the speedup due to use of the hybrid model is significant, ranging from a factor of near 10 to over 17. This large speedup is of course due to the smaller RANS mesh used in the hybrid. However this difference is two fold; a single iteration of the Poisson pressure solver requires less time with a smaller matrix, and a smaller matrix will converge in fewer iterations of the PPE solver. Note also the overtopping flux predictions presented in the Table 1. The slight discrepancies in these results are attributed to the small differences in the modeled physics and numerical accuracy in the nonbreaking part of the domain.

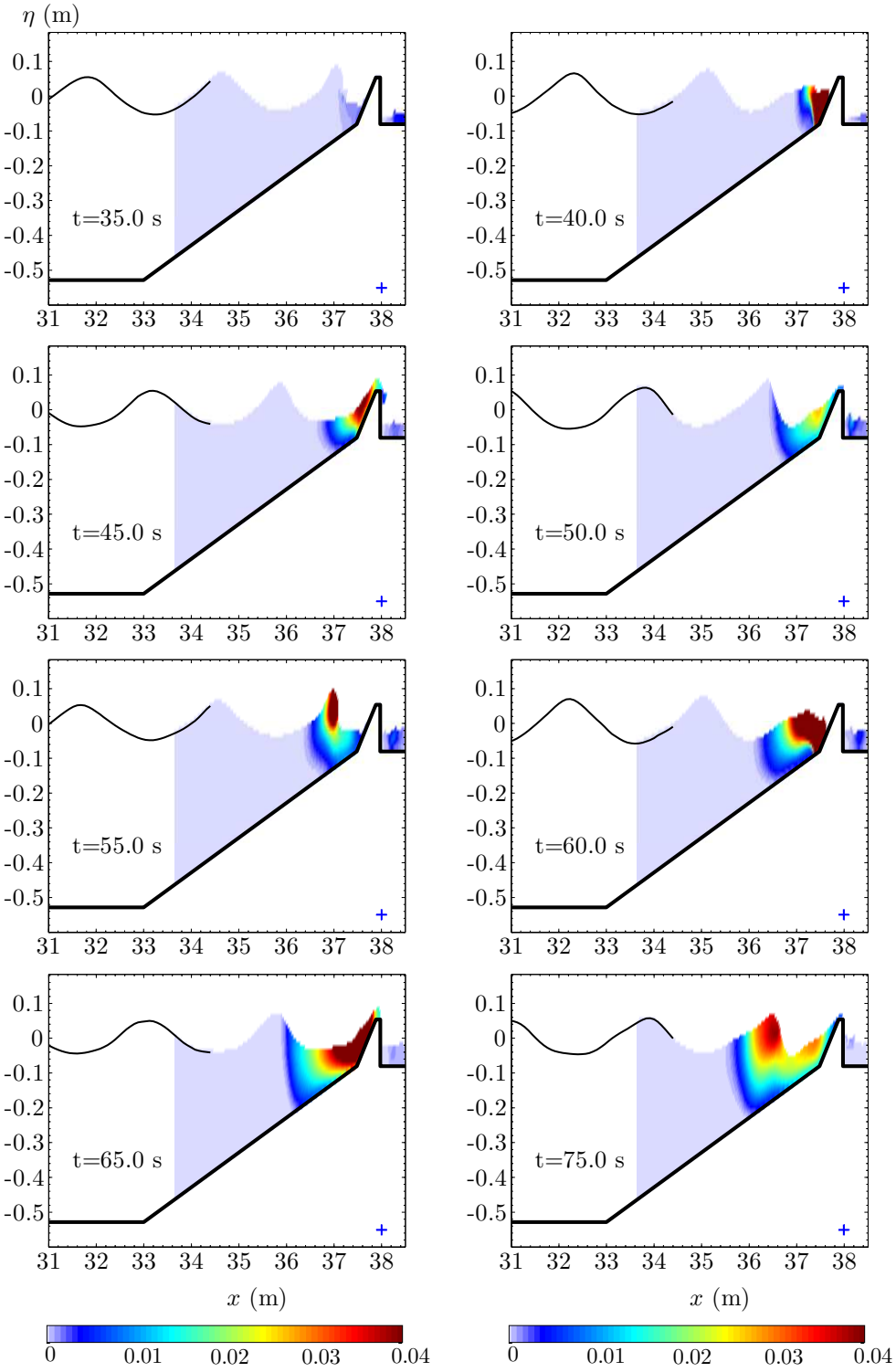


Figure 18. Turbulence kinetic energy distribution of the BEB sinusoidal wave overtopping, run 1.

Table 1. Experimental and simulated fluxes of the BEB sinusoidal wave overtopping.

Run[†]	h (m)	h_s (m)	h_c (m)	H (m)	T (s)	q_{data}[‡] $(\frac{\text{m}^2}{\text{s}})$	q_{KW} $(\frac{\text{m}^2}{\text{s}})$	q_{OTT} $(\frac{\text{m}^2}{\text{s}})$	q_{hyb} $(\frac{\text{m}^2}{\text{s}})$
1	0.529	0.081	0.054	0.107	1.549	0.0073	0.0030	0.0039	0.0032
2	0.529	0.081	0.107	0.107	1.549	0.0004	0.0003	0.0007	0.0003
3	0.609	0.161	0.054	0.107	1.549	0.0071	0.0058	0.0066	0.0075
4	0.609	0.161	0.107	0.107	1.549	0.0040	0.0015	0.0019	0.0055
5	0.609	0.161	0.054	0.081	1.858	0.0065	0.0058	0.0062	0.0062
6	0.529	0.081	0.054	0.107	2.616	0.0066	0.0060	0.0074	0.0071
7	0.529	0.081	0.107	0.107	2.616	0.0019	0.0018	0.0025	0.0030
8	0.529	0.081	0.161	0.107	2.616	0.0044	0.0002	0.0007	0.0012
9	0.609	0.161	0.054	0.107	2.616	0.0104	0.0100	0.0118	0.0128
10	0.609	0.161	0.107	0.107	2.616	0.0044	0.0050	0.0064	0.0069
11	0.609	0.161	0.161	0.107	2.616	0.0009	0.0018	0.0028	0.0024
12	0.529	0.081	0.054	0.081	3.634	0.0065	0.0070	0.0076	0.0063
13	0.609	0.161	0.054	0.081	3.634	0.0093	0.0081	0.0086	0.0086
14	0.609	0.161	0.107	0.081	3.634	0.0055	0.0037	0.0044	0.0026
15	0.609	0.161	0.161	0.081	3.634	0.0018	0.0011	0.0016	0.0015
16	0.609	0.161	0.215	0.081	3.634	0.0008	0.0011	0.0002	0.0004
17	0.529	0.081	0.054	0.107	2.616	0.0054	0.0073	0.0069	0.0064
18	0.529	0.081	0.161	0.107	2.616	0.0014	0.0009	0.0008	0.0016
19	0.448	0.000	0.054	0.107	2.616	0.0043	0.0044	0.0041	0.0030
20	0.448	0.000	0.107	0.107	2.616	0.0022	0.0008	0.0009	0.0004

[†] Run 1–16 used $s = 3$ and run 17–20 used $s = 1.5$.

[‡] Here, flux is presented in dimensional instead of dimensionless form as in Kobayashi and Wurjanto (1989).

Table 2. Computation time per wave period of the BEB sinusoidal wave overtopping, run 3, 5, 9, 18, and 19.

RUN	Hybrid			RANS				
	$\mathrm{n}x \times \mathrm{n}y$ (A_H)	t (sec)	Flux $\left(\frac{\mathrm{m}^2}{\mathrm{sec}}\right)$	$\mathrm{n}x \times \mathrm{n}y$ (A_H)	t (sec)	Flux $\left(\frac{\mathrm{m}^2}{\mathrm{sec}}\right)$	$\frac{\mathrm{A}_R}{\mathrm{A}_H}$	$\frac{t_\mathrm{R}}{t_\mathrm{H}}$
3	118×72	16.7	0.0075	798×86	269.8	0.0075	8.1	16.2
5	118×82	26.2	0.0062	798×98	454.7	0.0061	8.1	17.4
9	262×76	85.6	0.0128	826×122	1259.2	0.0122	5.1	14.7
18	149×72	25.5	0.0016	845×72	242.6	0.0010	5.7	9.5
19	115×52	10.5	0.0030	811×56	120.8	0.0032	7.6	11.5

Solitary Wave Overtopping of a Levee

In 1996, Hydraulic Research (HR) Wallingford in the United Kingdom performed an experiment on solitary wave overtopping of a breakwater. The experimental setup is given in Dodd (1999) and depicted here in Figure 19. The wave flume used in this experiment was 40 m long and 0.5 m wide and filled with water to $h_1 = 0.7$ or 0.6 m seaward of the breakwater and $h_2 = 0.3$ or 0.2 m behind the breakwater. A breakwater with $1 : 4$ or $1 : 2$ slope was built at the right end of the flume. This breakwater, which was 0.5 m high and 0.16 m wide on the top, was fronted by a $1 : 50$ inclined floor. To measure the free surface elevation of the overtopping water, a series of wave gages was installed on top of and behind the breakwater. The first gage, WG-13, was located 0.015 m behind the leading edge (A), the second, WG-14, and third, WG-15, gages were installed 0.055 and 0.11 m from the first gage respectively. Depending on whether the first or second depth was used, the fourth gage, WG-16,

was located 0.72 or 1.1 m behind the back edge (B) of the breakwater. The last gage was fixed 0.44 m behind the back toe (C) of the breakwater. The experiment was conducted for several solitary wave heights and water depths as given in Table 3.

The simulation setup is very similar to the one used in the BEB simulation where the uniform Boussinesq grid was coupled with the RANS nonuniform x- and uniform y-grids. The interface divides the domain into two segments, x_B and x_R, with ratio $x_B/x_R \approx 9$. A dynamic time step is also employed in the simulation. In all tests the initial location of the solitary wave was 10 m from the left boundary of the Boussinesq domain.

The time series of the free surface elevation of the hybrid simulations and the experimental data are presented in Figure 20 to 26. For comparison the same data from Dodd (1999) (called OTT) are also given in the same figures. In all the simulations, the hybrid wave model shows a clear bias towards overpredicting the water elevation on top of the structure. This is consistent with the OTT simulations, which show an even larger bias. On the lee side of the breakwater, the hybrid simulations in general agree quite well with the data, with a remarked improvement over the shallow water equation-based OTT.

The relative speedup is also measured for this case. The discretization of the pure-RANS domain was almost the same as in the previous BEB discretization. The only difference is that to save computational time in the pure-RANS, the domain offshore of the interface location of the hybrid model is discretized nonuniformly, with an increasingly coarse grid in the deeper water. The computation times of both the hybrid and pure-RANS simulations are summarized in Table 4. As with the speedups in the BEB tests, the reduction in CPU time shows a factor greater than the decrease in RANS domain size. This, again, is due to the Poisson solver requiring fewer iterations to converge with a smaller matrix size.

Table 3. Wave height and water depth of the HR solitary wave overtopping.

Test	H (m)	h_1 (m)	h_2 (m)	WG-16[†] (m)
4c7a	0.07	0.7	0.3	0.72
4c7b	0.10	0.7	0.3	0.72
4c7c	0.12	0.7	0.3	0.72
4c6a	0.07	0.6	0.2	1.10
4c6b	0.10	0.6	0.2	1.10
4c6c	0.12	0.6	0.2	1.10
4c6d	0.15	0.6	0.2	1.10

[†] Distance from the back edge B in Figure 3.

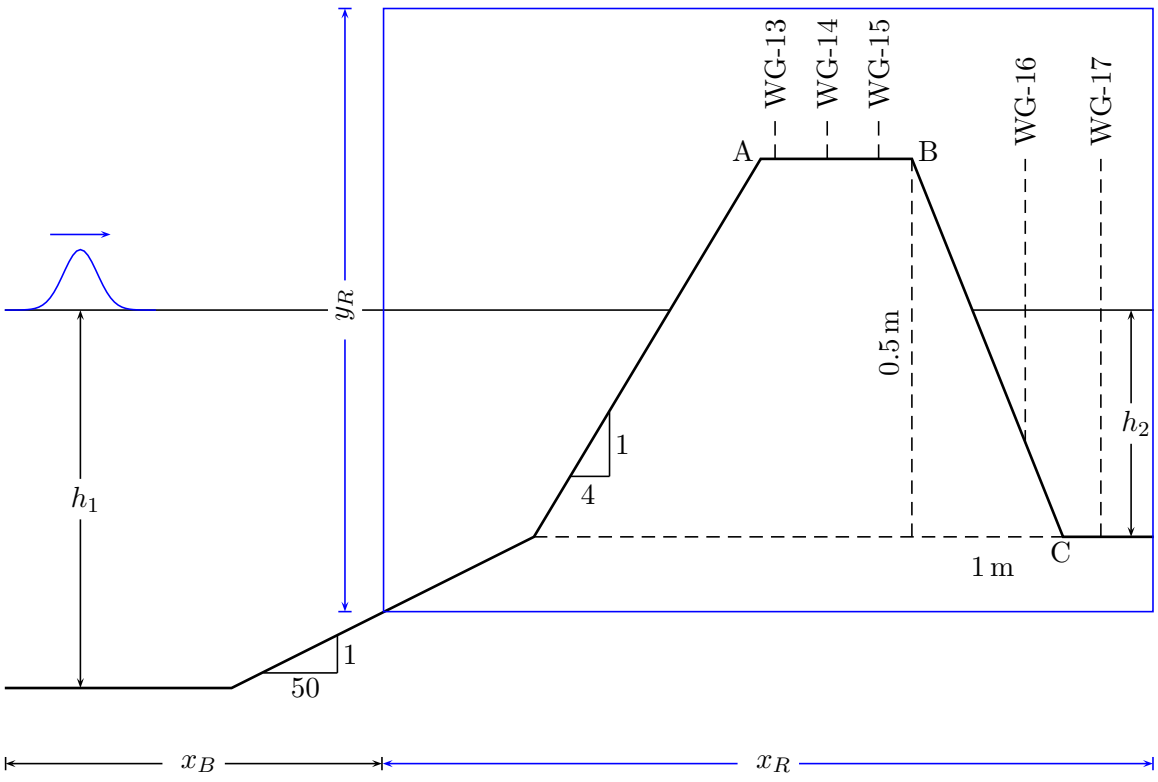


Figure 19. Experimental setup of the HR solitary wave overtopping.

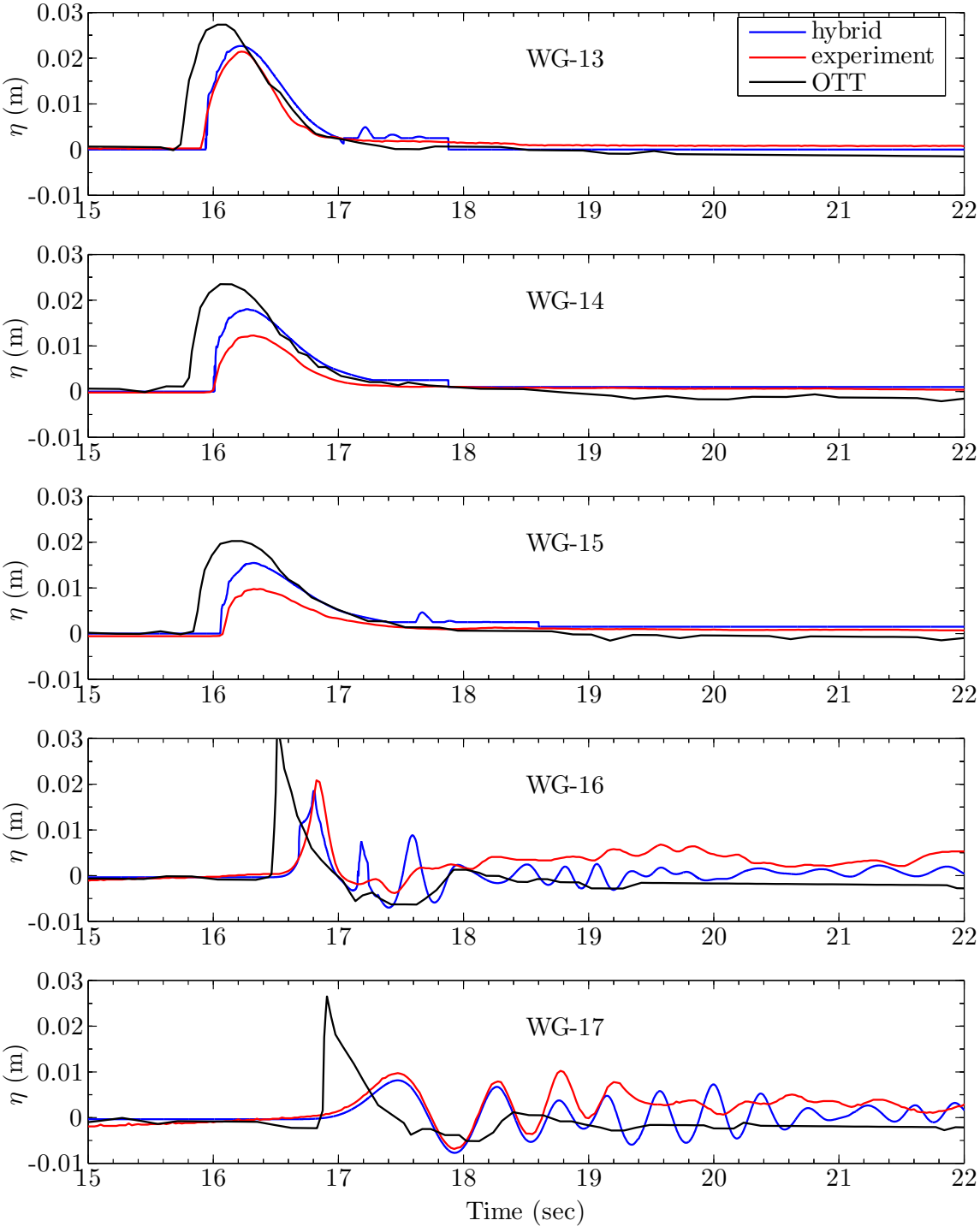


Figure 20. Time series of the free surface elevation for solitary wave overtopping, simulation-4c7a.

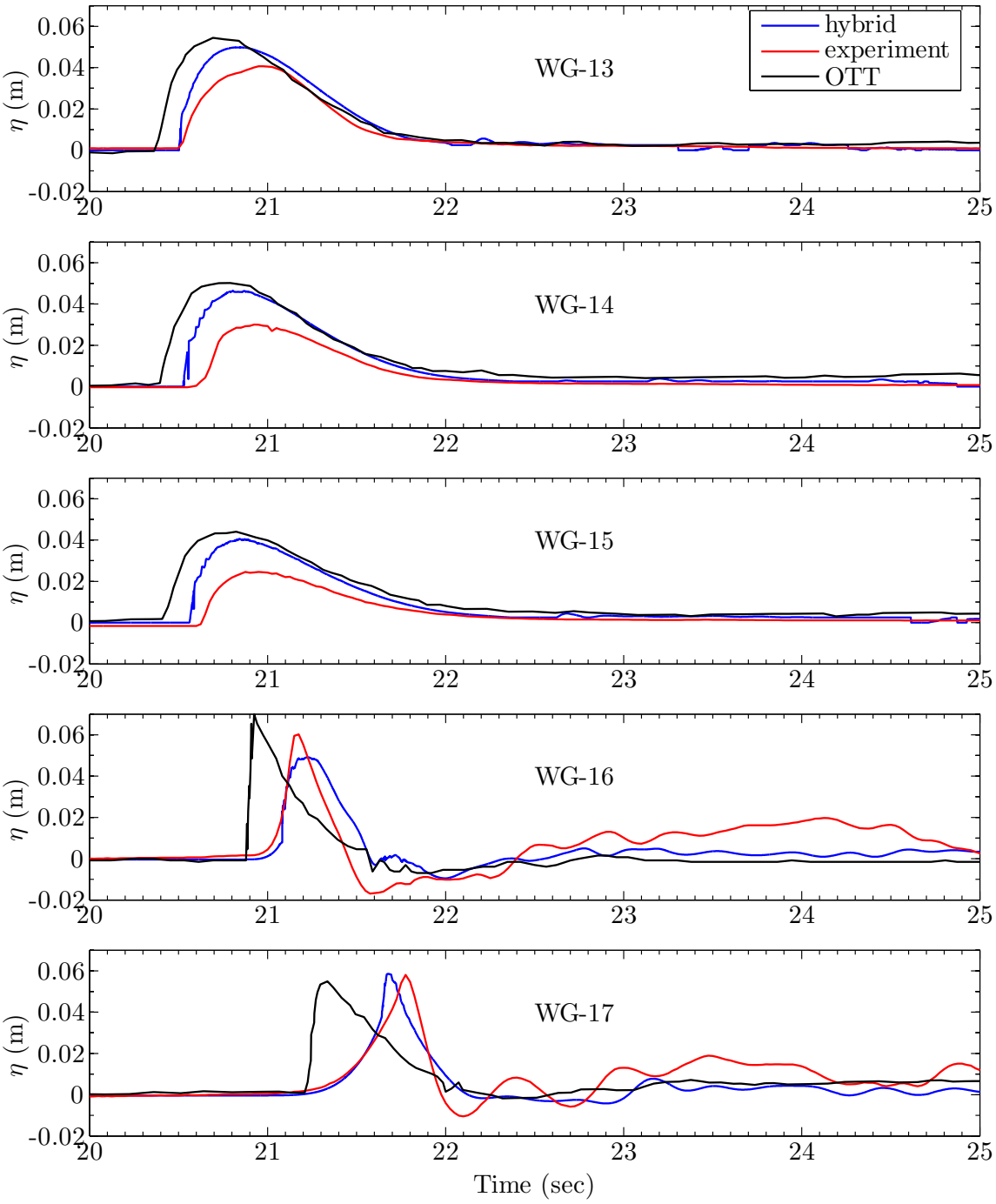


Figure 21. Time series of the free surface elevation for solitary wave overtopping, simulation-4c7b.

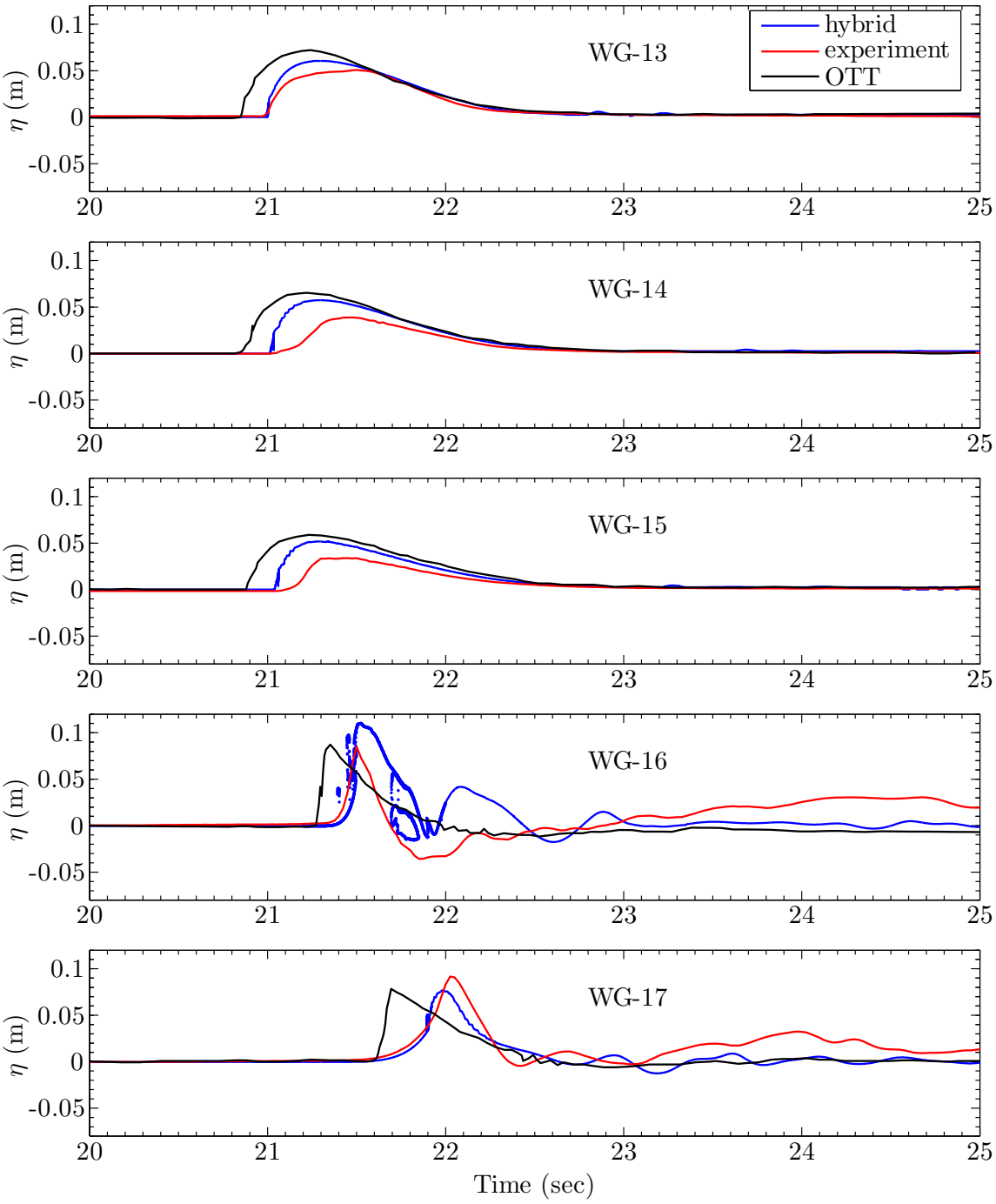


Figure 22. Time series of the free surface elevation for solitary wave overtopping, simulation-4c7c.

