INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2005; **49**:57–74
Published online 3 June 2005 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/fld.985

Parallel computation of a highly nonlinear Boussinesq equation model through domain decomposition

Khairil Irfan Sitanggang[‡] and Patrick Lynett[*,†]

Coastal and Ocean Engineering Division, Department of Civil Engineering, Texas A&M University, College Station, TX, U.S.A.

SUMMARY

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 is 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 is very satisfactory. A linear speedup is gained as the number of processors increases. These tests showed that the CPU time efficiency of the model is about 75–90%. Copyright © 2005 John Wiley & Sons, Ltd.

KEY WORDS: parallel Boussinesq model; tridiagonal solver; high-order scheme

1. INTRODUCTION

The calculation of wave propagation from deep to shallow water has long been a challenging problem among ocean/coastal engineers and scientists. As waves propagate 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.

*Correspondence to: Patrick Lynett, Coastal and Ocean Engineering Division, Department of Civil Engineering, Texas A&M University, College Station, TX, U.S.A.
†E-mail: plynett@tamu.edu
‡E-mail: irfan@tamu.edu

Contract/grant sponsor: National Science Foundation; contract/grant number: 0427014

Received 15 December 2004
Revised 23 March 2005
Accepted 1 April 2005

The Boussinesq equation model has been used for decades to simulate wave propagation from relatively deep to shallow water. Peregrine [1] derived the 'conventional', depth-averaged, Boussinesq equation that 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 wavenumber 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 [2] and Nwogu [3]. By modifying the depth-averaged Boussinesq model through manipulations of the dispersive terms, Madsen and Sørensen [2] created a model with good accuracy through the intermediate water regime. Nwogu [3] 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 $kh \sim 3$. While these works increased dispersive accuracy, they are still limited by the weakly nonlinear assumption.

Wei *et al.* [4] derived a highly nonlinear Boussinesq equation model by keeping nonlinear-dispersive terms in the model, which were truncated by Nwogu [3]. This model of Wei *et al.*, exhibits excellent linear dispersive properties to $kh \sim 3$, while shoaling, wave kinematics, and nonlinear interactions are generally well captured to $kh \sim 1$. A number of researchers have made modifications and enhancements to this model, including Kennedy *et al.* [5] to optimize the model nonlinearity and Lynett and Liu [6], 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 [6] use the high order finite difference method given by Wei *et al.* [4], however with slight differences in how some of the nonlinear dispersive terms are treated.

Further increasing the deep-water accuracy of the Boussinesq-type model are a number of 'high-order' derivations. Gobbi *et al.* [7] extended the model of Wei *et al.* [4] to the next order in $(kh)^2$, doubling the linear dispersion accuracy to $kh \sim 6$. Madsen *et al.* [8], building off the derivation of Agnon *et al.* [9], used multiple expansions at various elevations leading to a model with linear and nonlinear accuracy to $kh \sim 40$. Lynett and Liu [10, 11] created a 'multi-layer' concept, wherein the water column is 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 ($O(10\,\mathrm{km}^2)$), 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 sub-task 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 ($O(100\,\mathrm{km}^2)$) 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 [12].

This paper is organized as follows. Section 2 presents the highly nonlinear Boussinesq equations that govern wave propagation, followed by Section 3 that gives the finite difference discretization of the corresponding governing equations. In Section 4, the parallelization strategy is explained in detail. In Section 5, both the validity and performance of the parallel model are tested using idealized cases, where analytical solutions or experimental data exist.

2. GOVERNING EQUATIONS

The parallel Boussinesq model developed in this paper is based on its serial counterpart as can be found in Reference [6]. The governing equations that are used in this serial model (and also in this paper) consist of the two-dimensional depth-integrated continuity equation:

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

and the horizontal momentum equation:

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

where $S = \nabla\cdot\mathbf{u}_\alpha$, $T = \nabla\cdot(h\mathbf{u}_\alpha) + \partial h/\partial t$, h is depth, η is free surface elevation, $H = h + \eta$, $\mathbf{u}_\alpha$ is horizontal velocity vector, z_α is reference depth, and t is 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 [3]. Equations (1) and (2) differ from the equations given by Wei *et al.* [4] 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.

3. FINITE DIFFERENCE SOLUTION

The finite difference solution of the governing equations (1) and (2) is given in Lynett and Liu [6], which is based on the formulation presented in Wei *et al.* [4]. The finite difference scheme consists of the third-order in time explicit Adams–Bashford predictor step and fourth-order in time implicit Adams–Bashford corrector step [13]. The spatial derivatives in (1) and (2) are evaluated to fourth-order accuracy. Details of the finite difference method can be found in Reference [6]. Here, for convenience, the corresponding finite difference discretization is

given. The explicit predictor equations are

$$\eta_{i,j}^{n+1} = \eta_{i,j}^{n} + \tfrac{1}{12}\,\Delta t(23E_{i,j}^{n} - 16E_{i,j}^{n-1} + 5E_{i,j}^{n-2}) \tag{3a}$$

