INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2004; **44**:231–255 (DOI: 10.1002/fld.628)

Numerical solutions of fully non-linear and highly dispersive Boussinesq equations in two horizontal dimensions

David R. Fuhrman*,† and Harry B. Bingham

Department of Mechanical Engineering, Technical University of Denmark, DK-2800 Kgs. Lyngby, Denmark

SUMMARY

This paper investigates preconditioned iterative techniques for finite difference solutions of a high-order Boussinesq method for modelling water waves in two horizontal dimensions. The Boussinesq method solves simultaneously for all three components of velocity at an arbitrary z-level, removing any practical limitations based on the relative water depth. High-order finite difference approximations are shown to be more efficient than low-order approximations (for a given accuracy), despite the additional overhead. The resultant system of equations requires that a sparse, unsymmetric, and often ill-conditioned matrix be solved at each stage evaluation within a simulation. Various preconditioning strategies are investigated, including full factorizations of the linearized matrix, ILU factorizations, a matrix-free (Fourier space) method, and an approximate Schur complement approach. A detailed comparison of the methods is given for both rotational and irrotational formulations, and the strengths and limitations of each are discussed. Mesh-independent convergence is demonstrated with many of the preconditioners for solutions of the irrotational formulation, and solutions using the Fourier space and approximate Schur complement preconditioners are shown to require an overall computational effort that scales linearly with problem size (for large problems). Calculations on a variable depth problem are also compared to experimental data, highlighting the accuracy of the model. Through combined physical and mathematical insight effective preconditioned iterative solutions are achieved for the full physical application range of the model. Copyright © 2004 John Wiley & Sons, Ltd.

KEY WORDS: Boussinesq equations; non-linear waves; finite difference methods; sparse matrix preconditioners; mesh-independent convergence

1. INTRODUCTION

Boussinesq methods are widely used for predicting the propagation of non-linear wind-generated waves in harbors and along coast-lines. The principle behind Boussinesq formulations is to incorporate the effect of non-hydrostatic pressure while eliminating the vertical co-ordinate, thus significantly reducing the computational effort relative to a three-dimensional

*Correspondence to: D. R. Fuhrman, Department of Mechanical Engineering, Technical University of Denmark, Studentertorvet, Building 101E, DK-2800 Kgs. Lyngby, Denmark.
†E-mail: drf@mek.dtu.dk

Contract/grant sponsor: Danish Technical Research Council (STVF); contract/grant number 9801635

Received 27 December 2002
Revised 13 June 2003

solution. This principle was initially introduced by the French mathematician Boussinesq in 1872 [1], and since the late 1970s commercial numerical models based on this idea have been developed to solve practical engineering problems. Over the past 20 years, the international use of Boussinesq equations has steadily increased, as many researchers and engineers regard these formulations to be a balanced compromise between detailed results and affordable computational cost.

For many years Boussinesq methods have been limited to applications in fairly shallow water, to non-breaking waves without currents, and to weakly non-linear processes. As a result of an extensive world-wide research effort, the accuracy has improved dramatically over the past decade, pushing the range of applicability out to ever-deeper relative water depths. Boussinesq methods are traditionally formulated in terms of a surface elevation and a horizontal velocity variable, with the vertical velocity variable explicitly eliminated from the equations. Such two-equation systems (counting the horizontal velocity vector as one unknown) have been shown to correspond to Taylor series expansions with limited radii of convergence [2]. This effectively limits the range of applicability of such models to at best (wavenumber times depth) $kh \approx 6$ for surface quantities, and $kh \approx 4$ for vertical velocity distributions, regardless of the order included in the expansion. Examples of modern two-equation systems can be found in References [3, 4]. Madsen and Agnon [2] have shown, however, that by retaining the vertical velocity variable as an unknown, as in References [5, 6], an infinite radius of convergence is obtained. As a result a fifth-order, three-equation model can be used to treat highly non-linear waves to $kh \approx 25$, with accurate velocity profiles up to $kh \approx 12$ effectively removing any shallow water limitation.

In this paper efficient numerical solutions of the three-equation model of Reference [5] (see also Method III of Reference [6]) in two horizontal dimensions are considered. This model combines a time-stepping of the exact free surface boundary conditions with a truncated series expansion solution to the Laplace equation in the interior domain. The expansion is from an arbitrary z-level, and the solution is in terms of all three components of velocity at this level. The increased accuracy of this three-equation model over two-equation models comes at the expense of a rather complicated system of partial differential equations (PDEs). The complexity of the system is such that iterative methods are necessary to obtain efficient solutions for even moderately sized problems. The key to an efficient solution therefore lies in the preconditioning strategy adopted, and a number of possibilities are developed. These include complete/incomplete factorizations of the linearized matrix, a matrix-free (Fourier space) method and an approximate Schur complement approach. A detailed comparison of the methods is given for both rotational and irrotational formulations, and the strengths and limitations of each are discussed. Mesh-independent convergence is demonstrated with many of the preconditioners for solutions of the irrotational formulation, and solutions using the Fourier space and approximate Schur complement preconditioners are shown to require an overall computational effort that scales linearly with problem size (for large problems).

The outline of this paper is as follows: The Boussinesq formulation is outlined in Section 2, while Section 3 describes the numerical (finite difference) model used to solve the equations. Various preconditioning strategies are described in Section 4, and a detailed comparison of these methods is given in Section 5. The results of a variable depth simulation involving non-linear refraction and diffraction are given in Section 6 including comparison with experimental data. Conclusions are drawn in Section 7.

2. THE BOUSSINESQ FORMULATION

Consider the flow of an incompressible, inviscid fluid with a free surface. A Cartesian co-ordinate system is adopted, with the x- and y-axis located on the still-water plane, and with the z-axis pointing vertically upwards. The fluid domain is bounded by the sea bed at $z=-h(\mathbf{x})$, with $\mathbf{x}=\langle x, y\rangle$, and the free surface at $z=\eta(\mathbf{x},t)$, where t is time. It is computationally convenient to express the free surface conditions in terms of velocity variables at the free surface (see e.g. [5–7]). This leads to the following expressions for the kinematic and dynamic free surface conditions

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

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

where

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

Here $\tilde{\mathbf{u}}=\langle\tilde{u},\tilde{v}\rangle$ and $\tilde{w}$ are the horizontal and vertical velocities directly on the free surface, $g=9.81\ \mathrm{m/s^2}$ is the gravitational acceleration, and ∇ is the horizontal gradient operator i.e. $\nabla=\langle\partial/\partial x,\partial/\partial y\rangle$. Evolving η and $\tilde{\mathbf{U}}$ forward in time requires a means of computing the associated $\tilde{w}$, subject to the Laplace equation and the kinematic bottom condition

$$w+\nabla h\cdot\mathbf{u}=0,\quad z=-h \tag{4}$$

For this purpose the Boussinesq method described in detail in References [5, 6] is adopted. This method applies a truncated, Padé-enhanced Taylor series expansion of the velocity potential about an arbitrary level $z=\hat{z}$ in the fluid. In addition, the vertical component of velocity at this level is retained as an unknown, leading to an extremely accurate method (applicable to $kh\approx 25$ for surface quantities and $kh\approx 12$ for vertical velocity distributions). Thus, the vertical distribution of fluid velocity is approximated by

$$\mathbf{u}(\mathbf{x},z,t)=(1-\alpha_2\nabla^2+\alpha_4\nabla^4)\hat{\mathbf{u}}^*(\mathbf{x},t)+((z-\hat{z})\nabla-\beta_3\nabla^3+\beta_5\nabla^5)\hat{w}^*(\mathbf{x},t) \tag{5}$$

$$w(\mathbf{x},z,t)=(1-\alpha_2\nabla^2+\alpha_4\nabla^4)\hat{w}^*(\mathbf{x},t)-((z-\hat{z})\nabla-\beta_3\nabla^3+\beta_5\nabla^5)\hat{\mathbf{u}}^*(\mathbf{x},t) \tag{6}$$

where

$$\begin{aligned}\alpha_2&=\frac{(z-\hat{z})^2}{2}-\frac{\hat{z}^2}{18},\quad \alpha_4=\frac{(z-\hat{z})^4}{24}-\frac{\hat{z}^2(z-\hat{z})^2}{36}+\frac{\hat{z}^4}{504}\\ \beta_3&=\frac{(z-\hat{z})^3}{6}-\frac{\hat{z}^2(z-\hat{z})}{18},\quad \beta_5=\frac{(z-\hat{z})^5}{120}-\frac{\hat{z}^2(z-\hat{z})^3}{108}+\frac{\hat{z}^4(z-\hat{z})}{504}\end{aligned} \tag{7}$$

In (5) and (6) the quantities $\hat{\mathbf{u}}^*$, $\hat{w}^*$ are utility variables which have been introduced to allow Padé enhancement of the Taylor series operators. Optimal velocity distributions are obtained near $\hat{z}=-h/2$, and we adopt this value throughout. Note that terms multiplied by

$\nabla\hat{z}$ from References [5, 6] are neglected in this work. Note also that this formulation differs slightly from Reference [5] in that (5) and (6) are applied throughout the fluid domain. We emphasize that throughout this paper the interpretation of the power of ∇ depends on whether this operator is acting on a scalar or a vector, and in this context the following set of rules should be obeyed (see Reference [8], Chapter 5)

$$\nabla^{2n}\mathbf{u} = \nabla(\nabla^{2n-2}(\nabla\cdot\mathbf{u})), \quad \nabla^{2n+1}\mathbf{u} = \nabla^{2n}(\nabla\cdot\mathbf{u})$$
$$\nabla^{2n}w = \nabla^{2n}w, \quad \nabla^{2n+1}w = \nabla(\nabla^{2n}w)$$