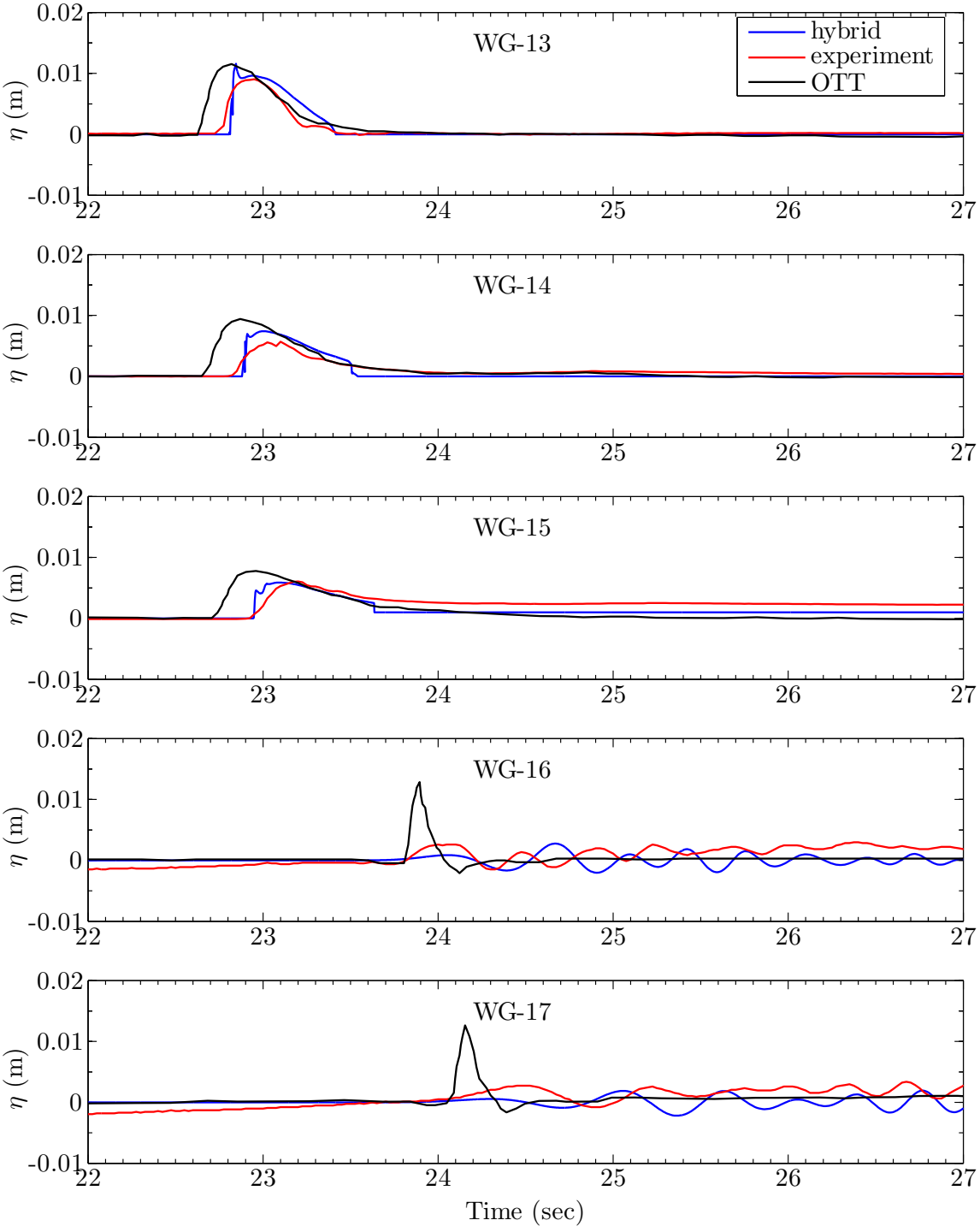


Figure 23. Time series of the free surface elevation for solitary wave overtopping, simulation-4c6a.

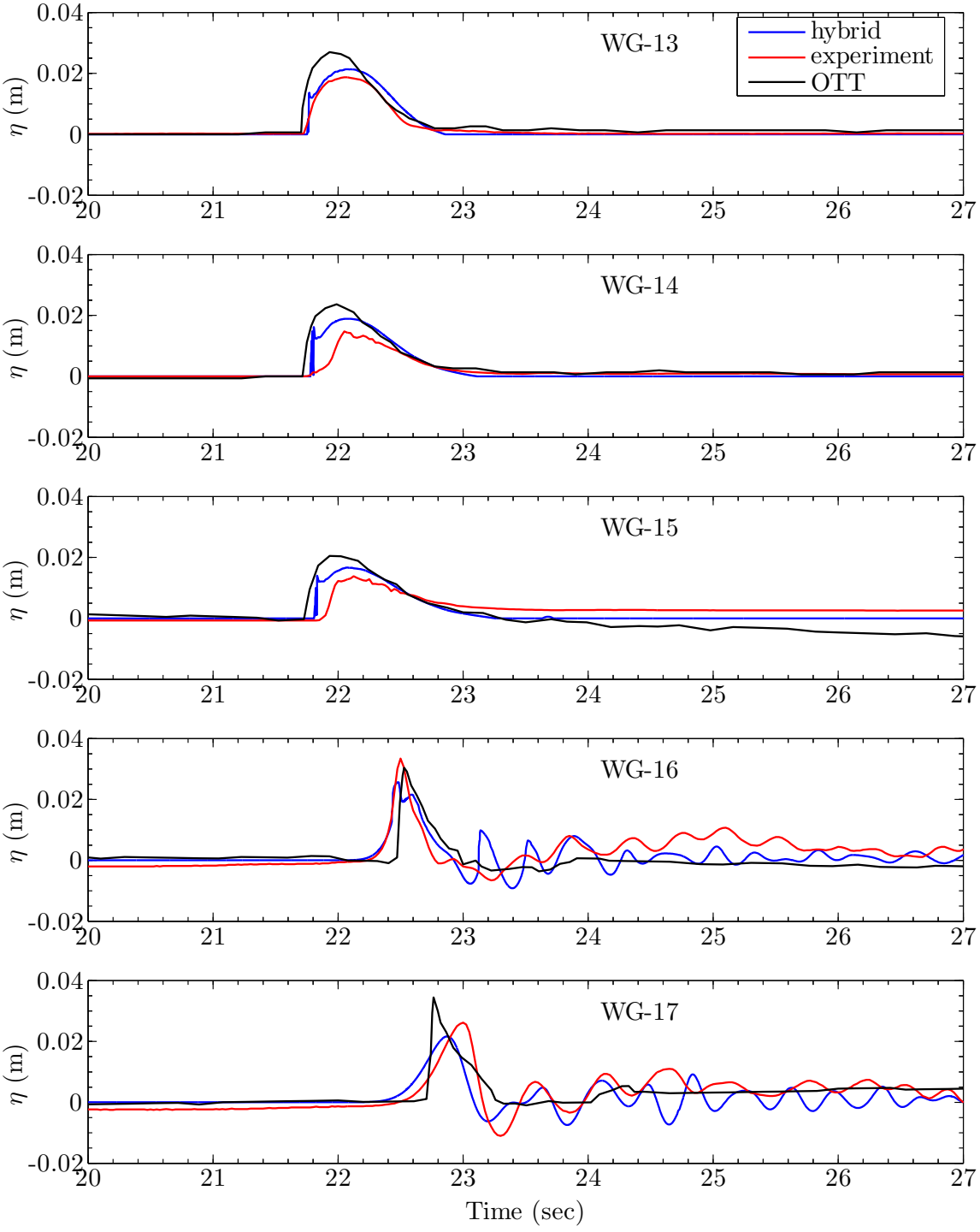


Figure 24. Time series of the free surface elevation for solitary wave overtopping, simulation-4c6b.

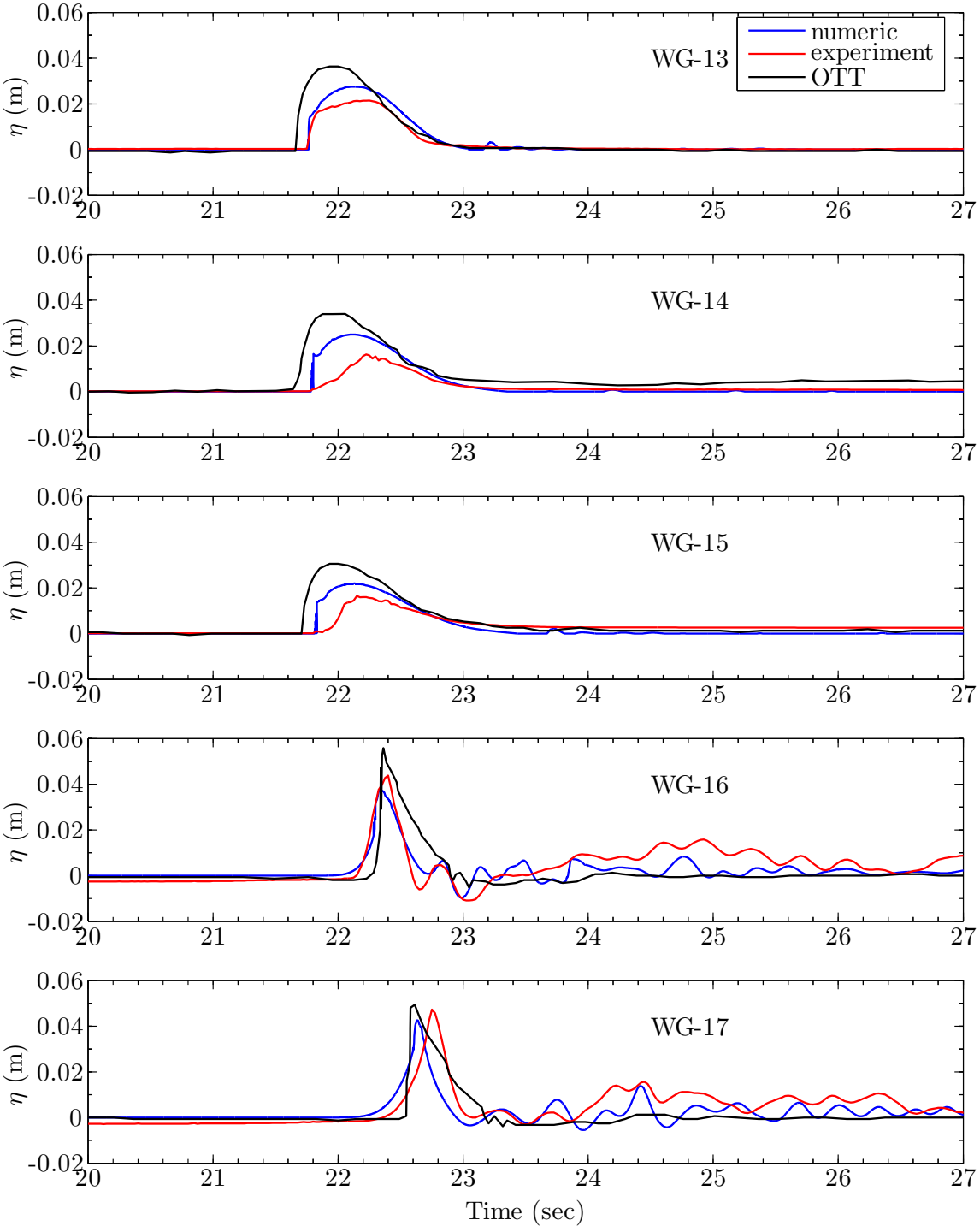


Figure 25. Time series of the free surface elevation for solitary wave overtopping, simulation-4c6c.

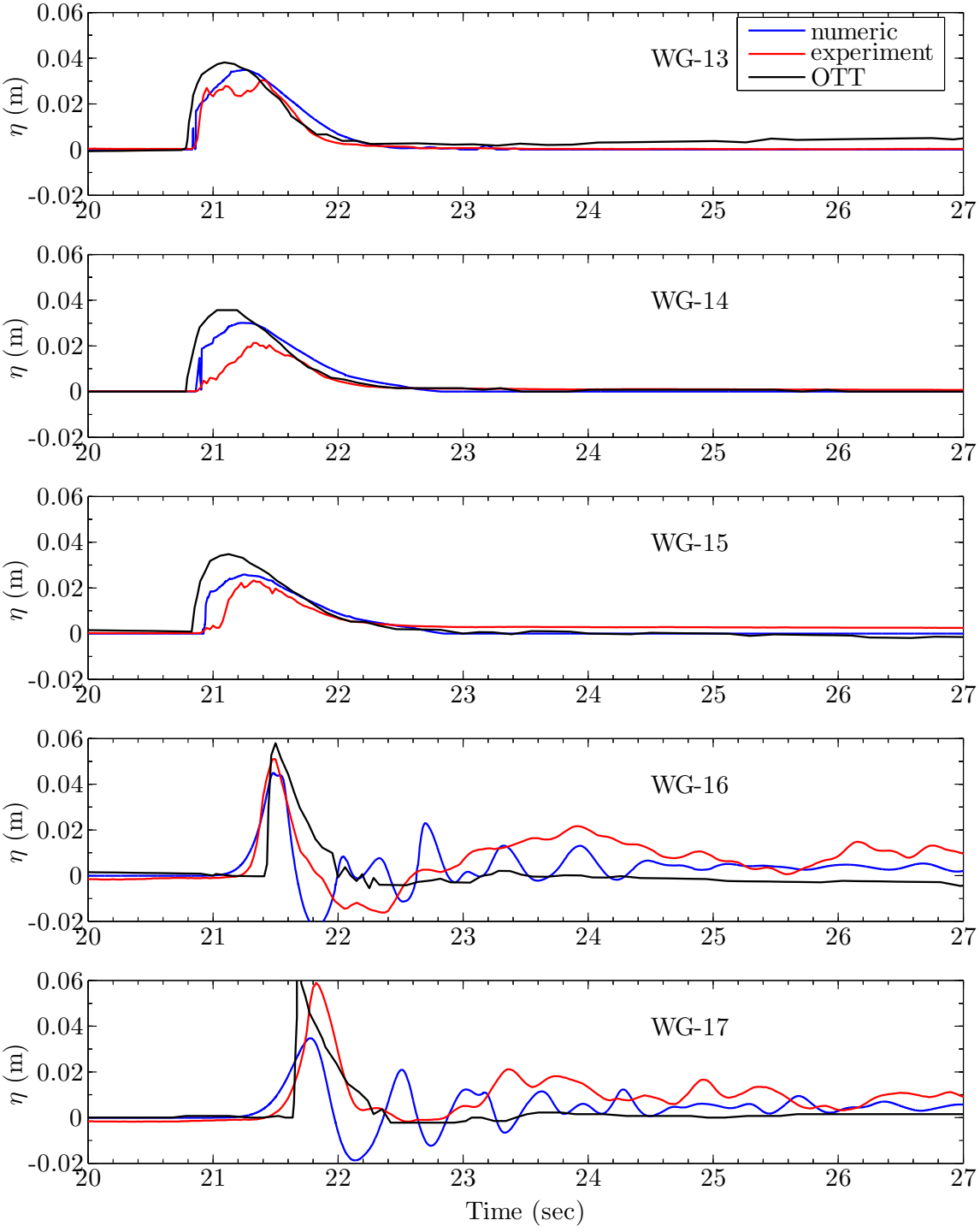


Figure 26. Time series of the free surface elevation for solitary wave overtopping, simulation-4c6d.

Table 4. Computation time of HR solitary wave overtopping.

Test	Hybrid		RANS			
	nx × ny (A_H)	t (sec)	nx × ny (A_H)	t (sec)	$\frac{A_R}{A_H}$	$\frac{t_R}{t_H}$
4c7a	303 × 122	746	610 × 202	3605	3.3	4.8
4c7b	419 × 122	1613	765 × 202	7580	3.0	4.7
4c7c	461 × 122	2494	862 × 202	9930	3.1	4.0
4c6a	395 × 142	2236	746 × 202	9134	2.7	4.1
4c6b	435 × 122	2367	821 × 202	12303	3.1	5.2
4c6c	435 × 122	2518	821 × 202	15126	3.1	6.0
4c6d	435 × 122	2573	821 × 202	10951	3.1	4.3

Wave Propagation Over Porous Structure

In this section, we present the employment of the hybrid model to simulate the wave propagation over a porous structure. The setup of this simulation is based on the data from a laboratory experiment conducted in the wave and current flume of the Coastal Laboratory of the University of Cantabaria as a part of the research conducted for the European Project DELOS "Environtmental DEsign of LOw Crested Coastal Defence Structures."

The flume is 24 m long, 0.6 m wide, and 0.8 m high. The wave is generated using a piston type wavemaker, integrated in the Active Wave Absorption System (AWACS) which allows the absorption of the reflected wave in the model. The experiment setup is shown in Figure 27. Although the length of the flume is 24 m, the domain length for the simulation was 17.4 m only. The rest of the domain is occupied by the absorbing

beach and false bottom that can partially or totally removed to set off a current in the flume. Built in the simulation domain was a porous breakwater with dimension given in the figure. The breakwater was composed of the 1 : 2 slope outer armor and core layers whose characteristics are given in Table 5. From Figure 28, the thickness of the outer armor layer is visually justified to be 0.083 m.

The breakwater was sitting on a 3.8 m horizontal floor connected with the front and rear 1 : 20 slopes. A total of 11 wave gages, numbered 1, 2, ..., 11 in the figure, were installed in the breakwater vicinity to measure the free surface elevations. These gages were horizontally located 1.13, 1.28, 1.5, 2.0, 2.25, 3.137, 3.602, 4.545, 4.985, 5.135, 5.335 m from bottom corner A. Along the bottom of the breakwater are three pressure gages, numbered 16, 17, and 18 and located horizontally 2.5, 3.0, 3.5 m from corner A, for pressure measurement. For this experiment, the flume was filled with water to a depth of 0.4 m, and the sinusoidal wave, $H = 0.035$ m and $T = 0.6$ s, was generated using the piston type wavemaker.

Table 5. Characteristics of the armor and core layers of the porous breakwater.

Layer	W_{50} (gr)	D_{50} (cm)	Porosity	γ (kg/m^3)
Armor	153	3.94	0.53	2,647
Core	4.31	1.18	0.49	2,607

For the numerical simulation, the domain was divided into the Boussinesq and RANS domains. The Boussinesq domain stretched from the wavemaker ($x = 0$ m) to near corner A, and was discretized into uniform $\Delta x = 0.03$ m grids. The RANS domain occupied the rest of the domain, and was discretized into 430 uniform horizontal $\delta x = 0.03$ m grids and 112 uniform vertical $\delta y = 0.005$ m grids. The simulation was

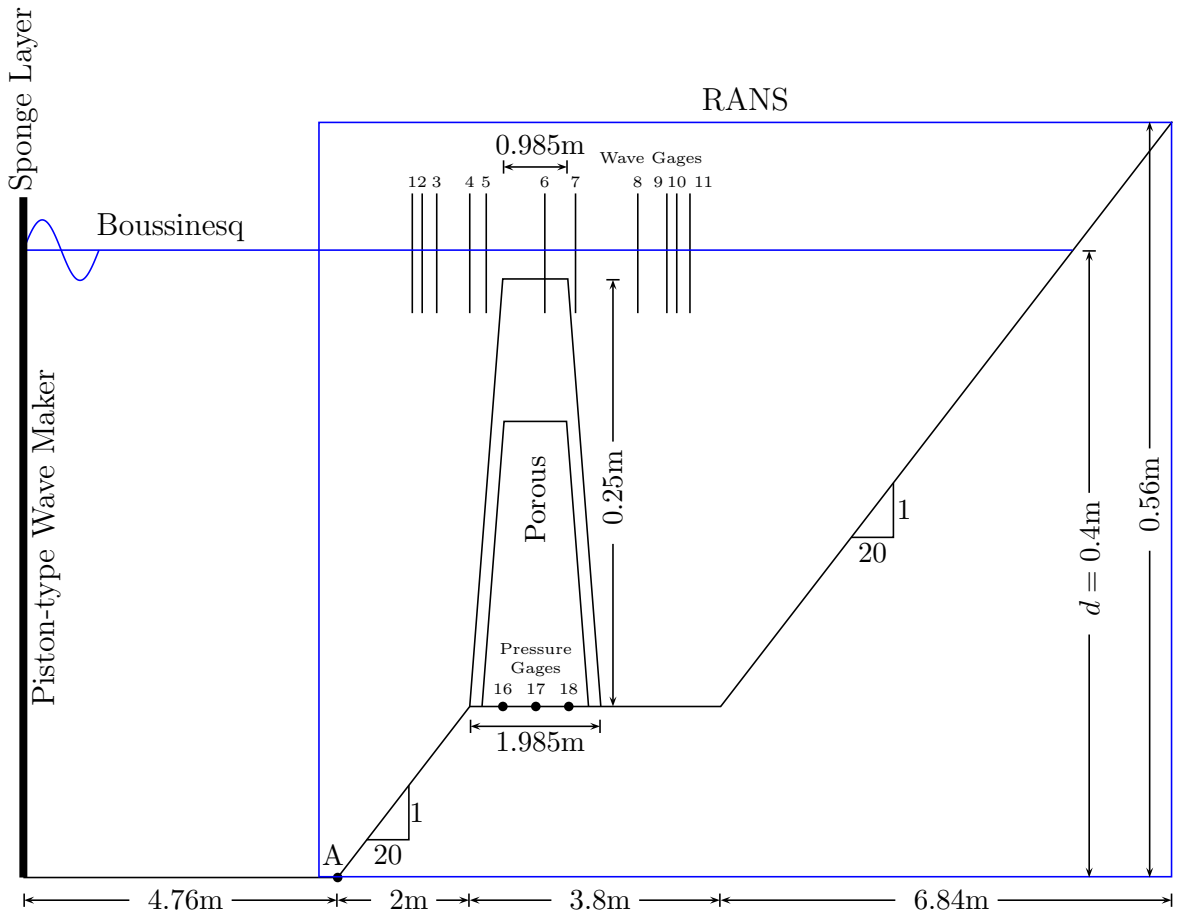


Figure 27. Experiment/numerical simulation setup of the wave propagation over porous structure.