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

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

and the implicit corrector equations:

$$\eta_{i,j}^{n+1} = \eta_{i,j}^{n} + \tfrac{1}{24}\,\Delta t(9E_{i,j}^{n+1} + 19E_{i,j}^{n} - 5E_{i,j}^{n-1} + E_{i,j}^{n-2}) \tag{4a}$$

$$U_{i,j}^{n+1} = U_{i,j}^{n} + \tfrac{1}{24}\,\Delta t(9F_{i,j}^{n+1} + 19F_{i,j}^{n} - 5F_{i,j}^{n-1} + F_{i,j}^{n-2}) + 2(F_1)_{i,j}^{n} + (F_1)_{i,j}^{n+1} - (F_1)_{i,j}^{n} \tag{4b}$$

$$V_{i,j}^{n+1} = V_{i,j}^{n} + \tfrac{1}{24}\,\Delta t(9G_{i,j}^{n+1} + 19G_{i,j}^{n} - 5G_{i,j}^{n-1} + G_{i,j}^{n-2}) + 2(G_1)_{i,j}^{n} + (G_1)_{i,j}^{n+1} - (G_1)_{i,j}^{n} \tag{4c}$$

where

$$U = u + \tfrac{1}{2}\,(z_\alpha^2 - \eta^2)u_{xx} + (z_\alpha - \eta)(hu)_{xx} - \eta_x[\eta u_x + (hu)_x] \tag{5}$$

$$V = v + \tfrac{1}{2}\,(z_\alpha^2 - \eta^2)v_{yy} + (z_\alpha - \eta)(hv)_{yy} - \eta_y[\eta v_y + (hu)_y] \tag{6}$$

$$\begin{aligned} E = {} & -h_t - [(\eta + h)u]_x - [(\eta + h)v]_y \\ & + \{(\eta + h)[\tfrac{1}{6}\,(\eta^2 - \eta h + h^2) - \tfrac{1}{2}\,z_\alpha^2]S_x + (\tfrac{1}{2}\,(\eta - h) - z_\alpha)T_x\}_x \\ & + \{(\eta + h)[\tfrac{1}{6}\,(\eta^2 - \eta h + h^2) - \tfrac{1}{2}\,z_\alpha^2]S_y + (\tfrac{1}{2}\,(\eta - h) - z_\alpha)T_y\}_y \end{aligned} \tag{7}$$

$$\begin{aligned} F = {} & -\tfrac{1}{2}\,[(u^2)_x + (v^2)_x] - g\eta_x - z_\alpha h_{xtt} - z_{\alpha t}h_{xt} + (\eta h_{tt})_x - [E(\eta S + T)]_x \\ & -[\tfrac{1}{2}\,(z_\alpha^2 - \eta^2)(uS_x + vS_y)]_x - [(z_\alpha - \eta)(uT_x + vT_y)]_x - \tfrac{1}{2}\,[(T + \eta S)^2]_x \end{aligned} \tag{8}$$

$$F_1 = \tfrac{1}{2}\,(\eta^2 - z_\alpha^2)v_{xy} - (z_\alpha - \eta)(hv)_{xy} + \eta_x[\eta v_y + (hv)_y] \tag{9}$$

$$\begin{aligned} G = {} & -\tfrac{1}{2}\,[(u^2)_y + (v^2)_y] - g\eta_y - z_\alpha h_{ytt} - z_{\alpha t}h_{yt} + (\eta h_{tt})_y - [E(\eta S + T)]_y \\ & -[\tfrac{1}{2}\,(z_\alpha^2 - \eta^2)(uS_x + vS_y)]_y - [(z_\alpha - \eta)(uT_x + vT_y)]_y - \tfrac{1}{2}\,[(T + \eta S)^2]_y \end{aligned} \tag{10}$$

$$G_1 = \tfrac{1}{2}\,(\eta^2 - z_\alpha^2)u_{xy} - (z_\alpha - \eta)(hu)_{xy} + \eta_y[\eta u_x + (hu)_x] \tag{11}$$

$$S = u_x + v_y, \quad T = (hu)_x + (hv)_y + h_t \tag{12}$$

The procedure to solve governing equations (1) and (2) is to first predict the solution (η^{n+1}, u^{n+1}, and v^{n+1}) via the explicit predictors (3), then solving (5) and (6) 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 (5) and (6), tridiagonal systems of linear equations must be solved. Next, the predicted values must be

iterated using the implicit correctors (4) until the solution converges. During each iteration of the corrector step, the tridiagonal systems of (5) and (6) must also be solved. For the iteration to halt, the maximum local relative error, which is defined as

$$\left| \frac{w^{n+1} - w_*^{n+1}}{w^{n+1}} \right| \tag{13}$$

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

4. PARALLELIZATION STRATEGY

The higher order finite difference scheme [6] 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 1). 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 (5) and (6). Here, a tridiagonal system of linear equations must be solved for each row of the computational matrix 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 [13] 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 carried out 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 [14] and difficult to efficiently parallelize.

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 (3) and corrector (4) steps are highly parallelizable. The tridiagonal system of the linear equations (5) and (6) that arises from the finite difference scheme is not

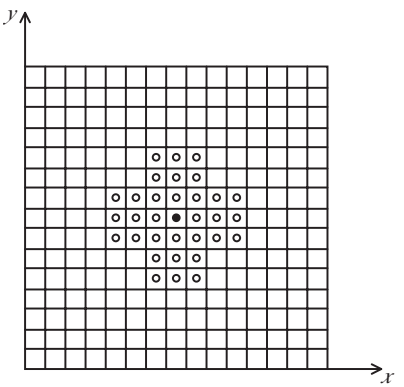

Figure 1. Finite difference stencil of the higher order finite difference solution of the Boussinesq equation.

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 2.

4.1. Domain decomposition

The physical/computational domain which is used in References [4, 6], and in this paper is rectangular in shape. In the domain decomposition method, the rectangular domain is divided into several smaller rectangular sub-domains, where the number of sub-domains 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 3. 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 1000 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 1, it is apparent that the computation at an arbitrary point requires values from at most three nodes from the left, right, bottom, and top. Nodes located within three 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 sub-domain is increased by three *imaginary* nodes in all directions. This is manifested in the sizes of all the related arrays.

4.2. 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 inter-processor communication occurring when the parallel model solves the tridiagonal systems of linear equations. The latter will be explained in the next section.

