INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2006; **52**:213–235
Published online 10 February 2006 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/fld.1175

Solving a fully nonlinear highly dispersive Boussinesq model with mesh-less least square-based finite difference method

Benlong Wang[‡] and Hua Liu[*,†]

School of Naval Architecture, Ocean and Civil Engineering, Shanghai Jiao Tong University, 200030, Shanghai, China

SUMMARY

Combining mesh-less finite difference method and least square approximation, a new numerical model is developed for water wave propagation model in two horizontal dimensions. In the numerical formulation of the method, the approximation of the unknown functions and their derivatives are constructed on a set of nodes in a local circular-shaped region. The Boussinesq equations studied in this paper is a fully nonlinear and highly dispersive model, which is composed of the exact boundary conditions and the truncated series expansion solution of the Laplace equation. The resultant system involves a sparse, unsymmetrical matrix to be solved at each time step of the simulation. Matrix solutions are studied to reduce the computing resource requirements and improve the efficiency and accuracy. The convergence properties of the present numerical method are investigated. Preliminary verifications are given for nonlinear wave shoaling problems; the numerical results agree well with experimental data available in the literature. Copyright © 2006 John Wiley & Sons, Ltd.

KEY WORDS: finite difference method; least square approximation; Boussinesq equations; dispersive wave

1. INTRODUCTION

In the last decades, mesh-less or mesh-free methods have attracted great attention in the field of computational mechanics, e.g. References [1, 2] and references therein. Apart from the studies in this field, applications of mesh-free methods in the area of geophysics and coastal engineering are some of the promising extensions. For example, Reference [3] introduced the mesh-less Galerkin method in hydraulics, where the stationary, shallow water flows in rivers

*Correspondence to: Hua Liu, Department of Engineering Mechanics, Shanghai Jiao Tong University, Hua Shan Road 1954#, 200030, Shanghai, China.
†E-mail: hliu@sjtu.edu.cn
‡E-mail: wblsjtu@hotmail.com

Contract/grant sponsor: National Science Foundation of China and Doctoral Program Foundation for Higher Education from the Ministry of Education of China; contract/grant numbers: No. 10172058 and No. 2000024817

Received 10 June 2005
Revised 27 November 2005
Accepted 28 November 2005

were simulated. Most modelling of free surface flows in the region of estuaries, coastal waters and seas, including tidal waves, sediment processes, environmental hydraulics and physical oceanography, etc. concern complex geometric boundaries. The design of numerical methods to solve the associated partial differential equations must take into account the geometrical complexities of the physical region. However, it is a difficult task to handle flows with complex boundaries effectively and efficiently.

Predicting the wave climates of large, near-shore regions of irregularly shaped shorelines, such as harbours, bays and tidal inlets, is one of the major problems in coastal engineering. As one of the phase resolving methods, the Boussinesq models are widely used. The classical Boussinesq theory provides a set of evolution equations for surface water waves in the combined limit of weak nonlinearity and weak dispersion, which measure the wave height to water depth ratio and water depth to wavelength ratio, respectively. Recent development of the Boussinesq wave models fall into two categories. One is the improvement of the model to treat a great number of nonlinear phenomenon, such as wave energy transfer and sideband instability of water waves. The other one is the improvement of the dispersion characteristics allowing the model to treat propagation of waves in a larger range of deep-water depths. More details are given in the review papers [4] and [5], and references therein. In the last decades, most Boussinesq wave models were solved with the finite difference method on Cartesian mesh. Difficulties are encountered when curved shorelines or man-made constructions such as wave-breakers are considered. To satisfy local resolution requirements, uniform grid spacing may become too expensive to be used in large near-shore regions. The nested mesh technique could partially handle these resolution difficulties. However, the stair-stepped boundaries associated with rectangular grids may cause the spurious scattering of waves from each of the corners and decrease the computational accuracy [6]. It does not seem possible for this difficulty to be removed due to the essential characteristic of the rectangular mesh.

To remedy this difficulty, a curvilinear grid method with finite difference discretization can be used. This method is widely used in numerical modelling of large-scale oceanographic problems involving the nonlinear shallow water equations. However, the application of curvilinearized mesh method to Boussinesq models is rather limited in the literature. One of the main reasons is the difficulty in treating the high order (at least 3rd order, even up to 5th order) derivative terms contained in the Boussinesq equations. An example of the use of a curvilinear grid system to solve the Boussinesq-type equations is given in Reference [7], where solitary wave scattering by a vertical cylinder was studied. Reference [6] rewrote the Boussinesq models in curvilinear coordinates based on a mapping method. A feature of the mapping method is that the calculations involve the use of the Jacobian. If the mapping is nearly singular at one point of the region, the Jacobian will be close to zero with a corresponding lack of accuracy. In spite of these drawbacks, solving the equations with a curvilinear mesh dose significantly extend the application range towards engineering practices.

Apart from the curvilinear grid method, mesh-less numerical methods seem appropriate for solving the Boussinesq equations in the computational domain with complex boundaries. The most important advantage of this method lies in the flexibility of treating the complex boundary. Another one of the key advantages is the ease of adding and subtracting nodes in the existing node system, which provides much more flexibility for the local grid refinement.

In the numerical formulation of mesh-free methods, the approximation of the unknown functions or its derivatives is constructed on a set of nodes in a local circular-shaped region. To achieve the same order of accuracy as the finite difference method on a Cartesian mesh,

more nodes are needed. Consequently, the bandwidth of the sparse matrix discretized from the PDEs is slightly expanded, resulting in the unavoidable decrease in computational efficiency. In this regard, the mesh-free methods achieve geometric flexibility at the cost of computational efficiency. For large-scale problems, the numerical cost becomes a difficulty and needs to be considered. However, many novel techniques have been designed to treat this problem. For example, the combination of the general finite difference method (usually in a mesh-less node system) and the conventional difference method (usually in a Cartesian mesh) could improve the computational efficiency significantly [8]. Along another track, following [9], a completely matrix-free, approximate moving least square algorithm employed the theory of approximations has been developed [10]. This work can dramatically reduce the computational effort for the mesh-less methods.

The aim of the present work is to develop an easy-to-use numerical wave tank for the coastal engineering practice. In this paper, the fully nonlinear and highly dispersive Boussinesq equations are solved, incorporating a least squares-based finite difference method. In Section 2, the Boussinesq theory involving exact boundary conditions and a truncated series expansion solution of the Laplace equation is presented. Combining the generalized finite difference method with the least squares approximation, the finite difference method is improved from the one studied by [11]. In the original work, the selection of proper nodes around a centre point was a complicated process. To simplify this manipulation, the least square approximation is taken to replace the Gauss–Jordan algorithm used in Reference [11]. At the same time, less information about the node-to-node connection is needed with the least square approximation. The formulation and the numerical implementation are presented in Section 3. In Section 4, the performance of the numerical method is analysed for both the linear and nonlinear problems. As a preliminary application, comparisons between the numerical results and experimental data for nonlinear wave shoaling combined with refraction and diffraction are carried out. Concluding remarks are made in Section 5.

2. BOUSSINESQ EQUATIONS

The Boussinesq equations used in this paper follow the work in References [12–14], where the fully nonlinear and highly dispersive Boussinesq model has been developed and proved to be valid and effective. Consider the flow of an incompressible, inviscid fluid with a free surface. A Cartesian coordinate system is adopted, with the x- and y-axis located on the still-water plane, and the z-axis pointing vertically up-wards. The fluid domain is bounded by the seabed at $z=-h(x,y)$, and the free surface at $z=\eta(x,y,t)$, where t is the time. The kinematic and dynamic free surface conditions are

$$\frac{\partial \eta}{\partial t} = (1+\nabla\eta\cdot\nabla\eta)\widetilde{w} - \widetilde{\mathbf{U}}\cdot\nabla\eta \tag{1}$$

$$\frac{\partial \widetilde{\mathbf{U}}}{\partial t} = -g\nabla\eta - \nabla\left(\frac{\widetilde{\mathbf{U}}\cdot\widetilde{\mathbf{U}}}{2} - \frac{\widetilde{w}^2}{2}(1+\nabla\eta\cdot\nabla\eta)\right) \tag{2}$$

where

$$\widetilde{\mathbf{U}} = \langle \widetilde{U}, \widetilde{V} \rangle = \widetilde{\mathbf{u}} + \widetilde{w}\nabla\eta \tag{3}$$