Inserting (5) and (6) into (4) gives the following expression of the kinematic bottom condition, which relates the utility velocity variables $\hat{\mathbf{u}}^*$, $\hat{w}^*$ to each other

$$\left(1 - \frac{4}{9}\gamma^2\nabla^2 + \frac{1}{63}\gamma^4\nabla^4\right)\hat{w}^* + \left(\gamma\nabla - \frac{1}{9}\gamma^3\nabla^3 + \frac{1}{945}\gamma^5\nabla^5\right)\hat{\mathbf{u}}^*$$
$$+\nabla h\cdot(1 - c_2\gamma^2\nabla^2 + c_4\gamma^4\nabla^4)\hat{\mathbf{u}}^* - \nabla h\cdot(\gamma\nabla - s_3\gamma^3\nabla^3 + s_5\gamma^5\nabla^5)\hat{w}^* = 0 \tag{8}$$

where $\gamma = (h + \hat{z})$. Here the slope term coefficients have been modified through numerical optimization with respect to the linear shoaling gradient (see Reference [6]) and in final form are $c_2 = 0.357739$, $c_4 = 0.00663819$, $s_3 = 0.0753019$, and $s_5 = -6.31532\times10^{-5}$. Combining (8) with (5) applied at $z = \eta$, while also invoking (3) gives a 3×3 system that can be solved for $\hat{\mathbf{u}}^*, \hat{w}^*$ in terms of $\tilde{\mathbf{U}}$ and η. The resulting system of PDEs is given in matrix form as

$$\begin{bmatrix} \mathscr{A}_{11} - \eta_x\mathscr{B}_{11} & \mathscr{A}_2 - \eta_x\mathscr{B}_{12} & \mathscr{B}_{11} + \eta_x\mathscr{A}_1 \\ \mathscr{A}_2 - \eta_y\mathscr{B}_{11} & \mathscr{A}_{22} - \eta_y\mathscr{B}_{12} & \mathscr{B}_{12} + \eta_y\mathscr{A}_1 \\ \mathscr{A}_{01} + h_x\mathscr{C}_{11} + h_y\mathscr{C}_{21} & \mathscr{A}_{02} + h_x\mathscr{C}_{12} + h_y\mathscr{C}_{22} & \mathscr{B}_0 - h_x\mathscr{C}_{13} - h_y\mathscr{C}_{23} \end{bmatrix} \begin{bmatrix} \hat{u}^* \\ \hat{v}^* \\ \hat{w}^* \end{bmatrix} = \begin{bmatrix} \tilde{U} \\ \tilde{V} \\ 0 \end{bmatrix} \tag{9}$$

Here the subscripts x and y denote partial differentiation. The system contains a number of operators, which are given in their entirety in the Appendix. For now it is sufficient to mention that each operator contains up to either fourth- or fifth-order mixed partial derivatives. This system of operators shall henceforth be referred to as $\mathscr{A}$, and upon discretization this system shall be referred to as $\mathbf{Ax} = \mathbf{b}$. This paper will in large part concentrate on iterative methods for solving discrete linear systems of this form. It is also worthwhile to note that under the assumption of potential (irrotational) flow such that

$$\frac{\partial u}{\partial y} - \frac{\partial v}{\partial x} = 0 \tag{10}$$

the system simplifies slightly to

$$\mathscr{A} = \begin{bmatrix} \mathscr{A}_1 - \eta_x\mathscr{B}_{11} & -\eta_x\mathscr{B}_{12} & \mathscr{B}_{11} + \eta_x\mathscr{A}_1 \\ -\eta_y\mathscr{B}_{11} & \mathscr{A}_1 - \eta_y\mathscr{B}_{12} & \mathscr{B}_{12} + \eta_y\mathscr{A}_1 \\ \mathscr{A}_{01} + h_x\mathscr{C}_1 & \mathscr{A}_{02} + h_y\mathscr{C}_1 & \mathscr{B}_0 - h_x\mathscr{C}_{13} - h_y\mathscr{C}_{23} \end{bmatrix} \tag{11}$$

This system has certain useful mathematical properties which shall be made apparent later. Water wave models are commonly formulated in terms of a velocity potential, thus not much is lost physically under this assumption. Solutions involving $\mathbf{A}$ stemming from (9) and (11) will both be considered in this work.

Having solved for the utility variables $\hat{\mathbf{u}}^*, \hat{w}^*$ from (9) or (11), $\tilde{w}$ can be computed from (6) applied at $z = \eta$, which closes the problem.

3. THE NUMERICAL MODEL

This section discusses various aspects of the numerical model used for solving the previously outlined system of PDEs. The system of PDEs is solved numerically using finite difference approximations, and the model is programmed in FORTRAN 90.

3.1. Boundary conditions

In the numerical solution of any system of PDEs appropriate boundary conditions must be specified. In the present model combined Dirichlet and Neumann boundary conditions are used to create closed boundaries on a rectangular domain. Specifically, this corresponds to imposing $u = 0$, $\partial v/\partial x = 0$, and $\partial w/\partial x = 0$ along x-boundaries; and $\partial u/\partial y = 0$, $v = 0$, and $\partial w/\partial y = 0$ along y-boundaries. These conditions are imposed simply by reflecting the finite difference coefficients evenly for Neumann boundary conditions and oddly for Dirichlet boundary conditions. This strategy has the advantage of keeping the overall model structure very regular, as all equations are considered in some fashion at each individual grid point.

3.2. Time integration

Throughout the present work the classical fourth-order, four-stage explicit Runge-Kutta method is used for time integration. Due to the wide-spread use of this scheme details are not given here (see e.g. Reference [9]). Other explicit time-stepping methods have also been considered, however this particular method has been found to give a good combination of accuracy and stability at reasonable computational costs. Given the complexity of this system of PDEs implicit methods are not felt to be very attractive.

3.3. Finite difference approximations

In this section we consider the relative merits of several discretizations of $\mathscr{A}$ using finite difference approximations. These are (a) second-order approximations for each partial derivative, (b) a 25-point (diamond) finite difference stencil, (c) a 37-point (octagon) stencil, and (d) a 49-point (square) stencil. Each of these stencils is shown in Figure 1. With the exception of (a), all finite difference approximations are allowed to have the maximum possible accuracy for the given stencil, which results in greater accuracy for the lower-order partial derivatives than for their higher-order counterparts. In particular, under finite difference stencil (b) mixed derivatives of a given order will be less accurate than corresponding pure x- or y-derivatives. Alternatively, stencil (c) gives equal accuracy for all terms of a given order, and stencil (d) results in equal accuracy for all x- and y-derivatives of a given order, regardless of the order

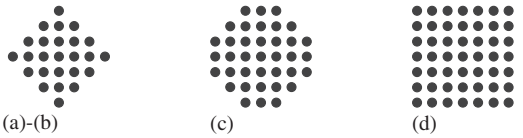

Figure 1. Stencils used for the various combinations of finite difference approximations. These stencils have (a)–(b) 25, (c) 37 and (d) 49 points.

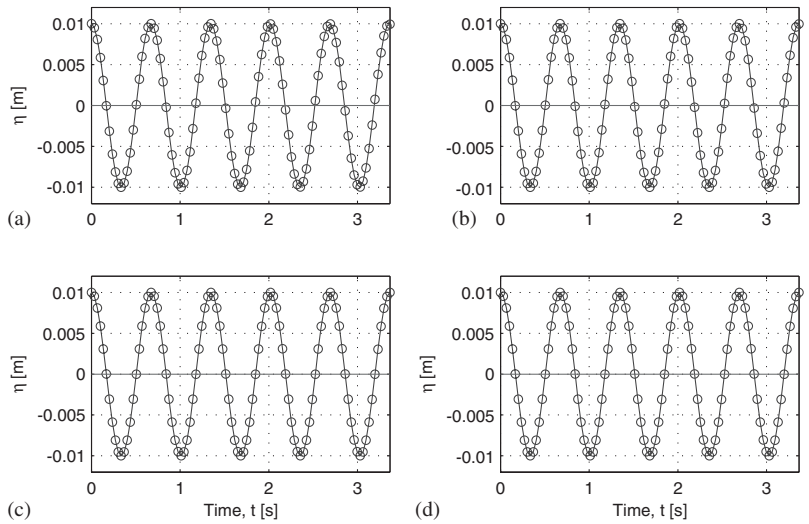

Figure 2. Computed time series of free surface elevations (at the center-point) from linear standing wave simulations using (a) second-order finite difference approximations, (b) a 25-point stencil, (c) a 37-point stencil and (d) a 49-point stencil.

of any accompanying derivatives in the other direction. The minimum stencil that can be used to (centrally) discretize mixed fifth-order partial derivatives is the 25-point stencil, however the larger stencils do not significantly affect the overall structure of **A**. To demonstrate the performance of the various finite difference approximations a linear standing wave in two horizontal dimensions is considered with the initial conditions

$$\tilde{\mathbf{U}}(\mathbf{x},0)=0 \tag{12}$$

$$\eta(\mathbf{x},0)=\frac{H}{2}\cos k_1 x\cos k_2 y \tag{13}$$

where H is the wave height, and k_1 and k_2 are components of the wavenumber vector $\mathbf{k}=\langle k_1,k_2\rangle=\langle 2\pi/L_1,2\pi/L_2\rangle$ (where L_1 and L_2 are wavelengths in the x- and y-directions, respectively). For the remainder of this paper $k=|\mathbf{k}|=\sqrt{k_1^2+k_2^2}$ and $L=L_1=L_2$ shall be used. Note that in the linear sense $\tilde{\mathbf{U}}=\mathbf{u}_0$, where $\mathbf{u}_0=\langle u_0,v_0\rangle$ are horizontal velocity variables at $z=0$. These simulations are on a 21×21 grid, with $H=0.02$ m, $k_1=k_2=2\pi$ m^{-1} (i.e. $H/L=0.02, kh=2\pi$), and $\Delta x=\Delta y=0.05$ m. This gives a linear period $T=0.6730$ s, and the

Table I. RMSE of computed free surface elevations (compared with linear theory) using various finite difference approximations in linear standing wave simulations.