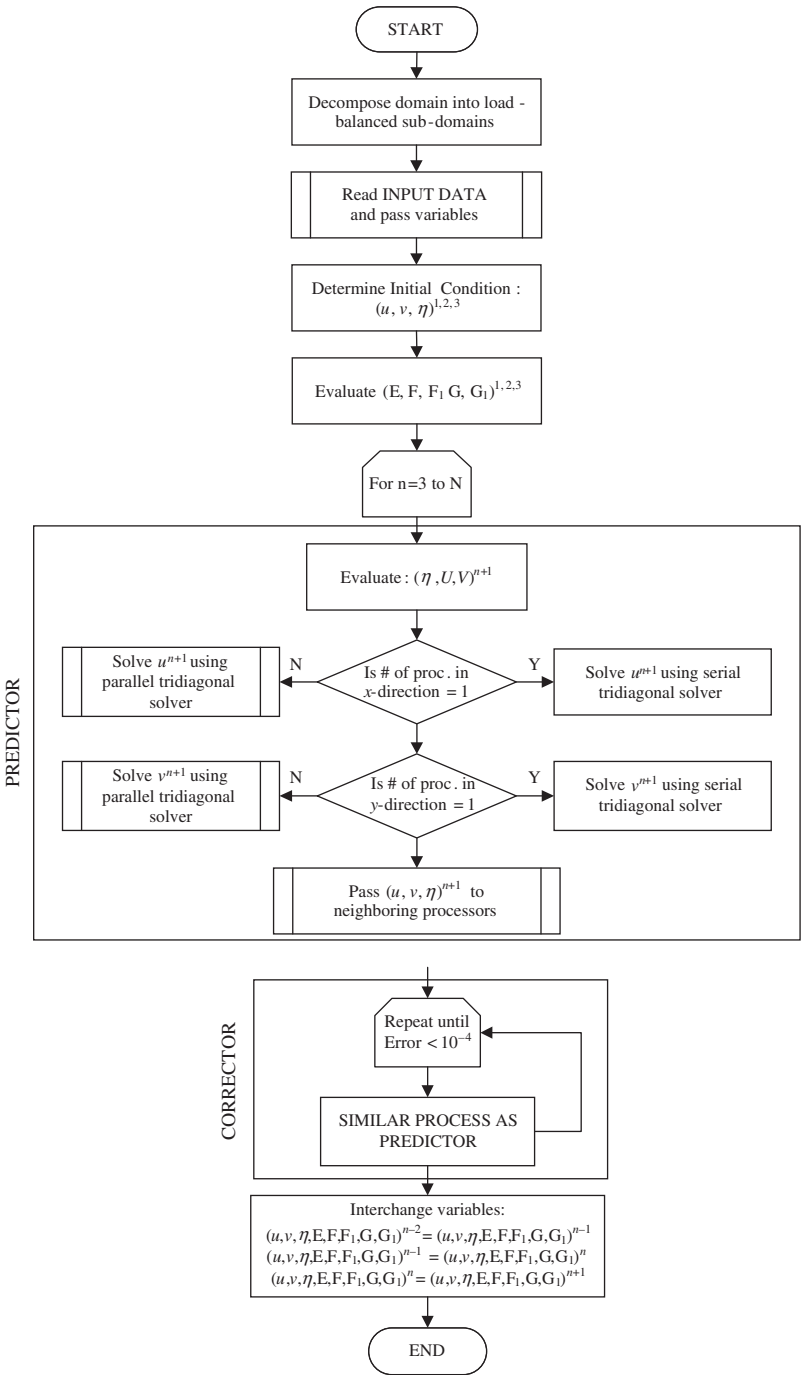

Figure 2. Parallel algorithm flowchart. Rectangle with double left/right boundary involves communication between processors.

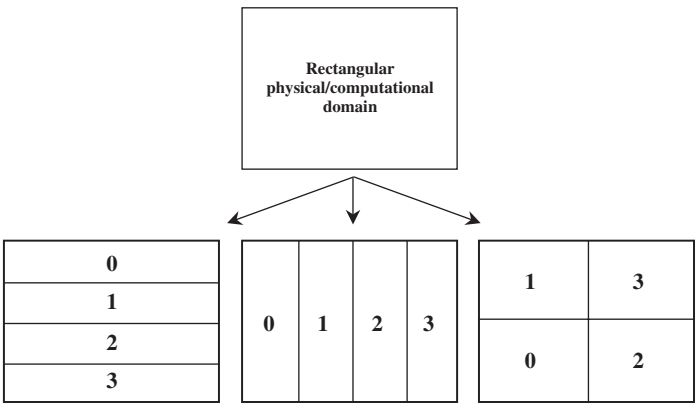

Figure 3. Three possible ways of decomposing a rectangular domain. The numbers in the sub-domains represent the processor numbers.

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

To achieve a safe communication process, we use the non-blocking communication `mpi_isend` and `mpi_ireceive`, 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 non-blocking communication can prevent the system from 'memory starving' that may cause deadlock. With a large number of grids, the use of non-blocking communication may potentially improve the performance of a parallel program [12].

Since the problem at hand is two-dimensional, the communication takes place in both the x- and y-direction (Figure 4). In this parallel model, the communication in the x-direction is first carried out 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 $3 \times \mathrm{n}y$ points ($\mathrm{n}y =$ number of 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 $3 \times (\mathrm{n}x + 3)$ points ($\mathrm{n}x =$ number of grids in the x-direction in one processor).