Here $\tilde{\mathbf{u}} = \langle \tilde{u}, \tilde{v} \rangle$ and $\tilde{w}$ are the horizontal and vertical velocities evaluated at the free surface, $g = 9.81\ \mathrm{m/s^2}$ is the gravitational acceleration, and ∇ is the horizontal gradient operator i.e. $\nabla = \langle \partial/\partial x, \partial/\partial y \rangle$. The kinematic condition on the bottom reads as

$$w_b + \nabla h \cdot \mathbf{u_b} = 0, \quad z = -h(x, y) \tag{4}$$

The present method applies a truncated, Padé-enhanced Taylor series expansion of the velocity potential about an arbitrary level $z = \hat{z}$ in the fluid layer. To close the problem, the vertical distribution of the fluid velocity is approximated by

$$\mathbf{u}(x, y, z, t) = (1 - \alpha\nabla^2)\hat{\mathbf{u}}(x, y, t) + ((z - \hat{z})\nabla - \beta\nabla^3)\hat{w}(x, y, t) \tag{5}$$

$$w(x, y, z, t) = (1 - \alpha\nabla^2)\hat{w}(x, y, t) - ((z - \hat{z})\nabla - \beta\nabla^3)\hat{\mathbf{u}}(x, y, t) \tag{6}$$

where

$$\alpha = \frac{(z - \hat{z})^2}{2} - \frac{\hat{z}^2}{10}, \quad \beta = \frac{(z - \hat{z})^3}{6} - \frac{\hat{z}^2(z - \hat{z})}{10} \tag{7}$$

In Equations (5) and (6) the quantities $\hat{\mathbf{u}}$ and $\hat{w}$ are utility variables which have been introduced for the approximate solution of the Laplace equation. We denote Boussinesq formulations for Equations (5) and (6). With the Boussinesq formulation, the velocity components at the free surface and bottom could be easily obtained by substituting $z = \eta$ and $z = -h$. Inserting the Boussinesq formulation Equations (5) and (6) into the bottom boundary condition Equation (4) gives the following expression for the kinematic bottom condition, which relates the utility velocity variables $\hat{\mathbf{u}}$ and $\hat{w}$ to each other

$$(1 - \tfrac{2}{5}\gamma^2\nabla^2)\hat{w} + (\gamma\nabla - \tfrac{1}{15}\gamma^3\nabla^3)\hat{\mathbf{u}} + \nabla h \cdot [(1 - c\gamma^2\nabla^2)\hat{\mathbf{u}} - (\gamma\nabla - s\gamma^3\nabla^3)\hat{w}] = 0 \tag{8}$$

where $\gamma = h + \hat{z}$. Here, coefficients of the slope terms have been modified through numerical optimization with respect to the linear shoaling gradient and the optimized coefficients are $c = 0.605$ and $s = 0.016$ for $kh \leqslant 10$. Combining the definition of $\tilde{\mathbf{U}}$ (Equation (3)) and the bottom boundary condition Equation (8) results in the following linear system:

$$\mathbf{M}(\hat{u}\ \ \hat{v}\ \ \hat{w})^{\mathrm{T}} = (\tilde{U}\ \ \tilde{V}\ \ 0)^{\mathrm{T}} \tag{9}$$

where

$$\mathbf{M} = \begin{pmatrix} \mathscr{A}_{11} - \eta_x\mathscr{B}_{11} & \mathscr{A}_2 - \eta_x\mathscr{B}_{12} & \mathscr{B}_{11} + \eta_x\mathscr{A}_1 \\ \mathscr{A}_2 - \eta_y\mathscr{B}_{11} & \mathscr{A}_{22} - \eta_y\mathscr{B}_{12} & \mathscr{B}_{12} + \eta_y\mathscr{A}_1 \\ h_x\mathscr{C}_{11} + h_y\mathscr{C}_{21} + \mathscr{A}_{01} & h_x\mathscr{C}_{12} + h_y\mathscr{C}_{22} + \mathscr{A}_{02} & \mathscr{B}_0 - (h_x\mathscr{C}_{13} + h_y\mathscr{C}_{23}) \end{pmatrix}$$

Here, the subscripts x and y denote partial differentiation. Applying the Boussinesq formulas Equations (5) and (6), the operators read:

$$\mathscr{A}_{11} = 1 - \alpha\frac{\partial^2}{\partial x^2}, \quad \mathscr{A}_2 = -\alpha\frac{\partial^2}{\partial x \partial y}, \quad \mathscr{A}_{22} = 1 - \alpha\frac{\partial^2}{\partial y^2}, \quad \mathscr{A}_1 = 1 - \alpha\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)$$

$$\mathscr{B}_{11} = (\eta - \hat{z})\frac{\partial}{\partial x} - \beta\left(\frac{\partial^3}{\partial x^3} + \frac{\partial^3}{\partial x \partial y^2}\right), \quad \mathscr{B}_{12} = (\eta - \hat{z})\frac{\partial}{\partial y} - \beta\left(\frac{\partial^3}{\partial y^3} + \frac{\partial^3}{\partial x^2 \partial y}\right)$$

for the quantities at the free surface $z=\eta$ and

$$\mathscr{A}_{01}=\gamma\frac{\partial}{\partial x}-\frac{1}{15}\gamma^3\left(\frac{\partial^3}{\partial x^3}+\frac{\partial^3}{\partial x\partial y^2}\right),\quad \mathscr{A}_{02}=\gamma\frac{\partial}{\partial y}-\frac{1}{15}\gamma^3\left(\frac{\partial^3}{\partial y^3}+\frac{\partial^3}{\partial x^2\partial y}\right)$$

$$\mathscr{B}_0=1-\frac{2}{5}\gamma^2\left(\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}\right)$$

$$\mathscr{C}_{11}=1-c\gamma^2\frac{\partial^2}{\partial x^2},\quad \mathscr{C}_{21}=-c\gamma^2\frac{\partial^2}{\partial x\partial y},\quad \mathscr{C}_{12}=\mathscr{C}_{21},\quad \mathscr{C}_{22}=1-c\gamma^2\frac{\partial^2}{\partial y^2}$$

$$\mathscr{C}_{13}=\gamma\frac{\partial}{\partial x}-s\gamma^3\left(\frac{\partial^3}{\partial x^3}+\frac{\partial^3}{\partial x\partial y^2}\right),\quad \mathscr{C}_{23}=\gamma\frac{\partial}{\partial y}-s\gamma^3\left(\frac{\partial^3}{\partial y^3}+\frac{\partial^3}{\partial x^2\partial y}\right)$$

for the bottom boundary, i.e. the seabed $z=-h$.

The utility velocity components $\hat{\mathbf{u}}$ and $\hat{w}$ could be solved from Equation (9) in terms of $\widetilde{\mathbf{U}}$ and η. Having solved the utility variables, the vertical velocity at the free surface, $\widetilde{w}$, can be computed from the Boussinesq formulation Equation (6), i.e.

$$\widetilde{w}=\mathscr{A}_1\hat{w}-\mathscr{B}_{11}\hat{u}-\mathscr{B}_{12}\hat{v} \tag{10}$$

which is used to close the governing equations and forms the time stepping problem for the fully nonlinear free surface boundary conditions, Equations (1) and (2).

3. NUMERICAL METHOD

This section discusses various aspects of the numerical method used in the proceeding for solving the previously outlined system of PDEs. The system of PDEs is solved using a mesh-less least square-based finite difference method, and the numerical code is programmed in FORTRAN 90 with double precision.

3.1. Principle of the least square-based finite difference method

To obtain an explicit difference formulae, the influence polynomial is introduced in the same way as reported in Reference [11]. In general, we need $m=(q+1)(q+2)/2$ coefficients to determine the qth-order polynomial

$$P_q(x,y)=a_1+a_2x+a_3y+a_4x^2+a_5xy+a_6y^2+\cdots+a_{m-1}xy^{q-1}+a_my^q \tag{11}$$

Let $P_{q,i}(x,y)$ be the influence polynomial of order q. The value of the influence polynomial at nodal point i is

$$P_{q,i}(x,y)=\begin{cases}1\ (x_j,y_j) & \text{if } i=j\\ 0\ (x_i,y_i) & \text{if } i\neq j\end{cases} \tag{12}$$

where nodal point j is in the same element as nodal point i. The influence polynomials serve the same role as the shape functions in the finite element method, but used in a quite different context.

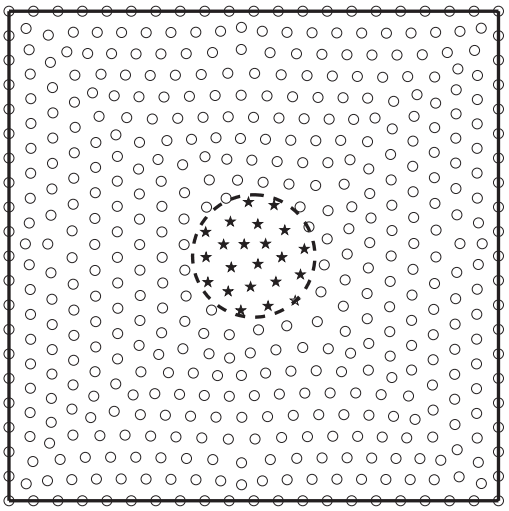

Figure 1. Distribution of the computational nodes.