Stencil	Grid	$t = 4.75T$	$t = 5T$	CPU (s)
2nd-order	21×21	3.94×10^{-4}	2.00×10^{-5}	5.99
25-point	21×21	1.74×10^{-4}	6.68×10^{-6}	5.98
37-point	21×21	7.59×10^{-6}	3.46×10^{-6}	8.41
49-point	21×21	1.08×10^{-5}	3.46×10^{-6}	8.60
2nd-order	81×81	1.31×10^{-5}	3.36×10^{-6}	158
25-point	41×41	3.24×10^{-5}	3.50×10^{-6}	29.8

time step is taken to be $\Delta t = T/20 = 0.03365$ s, which has been found to provide sufficient accuracy. Non-linear terms are switched off for the simulations so that results should match linear theory. Resulting time series of surface elevations at the center-point under each finite difference stencil are shown in Figure 2. Here particular attention should be paid to the zero crossings, as these should theoretically correspond to a point from the time series. In Figures 2(a) and (b) the period is seen to be noticeably off, whereas in (c) and (d) it is visually exact under this discretization. Table I gives quantitative results of the root-mean-squared-error (RMSE) over the entire domain at $t = 4.75T$ and at $t = 5T$, where the free surface should theoretically be flat and back to its initial condition, respectively. Some results on larger grids covering the same domain are also shown for comparison. It is again seen that the 37- and 49-point stencils give a substantial reduction in the accumulated error. Indeed, achieving similar accuracy with stencils (a) and (b) requires roughly 15 and 4 times as many grid points, respectively! The reason for the superior accuracy of these two stencils is due to the relative increase in the formal accuracy of the mixed derivatives. Due to the inclusion of mixed fifth-order partial derivatives, this model inevitably requires a fairly large finite difference stencil. Correspondingly, it rather naturally lends itself to higher-order finite difference approximations (for lower-order terms), which can give significant reductions in the overall computational expense (as well as the storage) required for a desired accuracy. Because the 37-point stencil seems to provide essentially the same accuracy as the 49-point stencil, it will be used exclusively in the remainder of this work.

3.4. Matrix properties

The matrix, $\mathbf{A}$, arising upon discretization of $\mathscr{A}$ (using centred finite difference approximations) is unsymmetric, but has a symmetric block structure. The matrix can have a variable sparsity pattern depending on the natural ordering of the equations (i.e. whether the equations are grouped strictly by PDE or by grid point). Both scenarios are shown in Figure 3. It is seen that grouping the equations by PDE, as in Figure 3(a), leads to a natural block structure as seen e.g. in (9). It is often useful to consider such a structure when implementing a preconditioner so that the natural operator structure is maintained. It is likewise seen that grouping the discrete equations by grid point results in a much smaller bandwidth as in Figure 3(b). This ordering results in larger, more concentrated blocks, which allows for a more efficient implementation of sparse matrix–vector multiplication within various iterative solution strategies. Regardless of the ordering used, the matrix is generally far from diagonally dominant.

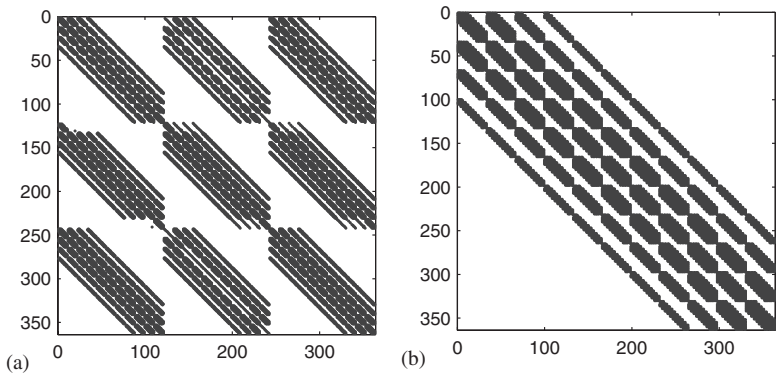

Figure 3. Sparsity patterns of **A** using a 37-point finite difference stencil on an 11×11 grid when grouping the discrete equations (a) by PDE and (b) by grid point.

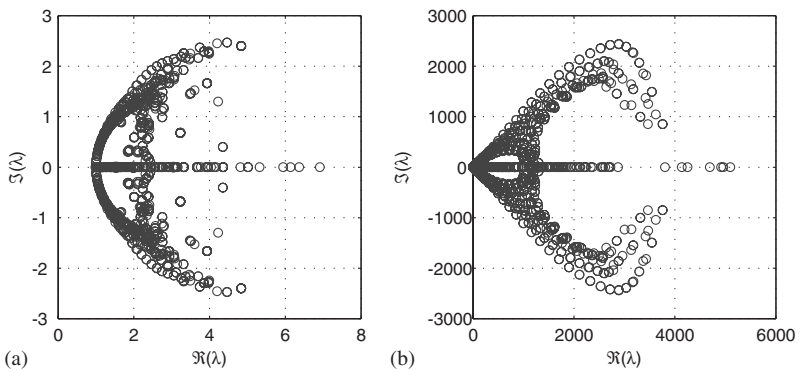

Figure 4. Eigenvalues of (a) a shallow-water matrix with $kh = \dfrac{\pi}{5}$ and (b) a deep-water matrix with $kh = 2\pi$.

The matrix is also somewhat unusual in that, while certainly sparse, it contains a substantial number of non-zeros per row. For example, a 37-point finite difference stencil results in up to $37 \times 3 = 111$ non-zeros per row.

The properties of the matrix also vary widely depending on the physical situation, with the most important parameters being the ratios $h/\Delta x$ and $h/\Delta y$. Note that under a constant discretization (in terms of grid points per wavelength) these parameters are directly proportional to the dimensionless measure of water depth $\mathbf{k}h = \langle k_1 h, k_2 h \rangle$. To illustrate this dependence the spectrum of eigenvalues, $\boldsymbol{\lambda}$, for two matrices having different depths (but with identical free surfaces) are shown in Figure 4. Both matrices are generated from the discretization of (9) on a 21×21 grid, with the free surface given by (13) with $H = 0.05\,\mathrm{m}$, $k_1 = k_2 = 2\pi\,\mathrm{m}^{-1}$, and $\Delta x = \Delta y = 0.05\,\mathrm{m}$. The shallow-water matrix, Figure 4(a), uses $h = 0.07071\,\mathrm{m}$ (i.e. $kh = \pi/5$, $H/h = 0.7071$), while the deep-water matrix, Figure 4(b), uses $h = 0.7071\,\mathrm{m}$ (i.e. $kh = 2\pi$, $H/L = 0.05$). Both matrices have a minimum eigenvalue near unity. The eigenvalues of the

shallow-water matrix are reasonably well clustered, which gives evidence that preconditioning is not so crucial in these situations. The spectrum of the deep-water matrix is dramatically different, having a much larger spread of eigenvalues throughout the right half of the complex plane. Preconditioning deep-water problems therefore seems to be much more critical. These conclusions are further reflected in the respective condition numbers of the two matrices, which are 11.5 and 5.47×10^3 for the shallow- and deep-water matrices, respectively. The matrices become even more ill-conditioned as the depth is further increased (or the grid refined), but these matrices illustrate the general nature of this particular linear system.

3.5. Direct matrix factorizations

Due to the structure of **A** (with a bandwidth that continually increases with problem size) direct matrix factorizations have been found to be uncompetitive as a general solution procedure. However, many of the strategies developed herein use direct methods within a greater iterative solution strategy. For all direct matrix factorizations and corresponding solutions the MA41 package from the well-known Harwell Subroutine Library (HSL) is employed. The factorization method used is a potentially parallel sparse multi-frontal variant of Gaussian elimination, which is particularly effective on matrices whose sparsity pattern is symmetric, or nearly so. The method chooses pivots from the diagonal using the approximate minimum degree algorithm of Reference [10]. When solving systems with a single right-hand side (RHS) the routine also makes efficient use of level 2 Basic Linear Algebra Subprograms (BLAS), which have been optimized using the Automatically Tuned Linear Algebra Software (ATLAS, see e.g. Reference [11]). For full details on the factorization method see References [12–14]. In the present work only the serial version of the code is used (a parallel OpenMP version is also available), however, the potential for parallelism is duly noted here.

3.6. Krylov subspace method

Due to the large number of non-zeros per row in **A**, the Krylov subspace method best suited to solving this linear system is arguably the Generalized Minimal RESidual (GMRES) algorithm of Reference [15]. Indeed, as long as the number of iterations required are kept reasonable (through effective preconditioning), the additional storage required by GMRES is generally the same order of magnitude or less than the matrix itself. Furthermore, the number of iterations must become fairly large before an increase in the number of matrix-vector products can be warranted by restarting the iteration procedure. For these reasons unrestarted GMRES is used throughout this work.

4. PRECONDITIONING METHODS

The key to the success of any Krylov subspace method used to solve non-trivial linear systems lies in effective preconditioning. A good preconditioner must satisfy the often-conflicting criteria of approximating **A** well, while at the same time being somehow ‘easy’ to solve. This section introduces a number of preconditioning strategies that have proven to be effective in solving $\mathbf{Ax} = \mathbf{b}$. Throughout this work the preconditioning matrix shall be denoted

as $\mathbf{M}$, and the preconditioning operation consists of solving systems of the form $\mathbf{Mz} = \mathbf{y}$, where $\mathbf{z}^{\mathrm{T}} = [\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3]$ and $\mathbf{y}^{\mathrm{T}} = [\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3]$. All preconditioning in the present work is done from the left, though in limited testing preconditioning from the right has been found to be equally effective. For recent reviews on iterative methods and preconditioning techniques see References [16, 17], respectively.