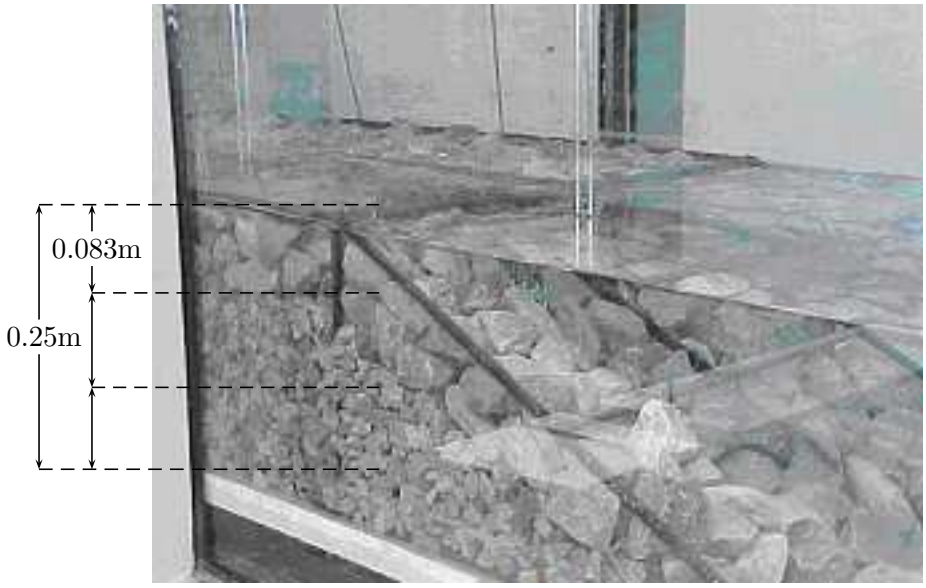


Figure 28. Porous breakwater layers. Adapted from Vidal and Losada (2007).

run for 100 s using the dynamic time step. Referring to Garcia et al. (2004), we used $\alpha = 1000$ for both the armor and core layers, $\beta = 0.8$ for the armor and 1.2 for the core layer.

For time benchmark, we also simulate this scenario using the RANS model alone. The domain size for the pure-RANS simulation was 17.4 m long and 0.56 m high. The domain was discretized as in the hybrid RANS model, resulting in 580 horizontal grids and 112 vertical grids. The scenario was run for 100 s, and took 88 s wall clock time per wave period to complete the simulation cycles. The hybrid model spends 50 s clock time for this simulation. Hence, we gain 1.76 time speedup over 1.34 $\frac{\text{HYBRID}}{\text{RANS}}$ grid ratio, a 131% efficiency, in employing the hybrid model.

Figure 29 depicts the data and the calculated free surface elevation time series at wave gages 1–11. The calculation agrees very well with the data for gages l–5, in front of the structure. Due to the wave-structure interaction, the higher harmonics

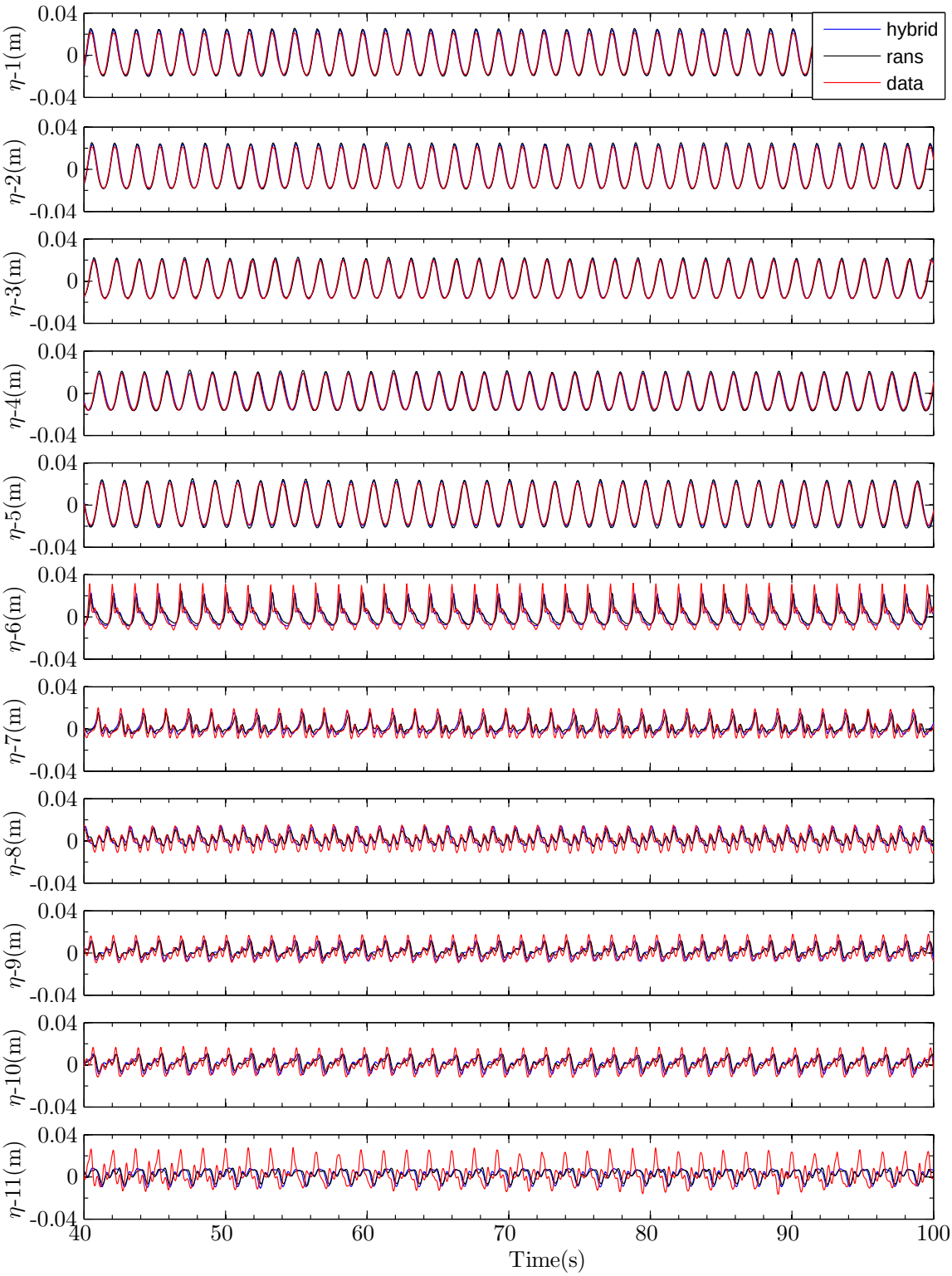


Figure 29. Free surface elevation at gages 1–11.

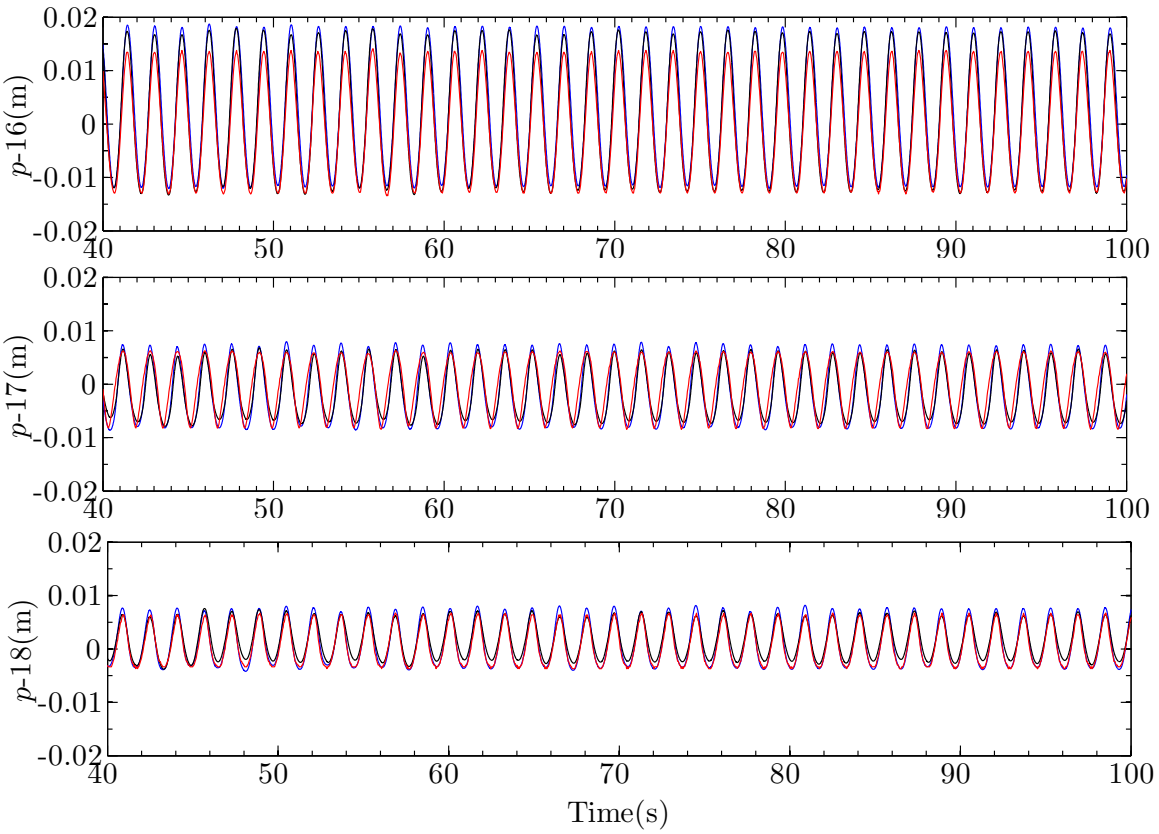


Figure 30. Pressure at stations 16–18.

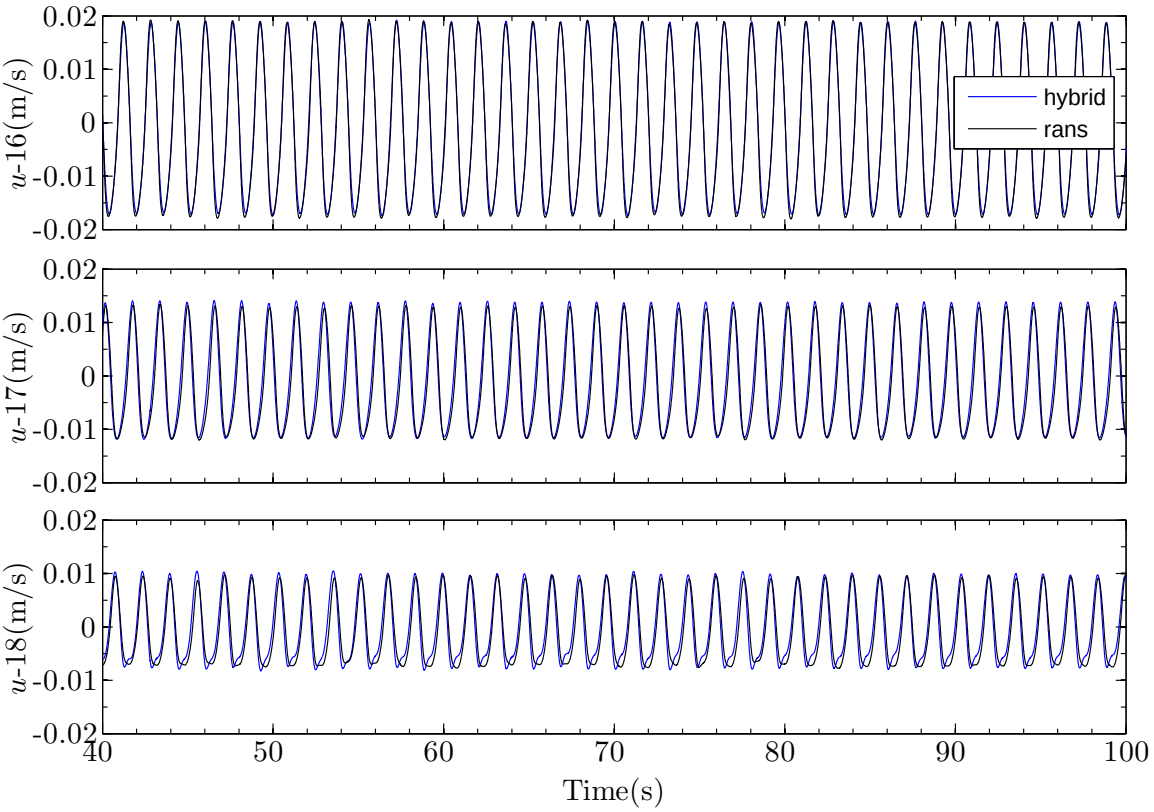


Figure 31. Horizontal velocity at gages 16–18.

are created when the wave passes the structure. Both the hybrid and pure-RANS simulation underpredict the higher harmonics. As the wave moves farther, the higher harmonics dissipates. Similar observation was also reported in Garcia et al. (2004) in their numerical simulation of the near-field flow employing RANS model and the same setup with higher wave height of 0.07 m. Comparison is also made for the pressure along the bottom of the structure at gages 16, 17, and 18 as shown in Figure 30. Both the hybrid and pure-RANS simulations agree quite well with the data. Due to the data unavailability, similar comparison cannot be made for the velocity; however, model-to-model comparison, shown in Figure 31, is made, and shows that both the hybrid and pure-RANS simulations are in very good agreement.

Hypothetical Tsunami Simulation

In many tsunami events, such as the devastating tsunami which struck the West Sumatera Coast in 2004, the wave is generated by a sudden uplift or subsidence of the seafloor following a massive tectonic earthquake. The vertical displacement of the seafloor disturbs the equilibrium of the water column above it and in consequence the the water mass spreads as a long wave to attain a new gravitational equilibrium.

In this, the last model application, the hybrid wave model is used to simulate the propagation of a tsunami-type wave from the open ocean to the coast. The simulation setup is given in Figure 32. The domain consists of a 1 km deep ocean connected with a $1 : 50$ seafloor. A 4 m elevation breakwater was placed along the coast in a shallow water depth of approximately 3 m. For the hybrid simulation, the Boussinesq model occupied 93% of the total horizontal domain length; the RANS model occupies only 7%, which was located in the nearshore region. Both the Boussinesq and the RANS domains were uniformly discretized into $\Delta x = 15$ m grids for the Boussinesq

and $\delta x = 4$ m and $\delta y = 0.5$ m grids for the RANS. The simulation was run with a dynamic time step for 1,500 s of simulation time. The wave was generated 100 km offshore by uplifting the sea surface to form a gaussian-shape wave with zero initial velocity. The generated wave had an offshore wavelength of roughly 10 km, and thus might represent the leading wave of a leading elevation tsunami.

Figure 33 shows snapshots of the wave at three different times. At $t = 1,070$ s the wave has just reached the detached breakwater with a turbulent wave front that appears as an 18 m high wall of water moving at a speed of nearly 10 m/s. The second snapshot in Figure 34 gives the wave at $t = 1,140$ s. Just 70 s after the wave reaches the breakwater, the coast is flooded up to 1.6 km inland. The average flow depth in the flooded area is 13 m, with a 16 m/s average speed. The bottom part of the figure shows the detail of the flow around the breakwater. At the upstream side the flow moves 6 m/s and due to the breakwater acting as a sill, the velocity increases nearly three times to 17 m/s at the leeward side of the breakwater. Also note the regions of separation at the breakwater corners, indicated by a relatively low fluid speed. Figure 35 shows the tsunami at maximum run up. At this stage, about 4 minutes after the wave front first reaches the breakwater, the water has inundated 2.3 km of the coastal area.

For this model test, a time benchmark is not done as in the previous model tests. With current computing limitations, it is not practically possible to run the whole simulation with the RANS model without some type of parallel implementation of the model. Thus, this example demonstrates the potential of the hybrid model to routinely tackle multi-scale problems which would otherwise require extensive, expensive, and sophisticated computational capacity.

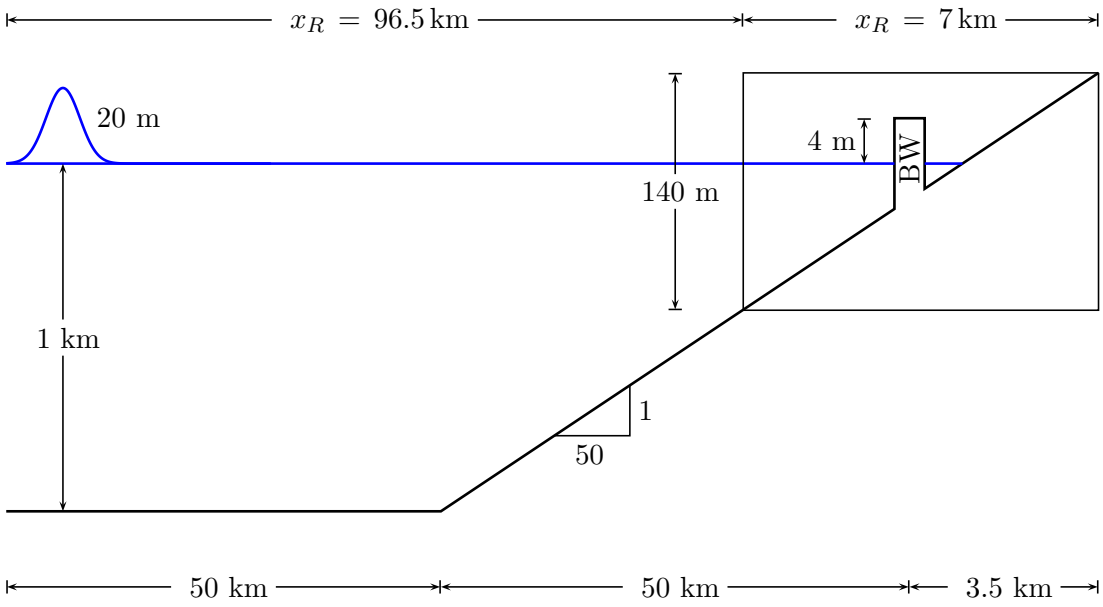


Figure 32. Simulation setup of the hypothetical tsunami generation and propagation; figure is not scaled.

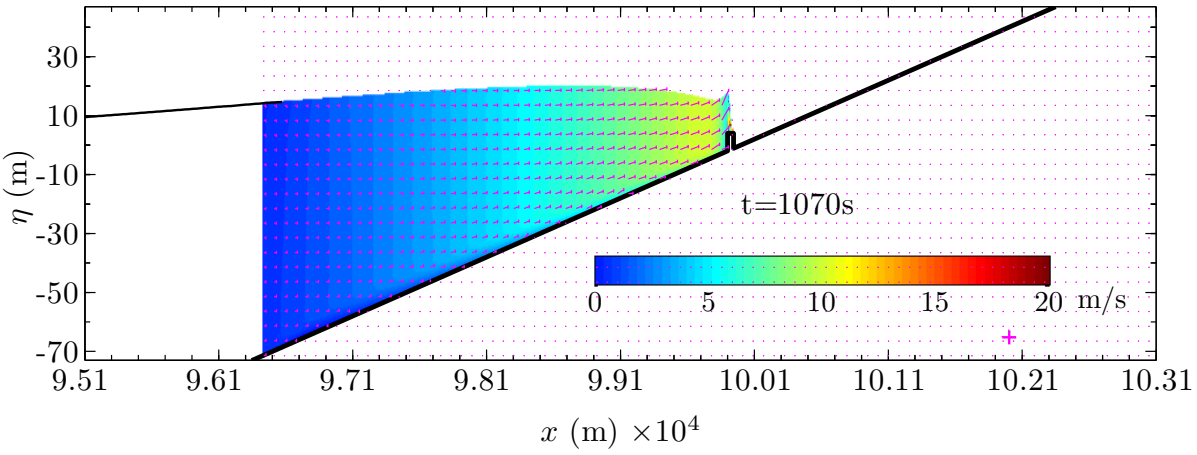


Figure 33. Snapshot of tsunami wave reaching the breakwater.

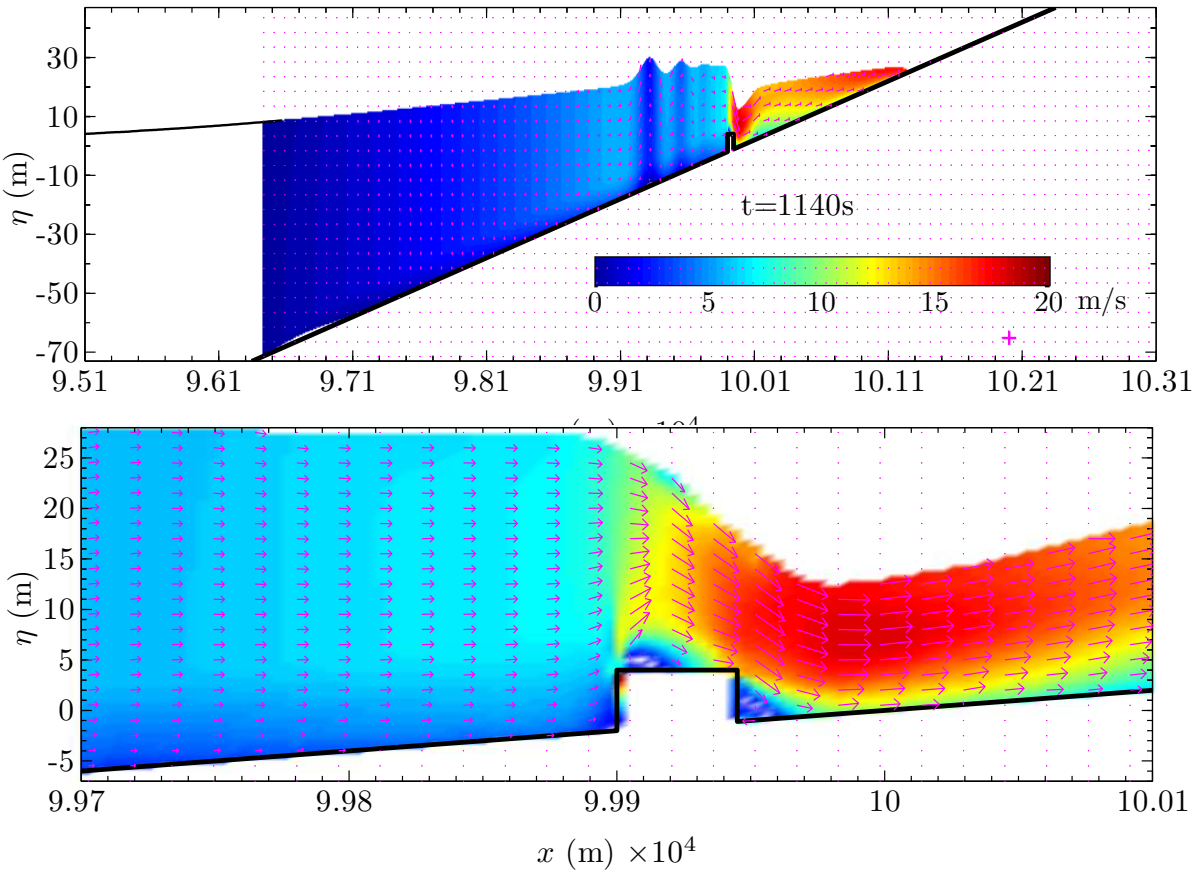


Figure 34. (Top) Snapshot of tsunami wave passing the detached breakwater (Bottom) Close-up look of velocity near the breakwater.

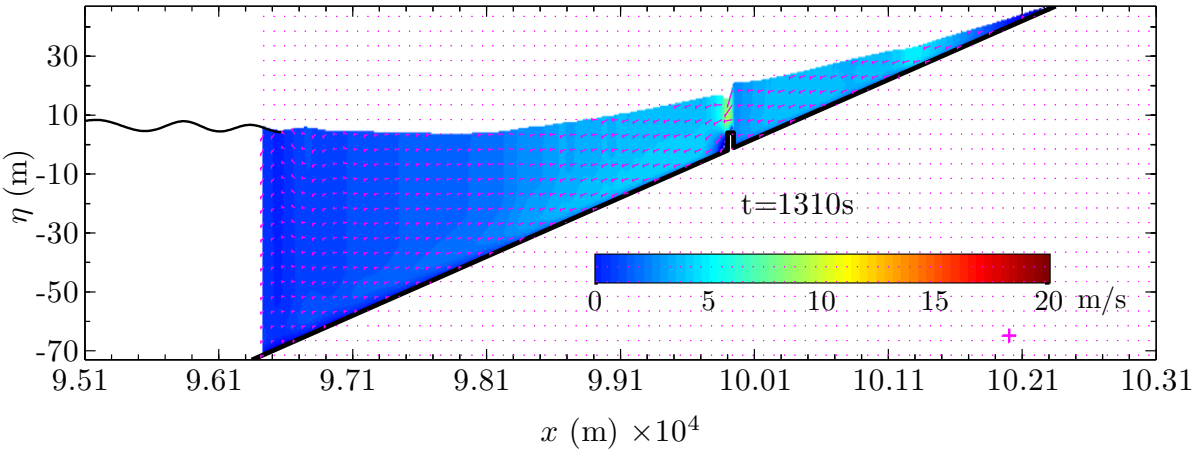


Figure 35. Snapshot of tsunami wave inundating the coast at time of maximum runup.