We must first decide which points we want to include in the approximation. Instead of selecting m appropriate points from the $m+r$ surrounding nodes (including the centre node) in the element through the Gauss–Jordan algorithm approach [11], all the $m+r-1$ nearest nodes to the centre node (as the stars shown in Figure 1) are chosen to determine the m polynomial coefficients by the least square approximation. Herein, r is the number of additional nodal points.

To determine the coefficients of each influence polynomial $P_{q,i}(x, y)$, we put the node coordinates x_j and y_j of the selected $m+r$ nodes around the centre points i into the function $P_{q,i}(x, y)$ which forms a linear system of order $m+r$ but with m unknowns:

$$M_{(m+r)\times m} A_{m\times(m+r)} = I_{m+r} \tag{13}$$

where I_{m+r} is $(m+r)$-by-$(m+r)$ identity matrix and matrix M is

$$M = \begin{bmatrix} 1 & x_1 & y_1 & x_1^2 & x_1 y_1 & y_1^2 & \cdots & y_1^q \\ 1 & x_2 & y_2 & x_2^2 & x_2 y_2 & y_2^2 & \cdots & y_2^q \\ & & & \vdots & & & & \vdots \\ 1 & x_{m+r} & y_{m+r} & x_{m+r}^2 & x_{m+r} y_{m+r} & y_{m+r}^2 & \cdots & y_{m+r}^q \end{bmatrix} \tag{14}$$

This linear system could be solved with the least square method.

When solving the coefficients of the influence polynomials, the local matrix M will become singular or ill-conditioned for inversion if too few additional nodal points are included. This drawback is not easy to remove because little is known about the effects of node distribution on the conditioning of the local matrix. In the numerical simulation, this can be done by a trial-and-error process. In the present work, the least square technique allows an optimized

approximation, derived from an over-determined set of equations. The resultant coefficient matrix has good properties such as positive and definite. More details are given in Section 4.1 for appropriate choices of the number $m+r$.

With the influence polynomial we immediately have an interpolating polynomial for the $m+r$ node values f_i. The discretized expression of the local function f_{d} is

$$f_{\mathrm{d}} := P_q(x,y) = \sum_{i=1}^{m+r} f(x_i,y_i)\cdot P_{q,i}(x,y) \tag{15}$$

The operation on the influence polynomial is needed for building the difference formulae, e.g. f_x:

$$f_{x,\mathrm{d}} := \frac{\partial P_q(x,y)}{\partial x} = \sum_{i=1}^{m+r} f(x_i,y_i)\,\frac{\partial P_{q,i}(x,y)}{\partial x} \tag{16}$$

The determination of other derivatives is analogous and the general formulation could be expressed as

$$\frac{\partial^{m+n} f_{\mathrm{d}}(x,y)}{\partial x^m \partial y^n} = \sum_{i=1}^{m+r} f(x_i,y_i)\,\frac{\partial^{m+n} P_{q,i}(x,y)}{\partial x^m \partial y^n} \tag{17}$$

Since these influence polynomials depend only on the geometric position of the nodes, the coefficients of the influence polynomial need to be evaluated at the very beginning of the numerical computation or at the time when mesh refinement is necessary.

3.2. Weighted least square method

To increase the influence of the surrounding nodes which are close to the centre node, the following weight function multiplies on each row of the influence polynomial matrices M and I, i.e. Equation (13).

$$w(s_i) = 1 - 6s_i^2 + 8s_i^3 - 3s_i^4 \quad (0 \leqslant s_i \leqslant 1) \tag{18}$$

where s_i is the distance between the selected node i and the centre node normalized by the largest distance value in the same element. The weighted least square problem becomes

$$M^{(w)}_{(m+r)\times m} A_{m\times(m+r)} = I^{(w)}_{m+r} \tag{19}$$

where matrix $M^{(w)}$ is

$$M^{(w)} = \begin{bmatrix} w(s_1) & w(s_1)x_1 & w(s_1)y_1 & \cdots & w(s_1)y_1^q \\ w(s_2) & w(s_2)x_2 & w(s_2)y_2 & \cdots & w(s_2)y_2^q \\ & \vdots & & & \vdots \\ w(s_{m+r}) & w(s_{m+r})x_{m+r} & w(s_{m+r})y_{m+r} & \cdots & w(s_{m+r})y_{m+r}^q \end{bmatrix} \tag{20}$$

and the diagonal elements of $I^{(w)}$ becomes $w(s_i)$, $i = 1, m+r$. It is called a weighted least square method in this paper. To obtain a better accuracy, we can tune the weight function.

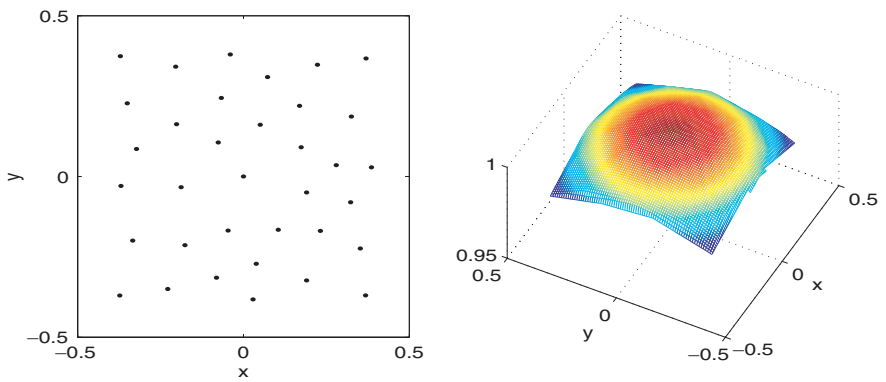

Figure 2. Node distribution (left) and surface of function $f(x, y) = \cos(kx)\cos(ky)$ (right) with $k = \pi/8$.

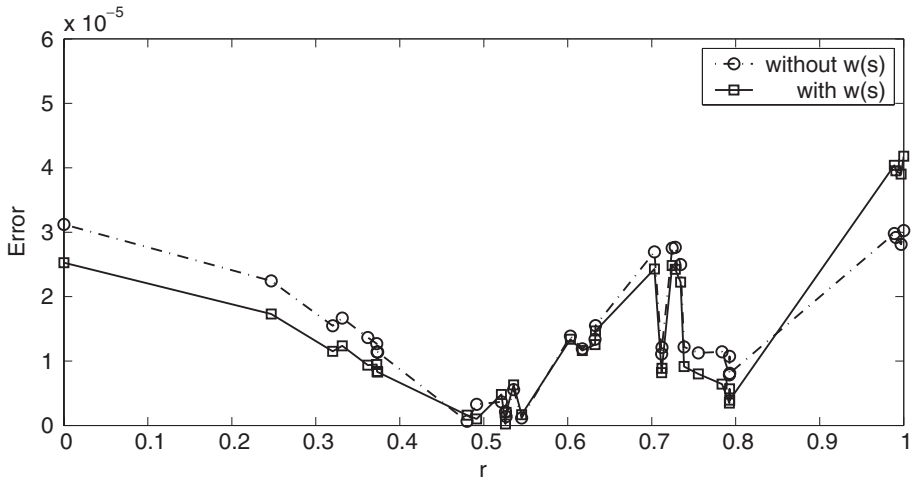

Figure 3. Comparison on the absolute error of each nodes versus the distance to the centre node.

However, the weight function Equation (18) gives a rather good performance in the present work.

To investigate the effects of the weighted least square method, the following numerical experiment is carried out. The local node system is illustrated in the left panel of Figure 2, the actual position of each node is given in the appendix for easy reference. To examine the effect of the weighted least square method quantitatively, let $f(x, y) = \cos(kx)\cos(ky)$ with $k = \pi/8$. The exact surface of function f is plotted in the right panel of Figure 2. Now let us estimate the reconstructed function value at the node position using Equation (15); the absolute error $|f_d - f|$ versus the distance to the centre node is plotted in Figure 3. When the weighted least square method is used, it is seen that the absolute error is much smaller for most nodes in Figure 3 except a few nodes far away from the centre node (i.e. $r \sim 1$). The

Table I. Comparison on the estimated function derivatives by interpolation, $k = \pi/8$.

	f	f_x	f_{xx}	f_{xy}	f_{xxx}	f_{xxy}
Exact	1	0	−0.1542	0	0	0
Without $w(s)$	1.0000	0.7174e − 5	−0.1531	−0.0319e − 6	−0.5989e − 3	−0.5097e − 4
With $w(s)$	1.0000	0.3226e − 5	−0.1532	0.0344e − 6	0.3888e − 3	0.6574e − 4

reconstructed function value, as well as the derivatives by the interpolation basis, are listed in Table I for the cases without or with the weighted function $w(s)$. From the comparison, better estimated derivative values are obtained when applying the weighted least square method.