4.3. Parallel solver of tridiagonal system

The evaluation of u^{n+1}, and v^{n+1} in (5) and (6) 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 sub-domains in one or both directions, the

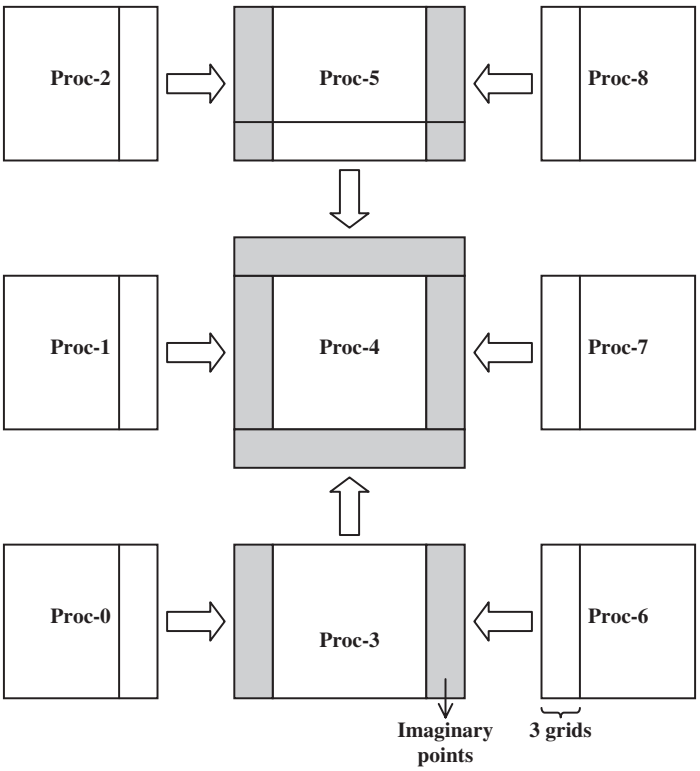

Figure 4. Message passing scheme in parallel Boussinesq model.

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 ([15–17] among others).

To solve for u and v from U and V in (5) and (6), which is the primary challenge to parallelize this model, we use the algorithm proposed by Mattor *et al.* [17]. 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:

$$\mathbf{x}_p = \mathbf{x}_p^{\mathrm{R}} + \xi_p^{\mathrm{UH}} \mathbf{x}_p^{\mathrm{UH}} + \xi_p^{\mathrm{LH}} \mathbf{x}_p^{\mathrm{LH}} \tag{14}$$

where $\mathbf{x}_p$ is the solution of the system, $\mathbf{x}_p^{\mathrm{R}}$, $\mathbf{x}_p^{\mathrm{UH}}$, and $\mathbf{x}_p^{\mathrm{LH}}$ are the particular, upper homogeneous, and lower homogeneous solutions of the inhomogeneous differential equation analogy and ξ_p^{UH}, ξ_p^{LH} are the coefficients which depend on coupling to the neighbouring solutions.

The subscript p indicates that the corresponding solution is local to processor p. Detail of the procedure is given in Mattor *et al.* [17].

Prior to evaluating the coefficients ξ_p^{UH}, ξ_p^{LH}, the first and last elements of $\mathbf{x}_p^{\mathrm{UH}}$, $\mathbf{x}_p^{\mathrm{LH}}$, and $\mathbf{x}_p^{\mathrm{R}}$, of all processors are concatenated to form a $(2P-2)\times(2P-2)$ tridiagonal system with ξ_p^{UH}, ξ_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 employ the collective communication routine '`mpi_allgather`' of MPI with 'OutData' variable (see Reference [17]), which carries the first and last elements of $\mathbf{x}_p^{\mathrm{UH}}$, $\mathbf{x}_p^{\mathrm{LH}}$, and $\mathbf{x}_p^{\mathrm{R}}$, 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 receive 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 [18].

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 are first evaluated, followed by distribution of the corresponding $\mathbf{x}_p^{\mathrm{UH}}$'s, $\mathbf{x}_p^{\mathrm{LH}}$'s, and $\mathbf{x}_p^{\mathrm{R}}$'s to all processors using the collective communication, and completed through evaluation of the final solution via (14). 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.

5. PARALLEL MODEL TESTING

The present parallel model is tested for both accuracy and performance. To examine accuracy, the linear and weakly nonlinear versions of the model are tested using two idealized scenarios having known analytical solutions. The first idealized case is linear wave evolution in a closed rectangular wave basin. The parallel model is run under the same condition as the idealized case and the result is 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 is considered. The nonlinear mode of the parallel model is expected to produce a solitary wave propagating along the channel with no change in wave form.

To test performance, the parallel model is 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 (1) and (2)). The model is run using different numbers of processors and the run time for each run is recorded to observe scalability of the model. As final test of the parallel model, the experimental setup of Vincent and Briggs [19], for a regular wave propagating over a 3D shoal, is simulated. The experimental setup is known to be very nonlinear (e.g. [10]). This practical application of the model will utilize the highly nonlinear equations, and the efficiency of the model is discussed.

The computer system used for testing the accuracy and performance of the parallel model is 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 [19] comparison, an additional, small, cluster will be used for benchmarking. This small cluster consists of 8 2.2 GHz Opteron processors in 4 2-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 are: $-fastsse-O4-tp=amd64$.

5.1. Model accuracy test

5.1.1. Wave evolution in a closed rectangular wave basin. In this idealized case, the wave evolution in a closed rectangular wave basin of constant depth is considered. The physical setup of this test is similar to the one used in Reference [20]. The wave basin is a square 7.5 m × 7.5 m basin with a constant depth of 0.45 m. The initial wave profile is a Gaussian hump shape profile

$$\eta_0 = H_0 \mathrm{e}^{-2[(x-3.75)^2+(y-3.75)^2]} \tag{15}$$

where η_0 is the initial free surface elevation, H_0 is wave amplitude ($=0.45$ m in this test) and the initial velocity is zero. The wave basin wall is an impermeable and reflecting wall. Detail regarding the numerical implementation of this boundary condition can be found in Wei *et al.* [4].