CHAPTER V
PARALLEL HYBRID WAVE MODEL

Introduction

The hybrid wave model developed herein is intended for wave simulation which employs relatively coarse grid system for offshore domain and fine grid system nearshore. For breaking wave simulation, very fine grids of a few centimeters in size must be used to discretize the nearshore domain. Depending on the problem at hand, the nearshore domain may stretch an area from the shoreline to a point a few hundred meters offshore, beyond the breaking point. With such a fine grid resolution, hundreds or thousands of grids should be employed in both the x- and z-directions, and in consequence a large system of linear equations which arises from the finite difference formulation of the PPE equation must be solved at each time step. In such a case, the hybrid wave model computation is expensive and not feasible to carry out in one computer.

To allow for the large scale wave simulation, we parallelize the current hybrid wave model to run on a cluster with distributed memory system. In this chapter, we explain the parallelization of the current hybrid wave model. In parallelizing the model, the whole job is distributed into several computers connected with a network system. In distributing the job, each computer carries the same amount of computational load. All the computers are running simultaneously and communicating pertinent data with other computers. Depending on the number of computers used, the parallel algorithm, and the architecture of the system, the computational time can be reduced to smaller time. Since the load is distributed into several computers, the memory used in each computer is approximately $1/P$ (P is the number of computers

used) of the memory requirement in the serial run. Therefore, the problem with the huge memory requirement hampering the serial hybrid simulation can be resolved by parallelizing the hybrid model.

Parallelization Strategy

In a typical hybrid wave simulation, the Boussinesq model uses only a fraction of the total computational time. Most of the computational time is used by the RANS model for solving the linear system of equations arising from the finite difference solution of the pressure Poisson equation. Therefore, in parallelizing the hybrid wave model, only the RANS model is parallelized, while the Boussinesq model remains serial, as shown shown in Figure 36. This figure shows that the system consists of P processors. Here, the RANS domain is distributed evenly to all the processors involved in the calculation, while the Boussinesq domain is assigned to processor-1 only.

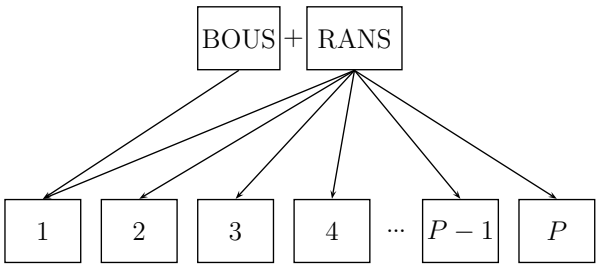


Figure 36. Load distribution in parallel hybrid wave computation. P is the number of processors involved in the calculation.

In such an implementation as shown in Figure 36, processor-1 carries more load than the other processors. In general, for each time step, processors 2 to P complete the calculation earlier than processor-1. Therefore, these processors must wait for processor-1 to complete its calculation before the computation can proceed to the

next time level.

Different from the serial hybrid model, in the parallel hybrid model the interface is located in the processor where the Boussinesq model is assigned to. In the case exemplified in the figure, this interface is located in processor-1. Accordingly, the Boussinesq model exchange the data with the RANS subdomain residing in the processor-1 only.

The algorithm to parallelize the hybrid wave model is very similar to the serial hybrid algorithm. Here, we assume that there are P processors available for the parallel computation.

1: **while** $t^n < t_{\text{end}}$ **do**

2: Calculate in parallel the RANS provisional velocities, $\tilde{u}^{n+1}$ and $\tilde{v}^{n+1}$, from (2.23), using Boussinesq boundary values from time level n.

3: Calculate, iteratively, the RANS fluid pressure, p^{n+1}, from the parallel version of (2.25).

4: Calculate in parallel the RANS final velocities, u^{n+1} and v^{n+1}, from (2.24).

5: In processor-1, calculate the RANS turbulence intensity and dissipation, k^{n+1} and ϵ^{n+1}, using Boussinesq boundary values from time level n.

6: In processor-1, calculate the RANS VOF function, F^{n+1}, using Boussinesq boundary values from time level n. This happens in processor-1 only.

7: In processor-1, calculate η^{n+1} and u_α^{n+1} from the Boussinesq predictor (2.3) and (2.4), using RANS boundary values from time level n.

8: In processor-1, calculate, iteratively, η^{n+1} and u_α^{n+1} from the Boussinesq corrector (2.5) and (2.6), using RANS boundary values at time level n+1. At this point, all fluid variables in both models have completed calculations for time level n+1.

9: In parallel, change δt if flow in RANS exceeds Courant stability constraints;

dynamic time-stepping.

10: **if** $\delta t_{\text{new}} \neq \delta t_{\text{old}}$ **then**

11: In processor-1, interpolate the new η^{n-1}, η^{n-2}, u_α^n, u_α^{n-1}, u_α^{n-2} at all nodes on the Boussinesq grid.

12: **end if**

13: $t^{n+1} = t^n + \delta t_{\text{new}}$

14: $n = n + 1$

15: **end while**

Prior to the calculation, the RANS domain is decomposed and distributed evenly to all the processors. The decomposition is done in the x-direction only (see Figure 37). For instance, if the original domain consists of 1000 grids in the x-direction and 100 grids in the z-direction, with $P = 4$, each subdomain will have $1000/4 + 2 = 252$ and 100 grids in the x and z directions respectively. The 2 additional (ghost) grids are added for receiving the data from the neighboring processors. Note that, we do not decompose the domain in the z-direction. Hence, the data exchange between the processors occurs in the x-direction only.

In step 2–4 and 9, the computations are in parallel, where all the processors perform the same operations. After the completion of the parallel operations, the pertinent variables are exchanged between the neighboring processors.

In the next sections, three important aspects of the parallel hybrid algorithm will be discussed. The three aspects are:

1. Parallelization of the loop of arithmetic computation.

2. Communication.

3. Parallelization of the preconditioned Conjugate Gradient solver.

Parallelization of the Loop of Arithmetic Computation

Although the RANS model is 2-D, all the major arrays in the model such as velocity, pressure, volume of fluid, etc. are stored in one dimensional arrays. The array is indexed row-wise, with the indices increasing from the left to the right in each row. This indexing is shown in Figure 37. This figure shows that the original domain consisting of 4 rows and 20 columns, a total of 80 cells, is divided into 3 subdomains, each consists of 4 rows and 8 columns, a total of 32 cells. Here, for instance, the pressure array is indexed as $p\,(1)$, $p\,(2)$, ..., $p\,(80)$ in the original domain, and $p\,(1)$, $p\,(2)$, ..., $p\,(32)$ in each subdomain. Note that the array elements $p(2)$, $p(10)$, $p(18)$, $p(26)$ in processors 2 and 3 contain the same values as the elements $p(8)$, $p(16)$, $p(24)$, $p(32)$ in the processors 1 and 2 respectively. Similarly, the elements $p(1)$, $p(9)$, $p(17)$, $p(25)$ in processors 2 and 3 are identical to elements $p(7)$, $p(15)$, $p(23)$, $p(31)$ in processor-1 and 2 respectively.

With such a decomposition and array indices in the original code, a typical serial loop in the code:

```
ij=1
do  j=1,4
   do  i=1,20
      p(ij)=f(a,b,c,...)
      q(ij)=g(a,b,c,...)
         .
         .
         .
      ij=ij+1
   enddo
enddo
```

is decomposed into several parallel loops in all the computers:

```
ij=1
do  j=1,4
   do  i=1,8
```

61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60
21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

(a)

Proc-1

25	26	27	28	29	30	31	32
17	18	19	20	21	22	23	24
9	10	11	12	13	14	15	16
1	2	3	4	5	6	7	8

Proc-2

25	26	27	28	29	30	31	32
17	18	19	20	21	22	23	24
9	10	11	12	13	14	15	16
1	2	3	4	5	6	7	8

Proc-3

25	26	27	28	29	30	31	32
17	18	19	20	21	22	23	24
9	10	11	12	13	14	15	16
1	2	3	4	5	6	7	8

(b)

Figure 37. 1-D array indices in the (a) serial RANS model (b) parallel RANS model. Gray area is ghost cells.

```
        p(ij)=f(a,b,c,...)
        q(ij)=g(a,b,c,...)
           .
           .
           .
        ij=ij+1
      enddo
    enddo
```

Once the loop is completed, the arrays to which the new values are assigned must exchange their pertinent elements with other arrays from the neighboring processors. This will be explained in the next section.

Communication

Due to the distributed memory used in the cluster, each processor must communicate the data with other processors during the simulation run. In the current parallel RANS model, there are two types of communication:

1. Communication with adjacent processors.
2. Communication with all processors.
3. Communication with *distant* processors.

The first communication is done after any operation which affects the variables in the left/right boundary cells. The communication copies the data from these cells into the right/left ghost cells in the adjacent processors. In processor-1, 2, and 3 of Figure 37, for example, the right boundary cells are cells 7, 15, 23, and 31 and the left boundary cells are cells 2, 10, 18, and 26. The affected variables in the right boundary cells of processor-1 and 2 are copied into the left ghost cells 1, 9, 17, and 25 in processor-2 and 3. The communication is done using the nonblocking

communication, with a receiving-process is posted first, followed by the sending-process. The left-to-right communication, for example, executes the following code snippet, to pass the variable p from the left to right processors:

```
      if(idproc > 0)then
        call mpi_irecv(p(1),4,mpi_doubleprecision,
     &  left,10,mpi_comm_world,request,error)
        call mpi_wait(request,status,err)
      endif

      if(idproc < 2)then
        call mpi_send(p(1),4,mpi_doubleprecision,
     &  right,10,mpi_comm_world,err)
      endif
```

and the following to pass the variable from the right to left processors:

```
      if(idproc < 2)then
        call mpi_irecv(p(1),4,mpi_doubleprecision,
     &  right,10,mpi_comm_world,request,error)
        call mpi_wait(request,status,err)
      endif

      if(idproc > 0)then
        call mpi_send(p(1),4,mpi_doubleprecision,
     &  left,10,mpi_comm_world,err)
      endif
```

where `idproc` is the processor's ID (0 for processor-1, 1 for processor-2, and 2 for processor-3). Most of the loops in the RANS model contain more than one array to be processed and passed to adjacent processors. To avoid the unnecessary huge latency time with multiple posts of `mpi_send`, these arrays are first concatenated into a single array which is then passed to adjacent processors. This way, only one latency time is required to pass all the arrays. After completing the send-receive process, each processor distributes the content of the array into the original arrays.

The second communication occurs when each processor requires some variable from all the processors. For instance, in calculating the inner product of two arrays

(vectors), the sum of the element-wise products of the array elements is calculated. Since each processor has only some portion of those products, it must get the rest from the other processors. This process, which is done through the call to `mpi_allreduce`, requires the communication with all the processors.

The third communication passes the data from one processor to others which are not the adjacent processors. For instance, if there are 5 processors involved in the parallel calculation, processor-1 may need to pass the data to processor-3, 4, or 5. This type of communication is specifically invoked in the the parallel conjugate gradient solver, which is used to solve the PPE equation. This will be discussed in detail in the next section.

Parallelization of the Preconditioned Conjugate Gradient Solver

The discretization of the PPE equation, Kothe et al. (1994), results in a system of linear equations of the form:

$$\mathbf{M}\mathbf{p}^{n+1} = \mathbf{S}, \tag{5.1}$$

where

$$\mathbf{p}^{n+1} = \begin{pmatrix} p_1^{n+1} \\ \vdots \\ p_k^{n+1} \end{pmatrix}, \quad \mathbf{S}^{n+1} = \begin{pmatrix} S_1 \\ \vdots \\ S_k \end{pmatrix}, \tag{5.2}$$

$$\mathbf{M} = \begin{pmatrix} A_1 & B_{u1} & \cdots & D_{u1} & & & & \\ B_{l2} & A_2 & B_{u2} & & D_{u2} & & & \\ \vdots & B_{l3} & \ddots & \ddots & & \ddots & & \\ D_{l\mathrm{IBAR}} & & \ddots & & & & D_{u\mathrm{IBAR}} & \\ & \ddots & & & & & \vdots & \\ & & & & & & B_{uN-1} & \\ & & & D_{lN} & \cdots & B_{lN} & A_N & \end{pmatrix}. \tag{5.3}$$

$\mathbf{p}^{n+1}$ is the pressure at time level $(n+1)$ and $\mathbf{S}$ is the source term which depends on the provisional velocity at time level $n+1$. The matrix $\mathbf{M}$ is a pentadiagonal symmetric positive definite matrix, where IBAR is the number of real cells in the x-direction and subscripts u and l denote the upper and lower diagonals respectively. The size of the system N is the product of the numbers of the real cells in the x- and y-directions. The diagonal $\mathbf{B}_l/\mathbf{B}_u$ are located next to the main diagonal $\mathbf{A}$ and the diagonal $\mathbf{D}_l/\mathbf{D}_u$ are offset IBAR from the diagonal $\mathbf{A}$. For the serial model, where the row-wise indexing is used, p_1^{n+1} corresponds to cell $(2,2)$, p_1^{n+1} to $(2,3)$, $p_{\mathrm{IBAR}+1}^{n+1}$ to $(3,2)$, $\cdots$, and p_N^{n+1} to $(\mathrm{IBAR}+1, \mathrm{JBAR}+1)$. In the parallel RANS model, however, to have a monotonically-increasing indices across the processors, we used column-wise indices for numbering the unknowns. In such a column-wise indexing, the first lower/upper diagonal of the serial (5.1) becomes the second lower/upper diagonal and vice versa. The offset of the second lower/upper diagonal in the parallel system is JBAR from the main diagonal. The unknowns p_1^{n+1} corresponds to cell $(2,2)$, p_2^{n+1} to $(3,2)$, $p_{\mathrm{JBAR}+1}^{n+1}$ to $(2,3)$, $\cdots$, and p_N^{n+1} to $(\mathrm{IBAR}+1, \mathrm{JBAR}+1)$, and similarly for the source $\mathbf{S}^{n+1}$. After (5.1) is solved, the unknowns are mapped into the row-wise indexed pressure array.

In the serial RANS code, the system (5.1) is solved using the CG (Conjugate

Gradient) method, combined with the Incomplete Cholesky decomposition, Kershaw (1978), to accelerate the convergence. This decomposition transforms the matrix $\mathbf{M}$ into $\mathbf{LDL}^{\mathrm{T}} + \mathbf{E}$, where $\mathbf{L}$ is the lower triangular matrix, $\mathbf{D}$ is an approximate identity diagonal matrix, and $\mathbf{E}$ is the error matrix. In decomposing $\mathbf{M}$, the entries of $\mathbf{L}$ are forced to have the sparsity pattern as matrix $\mathbf{M}$. The CG method combined with this type of decomposition is referred to as the ICCG(0) method in Meijerink and Vorst (1977). Using this decomposition, the original system is transformed into an equivalent one:

$$\left[\mathbf{L}^{-1}\mathbf{M}\left(\mathbf{L}^{\mathrm{T}}\right)^{-1}\right]\left(\mathbf{L}^{-1}\mathbf{p}\right) = \left(\mathbf{L}^{-1}\right)\mathbf{S} \tag{5.4}$$

which is then solved using the iterative conjugate gradient as follows (Kershaw (1978) and Saad (2003)):

1: Compute $\mathbf{r}_0 = \mathbf{S} - \mathbf{M}\mathbf{p}_0$, $\mathbf{q}_0 = \left(\mathbf{L}\mathbf{L}^{\mathrm{T}}\right)^{-1}\mathbf{r}_0$, and $\mathbf{q}_0 = \mathbf{z}_0$
2: **for** $j = 0, 1, \ldots$ until convergence **do**
3: $\quad \alpha_j = (\mathbf{r}_j, \mathbf{z}_j)/\left(\mathbf{M}\mathbf{q}_j, \mathbf{q}_j\right)$
4: $\quad \mathbf{p}_{j+1} = \mathbf{p}_j + \alpha_j \mathbf{q}_j$
5: $\quad \mathbf{r}_{j+1} = \mathbf{r}_j - \alpha_j \mathbf{M}\mathbf{q}_j$
6: $\quad \mathbf{z}_{j+1} = \left(\mathbf{L}\mathbf{L}^{\mathrm{T}}\right)^{-1}\mathbf{r}_{j+1}$
7: $\quad \beta_j = (\mathbf{r}_{j+1}, \mathbf{z}_{j+1})/(\mathbf{r}_j, \mathbf{z}_j)$
8: $\quad \mathbf{q}_{j+1} = \mathbf{z}_{j+1} + \beta_j \mathbf{q}_j$
9: **end for**

As long as the $\left(\mathbf{L}\mathbf{L}^{\mathrm{T}}\right)^{-1}$ serves as a good approximate inverse of $\mathbf{M}$, the ICCG(0) method converges quickly.

While the pure CG solver is readily parallelizable (although not so easy procedure) in the distributed memory machine, it is not the case with the ICCG above. Prior to the iteration in the above algorithm, the decomposition $\mathbf{L}\mathbf{L}^{T}$ must be cal-

culated. This procedure is inherently sequential. di Brozolo and Robert (1989) split the factorization into blocks that can overlap and solve in parallel one subsystem for each block. Boehm et al. (1991) also employed this technique for solving the linear system arising from the neutron diffusion equation.

In the parallel RANS model, we employ the nonoverlapping split technique to parallelize the ICCG(0). With 3 processors, for instance, this method split the matrices as in Figure 38. Note that the diagonal elements of $\mathbf{M}$, i.e. $\mathbf{A}$, $\mathbf{B}$, $\mathbf{D}$, pressure $\mathbf{p}$, and source $\mathbf{S}$ are stored in column matrices. Hence, for instance, if the size of the original system is 12,000, $\mathbf{A}\,(1:4000)$ of p1, p2, and p3 in the nonoverlapping partitions will store the elements $\mathbf{A}\,(1:4000)$, $\mathbf{A}\,(4001:8000)$, and $\mathbf{A}\,(8001:12000)$ of the serial sytem, respectively, and similarly for the other column matrices.

The preconditioner in each subsystem is evaluated based on the elements of the submatrix ($\mathbf{M}_1$, $\mathbf{M}_2$, $\mathbf{M}_3$) residing in the rectangle drawn in Figure 38. Note that the further the second subdiagonals are from the main one, the worse is the local preconditioner. For such a matrix, the overlapping-splitting may be used to get a better preconditioner. Depending on the distribution of the fluid in RANS cells, the pattern (values and locations of nonzero elements) of $\mathbf{M}$ changes in each time level. Therefore, in our implementation, at each new time level the preconditioner is calculated.

The arithmetic operations involved in the ICCG algorithm may be grouped into 3 kernels:

1. Vector-vector inner product.

2. Matrix inversion.

3. Matrix-vector multiplication.

This will be discussed in the next sections.

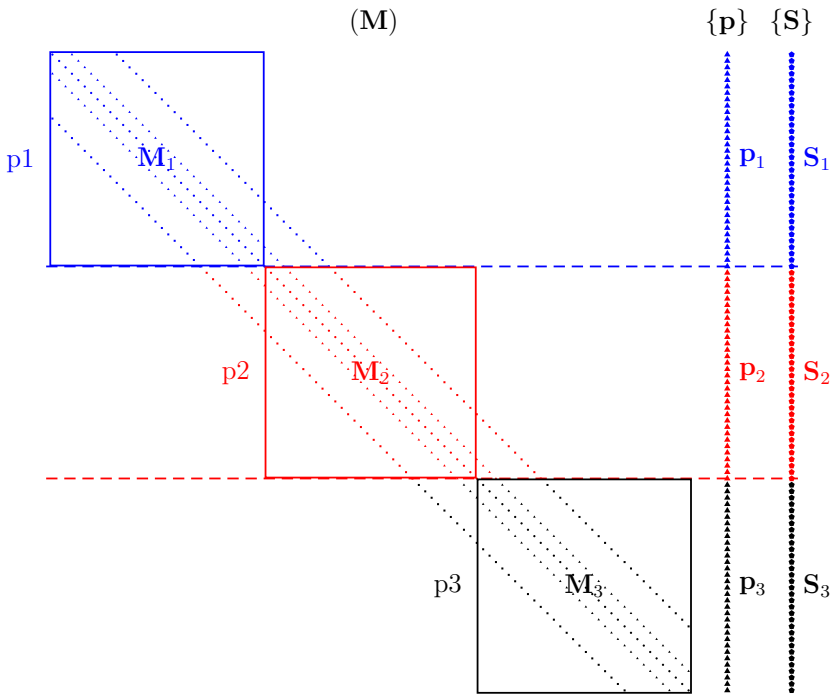


Figure 38. Partitioning of the PPE equation into 3 blocks.

Vector-Vector Inner Product

Since the scalar product of two vectors is simply the sum of all the element-wise multiplications of all elements, and moreover, since each processor has all the elements necessary to perform the local products, this kernel is relatively straightforward to parallelize as follows:

```
   dotp=dot_product{r,z}
   call mpi_allreduce(dotp,dot,1,mpi_double_precision,
&  mpi_sum,mpi_comm_world,err)
```

In the code snippet, `r` and `z` are vectors, `dotp` is local inner products, and `dot` is the sum of all `dotp`'s, and is broadcast to all processors.

Although there are 3 different such products in the ICCG algorithm, i.e. $(\mathbf{r}_j, \mathbf{z}_j)$ in lines 3 and 7, $\left(\mathbf{M}\mathbf{q}_j, \mathbf{q}_j\right)$ in line 3, and $(\mathbf{r}_{j+1}, \mathbf{z}_{j+1})$ in line 7, only 2 of them are calculated per iteration. Notice that the product $(\mathbf{r}_{j+1}, \mathbf{z}_{j+1})$ in line 7 can be used for the same product in line 2 in the next iteration.

Matrix Inversion

In the ICCG method, a matrix inversion $\left(\mathbf{L}\mathbf{L}^{\mathrm{T}}\right)^{-1}$ should be calculated per ICCG iteration to solve for $\mathbf{q}_0$ in line 1 and $\mathbf{z}_{j+1}$ in line 6 of the ICCG algorithm. The solution to this system can be easily calculated using the Gaussian elimination.

Matrix-Vector Multiplication

Consider a simple example of multiplying a 20×20 matrix $\mathbf{M}$ and a 20×1 vector $\mathbf{r}$ distributed into 5 processors as depicted in Figure 39. From the figure, elements 1–4

of $(\mathbf{M})\{\mathbf{r}\}$ in p3, for instance, are calculated as:

$$\begin{aligned}
1 &\rightarrow e_9 r_2 + c_9 r_8 + a_9 r_9 + b_9 r_{10} + d_9 r_{16} \\
2 &\rightarrow e_{10} r_3 + c_{10} r_9 + a_{11} r_{10} + b_{10} r_{11} + d_{10} r_{17} \\
3 &\rightarrow e_{11} r_4 + c_{11} r_{10} + a_{12} r_{11} + b_{11} r_{12} + d_{11} r_{18} \\
4 &\rightarrow e_{12} r_5 + c_{12} r_{11} + a_{13} r_{12} + b_{12} r_{13} + d_{12} r_{19}.
\end{aligned} \tag{5.5}$$

Note that elements r_2, r_3, r_4, r_5, r_8, r_{13}, r_{17}, r_{18}, and r_{19} are not available in p3. To get these variables, communication with other processors that have these elements should be done. Through this communication, p3 receives r_2, r_3, and r_4 from p1, r_5 and r_8 from p2, r_{13} from p4, and r_{17} and r_{18} from p5. On the other hand, when p1, p2, p4, and p5 perform similar computation, these processors receive certain elements from p5 and other processors. To efficiently perform all these communications, prior to the computation, each processor should have a list of processors, from where the data is received and to where the data is sent. The processors in the list could be located adjacent or away from and above or below the processor. Based on the location of the processors in the list, the communication can systematically be done as follows:

```
! receive data from the top, adjacent processors
  if(topadj>=0)then
    call mpi_irecv(recvtopadj(1),n,mpi_doubleprecision,
&   topadj,10,mpi_comm_world,request,error)
    call mpi_wait(request,status,err)
  endif

! send data to bottom, adjacent processors
  if(botadj<=nprocs)then
    call mpi_send(sendbotadj(1),n,mpi_doubleprecision,
&   botadj,10,mpi_comm_world,err)
  endif

! receive data from the bottom, adjacent processors
  if(botadj<=nprocs)then
    call mpi_irecv(recvbotadj(1),n,mpi_doubleprecision,
&   botadj,10,mpi_comm_world,request,error)
```

```
      call mpi_wait(request,status,err)
    endif

! send data to top, adjacent processors
    if(topadj>=0)then
      call mpi_send(sendtopadj(1),n,mpi_doubleprecision,
&     topadj,10,mpi_comm_world,err)
    endif
```

where `topadj` and `botadj` are the adjacent processors from the list, `nprocs` is the number of processors, and `n` is the number of data to be sent/received. The communication with distant processors is done similarly. If there are two distant processors in the list, as in the given example, the above `send-receive` is done twice. In the RANS mesh, all the subdiagonal elements associated with empty cells are zero and associated with the filled cells are nonzero. For efficiency, only the nonzero elements are communicated between the processors. Since the empty and filled-cells change with time, the number of elements `n` in the `send-receive` command changes from one time level to the other.