To provide more flexibility on the choice of $m + r$, as well as improve the accuracy, the node position is disturbed for a small magnitude off its original location obtained from grid generation software. This technique is studied in Reference [15] in which it is used to improve the finite difference accuracy. For an extreme example, if the nodes in an influence domain are collinear, the influence coefficient matrix may become singular and no solution could be obtained. With the nodes disturbing technique, the value of $m + r$ could be reduced a little without assembling an ill-conditional matrix. For implementation, two random parameters are generated for each node, one for the magnitude of the disturbance and the other one for the angular degree of rotation. From the numerical test cases we have carried out, a small disturbance at the order 1% of the local node-to-node distance gives satisfied performance.

3.3. Boundary conditions

In the numerical solution of any system of PDEs appropriate boundary conditions must be specified. Regarding the wave problem, a wave-maker zone is necessary to generate the specified incident waves. A sponge layer is applied at the open boundary to completely absorb the out-going waves. Both the wave maker and wave absorber could be set up with the concept of a relaxation zone, which has been verified to be a valid approach [12]. Generation and/or absorption of waves is achieved by simply defining a relaxation coefficient $0 \leqslant c_r(x, y) \leqslant 1$, and an exact desired solution $\eta^{(e)}$ and $\widetilde{\mathbf{U}}^{(e)}$. At every time step of the calculation, the solution in the relaxation zone is updated by

$$f(x, y, t) = c_r f(x, y, t) + (1 - c_r) f^{(e)}(x, y, t) \quad f = \eta \text{ or } \widetilde{\mathbf{U}} \tag{21}$$

With this technique, the sponger layer and wave-maker could be obtained by setting $f^{(e)}$ to be zero and the desired incident wave quantities, separately. This treatment is quite effective, as is demonstrated in the formation of standing waves in the next section.

With almost perfect wave maker and absorber, the still-water level will be maintained at the boundaries. Therefore, a symmetry boundary condition or a reflection boundary could be applied at these boundaries. The reflection boundary is imposed by flipping the selected nodes versus the local boundary evenly or oddly depending on the physical process. Figure 4 illustrates the implementation of this boundary condition. The ghost points (shown in stars) are distributed outside of the approximate boundary (the dash–dot line) by reflecting the interior points (shown in hollow squares). For a small curved boundary, the offset between the approximate boundary and the real boundary (the solid line) is small and satisfactory results could be obtained, as discussed in Section 4.2. For a boundary with large curvature or

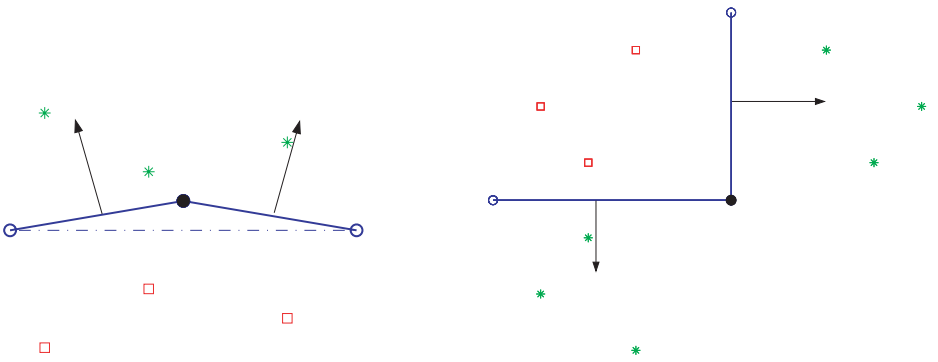

Figure 4. Approximation of the treatment of the general curved boundary.

sharp corner, the interior nodes are flipped for both boundaries. The difference coefficients are computed from the composite system using both the computational nodes and ghost nodes. This strategy has the advantage of keeping the overall model structure regular, as all the nodes in the computation domain could be treated in the same way.

The classical 4th-order, four-stage explicit Runge–Kutta method is used for time integration of the time stepping problem, Equations (1) and (2).

3.4. Grid generation and reordering technique

The nodal points are produced by the grid generation software Gambit (Fluent Inc.). The node indices need to be reordered after the selection of the appropriate nodes in the element to reduce the linear system bandwidth. When solving sparse systems, the triangular factors of the matrix **M** are also sparse. Indeed, if **M** has a band of width p and can be decomposed into **LU**, then both the triangular factors **L**-matrix and **U**-matrix are banded with bandwidth p. The bandwidth of the matrix is related directively to the computational efforts. The decomposition algorithm requires $O(np^2)$ operations, where n is the matrix dimension. As a matter of fact, a decomposition method applied to a sparse matrix without caring about its structure can lead to a substantial *fill-in*, i.e. generate a considerable number of new non-zero elements, which results in increasing storage and computational effort.

The reverse Cuthill–McKee ordering is frequently used when a matrix is to be re-numbered to reduce the bandwidth. This is discussed in Reference [16].

Figure 5 illustrates the sub-matrix structure of the coefficient matrix with and without reverse Cuthill–McKee reordering in the square region with 13 nodes on each side and a total of 218 nodes in the computation domain. The sparsity pattern, i.e. the position of the non-zero elements, of the sub-matrix of **M** (e.g. $\mathscr{A}_{11} - \eta_x \mathscr{B}_{11}$) is plotted as a dark square in Figure 5. This pattern indicates whether one element of the matrix is a non-zero value or not, but not the value itself. Applying the reverse Cuthill–McKee reordering, the position of the non-zero elements are changed in each row and column, while the value of the element does not change in this process. The positions of the non-zeros become closer to the matrix diagonal. Consequently, the bandwidth of the matrix is reduced.

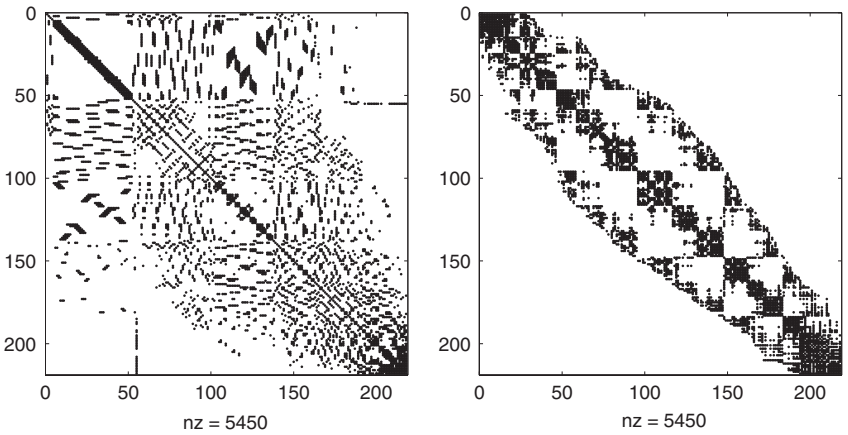

Figure 5. Comparison on the distributions of non-zero elements in the coefficient sub-matrix, with (right) and without (left) reverse Cuthill–McKee reordering: 218×218.

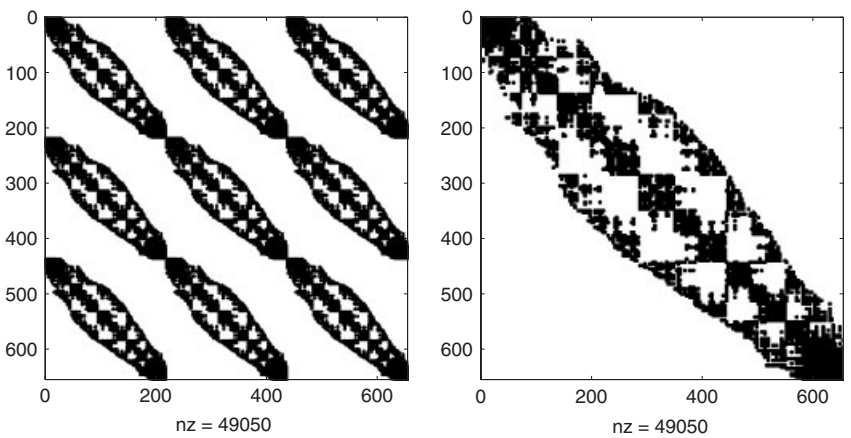

Figure 6. Comparison on the distributions of non-zero elements in the full coefficient matrix, grouping the discrete equations by PDEs (left) and nodes (left).