4.1. Factored linear preconditioner

A relatively straight-forward method for preconditioning $\mathbf{A}$ is to simply neglect the non-linear terms (with $\eta = 0$ in the remaining operators), which leaves for the rotational system

$$\mathbf{M} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_2 & \mathbf{B}_{11} \\ \mathbf{A}_2 & \mathbf{A}_{22} & \mathbf{B}_{12} \\ \mathbf{A}_{01} + h_x\mathbf{C}_{11} + h_y\mathbf{C}_{21} & \mathbf{A}_{02} + h_x\mathbf{C}_{12} + h_y\mathbf{C}_{22} & \mathbf{B}_0 - h_x\mathbf{C}_{13} - h_y\mathbf{C}_{23} \end{bmatrix} \tag{14}$$

and similarly for the irrotational system

$$\mathbf{M} = \begin{bmatrix} \mathbf{A}_1 & & \mathbf{B}_{11} \\ & \mathbf{A}_1 & \mathbf{B}_{12} \\ \mathbf{A}_{01} + h_x\mathbf{C}_1 & \mathbf{A}_{02} + h_y\mathbf{C}_{22} & \mathbf{B}_0 - h_x\mathbf{C}_{13} - h_y\mathbf{C}_{23} \end{bmatrix} \tag{15}$$

The physical justification for neglecting the non-linear terms is related to the fact that in deep water (where the matrix becomes ill-conditioned) the maximum wave steepness physically possible before breaking is $H/L \approx 0.141$ [18, 19]. This value gives a rough upper estimate for the relative significance of the non-linear terms in deep water. More precisely, as h becomes large $(\eta - \hat{z}) \approx -\hat{z}$, and $\mathbf{M}$ should quite closely resemble $\mathbf{A}$. Unfortunately, $\mathbf{M}$ generally has the same structure as $\mathbf{A}$, and after factorization will have essentially the same storage demands as would a direct method. The advantage of this approach lies simply in the fact that $\mathbf{M}$ is time-constant. The preconditioner can therefore be factored a single time at the beginning of a simulation, with the preconditioning operation consisting only of a solve step. This is quite significant, as a solve step for this system is significantly less expensive than a factorization step, typically by a factor 10–100 for the range of problem sizes considered in this work. To combat the sometimes excessive storage demands associated with this approach single precision (SP) factorizations of $\mathbf{M}$ are also considered (still to precondition $\mathbf{A}$ in double precision), thus reducing the storage by roughly a factor of two.

4.2. ILUT preconditioner

As a lower-storage alternative to the full factorizations described in Section 4.1, incomplete factorizations of $\mathbf{M}$ in (14) or (15) will also be considered. For these purposes the well-known ILUT factorization of Reference [20] is used. This software uses a dual-threshold dropping strategy, and is freely available as part of the SPARSKIT package [21]. In earlier testing (incremental) incomplete factorizations of $\mathbf{A}$ have also been considered. However, the use of incomplete factorizations of the time-constant linear matrix has proven to be a much more efficient alternative. Throughout this work a drop-tolerance of 0.005 is used, combined with

a maximum fill-in of 200 elements per row (in the factors), which have been found to be good general parameters for this problem class.

4.3. Fourier space preconditioner

The linearized version of **A**, as discussed in Section 4.1, should provide an effective preconditioner for this system. Unfortunately the high storage requirements for such complete factorizations can be quite limiting. In search of a more efficient means of applying this idea, we also consider an equivalent operation in Fourier space. In the linear sense (i.e. neglecting non-linear terms), $\mathscr{A}$ relates the (utility) velocities at $\hat{z}$ to the horizontal velocities at $z=0$. According to Stokes' first-order theory for constant h, the relationship between $\hat{\mathbf{u}}$ and $\mathbf{u}_0$ is given as

$$\hat{\mathbf{u}} = \frac{\cosh k(h+\hat{z})}{\cosh kh}\,\mathbf{u}_0 = [\cosh(-k\hat{z}) + \sinh(-k\hat{z})\tanh(k(h+\hat{z}))]^{-1}\mathbf{u}_0 \tag{16}$$

To gain an expression consistent with the embedded properties of the Boussinesq formulation, the infinite operators must first be replaced by Taylor series expansions. Further Padé-enhancement of the resultant expansions corresponds to transforming from $\hat{\mathbf{u}}$ to the utility velocities $\hat{\mathbf{u}}^*$. This procedure ultimately leads to the following replacement operations

$$\cosh(-k\hat{z}) \Rightarrow 1 + k^2\alpha_2 + k^4\alpha_4 \tag{17}$$

$$\sinh(-k\hat{z}) \Rightarrow -k\hat{z} + k^3\beta_3 + k^5\beta_5 \tag{18}$$

$$\tanh(k(h+\hat{z})) \Rightarrow \frac{k\gamma + \frac{1}{9}k^3\gamma^3 + \frac{1}{945}k^5\gamma^5}{1 + \frac{4}{9}k^2\gamma^2 + \frac{1}{63}\gamma^4} \tag{19}$$

where setting $z=0$ in (7) gives $\alpha_2 = 4\hat{z}^2/9$, $\alpha_4 = \hat{z}^4/63$, $\beta_3 = -\hat{z}^3/9$, $\beta_5 = -\hat{z}^5/945$, and $\gamma = (h+\hat{z})$. Inserting (17)–(19) into (16) and setting $\hat{z} = -h/2$ gives the final relationship

$$\hat{\mathbf{u}}^* = \left[1 + \frac{k^2h^2}{9} + \frac{k^4h^4}{1008} + \frac{(15,120kh + 420k^3h^3 + k^5h^5)^2}{907,200(1008 + 112k^2h^2 + k^4h^4)}\right]^{-1}\mathbf{u}_0 \tag{20}$$

The corresponding preconditioning operation consists of firstly transforming the components of the preconditioning RHS $\mathbf{y}_1$, $\mathbf{y}_2$, and $\mathbf{y}_3$ into Fourier space (treating each as 2D arrays). For this operation 2D combinations of fast sine and cosine transforms are used as appropriate for Dirichlet and Neumann boundary conditions, respectively. The preconditioning is applied entirely in Fourier space, which uses $\Delta k_1 = \pi/(\Delta x(N_1-1))$ and $\Delta k_2 = \pi/(\Delta y(N_2-1))$, where N_1 and N_2 are the number of grid points in the x- and y-directions, respectively. The complete operation is given in Algorithm 1, which is seen to include the truncated relationship from (20). Note also that solutions for $z_3(i,j)$ are found simply through spectral differentiation i.e. replacing $\partial/\partial x$ and $\partial/\partial y$ with $\mathrm{i}k_1$ and $\mathrm{i}k_2$, respectively, in the flat-bottom operators $\mathscr{A}_{01}$, $\mathscr{A}_{02}$, and $\mathscr{B}_0$. Once the loops are complete, the 2D arrays corresponding to $\mathbf{z}_1$, $\mathbf{z}_2$, and $\mathbf{z}_3$ are inverse-transformed back to physical space, completing the preconditioning operation.

Algorithm 1 Algorithm for the Fourier space preconditioning operation.

$\mathbf{y}_1 = \mathscr{F}\{\mathbf{y}_1\};\ \mathbf{y}_2 = \mathscr{F}\{\mathbf{y}_2\};\ \mathbf{y}_3 = \mathscr{F}\{\mathbf{y}_3\}$
for $j = 1$ to N_2 **do**
  $k_2 = (j-1)\Delta k_2$
  for $i = 1,\ N_1$ **do**
    $k_1 = (i-1)\Delta k_1;\ k = \sqrt{k_1^2 + k_2^2}$

$$\varepsilon = \left(1 + \frac{k^2h^2}{9} + \frac{k^2h^2}{1008} + \frac{(15,120kh + 420k^3h^3 + k^5h^5)^2}{907,200(1008 + 112k^2h^2 + k^4h^4)}\right)^{-1}$$

$z_1(i,j) = \varepsilon y_1(i,j);\ z_2(i,j) = \varepsilon y_2(i,j)$

$$\hat{A}_{01} = \frac{k_1 h}{30,240}(15,120 + 420k^2h^2 + k^4h^4)$$

$$\hat{A}_{02} = \frac{k_2 h}{30,240}(15,120 + 420k^2h^2 + k^4h^4)$$

$$\hat{B}_0 = 1 + \frac{k^2h^2}{9} + \frac{k^4h^4}{1008}$$

$$z_3(i,j) = \frac{1}{\hat{B}_0}(y_3(i,j) - \hat{A}_{01}z_1(i,j) - \hat{A}_{02}z_2(i,j))$$

  end for
end for
$\mathbf{z}_1 = \mathscr{F}^{-1}\{\mathbf{z}_1\};\ \mathbf{z}_2 = \mathscr{F}^{-1}\{\mathbf{z}_2\};\ \mathbf{z}_3 = \mathscr{F}^{-1}\{\mathbf{z}_3\}$

This preconditioner should essentially provide the same operation as the factored (irrotational) linear matrix with constant h in (15). It is entirely matrix-free, however, thus any additional storage requirements are negligible. This preconditioner requires a global value for h to be applied in Fourier space. The hope was that simply taking an average value over the domain would still be effective in preconditioning the system on a variable bottom. Unfortunately this simple strategy does not appear to work, and it is not immediately clear how to apply the idea on a variable depth. Therefore, in the present work applications of this preconditioner will be limited to cases having constant depth. Because this method stems from potential theory it is also expected to be more effective in preconditioning the irrotational system than the rotational system.