For this setup, the parallel linear model is run for 50 s of model time. The spatial and time grid sizes for the run are 0.075 m and 0.0143 s, respectively, a total of 100 grids along both x- and y-sides and 3500 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 5. The temporal variation of the free surface elevation at the centre of the basin, $x = 3.75$, $y = 3.75$ m, is recorded and compared with the one calculated by the analytical solution. This comparison is given in Figure 6. The temporal free surface elevations calculated by the parallel model agree very well with the one calculated by the analytical model. The parallel model works 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 results in the best performance. With such configurations as 16×1 and 8×2, even if the load of arithmetic operations involved in each processor is equal to that in the 4×4 configuration, the communication load in the previous two is heavier than in the 4×4. Comparing the 4×4 parallel and serial run times, we gain a speedup of 3.7 or an efficiency of 23%. With a small number of grids ($\mathrm{n}x = 100$ and $\mathrm{n}y = 100$), the cost of communication is more expensive than the local arithmetic operational cost, hence results in small efficiency. This efficiency, as will be shown later, will increase as the number of grids increases.

5.1.2. Solitary wave propagation along 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 [20]. The velocity and the surface elevation of the solitary wave are analytically given by

$$u = A \operatorname{sech}^2[B(x - Ct)] \tag{16}$$

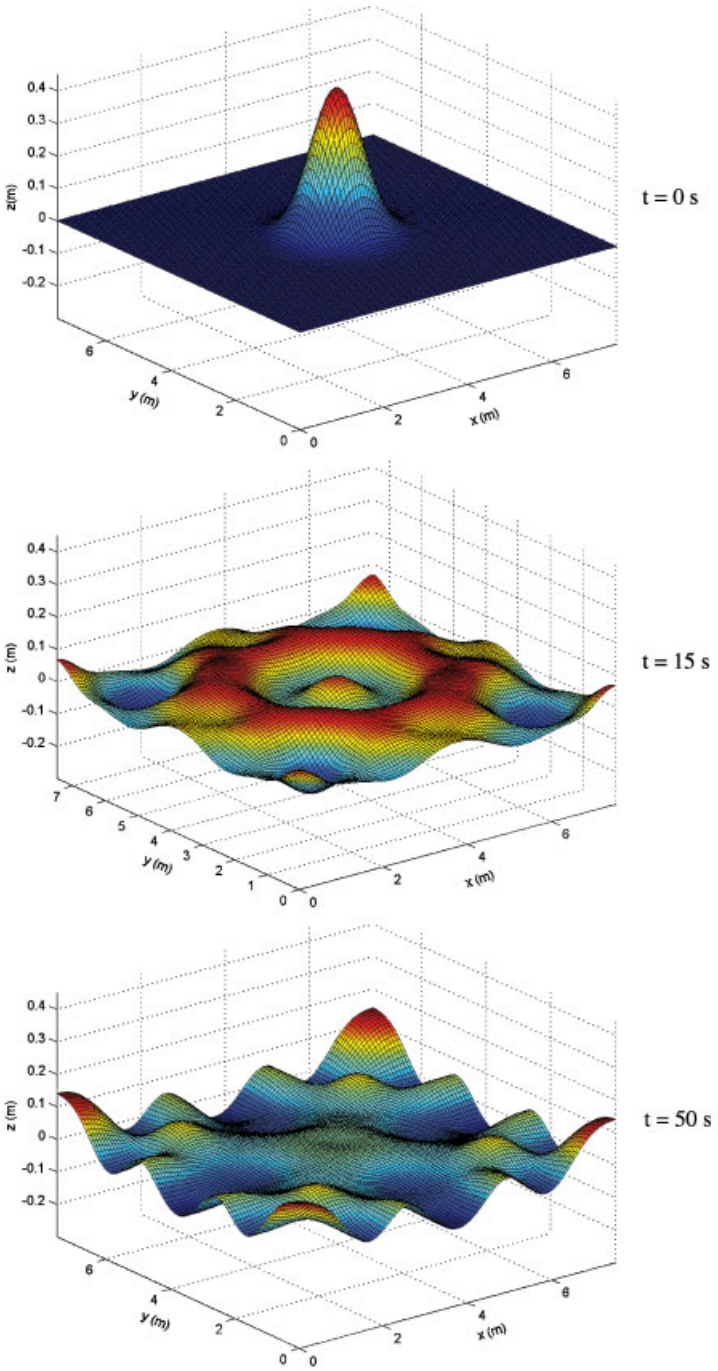

Figure 5. Linear Gaussian-wave profiles at three different times calculated using 16 processors.

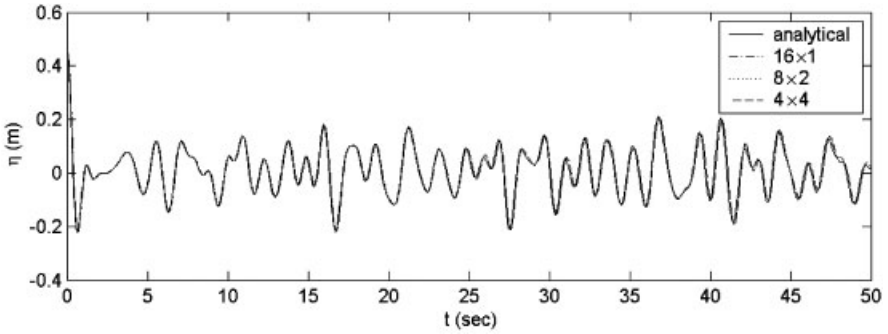

Figure 6. Temporal variation of free surface elevation at the centre of the wave basin calculated by the analytic and the parallel numerical models using 16 processors.

$$\eta = A_1 \operatorname{sech}^2[B(x - Ct)] + A_2 \operatorname{sech}^4[B(x - Ct)] \tag{17}$$

where A, B, C, A_1, and A_2 are constants that depend on the physical setup of the model [20]. In this case, the channel depth is 0.45 m, the wave amplitude is 0.04 m, and the length of the channel is 450 m. The 1-HD domain is discretized into 1500 equally spaced computational grids, each is 0.3 m long. The wave is initially located at $x = 80$ m.