It is seen that the reverse Cuthill–McKee reordering results in a much smaller bandwidth even in this simple test case. This kind of reordering technique could dramatically reduce the sub-matrix bandwidth from order $O(N)$ to $O(m + r)$ when a large computational domain is involved, and N is the total number of nodes. It should be noted that the reverse Cuthill–McKee reordering should be re-calculated when changing the selected node number in one element, even when the node positions are not changed.

As indicated by Equation (9), the full matrix to be solved is a 3×3 block matrix, and the aforementioned reordering is based on the sub-matrix. Although the sub-matrix has been diagonalized by reordering the node index, the full matrix is far from diagonally dominant as shown in the left panel of Figure 6. Before solving the linear systems, reordering the unknowns

is conducted to reduce the matrix to be a diagonal-like matrix. This can be accomplished by simply changing the unknowns from PDEs-order

$$[\hat{u}_1 \ \hat{u}_2 \ \cdots \ \hat{u}_N \ \hat{v}_1 \ \hat{v}_2 \ \cdots \ \hat{v}_N \ \hat{w}_1 \ \hat{w}_2 \ \cdots \ \hat{w}_N]^{\mathrm{T}} \tag{22}$$

to nodes-order

$$[\hat{u}_1 \ \hat{v}_1 \ \hat{w}_1 \ \hat{u}_2 \ \hat{v}_2 \ \hat{w}_2 \ \cdots \ \hat{u}_N \ \hat{v}_N \ \hat{w}_N]^{\mathrm{T}} \tag{23}$$

The full matrix is assembled according to the reordered unknowns. The iteration could greatly benefit from this simple operation due to the significant reduction of the bandwidth; the structure of the sparse matrix is shown in the right panel of Figure 6.

3.5. Solution of the sparse matrix

Two kinds of iteration methods are considered in present work. GMRES may be the most interesting method for the iterative solution of a large, sparse, non-symmetric matrix. Recently, Bi-CGSTAB as a variant of Bi-CG has been proposed for solving non-symmetrical linear systems, and its attractive convergence behaviour has been confirmed in many numerical experiments. For the cases studied so far, no significant difference could be found between these two kind iterative schemes. The necessary memory for a GMRES method is about $O(17N)$ while it is only $O(8N)$ for Bi-CGSTAB method, where N is the order of the linear system. On account of the practical application where large memory is needed, Bi-CGSTAB method is preferred in our work. All of the computations presented in this paper based on this iteration method.

The key to the efficient solution of the large-scale linear systems lies in effective preconditioning. The efficiency of various preconditioners was studied in Reference [14] for the solution of the high-order Boussinesq equation on a rectangular mesh. The ILUT preconditioner works quite well in shallow to intermediately deep water. SPARSKIT developed by [17] is used to solve the sparse linear equations in the present work. As a lower-storage method, the incomplete LU factorization with dual truncation mechanism is applied in each sub-time-step in the time integration.

4. MODEL VERIFICATION

4.1. Convergence tests

The simplest model in applied mathematics and mechanics is a linear system. It is also by far the most important, and we begin our tests with the linearized Boussinesq equations and a linear wave to gain insight into the accuracy and convergence of the numerical method. The oscillation of a linear standing wave in a square basin is considered with the initial conditions

$$\eta(x, y, 0) = \frac{H}{2} \cos kx \cos ky \quad \text{and} \quad \widetilde{\mathbf{U}}(x, y, 0) = 0 \tag{24}$$

where H is the wave height, and k is the wave number in each directions. The nonlinear terms are switched off for comparison against the exact solution. For a linear problem, the velocity components $\widetilde{\mathbf{U}}$ are the same as the still-water velocity, and the exact solution of the

problem could be obtained by multiplying $\cos(\omega t)$ on the initial condition for the free surface elevation. The horizontal dimension of the basin is 100 m × 100 m and the water depth is 10 m. To form the steady standing wave, a linear wave of length 100 m and wave height of 1 m is chosen as an initial condition. From the linear dispersion relation, the period of the input wave is 7.99 s.

Firstly, the space convergence is considered. It is hard to measure the node-to-node distance for it is a non-structured method. The grid generation software gives a nearly equilateral triangular mesh; hence, we can roughly estimate the node-to-node distance δ by considering the space step at the basin edge. Five mesh set-ups are generated with 31, 41, 61, 81 and 161 nodes at each edge. The corresponding δ could be roughly estimated as 3.3, 2.5, 1.67, 1.25 and 0.625 m. The notation s.p.p. denotes the time step per period in the following tables. The Euclidean norm of the approximation error l_2 reported in this paper is computed by

$$l_2 = \frac{1}{N}\sqrt{\sum_{i}^{N}\left(\frac{\eta_i - \eta_i^{\text{exact}}}{H}\right)^2} \tag{25}$$

where N is the total number of nodes. The l_2 error measures the averaged relative error over the computation domain. The Cauchy convergence rate is estimated by

$$\log(l_2^{(1)}/l_2^{(2)})/\log(\delta^{(2)}/\delta^{(1)}) \tag{26}$$

The error estimates are carried out at a time after five wave periods of oscillations.

We have tried the case where there are only 21 nodes at one edge; however, the numerical code breaks down. It implies that we need enough nodes in one wavelength to resolve the wave. From the numerical test cases, it seems that number of 20 nodes in one dominant wavelength is the lower limit, which corresponds to the case of 31 nodes on one edge, as there are wave components in both perpendicular directions. The first conclusion from the convergence tests is that the number of selected surrounding nodes has a large impact on the accuracy. It seems there is a minimum relative error at the region around $m + r$ equal to 30 and 40 and for q is 3 and 4, as shown in Tables II and III. In this case, the 3rd-order polynomial has $m = 10$ coefficients, and an additional $r = 20$ nodes are added to form the least square coefficient matrix. When the value of $m + r$ increases, the accuracy decreases because more nodes in one element will cause more dissipation. However, we cannot select too few surrounding points r otherwise a singularity will occurs when solving the influence coefficients, especially when the weight function has a rapid decay. In the numerical test cases, the space convergence rate is about 1.5–2 for most cases. This is reasonable because the 1st-order derivative difference should have a higher accuracy while the 2nd- and 3rd-order derivatives are lower under the selected order of polynomial approximation. Finally, a choice of 30 nodes per element has an overall best performance on the bases of the accuracy and computational expense when the 3rd-order polynomial is used. Nearly the same performance occurs for the case of $q = 4$. Comparing the choice of $q = 3$ and 4, no obvious advantage could be found for the accuracy with the time and space step at the order of $O(10)$ per wave period and wavelength. When taking the time and memory expense into account, the 3rd-order polynomial approximation gives an overall better performance.

Secondly, the time convergence is considered. The number of nodes 30 and 40 are chosen for the 3rd- and 4th-order polynomials. With regard to practical applications, choosing about 20 and 40 nodes in one dominant wavelength is acceptable and appropriate for linear and

Table II. Space convergence of the 3rd-order influence polynomial element.

		Nodes number $m+r$					
		25		30		35	
s.p.p.	Size	l_2 (e − 3)	order	l_2 (e − 3)	order	l_2 (e − 3)	order
	31	11.81		9.961		12.63	
	41	8.307	1.223	6.749	1.353	9.189	1.106
40	61	4.138	1.719	3.350	1.727	5.523	1.256
	81	2.740	1.433	1.926	1.924	3.320	1.769
	161	0.245	3.483	0.102	4.239	0.334	3.313
	31	10.41		8.560		10.86	
	41	7.068	1.346	5.563	1.498	7.676	1.206
80	61	3.312	1.870	2.724	1.761	4.486	1.325
	81	2.296	1.274	1.481	2.118	2.588	1.912
	161	0.251	3.193	0.111	3.738	0.243	3.413
	31	8.903		7.278		9.111	
	41	6.008	1.367	4.619	1.580	6.333	1.264
160	61	2.642	2.026	2.208	1.820	3.589	1.401
	81	1.794	1.346	1.104	2.409	1.964	2.096
	161	0.274	2.711	0.135	3.032	0.126	3.962

Table III. Space convergence of the 4th-order influence polynomial element.