4.4. Approximate Schur complement preconditioner

The derivation of an approximate Schur complement preconditioner shall begin with the irrotational, flat-bottom system given by

$$\begin{bmatrix} \mathscr{A}_1 & & \mathscr{B}_{11} \\ & \mathscr{A}_1 & \mathscr{B}_{12} \\ \mathscr{A}_{01} & \mathscr{A}_{02} & \mathscr{B}_0 \end{bmatrix}$$

The justification for neglecting the slope terms is that the formulation (with arbitrary $\hat{z}$) inherently includes a mild slope assumption (see again References [5, 6], as well as Reference [22]). Thus, terms multiplied by h_x and h_y in (9) and (11) should be of secondary importance. There

is nothing preventing the application of preconditioners based on this formulation on variable depth problems, however. Note that in this version, the upper-left 2×2 system is block diagonal, with the $\mathscr{A}_1$ operator on both diagonals. This rather unique structure will be taken advantage of in the following. The Schur complement with respect to this upper left 2×2 system is

$$\mathscr{S} = \mathscr{B}_0 - \mathscr{A}_{01}\mathscr{A}_1^{-1}\mathscr{B}_{11} - \mathscr{A}_{02}\mathscr{A}_1^{-1}\mathscr{B}_{12} \tag{21}$$

It seems natural to first simplify $\mathscr{S}$ through multiplication by $\mathscr{A}_1$, which leaves

$$\mathscr{S}_0 = \mathscr{A}_1\mathscr{B}_0 - \mathscr{A}_{01}\mathscr{B}_{11} - \mathscr{A}_{02}\mathscr{B}_{12} \tag{22}$$

This operation assumes commutivity for all the operators in $\mathscr{S}$, which is strictly true only when h is constant (i.e. on a flat bottom). However, experience has shown that it is still reasonable for preconditioning purposes on mildly sloping bathymetries. In full form $\mathscr{S}_0$ is the following 10th-order operator

$$\mathscr{S}_0 = 1 - \frac{17h^2\nabla^2}{36} + \frac{16h^4\nabla^4}{567} - \frac{h^6\nabla^6}{2240} + \frac{29h^8\nabla^8}{15,240,960} - \frac{h^{10}\nabla^{10}}{914,457,600} \tag{23}$$

At first glance inverting the discrete sub-matrix $\mathbf{S}_0$ might seem an insurmountable task (here a sub-matrix refers to a matrix with rank equal to the number of grid points N, i.e. $\frac{1}{3}$ that of $\mathbf{A}$); however it turns out, quite remarkably, that the operator can be factored into five second-order modified Helmholtz operators i.e.

$$\mathscr{S}_0 = \prod_{i=1}^{5} (1 - a_i h^2\nabla^2) \tag{24}$$

where $a_1 = 0.4052847276166189$, $a_2 = 0.045003441158841 87$, $a_3 = 0.015580175994150 51$, $a_4 = 0.005675885605119240$ and $a_5 = 0.000677991847491 6520$. Such a factorization can be shown to exist by considering (23) as a polynomial in $h^2\nabla^2$, which can in-turn be shown to have all real roots. Similarly, $\mathscr{A}_1$ can be factored as

$$\mathscr{A}_1 = \left(1 - \frac{h^2\nabla^2}{4(14 - \sqrt{133})}\right)\left(1 - \frac{h^2\nabla^2}{4(14 + \sqrt{133})}\right) \tag{25}$$

Thus, the preconditioning operation can be simplified to the quite-manageable task of inverting nine second-order sub-matrices and three sub-matrix-vector multiplications! The preconditioner is given by

$$\mathbf{M} = \begin{bmatrix} \mathbf{A}_1 & & \\ & \mathbf{A}_1 & \\ \mathbf{A}_{01} & \mathbf{A}_{02} & \mathbf{S}_0\mathbf{A}_1^{-1} \end{bmatrix} \tag{26}$$

where $\mathbf{S}_0$ and the two upper-left $\mathbf{A}_1$ sub-matrices correspond to discrete forms of (24) and (25), respectively. It should be stressed that the three $\mathbf{A}_1$ sub-matrices in (26) all have different boundary conditions corresponding to their respective column position. It is interesting to

mention that the preconditioned (linear, irrotational, flat-bottom) system (maintaining operator form) becomes

$$\mathscr{M}^{-1}\mathscr{A} = \begin{bmatrix} 1 & 0 & \mathscr{A}_1^{-1}\mathscr{B}_{11} \\ 0 & 1 & \mathscr{A}_1^{-1}\mathscr{B}_{12} \\ 0 & 0 & 1 \end{bmatrix} \tag{27}$$

Thus, even though $\mathbf{M}^{-1}$ in no sense approximates $\mathbf{A}^{-1}$, it should still have a similar clustering effect on the eigenvalues of $\mathbf{A}$. All sub-matrices in $\mathbf{M}$ are again time-constant, and can be built and (if necessary) factored a single time at the beginning of a simulation. Second-order (five-point) finite difference approximations for all Laplacian operators are used in practice as MA41 is extremely effective in limiting fill-in in matrices having a single outer band. As a result, this preconditioner should have very low storage demands, and as no truncations have been imposed, should not be limited by problem depth. It is again expected that this approach will be more effective in preconditioning the irrotational matrix than the full rotational matrix, however it shall be applied in either case.

Discussion on the use of Schur complement preconditioners on matrices with reasonably similar (2×2) block structures is given in References [23, 24]. For other applications see e.g. References [25, 26].

5. COMPARISON OF PRECONDITIONERS

To compare the preconditioning strategies outlined in Section 3 non-linear simulations using the initial conditions from (12) and (13) shall be used, again with $k_1 = k_2 = 2\pi\ \mathrm{m}^{-1}$ (i.e. $L = L_1 = L_2 = 1\ \mathrm{m}$). This provides a simple means for varying the non-linearity, water depth, and discretization. For ease of interpretation, all results will be reported in this section in terms of the dimensionless variables kh, and either H/h or H/L for shallow- and deep-water cases, respectively. As reference values, $kh \approx \pi$ is often taken as the practical deep-water limit, and (as noted previously in Section 4.1) the maximum deep-water wave steepness physically possible (before wave breaking) is $H/L \approx 0.141$. Similarly, at the shallow water limit breaking occurs at $H/h \approx 0.8$. All computations are performed on a Dell Pentium 4 1.8 GHz processor with 1024 MB DDR RAM, and all iterative solutions use a relative residual error tolerance $r = \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2 / \|\mathbf{b}\|_2$ of 10^{-6}. Within all simulations the previously found solution vector $\mathbf{x}$ is used as the starting guess for each successive iterative solution.

5.1. Performance versus relative water depth

Figure 5 demonstrates how the relative water depth, kh, affects the performance of the various preconditioning strategies under a constant discretization. These simulations use a 33×33 grid, are for 101 time steps, with $H/L = 0.05$ m, $\Delta x = \Delta y = 0.0625$ m (i.e. 16 grid points per wavelength), and $\Delta t = 0.03365$ s. The domain covers two wavelengths in both horizontal directions. Table II also provides a summary of the simulations at both the shallow and deep extremes, giving the range of iterations required with each method. In quite shallow water it can be seen that preconditioning is perhaps not so critical, as even the results with no preconditioning are reasonable. This is consistent with the expectations from Section 3.4. As

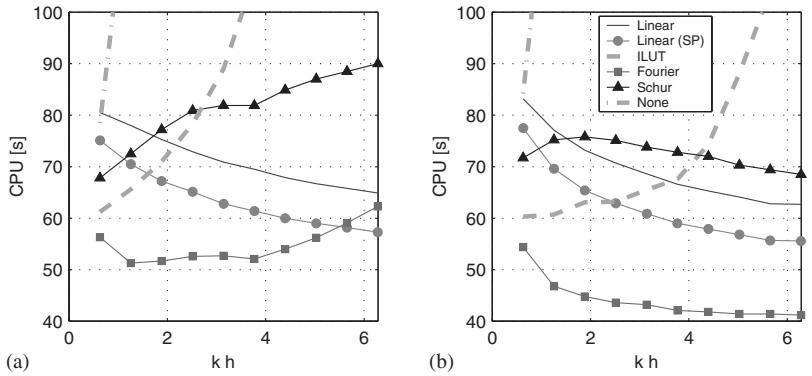

Figure 5. CPU times (101 time steps, $H/L = 0.05$) under variable depth for solving (a) the rotational system and (b) the irrotational system.

Table II. A summary of the simulations having the minimum and maximum kh values from Figure 5. All simulations are on a 33×33 grid, for 101 time steps, with $\Delta x = \Delta y = 0.0625$ m, and $\Delta t = 0.03365$ s. The description column corresponds to the entire list of simulations to the right.

		Rotational		Irrotational	
Description	Preconditioner	Iterations	CPU (s)	Iterations	CPU (s)
Shallow water:	Linear	3–12	80.5	3–13	83.2
$kh = \frac{\pi}{5}$	Linear (SP)	3–12	75.1	2–13	77.5
$\frac{H}{h} = 0.7071$	ILUT	3–14	61.2	3–16	60.3
	Fourier	5–14	56.3	4–15	54.4
	Schur	4–13	67.8	5–15	71.7
	None	8–23	78.5	9–27	84.2
Deep water:	Linear	2–9	64.9	2–9	62.7
$kh = 2\pi$	Linear (SP)	2–10	57.3	2–9	55.6
$\frac{H}{L} = 0.05$	ILUT	14–40	309	7–21	128
	Fourier	2–21	62.3	2–10	41.2
	Schur	3–20	90.0	4–14	68.5
	None	94–206	747	85–129	513

kh increases (even moderately), however, it is seen that some form of preconditioning becomes absolutely necessary. The ILUT preconditioner works quite well in shallow to intermediately deep water, however it rapidly loses effectiveness as the depth is further increased. This trend is more exaggerated when solving rotational problems, but the method eventually fails in either case. A somewhat similar loss in effectiveness is seen with the Schur complement and (to some degree) with the Fourier space preconditioner for the rotational matrix. The observed increase in CPU time is much more controlled in these instances, however. Both provide noticeably more efficient solutions for irrotational simulations, which was expected from Sections 4.3

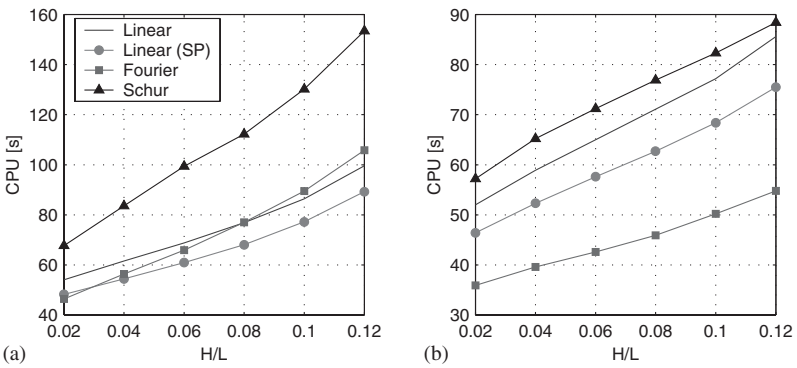

Figure 6. CPU times (101 time steps, $kh = 2\pi$) under variable deep-water non-linearity when solving (a) the rotational system and (b) the irrotational system.

and 4.4. Somewhat remarkably, many of the preconditioners actually become more effective as the depth is increased. This can again easily be explained physically by the fact that as h increases $(\eta - \hat{z}) \approx -\hat{z}$. These results will inevitably change under different discretizations, non-linearity, and problem size, however the trends seen here remain very consistent. Experience has shown that the results in Figure 5 are quite representative.

