A COMPUTER PROGRAM FOR TRANSIENT WAVE RUN-UP

by

STÉPHAN GRILLI

RESEARCH REPORT NO. CACR-93-02

June, 1993

CENTER FOR APPLIED COASTAL RESEARCH

Department of Civil Engineering
University of Delaware
Newark, Delaware 19716

A COMPUTER PROGRAM FOR TRANSIENT WAVE RUN-UP

by

S. Grilli and R. Subramanya

Department of Ocean Engineering, University of Rhode Island,

Kingston 02881, RI

RESEARCH REPORT NO. CACR-93-02
June 1993

Abstract

In this study, a computer software for two-dimensional wave propagation and runup over arbitrary bottom topography, was documented, and made available for use in the public domain. This software was developed over the past five years by Grilli *et al.* [31, 37], based on fully nonlinear potential flow equations, and although well validated as a research tool, it was not available in a form allowing easy use by others (see, e.g., [35, 36, 38]).

After an introduction covering theoretical aspects of the problem, numerical algorithms used in the solution are presented (Boundary Element Method, time updating), as well as detailed flowcharts for the software. A user's manual is finally provided, giving step by step instructions on how to use the software, along with a few typical applications of the program, that can be used for training and verification.

The source code for the software (FORTRAN 77) has also been documented as part of this project. Each of the 64 subroutines and functions in the source code has been given an internal description of its task and main variables, and each routine has also been listed and summarized in the user's manual. The software source code, too long for being attached to this report, is available on request through internet, as well as data and results for the few example presented in the report. Please send Email with inquiry to : grilli@mistral.oce.uri.edu.

Studies presented in this report have been supported by a grant from National Science Foundation's : Sitting and Geotechnical Systems, Division of Biological and Critical Systems (NSF award nb. 9111827).

Contents

1 Introduction

1.1 Background of the project

A numerical model for fully nonlinear water waves was developed and validated over the past five years, and used as a research tool for calculating various wave propagation and wave-structure interaction problems in coastal areas (Grilli [24,25]; Grilli *et al.* [29,40]; Grilli, Losada & Martin [26,27,28]; Grilli, Skourup & Svendsen [30,31]; Grilli & Svendsen [33,34,35,36,37,38]; Otta *et al.* [56]; Skourup *et al.* [65,66,67]; Grilli & Subramanya [32]; Svendsen & Grilli [72]).

This model solves unsteady two-dimensional potential flow equations in domains of arbitrary geometry, and can be used to calculate wave shoaling and runup on slopes, and wave interaction with coastal structures. Because of its Eulerian-Lagrangian description of the free surface, the model is also capable of modeling wave overturning over slopes and structures, up to the instant the tip of the breaking wave hits the free surface.

Waves are generated in the model, by imposing waves directly on the free surface, by simulating a piston wavemaker motion, as in laboratory experiments, or by using a line of internal sources. Except for the assumption of potential flow, no further approximation is made in the model, unlike in most wave theories. The model, therefore, is not restricted to special types of waves (e.g., short, long, periodic, non-periodic,...), and can be used for arbitrary incident wave conditions. Submerged or emerged structures of arbitrary shape, like obstacles on the bottom or breakwaters, can be introduced in the model, as well as gentle or steep bottom slopes.

Although well validated as a research tool, this computer model was not available in a form allowing easy use or modification by others. The purpose of the present project has been to document and maintain this computer software, in order to make it available for other researchers and scientists.

The present text constitutes the final report for this research project, along with the documented source code for the computer software, that is available on request.

1.2 Project achievements

The main task in this project was to prepare a user's manual detailing the theory underlying the computer model, the organization of the computer software (flowcharts), the algorithms implemented for solving wave propagation and runup problems (Boundary Element Method, time updating, wave generation,...), and the input data required for running the software.

Detailed technical achievements for the project are as follows :

i) Theory and numerical algorithms for the mathematical and numerical model are described in sections 2,3, and 4 of the report.