		Nodes number $m+r$					
		30		40		60	
s.p.p.	Size	l_2 (e − 3)	order	l_2 (e − 3)	order	l_2 (e − 3)	order
	31	7.825		9.495		13.47	
	41	5.245	1.391	6.923	1.098	10.79	0.771
40	61	4.322	0.477	4.044	1.326	7.420	0.923
	81	2.830	1.472	2.728	1.368	5.108	1.298
	161	0.820	1.787	0.748	1.867	1.994	1.357
	31	6.289		7.340		10.27	
	41	4.103	1.485	5.208	1.193	7.900	0.912
80	61	3.461	0.420	2.956	1.397	5.146	1.057
	81	2.207	1.564	1.943	1.459	3.400	1.441
	161	0.609	1.858	0.526	1.885	1.282	1.407
	31	5.135		5.454		7.493	
	41	3.362	1.472	3.684	1.364	5.311	1.196
160	61	1.738	1.627	1.916	1.612	3.060	1.360
	81	0.990	1.956	1.160	1.744	1.833	1.780
	161	0.379	1.385	0.240	2.273	0.532	1.785

Table IV. Comparison on the time convergence.

q		Time step per period							
$m+r$		40	60	80	160	320	640	1280	2560
3	l_2 (e − 3)	6.749	6.032	5.563	4.617	3.901	4.088		
30	order		0.277	0.281	0.269	0.243	−0.068		
4	l_2 (e − 3)	6.923	5.853	5.208	3.684	2.269	0.942	0.552	1.662
40	order		0.414	0.406	0.499	0.699	1.268	0.771	−1.590

Table V. The time convergence with the 4th-order polynomial element.

		Time step per period							
Size		40	60	80	100	160	320	640	1280
61	l_2 (e − 3)	4.044	3.398	2.956	2.618	1.916	0.884	0.218	1.322
	order		0.429	0.484	0.544	0.664	1.116	2.020	−2.600
81	l_2 (e − 3)	2.728	2.261	1.943	1.693	1.160	0.361	0.495	
	order		0.463	0.527	0.617	0.804	1.684	−0.455	

nonlinear wave simulations, respectively. The results of the convergence tests are shown in Table IV. For comparison, the choice of a very fine time step, e.g. 320 and 640, etc., was tested as well to investigate the final convergence rate. The numerical results imply that the convergence rate could reach up to 1 for those very-fine time step, while it is around 0.3 and 0.4 for the two selected polynomial function when choosing the same order of both the time step per wave period and space step per wavelength. To further investigate the temporal convergence of the present numerical method, a high space resolution for $q = 4$ are studied and the numerical results are shown in Table V. In this circumstance, the convergence rate could reach up to 2. These results indicate that we cannot improve the time convergence rate when the space resolution is not fine enough. It is generally well known that the time convergence and space convergence are coupled for time-relating PDEs. When further decreasing the time step, the l_2 error may increase, e.g. the case tested by 640 time step with 81 nodes in the edge, which is thought to be the accumulation of the discretization error.

Thirdly, the higher-order influence polynomials are considered for $q = 5$ and 6. The extension of the influence polynomial is straightforward. When the order of the influence polynomial increases, more surrounding nodes need to be included in the finite difference element to meet the demands of both the necessary polynomial and least square approximation. From the results of numerical experiments, shown in Tables VI and VII, increasing the influence polynomial order has not notably improved the numerical accuracy, e.g. the smallest value of l_2 is 2.728 for 4th-order polynomial but 4.107 and 4.063 for 5th- and 6th-order with the same time step s.p.p. 40 and size 81. Given the polynomial order q, the accuracy decreases when the surrounding node number increases. Fixing the surrounding node number $m+r$, the accuracy decreases as increasing the polynomial order. One possible reason may be the fact

Table VI. The convergence with the 5th-order influence polynomial element.

		Node number $m+r$					
		50		60		80	
s.p.p.	Size	l_2 (e − 3)	order	l_2 (e − 3)	order	l_2 (e − 3)	order
	41	8.522		9.664		10.39	
40	61	5.765	0.964	7.092	0.763	8.126	0.606
	81	4.107	1.179	5.039	1.188	6.386	0.838
	41	6.206		6.942		7.528	
80	61	3.999	1.084	4.779	0.921	5.461	0.792
	81	2.699	1.367	3.223	1.369	4.044	1.044

Table VII. The convergence with the 5th-order influence polynomial element.

		Node number $m+r$					
		50		60		80	
s.p.p.	Size	l_2 (e − 3)	order	l_2 (e − 3)	order	l_2 (e − 3)	order
	41	9.187		9.958		10.64	
40	61	5.535	1.250	7.140	0.820	8.472	0.562
	81	4.063	1.075	4.994	1.243	6.489	0.927
	41	6.955		7.277		7.647	
80	61	4.151	1.273	5.035	0.908	5.764	0.697
	81	2.922	1.220	3.380	1.385	4.207	1.095

that the large different element will introduce more numerical dissipation. With the viewpoint of practice applications, a high-order polynomial model will result in the rapidly increasing computational expense without improvement the accuracy. This is not a promising numerical model for engineering applications. Consequently, the higher order polynomial approximations are abandoned in the present work.

4.2. *Boundary fitting mesh*

One of the most important advantages of the mesh-less method is the ease of implement for an irregular coastal boundary. In this section, simulation of a Gaussian hump oscillating in a circular tank is conducted to illustrate this property.

Considering a circular tank with radius $R = 50$ m, the water depth is 10 m. The still water is located at the xoy plane and with the z axis upward. The initial condition is given by a motionless Gaussian hump of water with its centre located at the origin of the basin as Equation (27)

$$\eta(x, y, 0) = \mathrm{Exp}(-\tfrac{1}{100}(x^2 + y^2)) \tag{27}$$

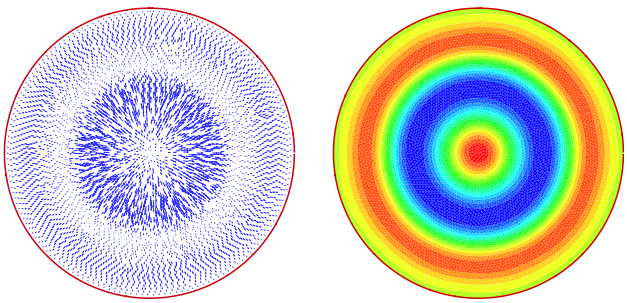

Figure 7. Top view of the still-water velocity field (left) and the surface elevation (right) at $T = 40$ s. In the left panel, the length of the vector indicates the velocity magnitude, and the colour represents the surface elevation in the right one.

The average node-to-node distance is 1 m. The time step is 0.1 s. After 40 s during which the hump oscillates around 8 times, the still-water level field and surface elevation are shown in Figure 7. A good circumferential symmetry property has been obtained with the present boundary treatment as shown in Figure 7, which verifies the effectiveness of the boundary implementation.

4.3. *Wave-maker and sponger*

To test the capability of the relaxation technique, the reflection of a nonlinear stream-function wave and formation of nonlinear standing wave are studied. The wave tank covers a length of 800 m and a width of 20 m. The water depth is 5 m and the length and height of the incident wave is 100 m and 0.5 m, separately. The average node to node distance δ is about 1.5 m by Gambit and total number of nodes is 8848. The time step of the computation is 0.2 s. The selection node number $m + r$ equals 30 and the polynomial order is $q = 3$ in this test case.

Figure 8 demonstrates the snapshots of surface elevation in one period of simultaneous wave generation and absorption from a relaxation zone at the centreline of the numerical wave tank. A nonlinear incident wave is specified at the left-hand boundary using a relaxation zone of one wavelength. The second relaxation zone covers another wavelength $x \in [-300, -200]$ to absorb the reflection wave from the computational domain. The right-hand boundary is a vertical wall. The steady-state results in a perfect standing wave outside the relaxation zone, i.e. the region of $x \in [-200, 400]$. In general, the absorbing of shallow water waves needs a long relaxation zone. In this test case, the incident wave belongs to the shallow water wave ($kh = \pi/10$). For intermediate water waves, about half of the wavelength is enough for the relaxation zone by appropriately adjusting the relaxation function $c_r(x, y)$.