The parallel weakly nonlinear model is run for 200 s using 16 processors. In the course of its propagation, the wave at $t = 0, 40, 80, 120$, and 160 s, are recorded. These snapshots are given in Figure 7. This figure shows that the numerically-calculated solitary wave propagates in the positive x-direction with constant speed, i.e. the distances between two consecutive wave forms are constant, and wave height and length and agrees well with the analytic solution. This example also clearly shows that information is passed correctly from sub-domain to sub-domain.

5.2. *Performance test*

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

The model is run under three different modes: linear, weakly nonlinear, and fully nonlinear. In all parallel runs, even numbers of processors are used: $2, 4, 6, 8, 10, 12, 16, 18, 20, 24, 30,$ and 32 processors. Figure 8 shows the speedup and efficiency of the parallel calculation for different numbers of processors used. Here, the speedup is defined as the ratio of the parallel run-time to the run time of the serial version of the Boussinesq model (using a single grid and the Thomas algorithm to solve the tridiagonal systems). Figure 8 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

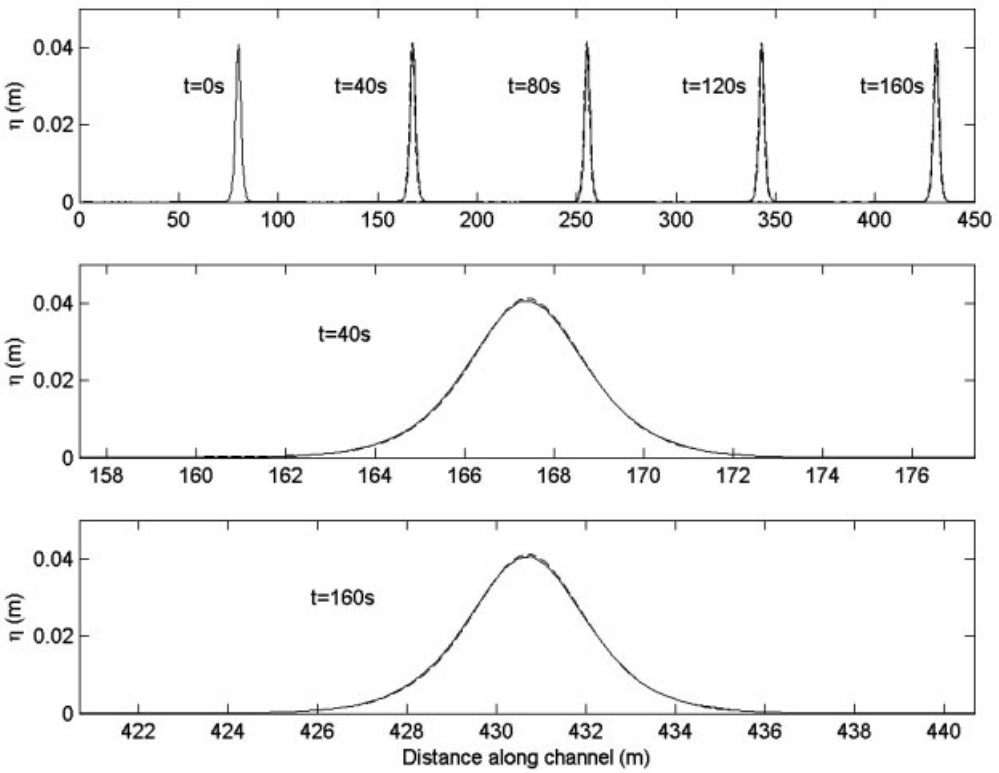

Figure 7. Solitary wave propagation along a long straight channel (full line = analytic solution and dashed line = parallel model) calculated using 16 processors.

Table I. Domain setup for parallel model performance test.

Test	LX* (m)	LY† (m)	NX‡	NY§
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.

model improves as the number of grids increases; a favourable 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 sub-grid sizes of 200×200 or greater.

Finally, simulation of a 3D experimental setup is presented. One of the most frequently studied 3D water wave problems is that of wave interaction with a submerged shoal (e.g. Reference [21]). Here, one of the experiments of Vincent and Briggs [19] will be numerically