Assumptions, accuracy and limitations of computations are clearly stated. Many checks of data and results, and error messages have been implemented at various stages of computations, to stop the program in case of computational errors or inaccuracies, and to inform users on why computations have been interrupted.

ii) The computer software is presented in section 5, along with instructions on how to use it.

Each of the 64 subroutines and functions in the source code were internally documented, and are summarized in the report. Flow chart diagrams, and general descriptions of sequences of functions corresponding to subroutines in the program are also included.

Parts of the computer program that were not yet written in "structured programming" have been reorganized and well commented. Descriptions of tasks, algorithms, input and output variables, called subroutines or functions, have been added as a header to each subroutine source code, following a standard format. This will make it possible for other researchers to easily understand the structure of the program, and to make their own modifications as required by their research project.

Input data for using the software are detailed in the user's manual, and typical input/output listing for a few typical problems are given in section 6. Inputs have been

reorganized to make it easier generating data for typical problems.

iii) A few sets of complete calculations have been prepared for typical applications of the program (long-wave and tsunami runup), and are presented in section 6 of this report.

A clear description and discussion of input/output data for these cases is given, and references to related specific publications is made, for further detail or information on the physical meaning of these applications.

Computer files, with complete source code (about 10,000 lines in FORTRAN 77 code), user's manual, and examples, for this software—too voluminous to be included in this report— are available on request through the internet computer network (send Email to : grilli@mistral.oce.uri.edu, for inquiry).

1.3 Modeling of highly nonlinear waves

Over the past fifteen years, accurate numerical methods have been developed for calculating propagation of two-dimensional (2D) space-periodic waves in deep water and over constant depth, up to initiation of breaking (Longuet-Higgins & Cokelet [52] 1976, Vinje & Brevig [76] 1981, New *et al.* [55] 1985, and Dold & Peregrine [14] 1986). These methods are based on potential flow equations, with full nonlinearity included in the free surface boundary conditions, and use a representation of the flow that allows for multi-valued free surface elevations appearing during breaking (i.e., a Lagrangian representation; see Fig. 1).

Propagation, shoaling, and runup of 2D waves over a slope have also been the object of numerous theoretical and numerical studies over the past thirty years, particularly for the case of long waves or swells (Carrier & Greenspan [6] 1958, Carrier [5] 1966, Camfield & Street [4] 1969, Hibberd & Peregrine [42] 1979, Kobayashi *et al.* [48] 1989, and Synolakis [73] 1990, using Linear or Nonlinear Shallow Water equations; Peregrine [59] 1967, Pedersen & Gjevik [58] 1983, Freilich & Guza [19] 1984, Liu *et al.* [51] 1985, Zelt & Raichlen [77] 1990, and Kirby [46] 1991,