4.4. *Nonlinear wave shoaling*

The experimental results in Reference [18] are used extensively in the literature to demonstrate the numerical wave models involving nonlinear refraction and diffraction (see, e.g. References [14, 19]). The topography connects deep- and shallow-water regions with a shoaling

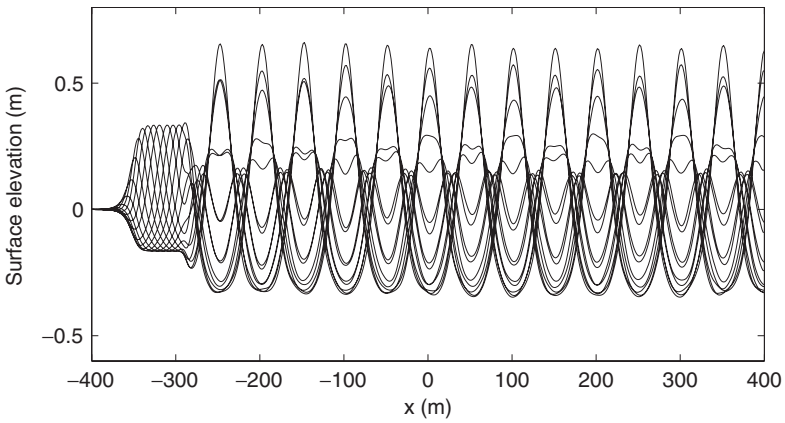

Figure 8. Nonlinear standing wave envelope in flat bottom flume.

region acting as a concave lens, and is described by

$$h(x,y)=\begin{cases}0.4572 & \text{if } 0\leqslant x<10.67-G\\ 0.4572+\frac{1}{25}(10.67-G-x) & \text{if } 10.67-G\leqslant x\leqslant 18.29-G\\ 0.1524 & \text{if } 18.29-G<x\leqslant 27\end{cases} \tag{28}$$

where

$$G(y)=\sqrt{y(6.096-y)} \tag{29}$$

The gradient of h is calculated analytically as input when building the bottom boundary condition Equation (4). Because the bathymetry is symmetrical about the centreline $y=3.048$ m, only the half of the domain is simulated. For waves of $T=2$ s, the wavelength is about 3.89 and 2.40 m in the deep-water and shallow-water part, respectively. Four meshes with approximately 31, 41, 49 and 61 nodes per wavelength, are generated for computation with Gambit. The total number of nodes in the computational domain are 10041, 16621, 24922 and 34591, respectively. The time step is fixed to be $\Delta t=T/50$, and simulations run for 1200 time steps to ensure the steady state for the harmonic analysis.

Almost identical results for the 1st-order harmonic are obtained with the mesh at these scales, as shown in Figure 9. These results are the relative amplitudes of harmonics along the centreline at $y=3.048$ m. The harmonic analysis was done for the time series of surface elevation from the last 500 time steps. For the coarsest mesh, i.e. 31 nodes per wavelength, the 2nd- and 3rd-order harmonics are slightly under-predicted when comparing the convergent harmonic magnitude, which is believed to be related to the mesh resolution and the numerical damping. If there are 31 nodes in one wavelength for the 1st-order harmonic, then there are only about 11 nodes per wavelength for the 3rd-order harmonic. When the resolution is increased, say 41 nodes per wavelength, almost convergent results could be obtained.

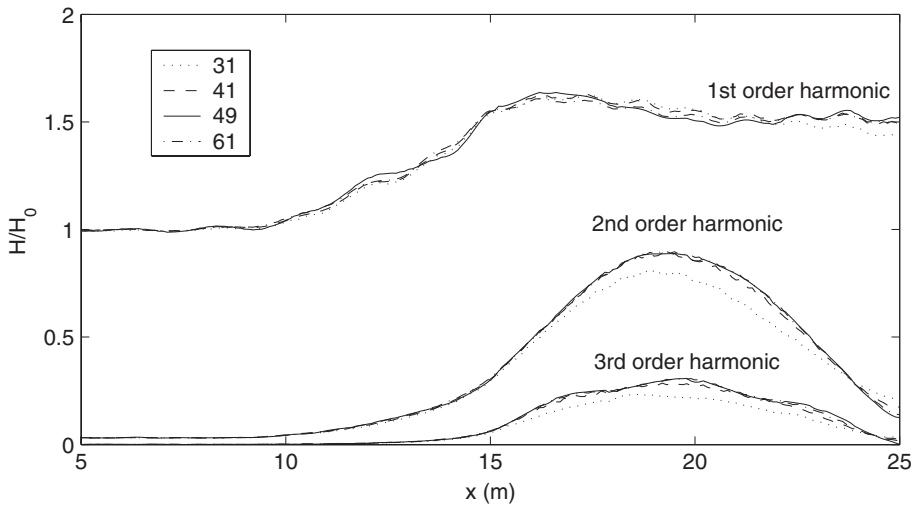

Figure 9. Comparison on the harmonic amplitude distribution with various space resolution: $q = 3$; $m + r = 30$.

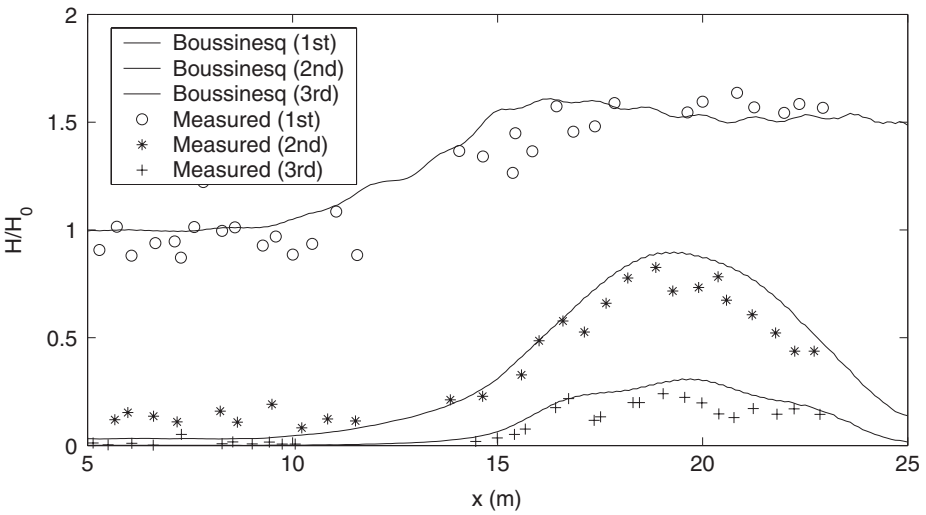

Figure 10. Computed and measured harmonic amplitudes for simulations modelling the experiments of Reference [18] with the 49 nodes per wavelength mesh: $q = 3$; $m + r = 30$.

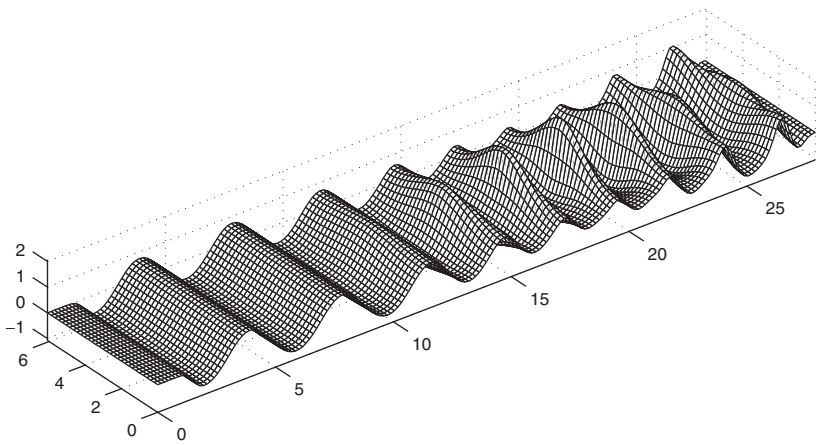

Figure 11. Instantaneous surface elevation with the mesh of 49 nodes per wavelength: $q=3$; $m+r=30$.

Comparisons between the numerical results with the 49 nodes per wavelength mesh and experimental results are presented in Figure 10. The match with the experimental data is good, and compares well with other numerical results in the literature. The focusing of the waves can be seen in Figure 11 shown as instantaneous surface elevation, in which the surface has been projected to a coarse rectangular mesh for clarity.

We remark that these results are preliminary and by no means comprehensive. They are merely to show the capability of the method itself and investigate the performance of the numerical model. In further work, we hope to study more problems involving complex boundaries which could not or are not easily solved in a rectangular mesh with traditional finite difference method.

4.5. Computational efficiency of the sparse matrix solver

During the ILUT preconditionor's factorization, each row of the L-matrix and U-matrix will have a maximum of *ifillin* elements (excluding the diagonal element). Comparing of *ifillin* is conducted to investigate the computational expanse for both the CPU time and memory. The numerical code is compiled with Intel Fortran 90 compiler (Version 7.0). The machine is a Linux Redhat 9.0 platform with 1024 MB RAM and Intel(R) Pentium(R)-4 CPU of 2.40 GHz. The details of the computer resources used are listed in Table VIII. The convergence criteria is chosen as $\|\text{residual}\| \leqslant 1.0\text{E}-20\|\text{rhs}\|$ for the iterative solver, and the threshold for the filling element magnitude takes the value of $1.0\text{E}-9$.