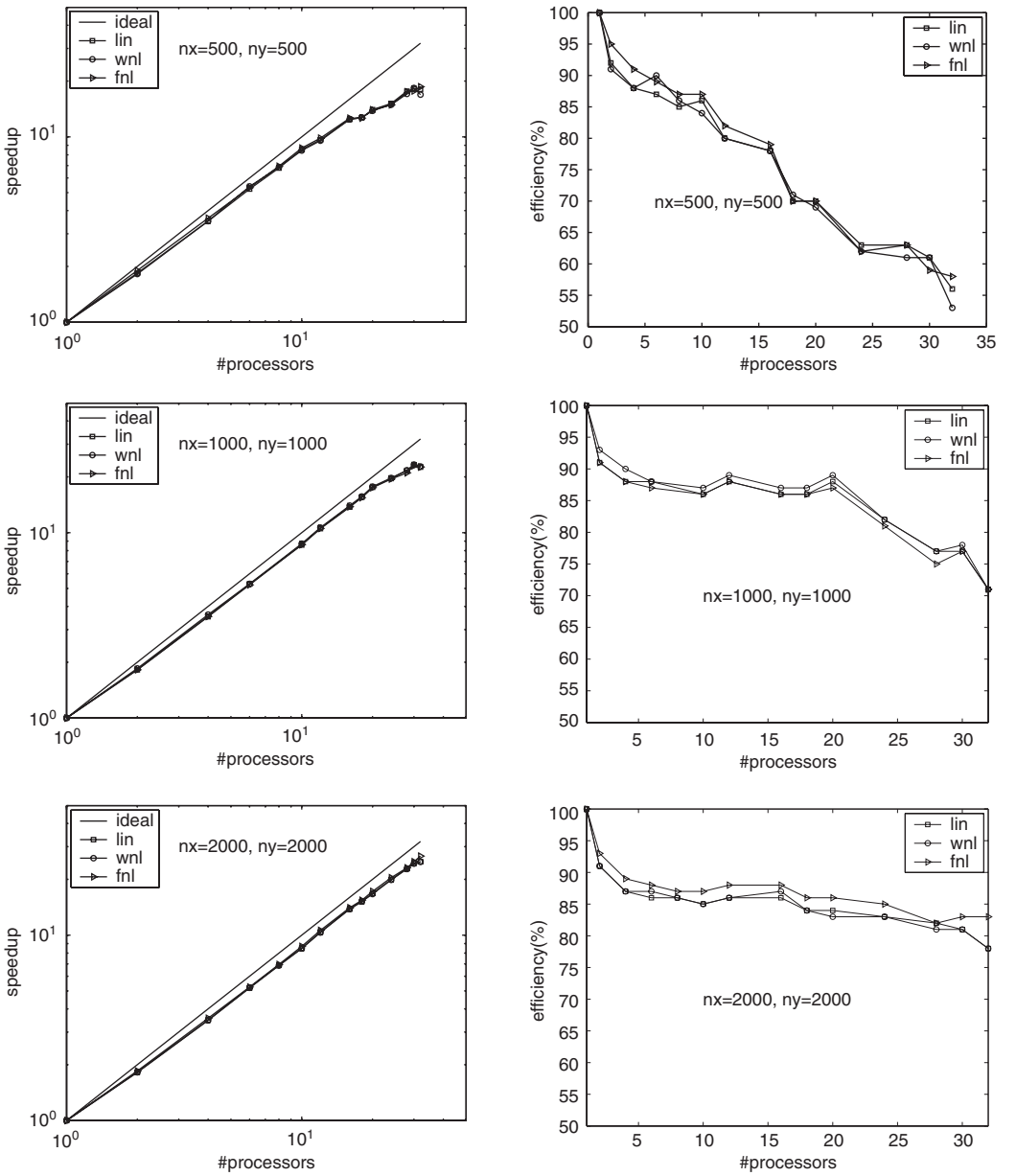

Figure 8. Parallel computational speedup/efficiency of the parallel Boussinesq model in computing the evolution of the Gaussian-wave in a rectangular basin. The squares are from the linear model, the circles from the weakly nonlinear model, and the triangles from the highly nonlinear model.

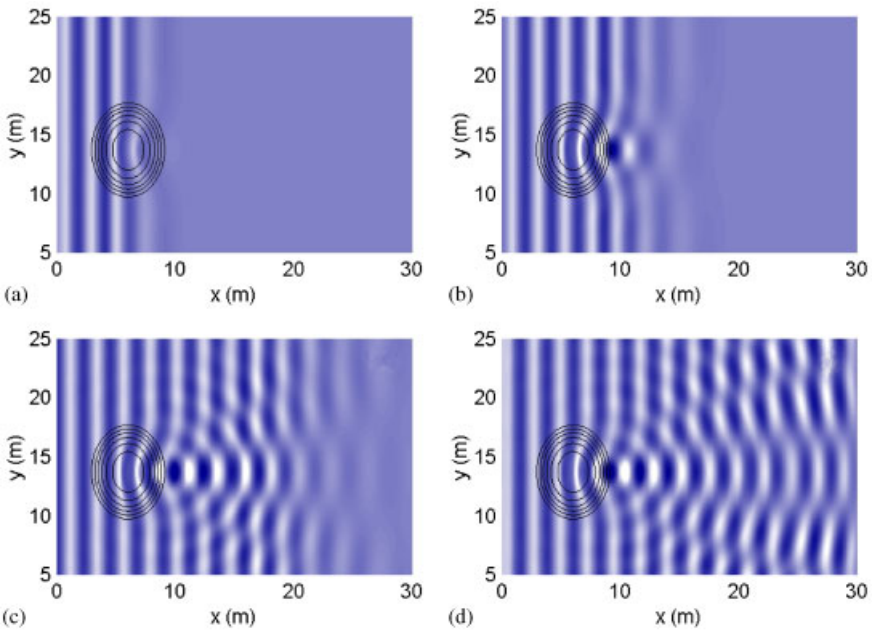

Figure 9. Plan view of regular wave propagation (period = 1.3 s, height = 4.8 cm) over a submerged shoal at: (a) time = 6 s; (b) time = 10 s; (c) time = 18 s; and (d) time = 50 s. The shoal location is given by the contours. Simulation uses 8 CPUs.

recreated. An elliptic shoal 6.1 m long and 7.92 m wide is placed in a wave tank that is 35 m wide and 29 m long. The shoal has a maximum height of 30.5 cm in 45.7 cm of water. The exact mathematical representation of the shoal can be found in Reference [19]. While many different wave conditions were examined experimentally, here only a single regular wave case is simulated with period = 1.3 s and height = 4.8 cm. The simulations use the highly nonlinear set of the Boussinesq equations.