After all the communications with the adjacent and distant processors are completed, all the necessary data for the matrix-vector multiplication are available in all the processors. Then local matrix-vector multiplication for each processor can be evaluated.

In the parallel ICCG algorithm, there is only one matrix-vector multiplication needs to be calculated per iteration, i.e. $\mathbf{Mq}_j$ in line 3. This result can be used later in line 5 for the same matrix-vector multiplication.

Parallel Hybrid Model Test

To test the accuracy and performance of the parallel hybrid model, we reran the standing wave and hypothetical tsunami simulations given in Chapter V using the

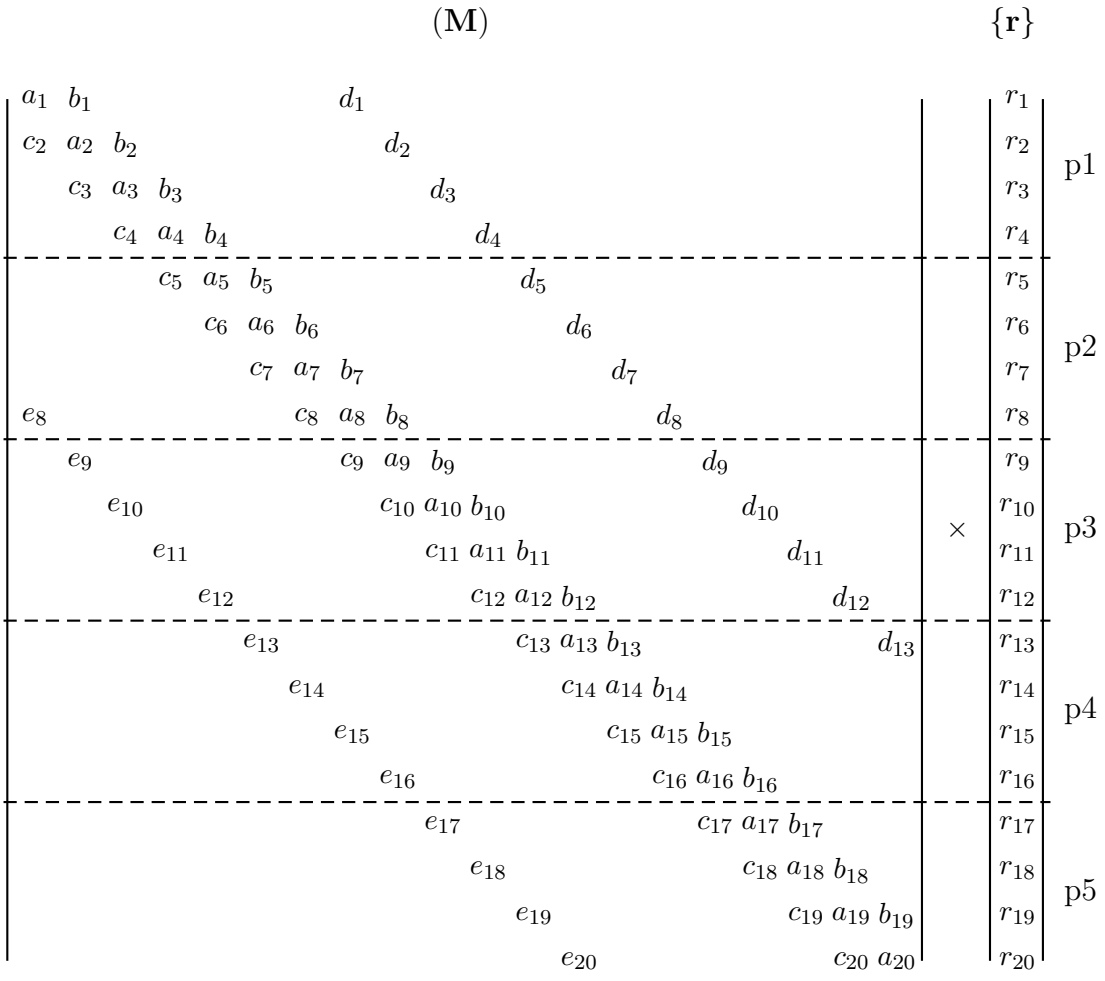


Figure 39. Matrix-vector multiplication.

parallel model. For this purpose, we use the Texas A&M University super computing facility, "hydra," an IBM HPC cluster which consists of 40 p5-575 nodes, each having 16 Power5+processors running at 1.9 GHz and 32 GB of DDR2 DRAM.

Parallel Standing Wave Simulation

The setup of this parallel model test is similar to the serial standing wave simulation in Chapter V. The wave height and period were $H = 0.05$ m and $T = 4$ s and the channel was 0.5 m deep and 72 m long. The channel was divided into the Boussinesq and RANS domains, each 36 m long. The vertical side of the RANS rectangular domain was 0.64 m high. The Boussinesq domain was discretized into 451 grids $\Delta x_B = 0.08$ m. The RANS domain was discretized into 960, $\delta x_R = 0.0375$ m, uniform horizontal grids and 64, $\delta y_R = 0.01$ m, uniform vertical grids. Associated with this setup is a system of 960×64=61,440 simultaneous linear equations, solved at each time level of the run.

The simulation was run for 100 s using 1, 2, 4, and 8 processors. The snapshots of the wave at different times are shown in Figure 40. The figure shows that the wave profiles calculated using different numbers of processors are in agreement, indicating that the information is properly exchanged between/among processors. In the performance test, the model was run using two different configurations of processors. In the first run, all the processors were located in the same node which has 16 processors sharing 32 GB of memory. In the second test, the processors were evenly distributed between two different nodes. For instance, in the 4-processor run, the first 2 processors were located in, say node-1, and the second 2 processors in node-2. Table 6 presents the performance of the parallel hybrid model in simulating the standing wave motion. In general, the model performs well in both the one-node and two-node runs. As expected, the one-node run outperforms the two-node runs because the communi-

cation between the processors in different nodes requires more time than that in the same node.

The iterations in the PPE equation for all parallel runs are twice as many as the the iterations in the serial run, which is due to less information is exchanged in the nonoverlapping blocks, di Brozolo and Robert (1989). The iteration number in pICCG solver is dependent on the number of grids used in the vertical direction as shown in the simple sensitivity analysis in Table 7. This table shows that as the number of grids in y-direction increases, the number of pICCG iteration also increases. Note that in the nonoverlapping block method, the number of points in the gap between two consecutive blocks is equal to the number of grids in the y-direction. Accordingly, the more grids in the y-directions the more information is not exchanged between processors, and expectedly the more iterations are required to reach convergence, as confirmed in the table. The number of iterations is relatively constant as the number of grids increases in the x-direction. Using the overlapping blocks reduces the number of iterations, as more information is exchanged between processors, di Brozolo and Robert (1989). For distributed memory machine, this method will require two extra communications for averaging the solution in the overlapped area between the blocks. Since the cases in the reference were tested on a vector multiprocessor, which presumably used shared memory system, such communication may not be an issue. In summary, the implementation of the overlapping blocks reduces the number of iterations, but adds two extra communications per iteration.

Parallel Hypothetical Tsunami Simulation

In this parallel model test, hypothetical tsunami simulation as presented in the (serial) hybrid model test was employed. The domain size was 101.5 km and divided into 95 km Boussinesq subdomain and 2 km RANS subdomain. The first subdomain

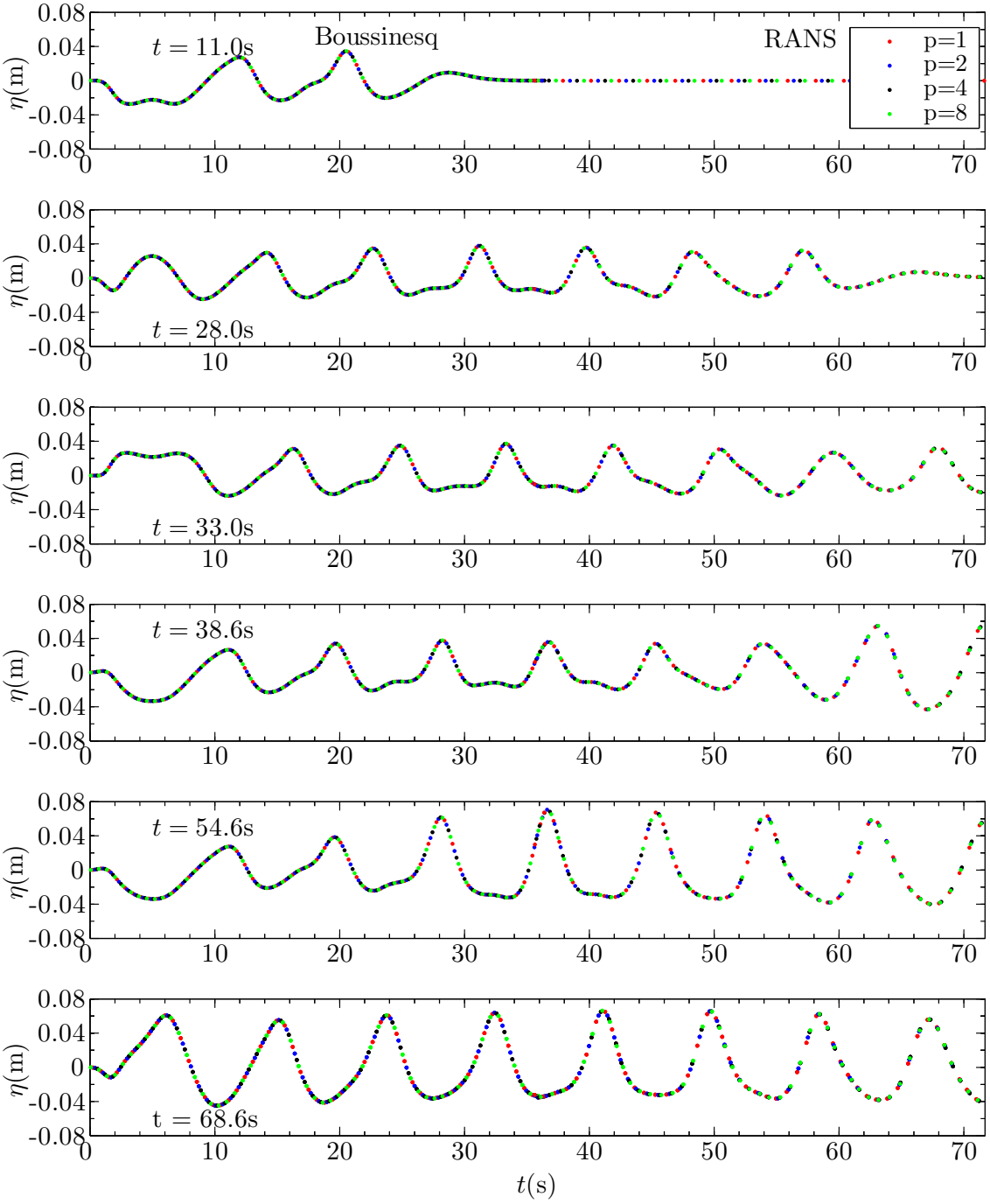


Figure 40. Snapshots of standing wave motion simulated using the parallel hybrid wave model.

Table 6. Performance of parallel hybrid model in standing wave simulation.

	1 Node†			2 Nodes‡			
Procs	time(s)	Speedup	Eff(%)	time(s)	Speedup	Eff(%)	Iter§
1	4,974	1.0	100	4,947	1.0	100	31
2	2,626	1.9	95	2,995	1.7	83	59
4	1,319	3.8	94	1,714	2.9	72	61
8	703	7.1	88	1,087	4.6	57	62

† All processors are in the same node with shared-memory.

‡ Processors are distributed into 2 nodes with distributed-memory.

§ Average number of iterations per time step in the pICCG solver.

Table 7. Average number of pICCG iterations per time step in the standing wave simulation for various combinations of number of grids in the x- and y-directions $(\mathrm{n}x, \mathrm{n}y)$.

Procs	$(480, 32)^{\dagger}$	$(480, 64)^{\ddagger}$	$(480, 128)^{\S}$	$(960, 64)^{\ddagger}$
1	24	23	22	22
2	31	40	50	40
4	32	41	50	40
8	33	43	51	40

† $\delta x = 0.0375\mathrm{m}$, $\delta y = 0.02\mathrm{m}$.

‡ $\delta x = 0.0375\mathrm{m}$, $\delta y = 0.01\mathrm{m}$.

§ $\delta x = 0.0375\mathrm{m}$, $\delta y = 0.005\mathrm{m}$.

was discretized uniformly into 7,000 grids, each 14.2 m long, and the second was also discretized uniformly into 1,000 horizontal and 100 vertical grids, each 2 m and 0.5 m long respectively. The simulation was initiated by 10 m high gaussian elevation offshore and was run for 1,350 s using 1, 2, 4, and 8 processors. As in the previous test, the subdomains were distributed into 1 node and evenly into 2 different nodes. In all the simulations, the turbulence was active.

Figure 41 shows the snapshots of the wave as it is approaching the breakwater. The snapshots are plotted based on the 8-CPU run. This figure clearly indicates that the information is properly passed between/among processors as the wave profiles are continuous across the subdomains. The clock times for the 4 runs are presented in Table 8.

Table 8. Performance of parallel hybrid model in hypothetical tsunami simulation.

	1 Node†			2 Nodes‡			
Procs	time(s)	Speedup	Eff(%)	time(s)	Speedup	Eff(%)	Iter§
1	8,123	1.0	100	8,123	1.0	100	20
2	5,427	1.5	74.8	5,819	1.4	69.8	38
4	2,903	2.8	70.0	3,312	2.5	61.3	38
8	1,637	5.0	62.0	2,066	3.9	49.1	38

† All processors are in the same node with shared-memory.

‡ Processors are distributed into 2 nodes with distributed-memory.

§ Average number of iterations per time step in the pICCG solver.

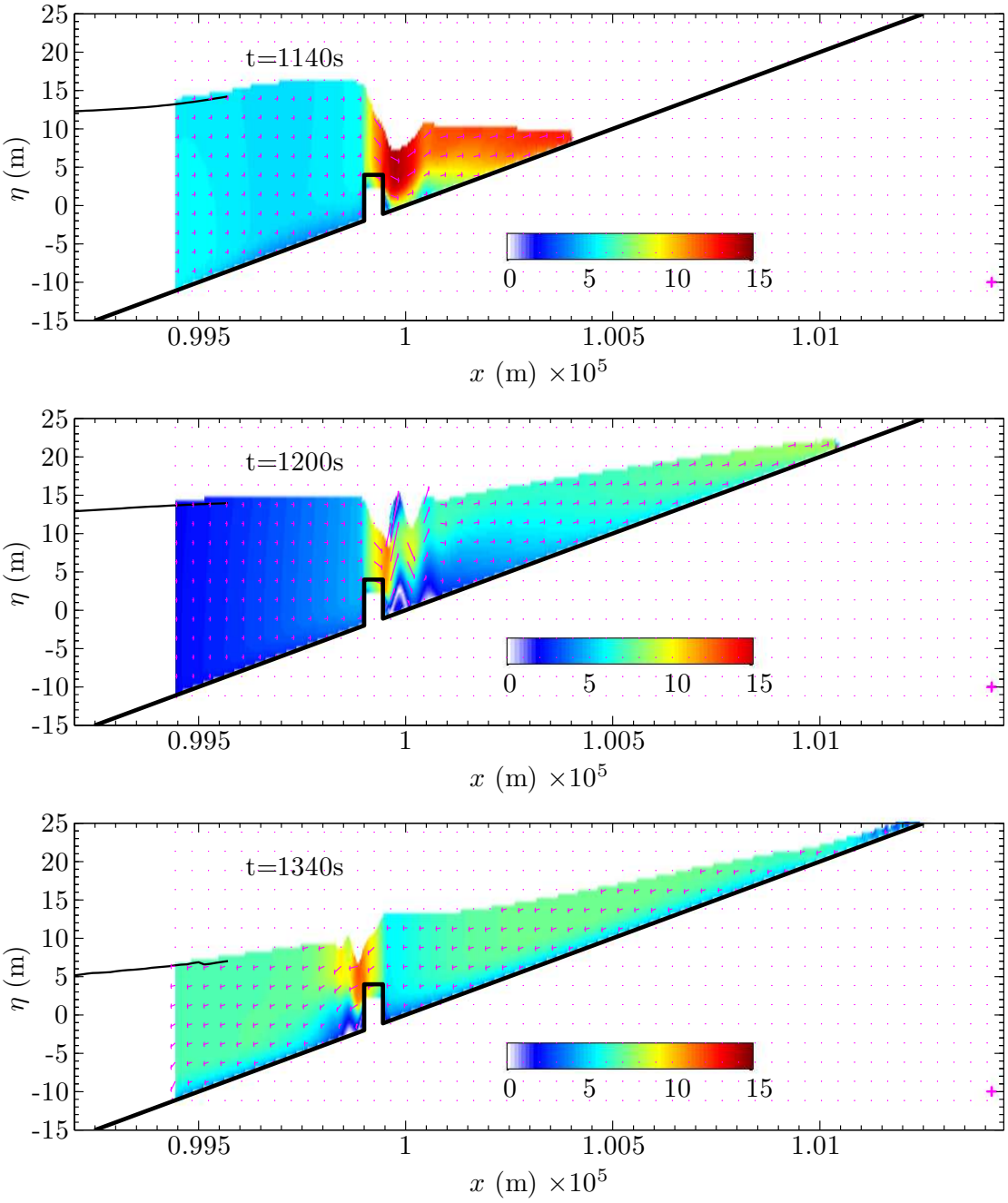


Figure 41. Snapshots of 8-CPU parallel simulation of hypothetical tsunami propagation.

CHAPTER VI
PARALLEL COMPUTATION OF A HIGHLY NONLINEAR BOUSSINESQ EQUATION MODEL THROUGH DOMAIN DECOMPOSITION*

Synopsis

Implementations of the Boussinesq wave model to calculate free surface wave evolution in large basins are, in general, computationally very expensive, requiring huge amounts of CPU time and memory. For large scale problems, it is either not affordable or practical to run on a single PC. To facilitate such extensive computations, a parallel Boussinesq wave model is developed using the domain decomposition technique in conjunction with the message passing interface (MPI). The published and well-tested numerical scheme used by the serial model, a high-order finite difference method, is identical to that employed in the parallel model. Parallelization of the tridiagonal matrix systems included in the serial scheme is the most challenging aspect of the work and was accomplished using a parallel matrix solver combined with an efficient data transfer scheme. Numerical tests on a distributed-memory supercomputer show that the performance of the current parallel model in simulating wave evolution was very satisfactory. A linear speedup is gained as the number of processors increased. These tests showed that the CPU time efficiency of the model was about 75−90%.

Introduction

The calculation of wave propagation from deep to shallow water has long been a challenging problem among the ocean/coastal engineers and scientists. As wave propagates from deep to shallow water, the wave field is transformed due to physical processes such as shoaling, refraction, and diffraction. The ability to accurately evaluate wave transformation depends not only on the computational method used to solve the equations governing the wave propagation, but also on the chosen governing equations themselves.

The Boussinesq equation model has been used for decades to simulate wave propagation from relatively deep to shallow water. Peregrine (1967) derived the "conventional," depth-averaged, Boussinesq equation which can be applied on variable bathymetry. This equation can be used for simulating nonlinear, multidirectional waves with kh values less than roughly 0.3, where k is the wave number and h the water depth. The application of this equation for larger kh does not produce accurate prediction of wave transformation due to a poor description of frequency dispersion. In the last decade, the accuracy limitations of the Boussinesq-type model have been pushed into deeper water, led by the works of Madsen and Sørensen (1992) and Nwogu (1993). By modifying the depth-averaged Boussinesq model through manipulations of the dispersive terms Madsen and Sørensen (1992) created a model with good accuracy through the intermediate water regime. Nwogu (1993) expressed the Boussinesq equations in terms of the velocity at some arbitrary elevation, and with the proper choice of this elevation developed a model with linear accuracy to $kh \approx 3$. While these works increased dispersive accuracy, they are still limited by the weakly nonlinear assumption. Wei et al. (1995) derived a highly nonlinear Boussinesq equation model by keeping nonlinear dispersive terms in the model, which were truncated by Nwogu

(1993). This model of Wei et al., exhibits excellent linear dispersive properties to $kh \approx 3$, while shoaling, wave kinematics, and nonlinear interactions are generally well captured to $kh \approx 1$. A number of researchers have made modifications and enhancements to this model, including Kennedy et al. (2001) to optimize the model nonlinearity and Lynett and Liu (2002) who included additional terms associated with the time dependency of the bathymetry, in order to examine the waves generated by submarine landslides. In solving the highly nonlinear equations, Lynett and Liu (2002) used the high order finite difference method given by Wei and Kirby (1995), however, with slight differences in how some of the nonlinear dispersive terms were treated.

Further increasing the deep-water accuracy of the Boussinesq-type model are a number of "high-order" derivations. Gobbi et al. (2000) extended the model of Wei et al. (1995) to the next order in $(kh)^2$, doubling the linear dispersion accuracy to $kh \approx 6$. Madsen et al. (2002), building off the derivation of Agnon et al. (1999), used multiple expansions at various elevations leading to a model with linear and nonlinear accuracy to $kh \approx 40$. Lynett and Liu (2004a) and Lynett and Liu (2004b) created a "multi-layer" concept, wherein the water column was divided into arbitrarily spaced layers. Accuracy of this model is dependent on the number of layers used, and can be extended into extremely deep water.

Application of the Boussinesq equations covers a broad spectrum of ocean and coastal problems of interest, from wind wave propagation in intermediate and shallow water depths to the study of tsunami wave propagation across large ocean basins. In many cases of practical interest, large physical domains $\left(O\left(10\,\mathrm{km}^2\right)\right)$, which require a huge number of finite difference computational grids, are inevitable. In such circumstances, not only can the PC memory size be too small to carry out the computations, but also a very large CPU time is required. To facilitate such computational

demand, computational tasks can be distributed into several processors so that each processor is responsible for a smaller computational subtask only. This idea underlies the present work of parallelizing a serial Boussinesq model. The parallel Boussinesq model to be developed will use the algorithm of the serial model, employing domain decomposition to create an efficient parallel model, capable of simulating wind waves in coastal basins $\left(O\left(10\,\mathrm{km}^2\right)\right)$ on modest-sized clusters. The implementation of the proposed parallel algorithm on the distributed system is done by employing the commonly used message passing interface (MPI) library, Snir et al. (1996).