The term Iter. represents the iteration numbers of each sub-time-step, and the value of CPU is the time used at each time step. The comparison results shown in Table VIII could give us a brief idea on the choice of *ifillin* value. The size of *ifillin* will significantly affects the iterations of the linear solver especially for a large-scale sparse matrix, e.g. the case of $ifillin=10$ for fine mesh in Table VIII. The iterate number drops down significantly when the *ifillin* number increases. However, the major time cost spent on the ILUT precondition of the coefficient matrix, which needs to be run at every sub-time-step during the time

Table VIII. Summary of the influence *ifillin* on the computational expanse, $q = 3$, $m + r = 30$.

	Coarse mesh : 16621 nodes			Fine mesh : 24922 nodes		
ifillin	Iter.	CPU (s)	RAM (MB)	Iter.	CPU (s)	RAM (MB)
10	56–68	18	195	196–216	90	411
20	32–34	24	208	64–68	82	425
30	18–20	39	216	32–34	108	451
60	10–12	81	253	14–16	259	508

integration. If the larger value of *ifillin* is chosen, the precondition matrix will become dense and more physical memory is needed; it will increase the calculation burden on the matrix and vector multiplication during iteration process. This conclusion can be easily checked when comparing *ifillin* = 30 and 60 for both the coarse and fine mesh cases. Although the iteration steps are reduced, the CPU time and memory used do increase. How to balance the two aspects on the choice of *ifillin* depends on the problem itself as well as the node optimization. For the test cases in our experience, it seems that the choice of *ifillin* that is about equivalent to $m + r$ gives the best overall performance or efficiency. The appropriate preconditioned method as well as other kinds of iteration scheme may provide significant acceleration on the solver of the present numerical model, which is under investigation in our work.

5. CONCLUSIONS

A new mesh-less least squares-based finite difference method has been developed to numerically solve the fully nonlinear and highly dispersive Boussinesq wave model. Details on the implementation are presented, especially for the techniques used to improve the efficiency. Various influence polynomials and the node numbering in each element are discussed. The numerical results imply that the choice of a 3rd-order influence polynomial is the best one regarding efficiency. While the size of the element has a large influence on the numerical dissipation, 30 surrounding nodes are appropriate for 3rd-order polynomial element. Comparisons between the present model and the experimental data are carried out. Good agreement was obtained, which proves the effectiveness of the present numerical model. More verifications involving complex geometric boundaries are in progress and the results will be published elsewhere.

Finally, we emphasize that the numerical method presented in this paper is by no means limited to the fully nonlinear Boussinesq equations. For some purposes it may be more attractive to use a depth-averaged Boussinesq model, e.g. References [4, 5], in which cases the computational effort will be reduced as they could be solved with an explicit scheme or semi-explicit scheme; no large linear system is involved. Furthermore, large-scale wave models, such as the nonlinear shallow water equations, the mild slope equation (both parabolic kind and hyperbolic kind), etc., could be solved with the mesh-less least square-based finite difference method.

Table AI. Locations of nodes.

ID	x	y	ID	x	y	ID	x	y
1	0.0000	0.0000	13	−0.0666	0.2441	25	0.1905	−0.0498
2	−0.3727	−0.3701	14	−0.3340	−0.1997	26	−0.0757	0.1058
3	0.3684	−0.3697	15	0.2231	0.3475	27	0.0729	0.3093
4	0.3698	0.3669	16	−0.3502	0.2276	28	0.3234	−0.0802
5	−0.3710	0.3741	17	−0.2279	−0.3501	29	−0.0812	−0.3149
6	−0.2017	0.1625	18	0.3523	−0.2241	30	0.1742	0.0910
7	0.1694	0.2197	19	0.1904	−0.3233	31	0.1052	−0.1657
8	0.2321	−0.1692	20	0.3255	0.1863	32	−0.3693	−0.0294
9	−0.1763	−0.2137	21	−0.2043	0.3416	33	−0.0396	0.3795
10	−0.1878	−0.0332	22	−0.3223	0.0857	34	0.0292	−0.3825
11	0.2799	0.0351	23	0.0505	0.1610	35	0.3859	0.0282
12	0.0388	−0.2717	24	−0.0462	−0.1684			

APPENDIX A

The position of each node in Section 3.2 is given in Table AI.

ACKNOWLEDGEMENTS

The present research was supported by a grant (No. 10172058) from the National Science Foundation of China and the Doctoral Program Foundation for Higher Education from the Ministry of Education of China (No. 2000024817). Many thanks due to instructive discussions with D.R. Fuhrman for precondition methods on numerical solving the linear equations, and P.A. Madsen for the enhanced Boussinesq equations when the authors visiting the Technical University of Denmark. Editorial changes by the referees are gratefully acknowledged.

REFERENCES

1. Belytschko T, Krongauz Y, Organ D, Fleming M, Krysl P. Meshless methods: an overview and recent developments. *Computer Methods in Applied Mechanics and Engineering* 1996; **139**:3–47.
2. Li S, Liu WK. Meshfree and practical methods and their applications. *Applied Mechanics Review* 2002; **55**:1–34.
3. Du CJ. An element-free Galerkin method for simulation of stationary two-dimensional shallow water flows in rivers. *Computer Methods in Applied Mechanics and Engineering* 2000; **182**:89–107.
4. Madsen PA, Schäffer HA. Review of Boussinesq-type equations for surface gravity waves. *Advances in Coastal and Ocean Engineering*, vol. 5, Chapter 1. World Scientific Publishing Co.: Singapore, 1999; 1–95.
5. Kirby JT. Boussinesq models and application to nearshore wave propagation, surf zone processes and wave-induced currents. *Advances in Coastal Modeling*, Chapter 1. Elsevier Science: New York, 2003; 1–41.
6. Shi F, Dalrymple RA, Kirby JT, Chen Q, Kenedy A. A fully nonlinear Boussinesq model in generalized curvilinear coordinates. *Coastal Engineering* 2001; **42**:337–358.
7. Wang KH, Wu TY, Yates GT. Three-dimensional scattering of solitary waves by vertical cylinder. *Journal of Waterway Port Coastal and Ocean Engineering* 1992; **118**:551–556.
8. Ding C, Shu H, Yeo KS, Xu D. Simulation of incompressible viscous flows past a circular cylinder by hybrid FD scheme and meshless least square-based finite difference method. *Computer Methods in Applied Mechanics and Engineering* 2004; **193**:727–744.
9. Maz'ya V, Schmidt G. On quasi-interpolation with non-uniformly distributed centers on domains and manifolds. *Journal of Approximation Theory* 2001; **110**:125–145.
10. Fasshauer GE. Toward approximate moving least squares approximation with irregularly spaced centers. *Computer Methods in Applied Mechanics and Engineering* 2004; **193**:1231–1243.
11. Schönauer W, Adolph T. How we solve PDEs. *Journal of Computational and Applied Mathematics* 2001; **131**:473–492.

12. Madsen PA, Bingham HB, Liu H. A new Boussinesq method for fully non-linear waves from shallow to deep water. *Journal of Fluid Mechanics* 2002; **462**:1–30.
13. Madsen PA, Bingham HB, Schäffer HA. Boussinesq-type formulations for fully nonlinear and extremely dispersive water waves: derivation and analysis. *Proceedings of the Royal Society of London Series A* 2003; **459**:1075–1104.
14. Fuhrman DR, Bingham HB. Numerical solutions of fully non-linear and highly dispersive Boussinesq equations in two horizontal dimensions. *International Journal for Numerical Methods in Fluids* 2004; **44**:231–255.
15. Gossler A. Moving least-squares: a numerical differentiation method for irregularly spaced calculation points. *Technical Report SAND2001-1669*, Sandia, 2001.
16. George A, Liu JW. *Computer Solution of Large Sparse Positive Deinite Matrices*. Prentice-Hall, Englewood Cliffs, NJ, 1981.
17. Saad Y. *SPARSKIT*: *A Basic Tool Kit for Sparse Matrix Computations*, *Version 2*. www.cs.umn.edu/research/arpa/SPARSKIT/sparsekit.html, 1994.
18. Whalin RW. The limit of applicability of linear wave refraction theory in a convergence zone. *Technical Report H-71*, U.S. Army Corps of Engineers, Waterway Experiment Station, Vicksburg, MS, 1971.
19. Beji S, Nadaoka K. A formal derivation and numerical modeling of the improved Boussinesq equations for varying depth. *Ocean Engineering* 1996; **23**(8):691–704.