5.2. Performance versus non-linearity

Figure 6 provides a similar comparison of CPU times for simulations where the deep-water non-linearity (or wave steepness), H/L, is varied. These simulations are again on a 33×33 grid, for 101 time steps, with $kh = 2\pi$, $\Delta x = \Delta y = 0.0625\,\text{m}$, and $\Delta t = 0.03365\,\text{s}$. As Figure 5 has shown simulations with the ILUT preconditioner to be uncompetitive at this depth, this preconditioner is not considered in the remainder of this section. As Figure 6 demonstrates, all of the preconditioning methods gradually lose some effectiveness as the non-linearity is increased. This is expected, as the non-linear terms have been neglected in the preconditioners. The growth is very acceptable, however, and the simulation time grows roughly linearly with the wave steepness. As can be seen, the preconditioners remain effective even when the non-linearity is quite high (results up to $H/L = 0.12$ are shown). Consistent with previous observations, the Fourier space and Schur complement methods are more sensitive to increases in non-linearity when solving the rotational system, as characterized by their steeper slopes.

5.3. Performance versus grid refinement

Figure 7 demonstrates how the preconditioning strategies perform when the mesh is refined under a constant depth and non-linearity. These simulations use $kh = 2\pi$, $H/L = 0.08$, and a constant fraction $\Delta x/\Delta t = \Delta y/\Delta t = 1.857\,\text{m/s}$. Each simulation uses a domain covering a single wavelength in each horizontal direction, and covers the equivalent of a linear period i.e. to $t = 0.6730\,\text{s}$. The reported iterations are the average from each simulation. These tests are quite demanding, as refinements in the mesh make $\mathbf{A}$ increasingly ill-conditioned. Solutions for the rotational system using the Schur complement and linear (SP) preconditioners can be seen to be rather sensitive to refinements in the mesh, as the number of required iterations increases significantly. The linear (SP) preconditioner is fairly robust with discretizations up

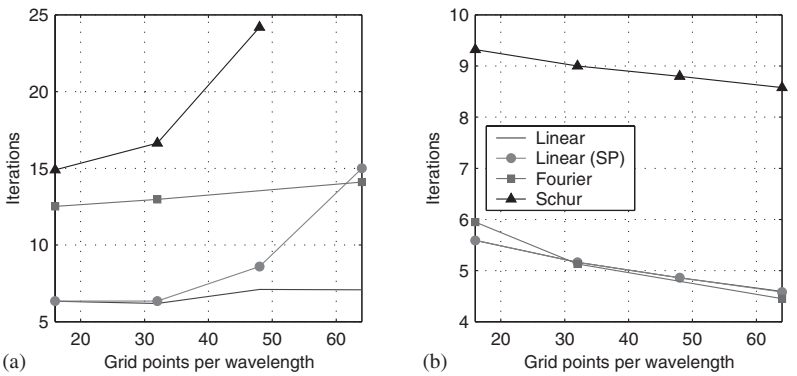

Figure 7. Average number of iterations required over a linear period ($kh = 2\pi$, $H/L = 0.08$) when solving (a) the rotational system and (b) the irrotational system. The domain in each simulation covers a single wavelength in each horizontal direction, and the time step is varied such that $\Delta x/\Delta t = \Delta y/\Delta t = 1.857$ m/s is constant. Note that the curves for both linear preconditioners in (b) are visually indistinguishable from one another.

to around 48 grid points per wavelength, however. The results for the irrotational system are most impressive, as the required number of iterations actually decreases as the mesh is refined. This decrease is simply due to the use of smaller time steps as Δx and Δy are reduced, thus the starting guess for each iterative solution becomes better as the mesh is refined. The results with both linear preconditioners are virtually indistinguishable from one another in Figure 7(b). Over this quite realistic discretization range the convergence using each of the preconditioners for solving the irrotational system appears to be mesh independent. The linear and (somewhat surprisingly) Fourier space preconditioners seem to be the most robust when solving the rotational system. In practice (see e.g. Section 3.3) solutions using as few as 15–20 grid points per wavelength have been found to give sufficient accuracy, perhaps making such fine discretizations unnecessary with this model. Mesh independence is a very desirable property nonetheless, and is rarely achieved with conventional (ILU or approximate inverse) preconditioning techniques [17]. The use of complete factorizations within the greater iterative strategy seems to have made this achievement possible.

5.4. Storage comparison

As hinted in Section 4, the storage required by each of the preconditioners varies significantly. To illustrate this point, Table III shows the length of real storage that must be allocated for each preconditioner for a variety of problem sizes. Also shown for comparison is the length of the real array required to store **A**. The linear preconditioners can be seen to have quite large storage demands, which are generally an order of magnitude more than for the matrix itself. The use of single precision factorizations has proven to be an effective method for reducing these demands, however even in this case the storage can be quite limiting. The Fourier and Schur complement methods, on the other hand, have much lower storage requirements. These results show that the additional storage required by the Fourier space preconditioner is essentially negligible, while that of the Schur complement preconditioner is roughly double that of **A** (for large problems), which is still very reasonable.

Table III. Length of real storage allocated for the various preconditioners compared with the matrix **A** for a wide range of problem sizes.

Grid	N	**A**	Linear	Fourier	Schur
17×17	289	8.1×10^4	4.0×10^6	8.7×10^2	1.1×10^5
33×33	1089	3.3×10^5	2.3×10^6	3.3×10^3	4.9×10^5
65×65	4225	1.3×10^6	1.5×10^7	1.3×10^4	2.3×10^6
129×129	16,641	5.4×10^6	8.4×10^7	5.0×10^4	1.1×10^7
257×129	33,153	1.1×10^7	1.8×10^8	9.9×10^4	2.4×10^7
257×257	66,049	2.2×10^7	4.3×10^8	2.0×10^5	5.3×10^7

Table IV. Percentage CPU time spent in major operations during irrotational simulations (101 time steps, $kh = 2\pi$, $H/L = 0.05$) on a 129×129 grid using the various preconditioners. The numbers in parentheses correspond to the percentage of the preconditioning operation spent in level 2 BLAS. The total CPU time for each simulation is also provided.

Operation	Linear	Linear (SP)	Fourier	Schur
Mat.-vec. prod.	19.8	24.6	50.5	40.3
Build **A**	12.3	15.1	28.4	15.3
Preconditioning	63.1 (89.0)	54.7 (88.7)	13.8	38.6 (25.1)
GMRES	1.2	1.8	3.7	2.9
Time int.	1.2	1.4	2.7	1.5
Misc.	2.4	2.4	0.9	1.4
Total CPU (s)	1587	1295	679	1268

5.5. *Breakdown of computational expenses*

Given the large differences in the storage demands (and therefore in the corresponding number of required floating point operations per iteration), it might seem surprising that the linear preconditioning methods are competitive at all with the Fourier and Schur complement approaches, as the required number of iterations do not differ nearly as significantly. The explanation is apparent upon a profiling of the simulations, a sample of which is given in Table IV. Here a breakdown of the computational expense of the major operations (in percentages) is provided for solutions with each preconditioner. The results shown are from irrotational simulations using 101 time steps on a 129×129 grid, with $kh = 2\pi$, $H/L = 0.05$, $\Delta x = \Delta y = 0.0625\,\mathrm{m}$, and $\Delta t = 0.03365\,\mathrm{s}$. It is seen that the preconditioning operation dominates the time spent in solutions using the linear preconditioners. Quite remarkably, nearly 90 per cent of this operation is spent inside level 2 BLAS routines. Thus, with these preconditioning strategies a high flop rate more closely associated with direct methods is achieved. Alternatively, the Fourier space preconditioning method is seen to be dominated by the sparse matrix-vector product, which is an inherently slower operation. Solutions using the Schur complement preconditioner require roughly the same time for the preconditioning operation and the sparse matrix-vector product, with a much smaller portion of the preconditioning operation spent in level 2 BLAS. Also noteworthy is the fairly small portion spent inside GMRES (regardless of the preconditioner), which seemingly confirms the arguments from Section 3.6. We stress

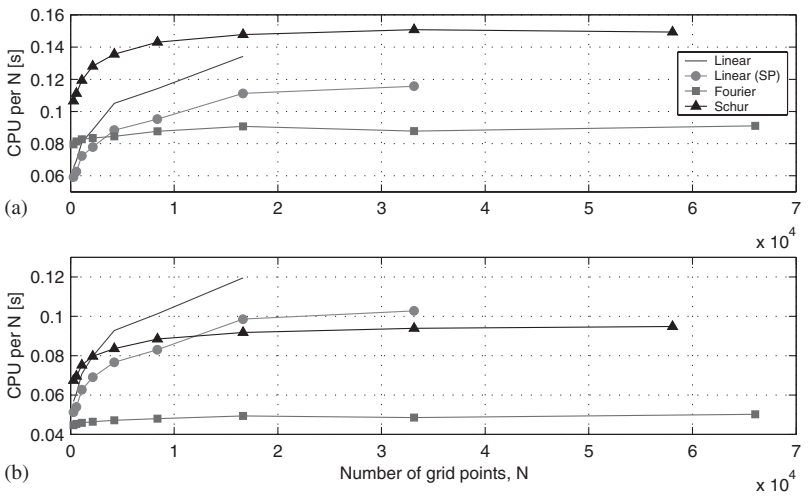

Figure 8. CPU time per grid point (101 time steps, $kh = 2\pi$, $H/L = 0.10$) for solving (a) the rotational system and (b) the irrotational system.

that the information in Table IV is only meant to provide a comparison of the relative expense of the major operations for simulations using each preconditioner. To obtain the overall expense of each operation, the percentages can be multiplied by the total CPU time given at the bottom of the table.