Governing Equations

The parallel Boussinesq model developed in this paper is based on its serial counterpart as can be found in Lynett and Liu (2002). The governing equations that are used in this serial model (and also in this paper) consist of the two-dimensional depth-integrated continuity equation:

$$\frac{\partial H}{\partial t}+\nabla\cdot\left(H\mathbf{u}_\alpha\right)-\nabla\cdot\left\{H\left[\left(\frac{1}{6}\left(\eta^2-\eta h+h^2\right)-\frac{1}{2}z_\alpha^2\right)\nabla S+\left(\frac{1}{2}\left(\eta-h\right)-z_\alpha\right)\nabla T\right]\right\}=0 \quad (6.1)$$

and the momentum equation:

$$\frac{\partial \mathbf{u}_\alpha}{\partial t}+\frac{1}{2}\nabla\left(\mathbf{u}_\alpha\cdot\mathbf{u}_\alpha\right)+g\nabla\eta+\frac{\partial}{\partial t}\left\{\frac{1}{2}z_\alpha^2\nabla S+z_\alpha\nabla T-\nabla\left(\frac{1}{2}\eta^2 S+\eta T\right)\right\}+\nabla\left\{\frac{\partial\eta}{\partial t}\left(T+\eta S\right)+\left(z_\alpha-\eta\right)\left(\mathbf{u}_\alpha\cdot\nabla\right)T+\frac{1}{2}\left(z_\alpha^2-\eta^2\right)\left(\mathbf{u}_\alpha\cdot\nabla\right)S+\frac{1}{2}\left(T+\eta S\right)^2\right\}=0 \quad (6.2)$$

where $S = \nabla \cdot \mathbf{u}_\alpha$, $T = \nabla \cdot (h\mathbf{u}_\alpha) + \partial h/\partial t$, h = depth, η = free surface elevation, $H = h + \eta$, $\mathbf{u}_\alpha$ = horizontal velocity at a reference level, z_α , and t = time.

In both equations, it is assumed that the frequency dispersion is weak and the nonlinearity can be large. The velocity variable, $\mathbf{u}_\alpha$, is evaluated at an arbitrary elevation, z_α (in the present work, $z_\alpha = -0.531h$), which is chosen such that the resulting frequency dispersion characteristics of the Boussinesq model agree well with linear theory, Nwogu (1993). Equations (6.1) and (6.2) differ from the equations given by Wei and Kirby (1995) in the inclusion of the time derivatives of the depth (h_t, h_{tt}) to account for temporal bottom profile changes that occur during landslide/earthquake, which is one of several possible sources of tsunami.

Finite Difference Solution

The finite difference solution of the governing equations (6.1) and (6.2) is given in Lynett and Liu (2002), which is based on the formulation presented in Wei and Kirby (1995). The finite difference scheme consists of the third-order in time explicit Adams-Bashforth predictor step and fourth-order in time implicit Adams-Bashforth corrector step, Press et al. (1992). The spatial derivatives in (6.1) and (6.2) are evaluated to fourth-order accuracy. Details of the finite difference method can be found in Lynett and Liu (2002). Here, for convenience, the corresponding finite difference discretization is given. The explicit predictor equations are

$$\eta_{i,j}^{n+1} = \eta_{i,j}^{n} + \tfrac{1}{12}\Delta t \left(23E_{i,j}^{n} - 16E_{i,j}^{n-1} + 5E_{i,j}^{n-2}\right) \tag{6.3}$$

$$U_{i,j}^{n+1} = U_{i,j}^{n} + \tfrac{1}{12}\Delta t \left(23F_{i,j}^{n} - 16F_{i,j}^{n-1} + 5F_{i,j}^{n-2}\right) + 2\left(F_1\right)_{i,j}^{n} - 3\left(F_1\right)_{i,j}^{n-1} + \left(F_1\right)_{i,j}^{n-2} \tag{6.4}$$

$$V_{i,j}^{n+1} = V_{i,j}^{n} + \tfrac{1}{12}\Delta t \left(23G_{i,j}^{n} - 16G_{i,j}^{n-1} + 5G_{i,j}^{n-2}\right) + 2\left(G_1\right)_{i,j}^{n} - 3\left(G_1\right)_{i,j}^{n-1} + \left(G_1\right)_{i,j}^{n-2} \tag{6.5}$$

and the implicit corrector equations:

$$\eta_{i,j}^{n+1} = \eta_{i,j}^{n} + \tfrac{1}{24}\Delta t \left(9E_{i,j}^{n+1} + 19E_{i,j}^{n} - 5E_{i,j}^{n-1} + E_{i,j}^{n-2}\right) \tag{6.6}$$

$$U_{i,j}^{n+1} = U_{i,j}^{n} + \tfrac{1}{24}\Delta t \left(9F_{i,j}^{n+1} + 19F_{i,j}^{n} - 5F_{i,j}^{n-1} + 5F_{i,j}^{n-2}\right) + 2\left(F_1\right)_{i,j}^{n} + \left(F_1\right)_{i,j}^{n+1} - \left(F_1\right)_{i,j}^{n} \tag{6.7}$$

$$V_{i,j}^{n+1} = V_{i,j}^{n} + \tfrac{1}{24}\Delta t \left(9G_{i,j}^{n+1} + 19G_{i,j}^{n} - 5G_{i,j}^{n-1} + 5G_{i,j}^{n-2}\right) + 2\left(G_1\right)_{i,j}^{n} + \left(G_1\right)_{i,j}^{n+1} - \left(G_1\right)_{i,j}^{n}, \tag{6.8}$$

where

$$E = -h_t - \left[\left(\eta + h\right)u\right]_x - \left[\left(\eta + h\right)v\right]_y + \left\{\left(\eta + h\right)\left[\tfrac{1}{6}\left(\eta^2 - \eta h + h^2\right) - \tfrac{1}{2}z_\alpha^2\right]S_x + \left(\tfrac{1}{2}\left(\eta - h\right) - z_\alpha\right)T_x\right\}_x + \left\{\left(\eta + h\right)\left[\tfrac{1}{6}\left(\eta^2 - \eta h + h^2\right) - \tfrac{1}{2}z_\alpha^2\right]S_y + \left(\tfrac{1}{2}\left(\eta - h\right) - z_\alpha\right)T_y\right\}_y \tag{6.9}$$

$$F = -\tfrac{1}{2}\left[\left(u^2\right)_x + \left(v^2\right)_x\right] - g\eta_x - z_\alpha h_{xtt} - z_\alpha h_{xt} + \left(\eta h_{tt}\right)_x - \left[E\left(\eta S + T\right)\right]_x - - \left[\tfrac{1}{2}\left(z_\alpha^2 - \eta^2\right)\left(uS_x + vS_y\right)\right]_x - \left[\left(z_\alpha - \eta\right)\left(uT_x + vT_y\right)\right]_x - \tfrac{1}{2}\left[\left(T + \eta S\right)^2\right]_x \tag{6.10}$$

$$F_1 = \tfrac{1}{2}\left(\eta^2 - z_\alpha^2\right)v_{xy} - \left(z_\alpha - \eta\right)\left(hv\right)_{xy} + \eta_x\left[\eta v_y + \left(hv\right)_y\right] \tag{6.11}$$

$$G = -\tfrac{1}{2}\left[\left(u^2\right)_y + \left(v^2\right)_y\right] - g\eta_y - z_\alpha h_{ytt} - z_\alpha h_{yt} + \left(\eta h_{tt}\right)_y - \left[E\left(\eta S + T\right)\right]_y - - \left[\tfrac{1}{2}\left(z_\alpha^2 - \eta^2\right)\left(uS_x + vS_y\right)\right]_y - \left[\left(z_\alpha - \eta\right)\left(uT_x + vT_y\right)\right]_y - \tfrac{1}{2}\left[\left(T + \eta S\right)^2\right]_y \tag{6.12}$$

$$G_1 = \tfrac{1}{2}\left(\eta^2 - z_\alpha^2\right)u_{xy} - \left(z_\alpha - \eta\right)\left(hu\right)_{xy} + \eta_y\left[\eta u_x + \left(hu\right)_x\right] \tag{6.13}$$

$$U = u + \tfrac{1}{2}\left(z_\alpha^2 - \eta^2\right)u_{xx} + \left(z - \eta\right)\left(hu\right)_{xx} - \eta_x\left[\eta u_x + \left(hu\right)_x\right] \tag{6.14}$$

$$V = v + \tfrac{1}{2}\left(z_\alpha^2 - \eta^2\right) v_{yy} + (z-\eta)\,(hv)_{yy} - \eta_y\left[\eta v_y + (hv)_y\right] \tag{6.15}$$

$$S = u_x + v_y \quad , \quad T = (hu)_x + (hv)_y + h_t. \tag{6.16}$$

The procedure to solve governing equations (6.1) and (6.2) is to first predict the solution (η^{n+1}, u^{n+1}, and v^{n+1}) via the explicit predictors (6.3, 6.4, 6.5), then solving (6.14) and (6.15) to determine u^{n+1} and v^{n+1} from the intermediate variables U and V. To find u^{n+1} and v^{n+1} from (6.14) and (6.15), tridiagonal systems of linear equations must be solved. Next, the predicted values must be iterated using the implicit correctors (6.6, 6.7, 6.8) until the solution converges. During each iteration of the corrector step, the tridiagonal systems of (6.14) and (6.15) must also be solved. For the iteration to halt, the maximum local relative error, which is defined as

$$\left|\frac{w^{n+1} - w_*^{n+1}}{w^{n+1}}\right|, \tag{6.17}$$

where w represents η, u, and v and w_* are the previous iterated values, must be less than 10^{-4}.

Parallelization Strategy

The higher order finite difference scheme, Lynett and Liu (2002), for solving the Boussinesq equations has an identical spatial finite difference stencil for each time level (i.e. $n-2$, $n-1$, n, and $n+1$) and for both the predictor and corrector steps (Figure 42). In both steps, the calculations of the free surface elevation, η^{n+1}, and the velocity groupings, U^{n+1} and V^{n+1}, are iterative and so are independent, and readily parallelizable, calculations. However, this is not the case with the computations of u^{n+1}'s and v^{n+1}'s in (6.14) and (6.15). Here, a tridiagonal system of linear equations must be solved for each row of the computational grids to get the corresponding u^{n+1}'s and for each column of the computational grids to get the corresponding v^{n+1}'s. With

the commonly used LU-decomposition (Thomas) algorithm, Krechel et al. (1989), for solving a tridiagonal system of linear equations, the lower and upper eliminations of the corresponding lower and upper diagonals of the system must be conducted in sequence, starting from the first element to the last for the lower diagonal elimination and in the reverse direction for the upper diagonal elimination. Hence, there are strong dependencies among all processes in this algorithm, which makes it suitable only for sequential calculation, Hockney and Jesshope (1981), and difficult to efficiently parallelize.

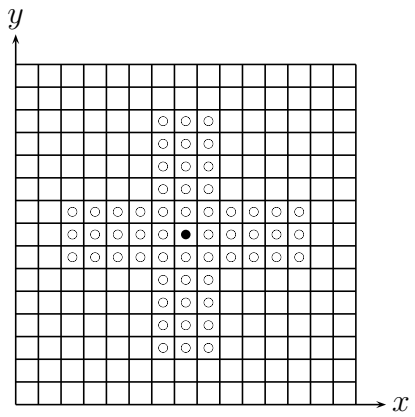


Figure 42. Finite difference stencil of the higher order finite difference solution of the Boussinesq equation.

Thus, with the Boussinesq model, there is a relatively straightforward and expectedly efficient parallelization, as well as an equally difficult one. The calculations of η^{n+1}, U^{n+1}, and V^{n+1} in both the predictor (6.3 to 6.5) and corrector (6.6 to 6.8) steps are highly parallelizable. The tridiagonal system of the linear equations (6.14) and (6.15) that arises from the finite difference scheme is not easily parallelized. However, comparing the amount of the arithmetic operations involved in the evaluations of η^{n+1}, U^{n+1}, and V^{n+1} via the predictor/corrector equations with those of u^{n+1} and

v^{n+1} from the tridiagonal equations, it is apparent that the former far outnumbers the latter. Hence, if an efficient parallel tridiagonal solver can be developed, the serial solution algorithm can be used without any significant modifications to parallelize the Boussinesq model.

In the present work, the domain decomposition method is used to parallelize the Boussinesq model. In this method, the parallel algorithm is very similar to the serial algorithm with some additional routines added to facilitate the communication between processors. Using this method, all the processors involved in the parallel calculations basically perform the same computational operations. The only difference is in the data being processed in each processor. There are three important aspects in our parallel algorithm: (1) domain decomposition, (2) communication, and (3) parallel solver of the tridiagonal system of the simultaneous linear equations. The three aspects are discussed in the following sections and the parallel algorithm is presented as a flowchart given in Figure 43.

Domain Decomposition

The physical/computational domain which is used in Wei et al. (1995) and Lynett and Liu (2002) and in this paper was rectangular in shape. In the domain decomposition method, the rectangular domain is divided into several smaller rectangular subdomains, where the number of subdomains is equal to the number of processors used. With 4 processors, for example, there are three possible ways of decomposing the domain into equal-area parts as depicted in Figure 44. The best decomposition depends on the architecture of the system being used and can be automatically determined in MPI.

An important aspect in decomposing the domain is the load balancing, i.e. all processors must have equal or almost equal amount of data to be processed. If the

number of grid points (nodes) is divisible by the number of processors, the nodes in each processor is simply the ratio of the number of nodes to processors. If it is not, we distribute the remainder on the first m processors, where m is the remainder. For instance, if there are $1,000$ nodes and three processors used along the x-direction, the first two processors will have one node more than the last processor, which results in a load balance in the corresponding direction. Load balancing must be created in both x- and y-directions.

From the finite difference stencil in Figure 42, it is apparent that the computation at an arbitrary point requires values from at most five nodes from the left, right, bottom, and top. Nodes located within five indices from a boundary must receive values from the processor on the opposite side of that boundary. To accommodate these near boundary nodes, the size of each subdomain is increased by five imaginary nodes in all directions. This is manifested in the sizes of all the related arrays.

Communication

Two types of communications occur in this parallel model. The first is the communication between two adjacent processors that occurs during the message passing, and the second is the interprocessor communication occurring when the parallel model solves the tridiagonal systems of linear equations. The latter will be explained in the next section.

In passing the data from one processor to another, an efficient and safe communication must be developed. To efficiently exchange the data between adjacent processors, the data, which consist of five arrays of horizontal and/or vertical grid points, are first stored in a contiguous memory (which can be facilitated in MPI) prior to executing the sending processes. At the same time contiguous memories of the same size as used in the sending processes are created to receive the data from

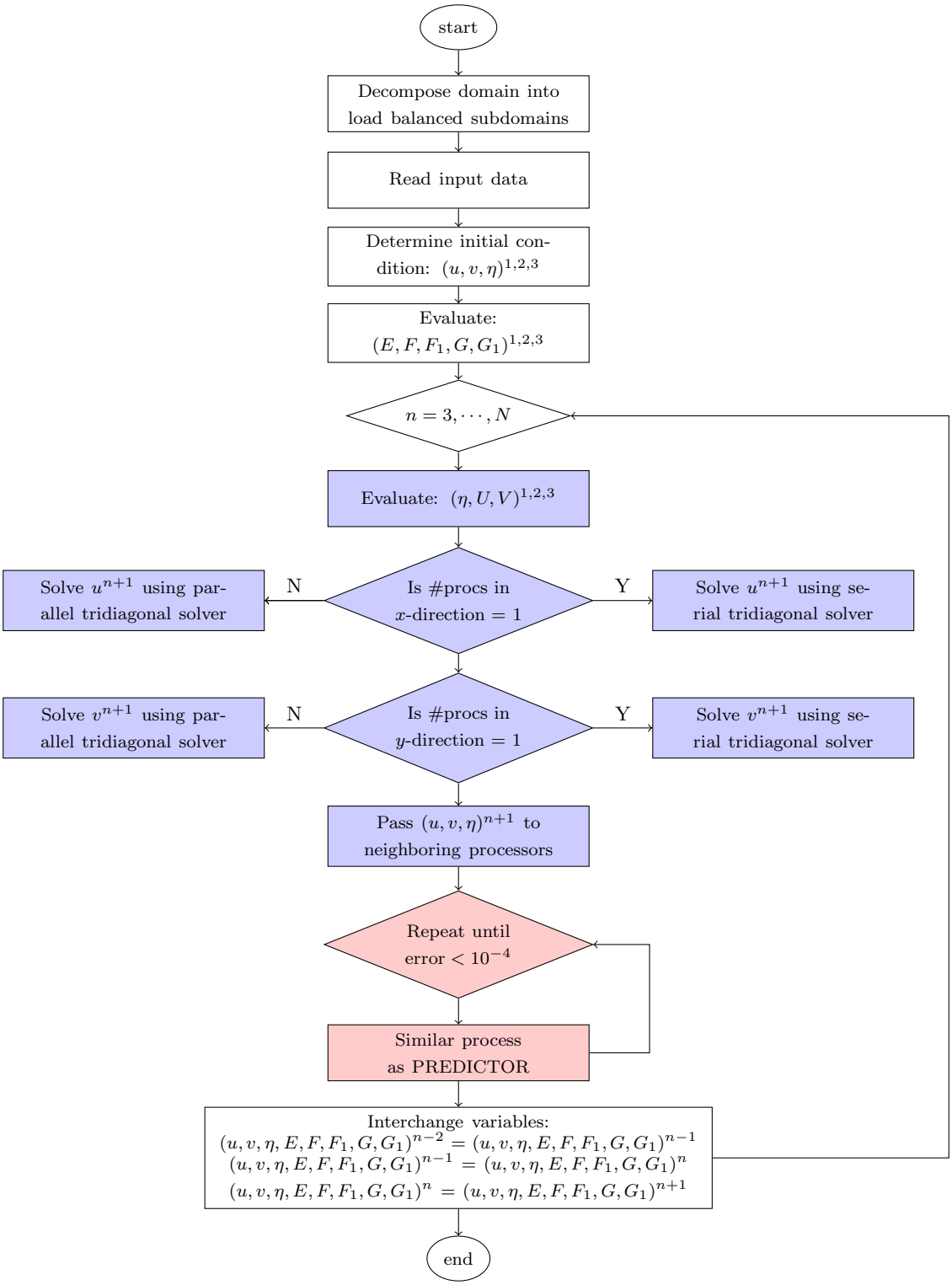


Figure 43. Flowchart of parallel Boussinesq model calculation.

the sending processes. At this point, the data are ready for sending and receiving processes.

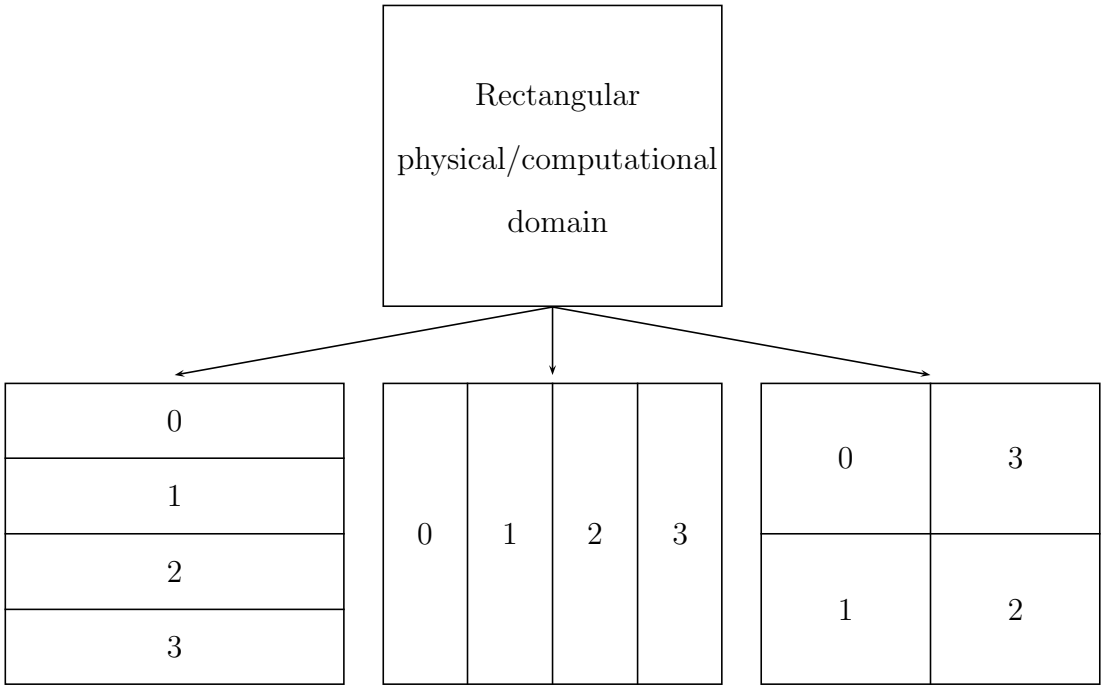


Figure 44. Three possible ways of decomposing a rectangular domain. The numbers in the subdomains represent the processor id's.

To achieve a safe communication process, we use the nonblocking communication `mpi_isend` and `mpi_receive`, with the latter being posted first and followed by the former. Since the computational domain may be very large, requiring a large number of grids and in consequence a large amount of memory, the use of nonblocking communication can prevent the system from "memory starving" that may cause deadlock. With a large number of grids, the use of nonblocking communication may potentially improve the performance of a parallel program, Snir et al. (1996).

Since the problem at hand is two-dimensional, the communication takes place in both the x- and y-directions (Figure 45). In this parallel model, the communication

in the x-direction is first conducted and then followed by the communication in the y-direction. To prevent overlapping communication at the corners of domain, the former communication conveys only the data in the real area, which is $5 \times \mathrm{n}y$ points ($\mathrm{n}y$ is the number of real grids in the y-direction in one processor) and the latter is responsible for the horizontal real area and the imaginary corner areas for a total of $5 \times (\mathrm{n}x + 5)$ or $5 \times (\mathrm{n}x + 10)$ points, depending on whether there is one or two processors on the corresponding sides of the processor ($\mathrm{n}x$ is the number of real grids in the x-direction in one processor).

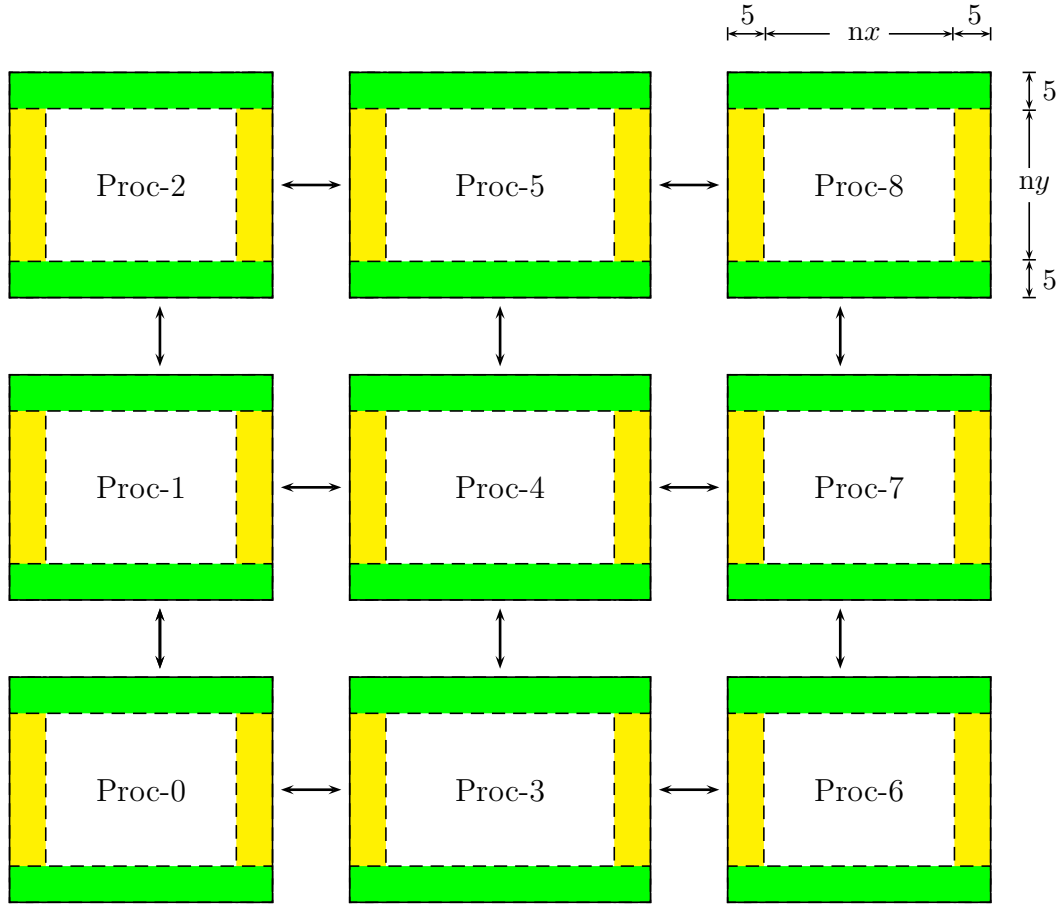


Figure 45. Message passing scheme in parallel Boussinesq model. White is real area, green/yellow is imaginary area; similar-color areas exchange data.