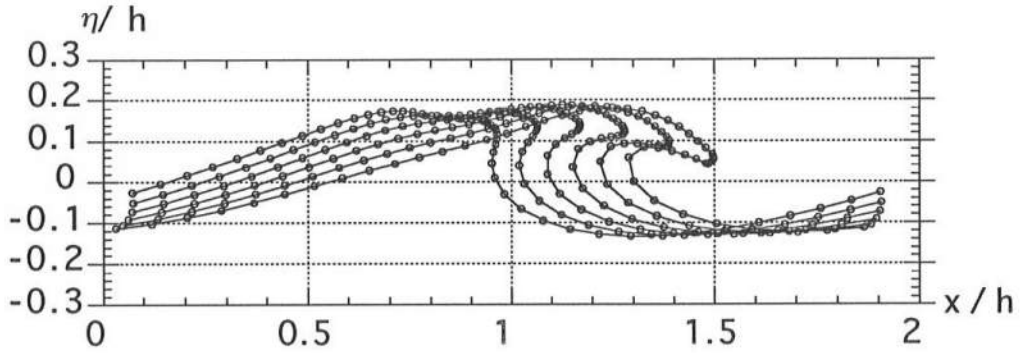


Figure 1: Instability by plunging breaking of a large periodic sine wave over constant depth h, as computed with the model by Grilli *et al.* [29] 1989. Initial wave height $\frac{H}{h} = 0.333$, length $\frac{L}{h} = 1.85$, and period $T\sqrt{\frac{g}{L}} = 2.50$. A periodicity condition is used in the model on lateral boundaries, to create a situation similar to that examined by Longuet-Higgins & Cokelet [49]. Symbols (o) denote BEM discretization nodes, identical to individual fluid particles whose motion is calculated in time.

using Boussinesq or parabolic approximations of Boussinesq equations). The reader can find details about various wave theories, and some of the above listed studies, in Mei [50] 1983, and Dean & Dalrymple [12] 1984. Most of the existing methods, however, are based on first- or low-order theories, whose assumptions—for instance, small amplitude mildly nonlinear waves, or mild bottom slope—may no longer be valid for waves that, due to shoaling, may be close to breaking (i.e., highly nonlinear), before they run up the slope

For predicting characteristics of shoaling and impending breaking waves on slopes (shoaling coefficients, profile, and kinematics), the state-of-the-art method has been to using the higher-order expansion methods, originally developed for waves of permanent form over constant depth (Stiassine & Peregrine [67] 1980, Peregrine [57] 1983, Sobey & Bando [66] 1991). These methods, however, by nature, cannot include effects of bottom slope or change of wave form during shoaling. In particular, shoaling waves may become strongly asymmetric when approaching breaking (e.g., Skjelbreia [61] 1987), an effect that is not included in the above approaches. Griffiths *et al.* [21] 1992, recently compared measurements of internal kinematics

of periodic waves shoaling up a 1:30 slope, with predictions of the 5th-order Stokes theory, of the 9th- and higher-order streamfunction theory, and of the full nonlinear model by New *et al.* [52] (see [12] for definitions of these wave theories). They found, horizontal velocities were correctly predicted by most theories below still water level. In the high crest region, low-order theories underpredicted velocities by as much as 50%, whereas predictions of the full nonlinear theory were quite good up to the crest. These comparisons, however, were only done for a mild slope (i.e, with limited bottom effect), and for cases in which breaking occurred by spilling. The authors pointed out "all theories are grossly in error when compared to severe plunging breakers".

Other fully nonlinear wave studies will be mentioned for completeness, that have either inherently been limited to non-breaking waves (Fenton & Rienecker [16] 1982, Kim *et al* [43] 1983, Nakayama [51] 1983), or have represented extensions (e.g., to axisymmetric or three-dimensional problems), or variant of existing methods— mostly by [49] and [13]—(Isaacson [41] 1982, Jansen [42] 1986, Dommermuth & Yue [14] 1987, Gravert [19] 1987, Greenhow [20] 1987, Tanaka *et al.* [72] 1987, Klopman [45] 1988, Cooker [10] 1990, Cointe [7] 1990, Romate [59] 1990, Seo & Dalrymple [60] 1990).

The 2D potential flow model developed by Grilli *et al.* [28, 29, 35] follows the strategy of deep water and constant depth nonlinear wave models mentioned above (e.g., Dold & Peregrine [13]). It is based on a mixed Eulerian-Lagrangian representation and includes full nonlinear terms in the free surface boundary conditions. Unlike most other approaches, however, this model works in the physical space and is valid for arbitrary bottom topography and incident wave conditions. It is therefore applicable to shallow water wave shoaling and breaking, and to wave runup over arbitrary slopes, without any approximation on the wave shape, or on the free surface boundary conditions. Development and verification of this model have been carried out under a 2D formulation. All elements in the model, however, were selected to allow implementation of a three-dimensional model, as a direct extension of the 2D formulation. This is unlike most other 2D models based on complex variable

formulations.