5.6. *Performance versus problem size*

To gain insight into how solutions using the various preconditioning strategies scale with problem size, Figure 8 shows the results of simulations where the number of grid points, N, is continually increased. Note that the rank of $\mathbf{A}$ is actually $3N$, as each grid point houses velocity variables in three directions. All simulations are for 101 time steps, with $kh = 2\pi, H/L = 0.10$, $\Delta x = \Delta y = 0.0625$ m, and $\Delta t = 0.03365$ s. Results solving both the rotational and irrotational systems are shown. Solutions with each preconditioner are carried out roughly to the maximum problem size possible on this machine (with 1024 MB RAM). The average number of iterations required for each simulation is essentially identical to that presented in Figure 6 for this non-linearity. The degree of non-linearity does not affect the general shape of the curves—it does, however, affect the relative solution times of the various methods, as should also be expected from Figure 6. Perhaps the most impressive of the methods considered is the Fourier space preconditioner, which gives a constant solution time per grid point for virtually any size of problem. The relative expense of the other preconditioning methods gradually flattens as the problem size is increased, which is perhaps more typical. Also noteworthy is the performance of the Schur complement preconditioner (especially in the irrotational simulations), which levels off much faster than the linear preconditioners. Indeed, it is seen that although this method is slower for small problems, it becomes the fastest of the variable-depth preconditioners for large potential flow problems. As mentioned previously, the method is less effective in preconditioning the rotational matrix, but for large variable

depth problems (moderately deep, where the non-linearity is not too large) it still seems to be a viable alternative. The linear preconditioners are equally effective in preconditioning both the rotational and irrotational systems, making them perhaps the most robust of the schemes devised.

5.7. *Overview*

As demonstrated throughout this section, each of the preconditioning methods presented has its own respective strengths and weaknesses. It is when viewed as complementary that they are seen to be very robust, as they efficiently cover the entire physical range of applicability of the Boussinesq model. Interestingly, many of the methods also serve as quite modern examples of the combined use of direct and iterative methods for sparse matrices, and as a result are robust in situations where more conventional ILU-based methods fail. In short, through combined physical and mathematical insight, the preconditioning methods successfully transform an extremely difficult problem to one 'whose solution can be approximated rapidly' [27], even in the most physically demanding situations. This is, of course, the very essence of preconditioning.

6. MODEL VERIFICATION

To demonstrate the effectiveness of the preconditioning strategies on a more realistic problem (i.e. with a variable depth) the experiments of Reference [28] involving non-linear refraction and diffraction shall be considered. Specific attention will be paid to the deep-water case with $T = 1$ s. This experiment has been used extensively in the literature to validate numerical models, and for a detailed description of the underlying physics the reader is referred to Reference [29]. The topography used connects deep and shallow regions with a shoaling region that acts as a focusing lens, and is described (in meters) by

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

where

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

Gradients of h in both horizontal directions are also calculated analytically. Because the bathymetry is symmetric about the centerline (i.e. at $y = 3.048$ m) only half of the domain is modeled. For all simulations $\Delta x = 0.0762$ m, $\Delta y = 0.1524$ m, and $\Delta t = 0.03906$ s are used. Because the variation is much weaker than in the x-direction, a coarser discretization in the y-direction is justified. A 50-point sponge layer is applied in the region after $x = 30$ m to prevent reflections. An analytical wave-maker is applied at the deep end (relaxed over 21 grid points, in the negative x-direction), where a stream-function solution (as in Reference [30]) is imposed. The solution uses $H = 0.039$ m, and (Stokes' drift velocity) $c_s = 0$ m/s to match the conditions of a closed flume, and gives a wavelength $L = 1.50$ m. The total computational

Table V. Summary of simulations modelling the experiment of Reference [28] with $T = 1$ s.

Formulation	Preconditioner	Iterations	CPU (h)
Rotational	Linear	4–10	4.39
Rotational	Linear (SP)	3–10	6.59
Rotational	ILUT	4–10	4.57
Rotational	Schur	6–17	5.13
Irrotational	Linear	3–10	4.75
Irrotational	Linear (SP)	3–10	6.39
Irrotational	ILUT	4–10	4.16
Irrotational	Schur	6–13	5.09

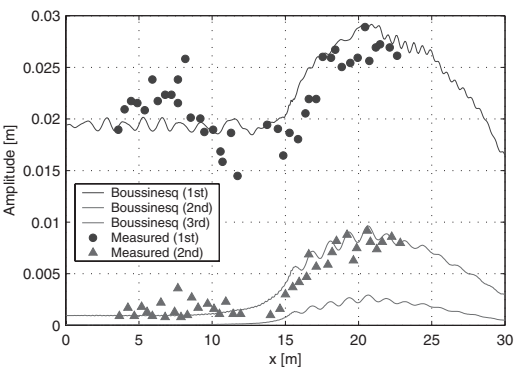

Figure 9. Computed and measured harmonic amplitudes for simulations modelling the experiments of Reference [28] with $T = 1$ s.

domain consists of a 467×21 grid, and simulations are for 2001 time steps. In each simulation a 10th-order, 109-point (octagon) Savitzky-Golay smoothing filter [31, 32] is applied every 20 time steps, which is necessary to remove high-frequency instabilities caused by the non-linear terms (this discretization can be shown to satisfy standard linear stability criterion). The end result is a very minor loss of accuracy.

Table V provides a summary of the simulations using the various preconditioning strategies for both rotational and irrotational simulations. As this problem is not extremely deep, all of the preconditioning methods are very effective. However, solutions of this problem are far from trivial and some form of preconditioning is necessary to achieve reasonable solution times. Notably, the Schur complement preconditioner remains effective on this variable-depth problem, even though it has neglected bottom slope terms. Curiously, the results using the linear (SP) preconditioner are slower than with the double precision alternative, which contradicts previous findings.

Figure 9 shows computed harmonic amplitudes along the centerline at $y = 3.048$ m along with experimental measurements. The harmonic analysis uses a linear least-squares fit with data from the last 500 time steps. The results shown are from the rotational simulation (using the linear preconditioner), however those from the other simulations are virtually identical.

The match with the experimental data is most acceptable, and compares well with others from the literature, thus highlighting the accuracy of the Boussinesq model. Other, more difficult problems involving non-linear, deep-water wave dynamics are currently under investigation.

7. CONCLUSIONS

This paper presents a new high-order finite difference method for fully non-linear and highly dispersive water waves in two horizontal dimensions. The method solves the high-order, three-equation Boussinesq formulation derived in References [5, 6]. High-order finite difference approximations are shown to be more efficient than low-order approximations (for a given accuracy), despite the additional overhead. The formulation requires that an often ill-conditioned sparse matrix arising from a non-linear system of PDEs be solved at each stage evaluation, and a number of different preconditioning strategies are developed for this purpose. These include complete factorizations of the linearized matrix, ILU factorizations, a matrix-free (Fourier space) method, and an approximate Schur complement approach. The preconditioners are tested under a variety of physical situations (i.e. varying the depth, discretization, and non-linearity), as well as on both rotational and irrotational formulations. With the exception of the ILU-based method, all preconditioners are found to be very effective in solving deep-water problems, which are by far the most difficult. In particular it is shown that the factored linear preconditioners are perhaps the most robust of the methods devised, as they are equally effective in solving both the rotational and irrotational systems. Their high storage demands, however, can limit the problem size to some degree. Alternatively, the Fourier space preconditioner has essentially negligible storage demands, and consistently produces the fastest solutions (for irrotational simulations) when it is applicable. Unfortunately, it is seemingly limited to solving constant depth problems on regular domains. Finally, the approximate Schur complement method has low storage demands, and is particularly effective in solving large potential-flow problems. Mesh-independent convergence is demonstrated with many of the preconditioners for solutions of the irrotational formulation, and solutions using the Fourier space and approximate Schur complement preconditioners are shown to require an overall computational effort that scales linearly with problem size (for large problems). As is evident, each of the methods have their own respective strengths and weaknesses, and should therefore be viewed as complementary. In general, the methods are quite robust, and are effective for the full physical range of applicability of the model. Results matching a well-known physical experiment demonstrate the applicability of the methods on a variable depth problem, and highlight the accuracy of the numerical model.

APPENDIX

This section includes the various operators in the system of PDEs denoted herein as $\mathscr{A}$. The enhanced free surface operators from the rotational system (9) are

$$\mathscr{A}_{11} = 1 - \alpha_2 \left(\frac{\partial^2}{\partial x^2} \right) + \alpha_4 \left(\frac{\partial^4}{\partial x^4} + \frac{\partial^4}{\partial x^2 \partial y^2} \right) \tag{A1}$$

$$\mathscr{A}_2 = -\alpha_2 \left(\frac{\partial^2}{\partial x \partial y} \right) + \alpha_4 \left(\frac{\partial^4}{\partial x^3 \partial y} + \frac{\partial^4}{\partial x \partial y^3} \right) \tag{A2}$$

$$\mathscr{A}_1 = 1 - \alpha_2 \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) + \alpha_4 \left(\frac{\partial^4}{\partial x^4} + 2\frac{\partial^4}{\partial x^2 \partial y^2} + \frac{\partial^4}{\partial y^4} \right)$$

$$\mathscr{B}_{11} = (\eta - \hat{z}) \left(\frac{\partial}{\partial x} \right) - \beta_3 \left(\frac{\partial^3}{\partial x^3} + \frac{\partial^3}{\partial x \partial y^2} \right) + \beta_5 \left(\frac{\partial^5}{\partial x^5} + 2\frac{\partial^5}{\partial x^3 \partial y^2} + \frac{\partial^5}{\partial x \partial y^4} \right) \tag{A3}$$

$$\mathscr{A}_{22} = 1 - \alpha_2 \left(\frac{\partial^2}{\partial y^2} \right) + \alpha_4 \left(\frac{\partial^4}{\partial x^2 \partial y^2} + \frac{\partial^4}{\partial y^4} \right) \tag{A4}$$

$$\mathscr{B}_{12} = (\eta - \hat{z}) \left(\frac{\partial}{\partial y} \right) - \beta_3 \left(\frac{\partial^3}{\partial x^2 \partial y} + \frac{\partial^3}{\partial y^3} \right) + \beta_5 \left(\frac{\partial^5}{\partial x^4 \partial y} + 2\frac{\partial^5}{\partial x^2 \partial y^3} + \frac{\partial^5}{\partial y^5} \right) \tag{A5}$$

with the α and β coefficients in (7) applied at $z = \eta$. The basic bottom operators are

$$\mathscr{A}_{01} = (h + \hat{z}) \left(\frac{\partial}{\partial x} \right) - \frac{1}{9}(h + \hat{z})^3 \left(\frac{\partial^3}{\partial x^3} + \frac{\partial^3}{\partial x \partial y^2} \right) + \frac{1}{945}(h + \hat{z})^5 \left(\frac{\partial^5}{\partial x^5} + 2\frac{\partial^5}{\partial x^3 \partial y^2} + \frac{\partial^5}{\partial x \partial y^4} \right) \tag{A6}$$

$$\mathscr{A}_{02} = (h + \hat{z}) \left(\frac{\partial}{\partial y} \right) - \frac{1}{9}(h + \hat{z})^3 \left(\frac{\partial^3}{\partial x^2 \partial y} + \frac{\partial^3}{\partial y^3} \right) + \frac{1}{945}(h + \hat{z})^5 \left(\frac{\partial^5}{\partial x^4 \partial y} + 2\frac{\partial^5}{\partial x^2 \partial y^3} + \frac{\partial^5}{\partial y^5} \right) \tag{A7}$$

$$\mathscr{B}_0 = 1 - \frac{4}{9}(h + \hat{z})^2 \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) + \frac{1}{63}(h + \hat{z})^4 \left(\frac{\partial^4}{\partial x^4} + 2\frac{\partial^4}{\partial x^2 \partial y^2} + \frac{\partial^4}{\partial y^4} \right) \tag{A8}$$