Parallel Solver of Tridiagonal System

The evaluation of u^{n+1} and v^{n+1} in (6.14) and (6.15) involves the process of solving a series of independent tridiagonal systems of linear equations in both the x- and y-directions. If the physical domain is divided into a number of subdomains in one or both directions, the tridiagonal systems of equations, which are now distributed over the processors, must be solved in parallel.

The parallel solution of a tridiagonal system of linear equations is much more difficult than its serial counterpart, which can easily and efficiently be solved by, for example, using the LU decomposition method. Much research has been done to solve the tridiagonal system of equations in parallel, Wang (1981), Krechel et al. (1989), Mattor et al. (1995), among others.

To solve for u and v from U and V in (6.14) and (6.15), which is the primary challenge to parallelize this model, we used the algorithm proposed by Mattor et al. (1995). The stability of this algorithm is similar to that of the serial LU decomposition, which is a desirable feature. The idea in this algorithm is analogous to the solution of a linear inhomogeneous ordinary differential equation, where the solution is the sum of the particular solution and the linear combination of the homogenous solutions:

$$\mathbf{x}_p = \mathbf{x}_p^R + \zeta_p^{UH}\mathbf{x}_p^{UH} + \zeta_p^{LH}\mathbf{x}_p^{LH}, \tag{6.18}$$

where $\mathbf{x}_p$ is the solution of the system, $\mathbf{x}_p^R$, $\mathbf{x}_p^{UH}$, and $\mathbf{x}_p^{LH}$ are the particular, upper homogeneous, and lower homogeneous solutions of the inhomogeneous differential equation analogy and ζ_p^{UH}, ζ_p^{LH} are the coefficients which depend on the coupling to the neighboring solutions. The subscript p indicates that the corresponding solution is local to processor p. Detail of the procedure is given in Mattor et al. (1995).

Prior to evaluating the coefficients ζ_p^{UH}, ζ_p^{LH}, the first and the last elements of

$\mathbf{x}_p^{UH}$, and $\mathbf{x}_p^{LH}$, and $\mathbf{x}_p^{R}$ of all processors were concatenated to form a $(2P-2)\times(2P-2)$ tridiagonal system with ζ_p^{UH}, ζ_p^{LH} as the unknowns and P is the number of processors. Although the algorithm to construct the tridiagonal system, which involved intercommunication among processors, was given in the original paper, here we employed the collective communication routine `mpi_allgather` of MPI with `OutData` variable, Mattor et al. (1995), which carries the first and last elements of $\mathbf{x}_p^{UH}$, $\mathbf{x}_p^{LH}$, and $\mathbf{x}_p^{R}$, acting as the sending variable and another variable of size $8P$ as the receiving variable. Note that, after the call to `mpi_allgather`, all processors received an identical $8P$ receiving-variable which contains all `OutData`'s from all processors, ordered from the smallest to the highest processor-ID. The use of collective communication simplifies the concatenation process and is more efficient on the employed computational platform than the hand-coded communication, Pacheco (1997).

The number of the systems of linear equations is equal to the number of grids used in the x- or y-direction. To efficiently solve those systems, the particular and homogenous solutions of all subsystems were first evaluated, followed by the distribution of the corresponding $\mathbf{x}_p^{UH}$'s, $\mathbf{x}_p^{LH}$'s, and $\mathbf{x}_p^{R}$ to all processors using the collective communication, and completed through evaluation of the final solution via (6.18). Note that since the calculation of u^{n+1}'s and v^{n+1}'s are independent to each other, the order of these calculations is not important, i.e. we can first calculate v^{n+1}'s followed by the u^{n+1}'s or vice versa.

Parallel Model Testing

The present parallel model was tested for both accuracy and performance. To examine accuracy, the linear and weakly nonlinear versions of the model were tested using two idealized scenarios having known analytic solutions. The first idealized case was the

linear wave evolution in a closed rectangular wave basin. The parallel model was run under the same condition as the idealized case and the result was compared with the analytically calculated profiles. In the second test, the propagation of a weakly nonlinear solitary wave along a long, straight, constant-depth channel was considered. The nonlinear mode of the parallel model was expected to produce a solitary wave propagating along the channel with no change in wave form.

To test the performance, the parallel model was used to calculate the wave evolution in a rectangular closed wave basin under three different modes: linear, weakly nonlinear (first order nonlinear terms only), and highly nonlinear (complete equations given by (6.1) and (6.2)). The model was run using different numbers of processors and the run time for each run was recorded to observe scalability of the model. As a final test of the parallel model, the experimental setup of Vincent and Briggs (1989) for a regular wave propagating over a 3-D shoal was simulated. The experimental setup was known to be very nonlinear (e.g. Lynett and Liu (2004a)). This practical application of the model utilized the highly nonlinear equations and the efficiency of the model was discussed.

The computer system used for testing the accuracy and performance of the parallel model was an SGI Altix 3700 supercomputer which consists of 128 1.3-GHz Itanium-2 processors in 32 four-CPU nodes connected through gigabit ethernet, with 256 Gigabytes of total distributed memory. The MPI software used on this platform is MPICH, and the Fortran compiler is Intel Fortran Compiler. The compiler switches used are `-O3 -tpp2`. For the Vincent and Briggs (1989) comparison, an additional small cluster was used for benchmarking. This small cluster consists of 8 2.2-GHz Opteron processors in 4 two-CPU nodes connected through dual gigabit Ethernet. The MPI software used on this platform is LAM and the Fortran compiler is PGI. The compiler switches used were `-fastsse -O4 -tp=amd64`.

Model Accuracy Test

Wave Evolution in a Closed Rectangular Wave Basin

In this idealized case, the wave evolution in a closed rectangular wave basin of constant depth was considered. The physical setup of this test was similar to the one used in Wei and Kirby (1995). The wave basin was a square 7.5×7.5 m^2 basin with a constant depth of 0.45 m. The initial wave profile was a Gaussian hump shape profile

$$\eta_0 = H_0 e^{-2\left[(x-3.75)^2+(y-3.75)^2\right]}, \tag{6.19}$$

where η_0 is the initial free surface elevation, H_0 was the wave amplitude (0.45 m in this test) and the initial velocity was zero. The wave basin wall was an impermeable and reflecting wall.

For this setup, the parallel linear model was run for 50 s of model time. The spatial and time grid sizes for the run were 0.075 m and 0.0143 s, respectively, a total of 100 grids along both x- and y-sides and 3,500 time steps. In each system, 16 processors are used with three different decompositions: 16×1, 8×2, and 4×4. Snapshots of the free surface evolution are shown in Figure 46. The temporal variation of the free surface elevation at the center of the basin, $x = 3.75$ m, $y = 3.75$ m, was recorded and compared with the one calculated by the analytic solution. This comparison is given in Figure 47. The temporal free surface elevations calculated by the parallel model agree very well with the one calculated by the analytical model. The parallel model worked well with the three very different configurations of processors. The run times for these three configurations were also recorded during the runs; the 16×1 configuration took about 46.1 s of CPU time, the 8×2 took about 21.6 s, and the 4×4 took 17.0 s. For comparison, similar run with 1 processor took about 62.2 s. The three parallel runs demonstrate that the 4×4 decomposition resulted in the best performance. With such

configurations as 16×1 and 8×2, even if the load of arithmetic operations involved in each processor was equal to that in the 4×4 configuration, the communication load in the previous two was heavier than in the 4×4. Comparing the 4×4 parallel and serial run times, we gained a speedup of 3.7 or an efficiency of 23%. With a small number of grids (nx = 100 and ny = 100), the cost of communication was more expensive than the local arithmetic operational cost, hence resulted in small efficiency. This efficiency, as will be shown later, will increase as the number of grids increases.

Solitary Wave Propagation Along a Straight Long Channel

Next, the weakly nonlinear mode of the parallel model is tested using an idealized case of solitary wave propagation in a straight long channel. This idealized case can be found in Wei and Kirby (1995). The velocity and the surface elevation of the solitary wave are analytically given by

$$\eta = A_1 \mathrm{sech}^2 \left[B\left(x - Ct\right)\right] + A_2 \mathrm{sech}^4 \left[B\left(x - Ct\right)\right] \tag{6.20}$$

$$u = A \mathrm{sech}^2 \left[B\left(x - Ct\right)\right], \tag{6.21}$$

where A, B, C, A_1, and A_2 are constants which depend on the physical setup of the model, Wei and Kirby (1995). In this case, the channel depth was 0.45 m, the wave amplitude was 0.04 m, and the length of the channel was 450 m. The domain was discretized into 1,500 equally spaced computational grids, each was 0.3 m long. The wave was initially located at x = 80 m.

The parallel weakly nonlinear model was run for 200 s using 16 processors. In the course of its propagation, the waves at t = 0, 40, 80, 120, and 160 s were recorded. These snapshots are given in Figure 48. This figure shows that the numerically-calculated solitary wave propagated in the positive x-direction with constant speed,

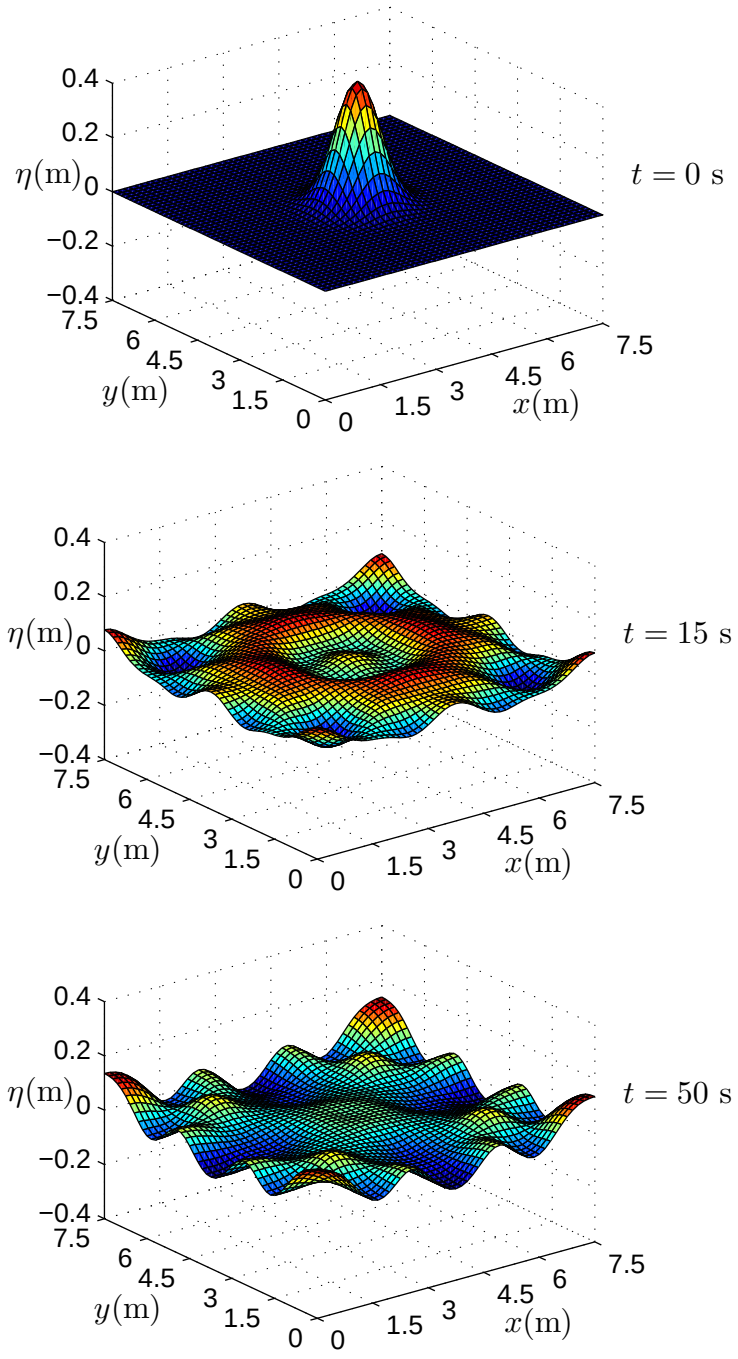


Figure 46. Linear Gaussian-wave profiles at three different times calculated using 16 processors.

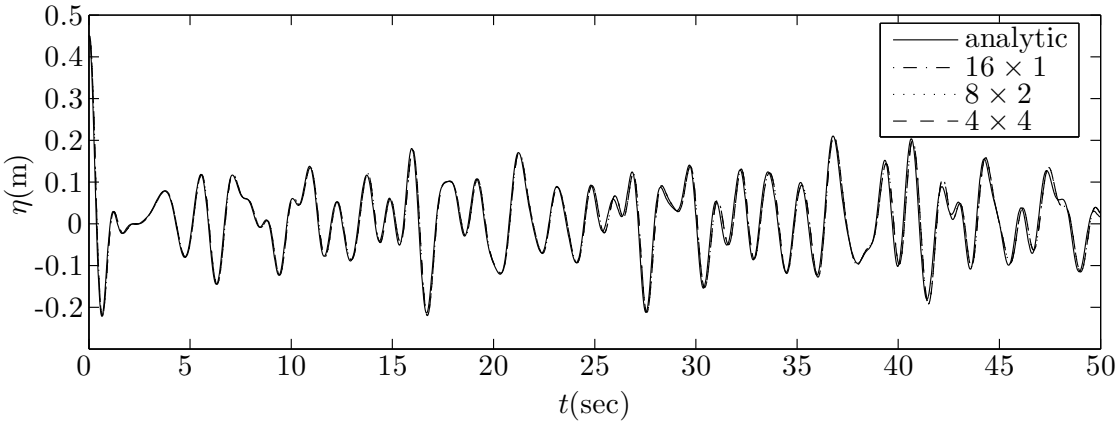


Figure 47. Temporal variation of the free surface elevation at the center of the wave basin calculated by the analytic solution and the parallel numerical models using 16 processors.

i.e. the distances between two consecutive wave forms were constant, and wave height and length agree well with the analytic solution. This example also clearly shows that information was passed correctly from subdomain to subdomain.

Performance Test

Performance of the parallel model was tested using a previous idealized case: wave evolution in a closed rectangular wave basin. The purpose of this performance test was to observe the scalability of the model for various domain sizes. Three different domain sizes were considered and presented in Table 9. For all simulations, the depth of the wave basin and the initial wave height were the same, $d = 0.45$ m and $H = 0.045$ m, respectively.

The model was run under three different modes: linear, weakly nonlinear, and fully nonlinear. In all parallel runs, even numbers of processors were used: 2, 4, 6, 8, 10, 12, 16, 18, 20, 24, 30, and 32 processors. Figure 49 shows the speedup and

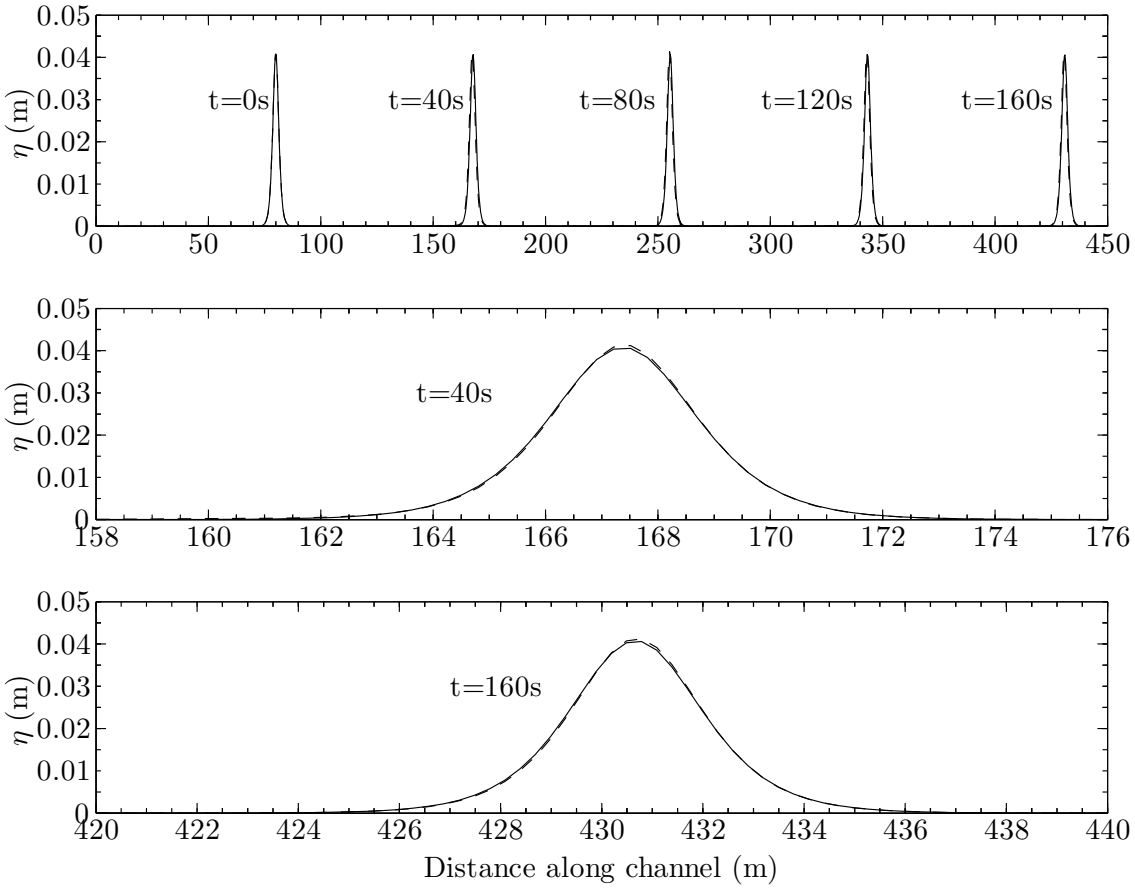


Figure 48. Solitary wave propagation along a long straight channel (full line is analytic solution, dashed line is parallel model) calculated using 16 processors.

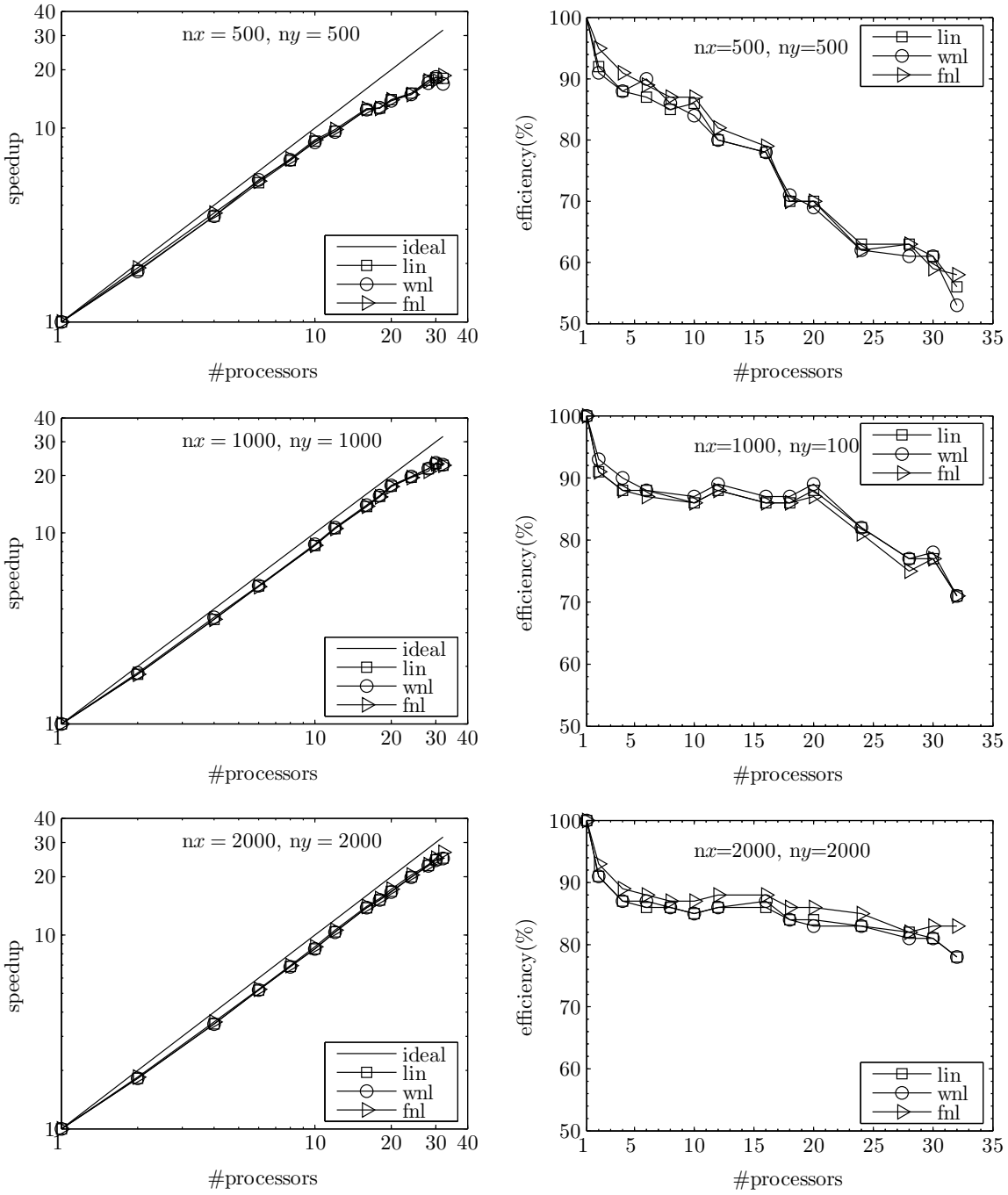


Figure 49. Parallel computational speedup/efficiency of the parallel Boussinesq model in computing the evolution of the Gaussian-wave in a rectangular basin.

Table 9. Domain setup for parallel model performance.

Test	lx^*(m)	l$y^\dagger$(m)	n$x^\ddagger$	n$y^\S$
1	50	50	500	500
2	100	100	1000	1000
3	2000	2000	2000	2000

* Length of the x-side of the basin.
† Length of the y-side of the basin.
‡ Number of grids in the x-direction.
§ Number of grids in the y-direction.

efficiency of the parallel calculation for different numbers of processors used. Here, the speedup was defined as the ratio of the parallel run-time to the run-time of the serial version of the Boussinesq model (using a single processor and the Thomas algorithm to solve the tridiagonal systems). Figure 49 shows that the overall performance of the model is very good. The efficiency of the model decreases as the number of processors increases which is apparent in the case of 500×500 and 1000×1000 domains. The rate of the efficiency decrease is faster for smaller domain. This is due to the ratio of arithmetic (addition/subtraction and multiplication/division) operation time to communication time decreasing faster for domains with smaller number of nodes. The performance of the model improves as the number of grids increases; a favorable feature of a parallel model which is intended for simulation on ever-increasing domain sizes. In general, it appears that the efficiency is at least 80% for subgrid sizes of 200×200 or greater.

Finally, simulation of a 3-D experimental setup is presented. One of the most frequently studied 3-D water wave problems is that of wave interaction with a sub-

merged shoal (e.g. Berkhoff et al. (1982)). Here, one of the experiments of Vincent and Briggs (1989) was numerically recreated. An elliptic shoal 6.1 m long and 7.92 m wide was placed in a wave tank which was 35 m wide and 29 m long. The shoal had a maximum height of 30.5 cm in 45.7 cm of water. The exact mathematical representation of the shoal can be found in Vincent and Briggs (1989). While many different wave conditions were examined experimentally, here only a single regular wave case was simulated with $T = 1.3$ s and $H = 4.8$ cm. The simulations used the highly nonlinear set of the Boussinesq equations.

Table 10. Vincent and Briggs shoal, fully nonlinear, $[\mathrm{n}x, \mathrm{n}y] = [622, 515]$.