Figure 9 gives snapshots of the waves as they transform over the shoal. The waves narrow and steepen as they move over the shoal, while refraction focuses 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 [10] for the 'one-layer' model (equivalent to the equations of Wei *et al.* [4]), exhibiting very good agreement. Hence, these identical comparisons will not be included here as well.

Tables II and III give the wall clock time per simulated wave period and efficiency of the parallel model, on two platforms for two different grid sizes. Table II shows the results using 40 grid points per incident wavelength, while the values in Table III use 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

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

	Opteron cluster		Itanium-2 cluster	
# CPUs	Wall clock time (s)/ wave period	Efficiency	Wall clock time (s)/ wave period	Efficiency
1	362		2888	
2	187	0.97	1586	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

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

	Opteron cluster		Itanium-2 cluster	
# CPUs	Wall clock time (s)/ wave period	Efficiency	Wall clock time (s)/ wave period	Efficiency
1	3994		20670	
2	2034	0.98	11351	0.91
4	1005	0.99	5802	0.89
6	686	0.97	4424	0.78
8	539	0.93	3332	0.78
16			1715	0.75
32			858	0.75

times for the two platforms, with the superior floating-point capabilities of the AMD chips showing their strength.

6. CONCLUSION

In the present work, the Boussinesq model of Wei *et al*. [4] is parallelized using the domain decomposition method, where each processor performs the same operations. The parallel algorithm is identical to its serial counterpart, based on an iterative predictor–corrector scheme also requiring a tridiagonal solution for each iteration. The model test indicates 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 computation 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 $O(100\,\mathrm{km}^2)$ and greater basins. Additionally, real-time simulation with Boussinesq equations becomes a possibility.

ACKNOWLEDGEMENT

This material is based upon work supported by the National Science Foundation under Grant No. 0427014.

REFERENCES

1. Peregrine DH. Long waves on a beach. *Journal of Fluid Mechanics* 1967; **27**(4):815–827.
2. Madsen PA, Sørensen OR. A new form of the Boussinesq equations with improved linear dispersion characteristics. Part 2. A slowly-varying bathymetry. *Coastal Engineering* 1992; **18**:183–204.
3. Nwogu O. Alternative form of Boussinesq equations for nearshore wave propagation. *Journal of Waterway, Port, Coastal, and Ocean Engineering* 1993; **119**(6):618–638.
4. Wei G, Kirby JT, Grilli ST, Subramanya R. A fully nonlinear Boussinesq model for surface waves. Part 1. Highly nonlinear unsteady waves. *Journal of Fluid Mechanics* 1995; **294**:71–92.
5. Kennedy AB, Kirby JT, Chen Q, Dalrymple RA. Boussinesq type equations with improved nonlinear behavior. *Wave Motion* 2001; **33**:225–243.
6. Lynett P, Liu PL-F. A numerical study of submarine-landslide-generated waves and run-up. *Proceedings of the Royal Society of London A* 2002; **458**:2885–2910.
7. Gobbi MF, Kirby JT, Wei G. A fully nonlinear Boussinesq model for surface waves. Part II. Extension to $O(kh)^4$. *Journal of Fluid Mechanics* 2000; **405**:182–210.
8. Madsen PA, Bingham HB, Liu H. A new Boussinesq method for fully nonlinear waves from shallow to deep water. *Journal of Fluid Mechanics* 2002; **462**:1–30.
9. Agnon Y, Madsen PA, Schaffer H. A new approach to high order Boussinesq models. *Journal of Fluid Mechanics* 1999; **399**:319–333.
10. Lynett P, Liu PL-F. A two-layer approach to water wave modeling. *Proceedings of the Royal Society of London A* 2004a; **460**:2637–2669.
11. Lynett P, Liu PL-F. Linear analysis of the multi-layer model. *Coastal Engineering* 2004b; **51**(6):439–454.
12. Snir M, Otto SW, Lederman SH-, Walker DW, Dongarra J. *MPI The Complete Reference*. The MIT Press Cambridge: Massachussets. ©1996 Massachussets Institute of Technology, 1996.
13. Press HW, Teukolsky SA, Vetterling WT, Flannery BP. *Numerical Recipes in Fortran 77* (2nd edn), ©1992 Cambridge University Press: Cambridge, 1992.
14. Hockney RW, Jesshope CR. *Parallel Computers*: *Architecture*, *Programming and Algorithms*, ©1981 Adam Hilger Ltd.: Bristol, 1981.
15. Wang HH. A parallel method for tridiagonal equations. *ACM Transactions on Mathematical Software* 1981; **7**(2):170–183.
16. Krechel A, Plum HJ, Stben K. Solving tridiagonal linear systems in parallel on local memory MIMD machines. *GMD Technical Report*, vol. 372, 1989.
17. Mattor N, Williams TJ, Hewett DW. Algorithm for solving tridiagonal matrix problems in parallel. *Parallel Computing* 1995; **21**:1769–1782.
18. Pacheco PS. *Parallel Programming with MPI*. ©1997 Morgan Kaufmann Publishers, Inc.: New York, 1997.
19. Vincent CL, Briggs MJ. Refraction–diffraction of irregular waves over a mound. *Journal of Waterway, Port, Coastal and Ocean Engineering* 1989; **115**:269–284.
20. Wei G, Kirby JT. A time-dependent numerical code for extended Boussinesq equations. *Journal of Waterway, Port, Coastal and Ocean Engineering* 1995; **121**:251–261.
21. Berkhoff JCW, Booij N, Radder AC. Verification of numerical wave propagation models for simple harmonic linear water waves. *Coastal Engineering* 1982; **6**(3):255–279.