The bottom slope operators (in the x-direction) are

$$\mathscr{C}_{11} = 1 - c_2(h + \hat{z})^2 \left(\frac{\partial^2}{\partial x^2} \right) + c_4(h + \hat{z})^4 \left(\frac{\partial^4}{\partial x^4} + \frac{\partial^4}{\partial x^2 \partial y^2} \right) \tag{A9}$$

$$\mathscr{C}_{12} = -c_2(h + \hat{z})^2 \left(\frac{\partial^2}{\partial x \partial y} \right) + c_4(h + \hat{z})^4 \left(\frac{\partial^4}{\partial x^3 \partial y} + \frac{\partial^4}{\partial x \partial y^3} \right) \tag{A10}$$

$$\mathscr{C}_{13} = (h + \hat{z}) \left(\frac{\partial}{\partial x} \right) - s_3(h + \hat{z})^3 \left(\frac{\partial^3}{\partial x^3} + \frac{\partial^3}{\partial x \partial y^2} \right) + s_5(h + \hat{z})^5 \left(\frac{\partial^5}{\partial x^5} + 2\frac{\partial^5}{\partial x^3 \partial y^2} + \frac{\partial^5}{\partial x \partial y^4} \right) \tag{A11}$$

and in the y-direction

$$\mathscr{C}_{21} = \mathscr{C}_{12} \tag{A12}$$

$$\mathscr{C}_{22} = 1 - c_2(h+\hat{z})^2 \left(\frac{\partial^2}{\partial y^2}\right) + c_4(h+\hat{z})^4 \left(\frac{\partial^4}{\partial x^2 \partial y^2} + \frac{\partial^4}{\partial y^4}\right) \tag{A13}$$

$$\mathscr{C}_{23} = (h+\hat{z}) \left(\frac{\partial}{\partial y}\right) - s_3(h+\hat{z})^3 \left(\frac{\partial^3}{\partial x^2 \partial y} + \frac{\partial^3}{\partial y^3}\right) + s_5(h+\hat{z})^5 \times \left(\frac{\partial^5}{\partial x^4 \partial y} + 2\frac{\partial^5}{\partial x^2 \partial y^3} + \frac{\partial^5}{\partial y^5}\right) \tag{A14}$$

The additional operator used in the irrotational system (11) is

$$\mathscr{C}_1 = 1 - c_2(h+\hat{z})^2 \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right) + c_4(h+\hat{z})^4 \left(\frac{\partial^4}{\partial x^4} + 2\frac{\partial^4}{\partial x^2 \partial y^2} + \frac{\partial^4}{\partial y^4}\right) \tag{A15}$$

It is also noted that an equivalent expression for $\tilde{w}$ from (6) applied at $z = \eta$ is

$$\tilde{w} = \mathscr{A}_1 \hat{w}^* - \mathscr{B}_{11} \hat{u}^* - \mathscr{B}_{12} \hat{v}^* \tag{A16}$$

ACKNOWLEDGEMENTS

The authors would like to thank Prof. L. Nick Trefethen and the Numerical Analysis Group at Oxford University for hosting part of this work. In particular Dr. Daniel Loghin is acknowledged for his help and advice in the Schur complement approach. We thank Prof. Iain Duff for recommending (and supplying) the HSL routine MA41, as well as for useful comments on an earlier manuscript. Prof. Per A. Madsen is also acknowledged for his numerous contributions throughout the work. Finally, we thank V.A. Barker, O. Axelsson, P.G. Thomsen, H.B. Nielsen, and H. van der Vorst for helpful discussions along the way. This work was financially supported by the Danish Technical Research Council (STVF grant no. 9801635). Their support is greatly appreciated.

REFERENCES

1. Boussinesq J. Theorie des ondes et des remous qui se propagent le long s'un canal rectangulaire horizontal, en communiquant au liquide contenu dans ce canal des vitesses sensiblement pareilles de la surface au fond. *Journal de Mathematiques Pures et Appliquees* 1872; **17**:55–108.
2. Madsen PA, Agnon Y. Accuracy and convergence of velocity formulations for water waves in the framework of Boussinesq theory. *Journal of Fluid Mechanics* 2003; **477**:285–319.
3. Madsen PA, Schäffer HA. Higher order Boussinesq-type equations for surface gravity waves—derivation and analysis. *Philosophical Transactions of the Royal Society of London, Series A* 1998; **356**:1–59.
4. Gobbi MF, Kirby JT, Wei G. A fully non-linear Boussinesq model for surface waves. Part 2. Extension to $O(kh)^4$. *Journal of Fluid Mechanics* 2000; **405**:181–210.
5. 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.
6. Madsen PA, Bingham HB, Schäffer HA. Boussinesq formulations for fully non-linear and extremely dispersive water waves. *Philosophical Transactions of the Royal Society of London Series A* 2003; **459**:1075–1104.
7. Zakharov VE. Stability of periodic waves of finite amplitude on the surface of a deep fluid. *Journal of Applied Mechanics and Technical Physics* 1968; **9**:190–194.
8. Madsen PA, Schäffer HA. A review of Boussinesq-type equations for gravity waves. In *Advances in Coastal and Ocean Engineering*, Liu P (ed). vol. 5. World Scientific: 1999; 1–95.
9. Iserles A. *A First Course in the Numerical Analysis of Differential Equations*. Cambridge University Press: Cambridge, 1996.

10. Amestoy PR, Davis TA, Duff IS. An approximate minimum degree ordering algorithm. *SIAM Journal on Mathematical Analysis and Applications* 1996; **17**:886–905.
11. Whaley RC, Petitet A, Dongarra JJ. Automated empirical optimization of software and the ATLAS project. 2000. Available: http://math-atlas.sourceforge.net/
12. Duff IS, Reid JK. The multifrontal solution of unsymmetric sets of linear systems. *SIAM Journal on Scientific Computing* 1984; **5**:633–641.
13. Duff IS. Parallel implementation of multifrontal schemes. *Parallel Computing* 1986; **3**:193–204.
14. Duff IS. Multiprocessing a sparse matrix code on the Alliant FX/8. *Journal of Computational and Applied Mathematics* 1989; **27**:229–239.
15. Saad Y, Schultz MH. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific Computing* 1986; **7**:856–869.
16. Saad Y, van der Vorst HA. Iterative solution of linear systems in the 20th century. *Journal of Computational and Applied Mathematics* 2000; **123**:1–33.
17. Benzi M. Preconditioning techniques for large linear systems: A survey. *Journal of Computational Physics* 2002; **182**:418–477.
18. Longuet-Higgins MS. Integral properties of periodic gravity waves of finite amplitude. *Philosophical Transactions of the Royal Society of London, Series A* 1975; **342**:157–174.
19. Williams JM. *Tables of Progressive Gravity Waves*. Pitman: London, 1985.
20. Saad Y. ILUT: a dual threshold incomplete ILU factorization. *Numerical Linear Algebra with Applications* 1994; **1**:387–402.
21. Saad Y. SPARSKIT: A basic tool kit for sparse matrix computations, Version 2. 1994. Available: www.cs.umn.edu/research/arpa/SPARSKIT/sparsekit.html
22. Madsen PA, Wang BL. A high order Boussinesq method extended to rapidly varying topography. Submitted for publication.
23. Murphy MF, Golub GH, Wathen AJ. A note on preconditioning for indefinite linear systems. *SIAM Journal on Scientific Computing* 2000; **6**:1969–1972.
24. Ipsen ICF. A note on preconditioning nonsymmetric matrices. *SIAM Journal on Scientific Computing* 2001; **23**:1050–1051.
25. Loghin D, Wathen AJ. Schur complement preconditioning for elliptic systems of partial differential equations. *Numerical Linear Algebra with Applications* 2003; **10**:423–443.
26. Loghin D, Wathen AJ. Schur complement preconditioners for the Navier–Stokes equations. *International Journal for Numerical Methods in Fluids* 2002; **40**:403–412.
27. Trefethen LN, Bau III D. *Numerical Linear Algebra*. SIAM: Philadelphia, 1997.
28. Whalin RW. The limit of applicability of linear wave refraction theory in a convergence zone. Res. Rep. H-71, U.S. Army Corps of Engineers, Waterways Expt. Station, Vicksburg, MS, 1971.
29. 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.
30. Fenton JD. The numerical solution of steady water wave problems. *Computers & Geosciences* 1988; **14**: 357–68.
31. Savitzky A, Golay MJE. *Analytical Chemistry* 1964; **36**:1627–1639.
32. Press WH, Teukolsky SA, Vetterling WT, Flannery BP. *Numerical Recipes in FORTRAN: The Art of Scientific Computing* (2nd edn). Cambridge University Press: Cambridge, 1992.