	Opteron cluster		Itanium-2 cluster	
# of CPUs	Wall clock time(s) wave period	Efficiency	Wall clock time(s) wave period	Efficiency
1	362		2,888	
2	187	0.97	1,586	0.91
4	94	0.96	831	0.87
6	71	0.85	549	0.88
8	56	0.81	425	0.85
16			244	0.74
32			153	0.59

Figure 50 gives snapshots of the waves as they transformed over the shoal. The waves narrowed and steepened as they moved over the shoal, while refraction focused wave energy behind the shoal. The result is a complex, highly nonlinear, and multi-directional wave field. The numerical comparisons with experimental data, for any number of processors used, are identical to those presented in Lynett and Liu (2004a)

Table 11. Vincent and Briggs shoal, fully nonlinear, $[\mathrm{n}x, \mathrm{n}y] = [1242, 1029]$.

# of CPUs	Opteron cluster		Itanium-2 cluster	
	Wall clock time(s) wave period	Efficiency	Wall clock time(s) wave period	Efficiency
1	3,994		20,670	
2	2,034	0.98	11,351	0.91
4	1,005	0.99	5,802	0.89
6	686	0.97	44,24	0.78
8	539	0.93	3,332	0.78
16			1,715	0.75
32			858	0.75

for the one-layer model (equivalent to the equations of Wei et al. (1995)), exhibiting very good agreement. Hence, these identical comparisons will not be included here as well.

Tables 10 and 11 give the wall clock time per simulated wave period and efficiency of the parallel model, on two platforms for two different grid sizes. Table 10 shows the results using 40 grid points per incident wavelength, while the values in Table 11 used 80 grid points per incident wavelength. The total numbers of grid points are given in the table captions. The Itanium-2 cluster shows efficiency similar to that given in Figure 8 for the like-sized matrix dimensions. The Opteron cluster yields slightly better efficiency, which may be attributed to the dual-gigabit Ethernet or the use of LAM. Also of significant note are the relative CPU times for the two platforms, with the superior floating-point capabilities of the AMD chips showing their strength.

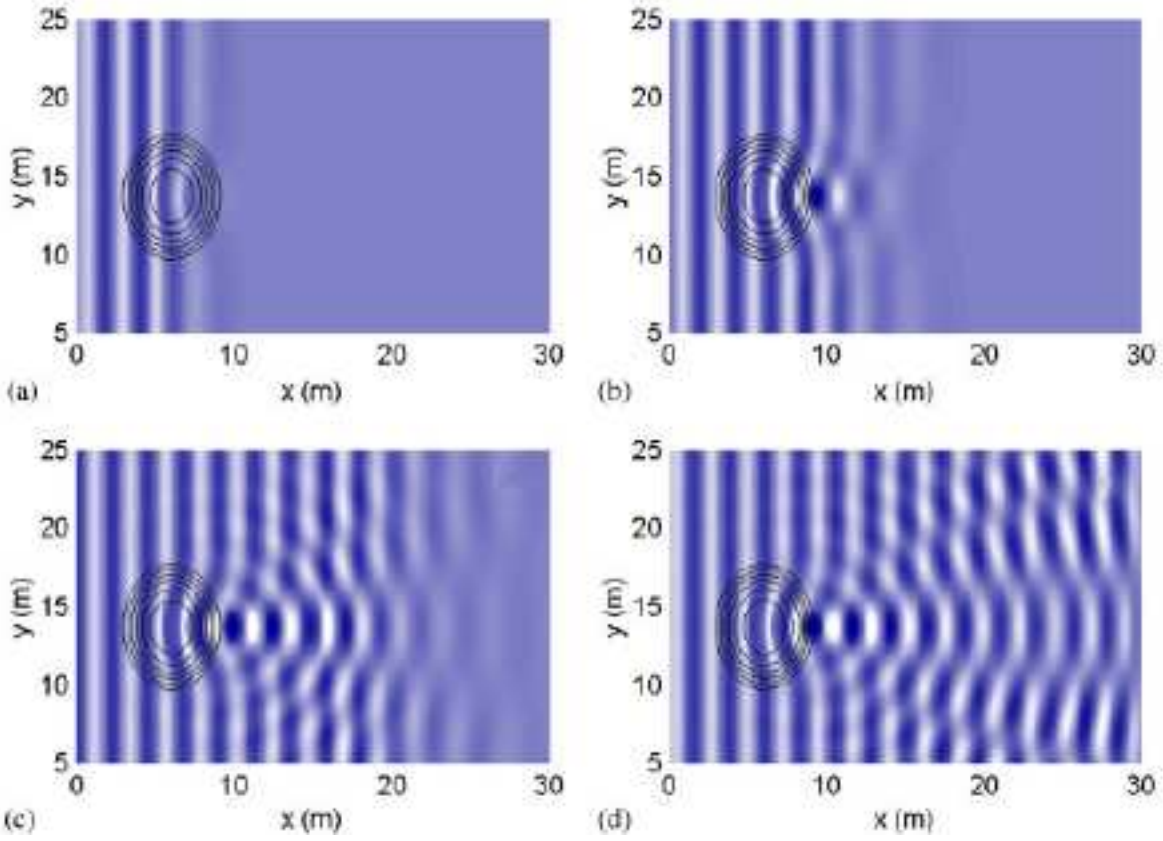


Figure 50. Plan view of regular wave propagation (period = 1.3 s, height = 4.8 cm) over a submerged shoal at: (a) time = 6 s, (b) time = 10 s, (c) time = 18 s, and (d) time = 50 s. The shoal location is given by the contours. Simulations used 8 CPUS.

Conclusion

In the present work, the Boussinesq model of Wei et al. (1995) was parallelized using the domain decomposition method, where each processor performed the same operations. The parallel algorithm was identical to its serial counterpart, based on an iterative predictor/corrector scheme also requiring a tridiagonal solution for each iteration. The model test indicated that both the validity and the performance of the model are excellent. The performance of the model may be further improved if a more efficient parallel tridiagonal solver is employed. Success at parallelizing the Boussinesq model will allow for large domain simulation which is not possible to run on a single PC due to limited memory size and large computational time. This parallel model provides the future opportunity for large wave-resolving simulations in the nearshore, with global domains of many millions of grid points, covering $O\left(10\,\mathrm{km}^2\right)$ and greater basins. Additionally, real-time simulation with Boussinesq equations becomes a possibility.

CHAPTER VII

SUMMARY

In the first part of this dissertation we present the development of the Boussinesq-RANS hybrid wave model which two-way couples numerical models based on the 1-D nonviscous Boussinesq-equations and the 2-D viscous turbulence-closed RANS-equations. The hybrid model is intended for large scale wave simulation, which from either the accuracy or computational point of view is not possible to carry out using either model alone. The Boussinesq model will typically solve a large spatial portion of the computational domain, from the wave generation area to the prebreaking zone with good accuracy and minor CPU needs. Coupled with the RANS model, turbulence and breaking waves in the nearshore can be simulated with high accuracy.

The model tests suggest that the current hybrid model is able to perform a broad range of nonbreaking/breaking wave tasks, from small scale analytic and laboratory experimental scenarios to large scale tsunami simulation, with good accuracy and efficient computational time. For future studies and to further validate the model, simulations based on field data may be conducted. The primary deficiency of the presented model is that the location of the interface must be specified a priori. As mentioned, this location should be situated where turbulence is very low, such that both models are correctly describing the local physics. Ideally, the interface would be dynamically located, and allowed to move either landward or seaward as the local conditions dictate. The implementation of such a dynamic interface relies more on coding flexibility than the implementation of a correct physical boundary condition, as is the focus of this paper, and is left as a future enhancement to be incorporated into the coupling presented here.

For future studies, the hybrid algorithm herein can be readily employed to ex-

tend the model coupling to the 2-D Boussinesq and 3-D RANS wave models, as the numerical algorithms of the higher dimension models remain the same.

Since the Boussinesq model in this work is based on the potential flow equation, the turbulence mechanism cannot be taken into account or approximated by the model, and hence the interface should be located in a low turbulence area. Incorporating the turbulence mechanism in the Boussinesq model such as in Karambas and Koutitas (1992) and Veeramony and Svendsen (2000) may push the location of the interface nearer to the turbulence area and hence reduce the computational cost. As suggested in Chapter II, the two-grid component on the interface is due to the small velocity discrepancy between the Boussinesq and RANS models, which often occurs in the simulation involving wave with higher nonlinearity. Employment of higher 4^{th} order Boussinesq model, Gobbi et al. (2000), may reduce the discrepancy, and therefore the occurrence of the two-grid component can be minimized, hence avoiding the use of filter.

For large scale simulation with detailed turbulence computation inside the breaker zone, the computational time is very high. This is of course due to the huge number of computational grids employed in the finer RANS mesh. Integrating a parallelized RANS solution scheme into the hybrid model could greatly reduce the wall clock time, and may further facilitate the regular use of the hybrid and RANS models by engineers and scientists. This is presented in second part of this dissertation.

In the second part of this dissertation, the parallel hybrid wave model is developed to optimize the computational speed of the model, especially intended for detail turbulence simulation covering large nearshore area. In parallelizing the model, the Boussinesq model remains serial and the RANS model is parallelized. One of the processors is responsible for doing both the Boussinesq and the RANS computations in one of the subdomains. The parallelization of the RANS model consists of two

main parts: first, the parallelization of all parts of the algorithm outside the PPE solver and second, the parallelization of the PPE solver.

To solve the PPE equation, we use the Conjutagate Gradient method preconditioned with the incomplete Cholesky decomposition with zero fill-in. While CG method is quite straightforward to parallelize, it is not the case with the preconditioner. In this work, we use the nonoverlapping decomposition technique where the preconditioner is calculated based on the the local block matrix. Since the decomposition does not overlap, there is no information exchange between two adjacent processors, and hence more iterations are required to reach convergence. Sensitivity analysis shows that the number of iterations in the parallel solver increases if the distance of the off-diagonal matrices from the main diagonal, which is equal to the number of grids in the y-direction, increases, and decreases otherwise, and are independent of the number of grids in the x-direction.

Two model tests, the standing wave simulation using 960×66 grids and the hypothetical tsunami simulation using 1000×100 grids, show that the parallel hybrid model can properly handle the data communication between/among the processors as indicated by the results of the simulations. The speedup of the runs with up to 8 processors is reasonably good.

For future development, the overlapping decomposition of the matrices may be used to reduce the number of iterations. This method, however, costs two additional communications per iteration to calculate the average of the solution in the overlapping area and the arithmetic operations in the averaging process itself.

The third part of this dissertation presents the work on the parallelization of the Boussinesq model. The parallel technique has been employed in the parallel Cornell University Long and Intermediate Wave Modeling Package (COULWAVE) and successfully employed for tsunami modeling, Lynett (2005), where large number

of computational grids are employed. In the presence of dry/wet region in the domain, some of the processors may occupy the dry region, while others occupy the wet region, as shown in an idealized case in Figure 51. To optimize the computational time for serial Boussinesq model run on such domain, it is relatively an easy practice by considering the wet region only. For parallel implementation, however, employing similar technique is not efficient since one of the processors (processor-1) is idle, while others are working. For future development, partitioning the domain so that all the processors are working on equal amount of load will optimize the parallel Boussinesq model. Figure 52 shows an implementation of the parallel simulation of 2004 Sumatera tsunami using the parallel Coulwave model, with nonoptimized decomposition.

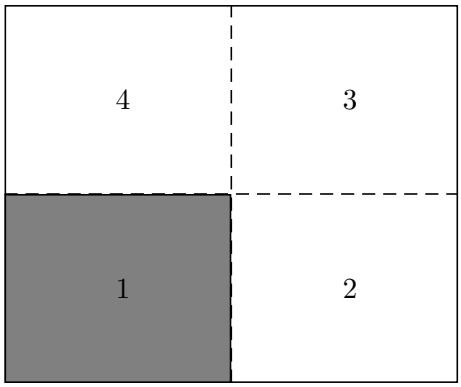


Figure 51. Domain decomposition on dry-wet area. Gray is dry, white is wet.

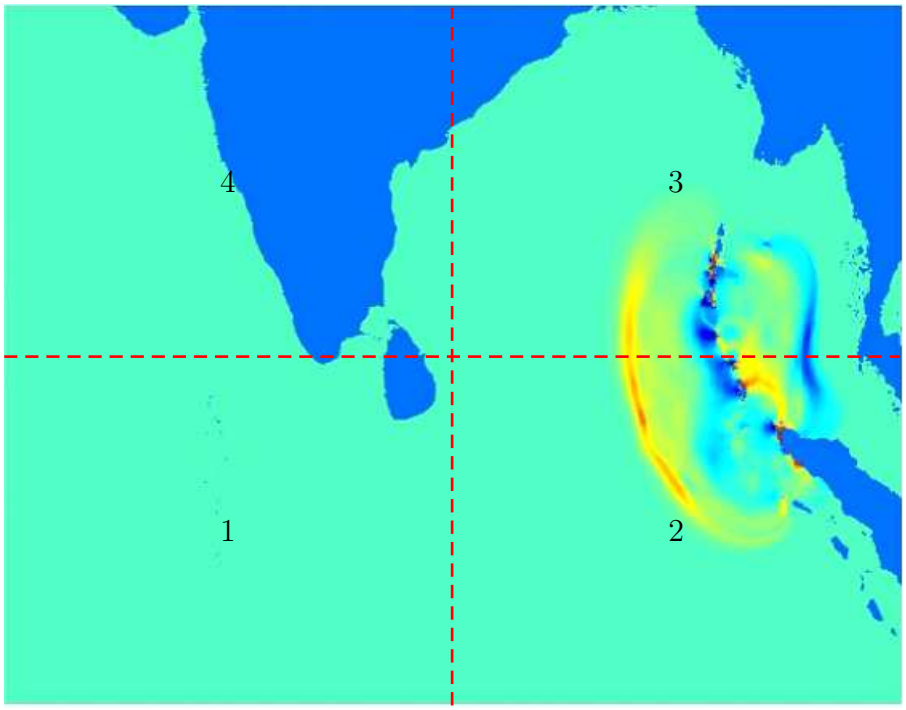


Figure 52. Domain decomposition in parallel simulation of the Sumatera 2004 tsunami. Adapted from Lynett (2005).

REFERENCES

Agnon, Y., Madsen, P. A., and Shäffer, H. (1999). "A new approach to high-order Boussinesq models." *Journal of Fluid Mechanics*, 399, 319–333.

Berkhoff, J. C. W., Booy, N., and Radder, A. C. (1982). "Verification of numerical wave propagation models for simple harmonic linear water waves." *Coastal Engineering*, 6(3), 255–279.

Biausser, B., Fraunie, P., Grilli, S. T., and Marcer, R. (2004). "Numerical analysis of the internal kinematics and dynamics of 3-D breaking waves on slopes." *International Journal of Offshore and Polar Engineering*, 14(4), 247–256.

Boehm, A., Brehm, J., and Finneman, H. (1991). "Parallel conjugate gradient algorithms for solving the neutron diffusion equation on SUPRENUM." *Proc., 5th ACM Int. Conf. Supercomput.*, Cologne, Germany. ACM, 163–171.

Chen, Q., Kirby, J. T., Dalrymple, R. A., Kennedy, A. B., and Chawla, A. (2000). "Boussinesq modeling of wave transformation, breaking, and runup. II: 2D." *Journal of Waterway, Port, Coastal, and Ocean Engineering*, ASCE, 126(1), 48–56.

Chen, Q., Kirby, J. T., Dalrymple, R. A., Shi, F., and Thornton, E. B. (2003). "Boussinesq modeling of longshore currents." *Journal of Geophysical Research*, 108, 1–18.

Dean, R. G. and Dalrymple, R. A. (1991). *Water Wave Mechanics for Engineers and Scientists*. World Scientific Publishing, Singapore. 353 pp.

di Brozolo, G. R. and Robert, Y. (1989). "Parallel conjugate gradient-like algorithms

for solving sparse nonsymmetric linear systems on a vector multiprocessor." *Parallel Computing*, 11(2), 223–239.

Dodd, N. (1999). "Numerical model of wave run-up, overtopping, and regeneration." *Journal of Waterway, Port, Coastal, and Ocean Engineering*, ASCE, 124(2), 73–81.

Fujima, K., Masamura, K., and Goto, C. (2002). "Development of the 2D/3D hybrid model for tsunami numerical simulation." *Coastal Engineering Journal*, 44(4), 373–397.

Garcia, N., Lara, J. L., and Losada, I. J. (2004). "2-D numerical analysis of near-field flow at low-crested permeable breakwaters." *Coastal Engineering*, 51(10), 991–1020.

Gobbi, M. F., Kirby, J. T., and Wei, G. (2000). "A fully nonlinear Boussinesq model for surface waves. Part 2. Extension to $O\left(kh\right)^4$." *Journal of Fluid Mechanics*, 405, 182–210.

Grilli, S. T., Gilbert, R. W., Lubin, P., Vincent, S., Astruc, D., Legendre, D., Duval, M., Kimmoun, O., Branger, H., Devrard, D., Fraunie, P., and Stephane, A. (2004). "Numerical modeling and experiments for solitary wave shoaling and breaking over a sloping beach." *Proc., 14th Int. Offshore Polar Eng. Conf.*, Toulon, France. International Society of Offshore and Polar Engineers, 306–312.

Hockney, R. W. and Jesshope, C. R. (1981). *Parallel Computers: Architecture, Programming and Algorithms*. Hilger Ltd., Bristol, UK.

Hsiao, S.-C., Lynett, P., Hwung, H.-H., and Liu, P. L.-F. (2005). "Numerical simulations of nonlinear short waves using a multilayer model." *Journal of Engineering Mechanics*, ASCE, 131(3), 231–243.

Karambas, T. V. and Koutitas, C. (1992). "A breaking wave propagation model based on the Boussinesq equations." *Coastal Engineering*, 18, 1–19.

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

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

Kershaw, D. S. (1978). "The incomplete Cholesky-Conjugate gradient method for the iterative solution of system of linear equations." *Journal of Computational Physics*, 26, 43–65.

Kobayashi, N. and Wurjanto, A. (1989). "Wave overtopping on coastal structures." *Journal of Waterway, Port, Coastal, and Ocean Engineering*, ASCE, 115(2), 235–251.

Kothe, D. B., Mjolsness, R. C., and Torrey, M. D. (1994). "Ripple: A Computer Program for Incompressible Flows with Free Surfaces." *Report No. LA-12007-MS*, Los Alamos National Laboratory, NM.

Krechel, A., Plum, H. J., and Stueben, K. (1989). "Solving tridiagonal linear systems in parallel on local memory MIMD machines." *GMD Technical Report 372*, St. Augustin, Germany.

Lachaume, C., Biausser, B., Grilli, S. T., Fraunie, P., and Guignard, S. (2003). "Modeling of breaking and post-breaking waves on slopes by coupling of BEM and VOF methods." *Proc., 13th Int. Offshore Polar Eng. Conf.*, Honolulu, HI. International Society of Offshore and Polar Engineers, 1698–1704.

Lara, J. L., Garcia, N., and Losada, I. J. (2006). "RANS modelling applied to random wave interaction with submerged permeable structures." *Coastal Engineering*, 53, 395–417.

Lin, P. and Liu, P. L.-F. (1998). "A numerical study of breaking waves in the surf zone." *Journal of Fluid Mechanics*, 359, 239–264.

Liu, P. L.-F. and Cheng, Y. (2001). "A numerical study of the evolution of a solitary wave over a shelf." *Physics of Fluids*, 13(6), 1660–1667.

Lynett, P. (2005). *Parallel Coulwave model for use on MPI-based clusters.* http://ceprofs.tamu.edu/plynett/COULWAVE/pCoulwave/default.htm. (accessed on 10/31/2007).

Lynett, P. and Liu, P. L.-F. (2002). "A numerical study of submarine-landslide-generated waves and run-up." *Proceedings of the Royal Society of London*, 458, 2885–2910.

Lynett, P. and Liu, P. L.-F. (2004a). "Linear analysis of the multi-layer model." *Coastal Engineering*, 51(6), 439–454.

Lynett, P. and Liu, P. L.-F. (2004b). "A two-layer approach to wave modeling." *Proceedings of the Royal Society of London*, 460, 2637–2669.

Madsen, P. A., Bingham, H. B., and Liu, H. (2002). "A new Boussinesq method for fully nonlinear waves from shallow to deep water." *Journal of Fluid Mechanics*, 462, 1–30.

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." *Coastal Engineering*, 18, 183–204.

Mattor, N., Williams, T. J., and Hewett, D. W. (1995). "Algorithm for solving tridiagonal matrix problems in parallel." *Parallel Computing*, 21, 1769–1782.

Meijerink, J. A. and Vorst, H. A. V. D. (1977). "An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix." *Mathematics of Computation*, 31(137), 148–162.

Nwogu, O. (1993). "Alternative form of Boussinesq equations for nearshore wave propagation." *Journal of Waterway, Port, Coastal, and Ocean Engineering*, ASCE, 119(6), 618–638.

Pacheco, P. S. (1997). *Parallel programming with MPI.* Morgan Kaufmann Publishers, Inc., San Fransisco, CA. 418 pp.

Peregrine, D. H. (1967). "Long waves on a beach." *Journal of Fluid Mechanics*, 27(4), 815–827.

Phillips, N. A. (1956). "The general circulation of the atmosphere: A numerical experiment." *Quarterly Journal of the Royal Meteorological Society*, 82(352), 123–164.

Press, H. W., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (1992). *Numerical Recipe in Fortran 77.* Cambridge University Press, Cambridge, 2nd edition.

Saad, Y. (2003). *Iterative Method for Sparse Linear Systems.* SIAM, USA. 528 pp.

Saville, T. J. (1955). "Laboratory data on wave run-up and overtopping on shore structures." *Tech. Memo. 64*, U.S. Army, Beach Erosion Board, Document Service Center, Dayton, Ohio.

Shapiro, R. (1970). "Smoothing, filtering, and boundary effects." *Reviews of Geophysics and Space Physics*, 8(2), 359–387.

Snir, M., Ottto, S. W., Lederman, S. H., Walker, D. H., and Dongarra, J. (1996). *MPI The Complete Reference*. The MIT Press, Cambridge: Massachussets Institute of Technology.

Veeramony, J. and Svendsen, I. A. (2000). "The flow in surf-zone waves." *Coastal Engineering*, 39, 93–122.

Vidal, C. and Losada, I. (2007). *Wave propagation over low crested structures (LCS)*. http://www.iahr.net/site/e_library/links/databases/CASE1/UCA_Tests2.doc. (accessed on 10/31/2007).

Vincent, C. L. and Briggs, M. J. (1989). "Refraction-diffraction of irregular waves over a mound." *Journal of Waterway, Port, Coastal and Ocean Engineering*, ASCE, 115(2), 269–284.

Wang, H. H. (1981). "A parallel method for tridiagonal equations." *ACM Transactions on Mathematical Software*, 7(2), 170–183.

Wei, G. and Kirby, J. T. (1995). "A time-dependent numerical code for extended Boussinesq equations." *Journal of Waterway, Port, Coastal, and Ocean Engineering*, ASCE, 121(5), 251–261.

Wei, G., Kirby, J. T., Grilli, S. T., and Subramanya, R. (1995). "A fully nonlinear Boussinesq model for surface waves. Part 1. Highly nonlinear unsteady waves." *Journal of Fluid Mechanics*, 294, 71–92.

Yuk, D., Yim, S. C., and Liu, P. L.-F. (2006). "Numerical modeling of submarine mass-movement generated waves using RANS model." *Computers & Geosciences*, 32(7), 927–935.

Zhao, Q., Armfield, S., and Tanimoto, K. (2004). "Numerical simulation of breaking waves by a multi-scale turbulence model." *Coastal Engineering*, 51(1), 53–80.

VITA

Khairil Irfan Sitanggang was born in Medan, Indonesia. He graduated from the Insitut Teknologi Bandung with his B.S. degree in Civil Engineering in April 1996, and his M.S. degree in July 2000. In Summer 2001, he attended Texas A&M University and received his M.S. degree from the Civil Engineering Department in December 2003. Then he continued to the doctoral program in the same department, working on the parallel Boussinesq and RANS hybrid modeling. He graduated with his Ph.D. degree in Ocean Engineering in May 2008.

Khairil Irfan Sitanggang may be contacted at the Coastal/Ocean Engineering Program, Zachry Department of Civil Engineering, Texas A&M University College Station, TX 77843 or his future address at the Ocean Engineering Department, Institut Teknologi Bandung, Labtek VI, 3rd Floor, Bandung, Indonesia.