Detailed equations and numerical procedures for this wave model are presented in sections 2,3, and 4. Applications of the model to cases of wave propagation in shallow water and wave runup on slopes are presented in section 6.

2 Mathematical model

Equations for the two-dimensional potential model by Grilli *et al.* [31, 37], and its most recent extensions, are presented in the next subsections. Full nonlinearity is maintained in the free surface boundary conditions, and time integration of these conditions is based on higher-order Taylor expansions, for both the free surface position and the potential. Laplace's equation is solved using a higher-order Boundary Element Method (Brebbia [2] 1978). No-flow boundary conditions are prescribed along solid boundaries of the domain (bottom, coastal structures), and arbitrary waves are generated in the model, either by specifying an initial wave on the free surface, by simulating a wavemaker at the open-sea boundary of the computational domain (as in laboratory experiments), or by uding a line of internal sources.

2.1 Governing equations and solid boundary conditions

The velocity potential $\phi(\mathbf{x}, t)$ is used to describe inviscid irrotational 2D flows in the vertical (x, z) plane, where the velocity is given by $\mathbf{u} = \nabla\phi = (u, w)$. The continuity equation in the fluid domain $\Omega(t)$ with boundary $\Gamma(t)$ is a Laplace equation for the potential (see Fig. 2 for definitions),

$$\nabla^2\phi = 0 \qquad \text{in } \Omega(t) \tag{1}$$

On the free surface $\Gamma_f(t)$, ϕ satisfies nonlinear kinematic and dynamic boundary conditions,

$$\frac{D\mathbf{r}}{Dt} = \mathbf{u} = \nabla\phi \qquad \text{on } \Gamma_f(t) \tag{2}$$

$$\frac{D\phi}{Dt} = -gz + \frac{1}{2}\nabla\phi\cdot\nabla\phi - \frac{p_a}{\rho} \qquad \text{on } \Gamma_f(t) \tag{3}$$

respectively, with $\mathbf{r}$ the position vector of a free surface fluid particle, g the acceleration due to gravity, z the vertical coordinate (positive upwards, and $z = 0$ at the undisturbed free

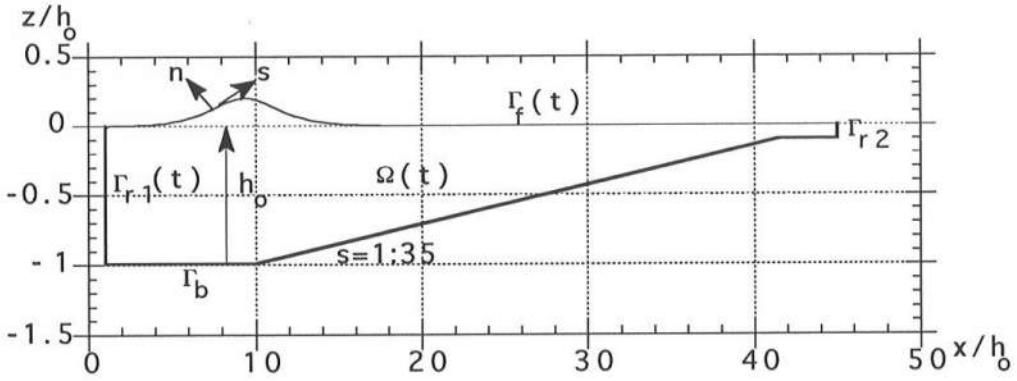


Figure 2: Typical computational domain for wave shoaling and runup on a slope, with definition of various boundaries. The domain is sketched with a slope s =1:35, terminated by a shelf of depth $h_1 = 0.1h_o$ at its upper part, and the free surface profile corresponds to a solitary wave, of initial height $\frac{H_o}{h_o} = 0.2$ generated by a piston wavemaker at boundary $\Gamma_{r1}(t)$.

surface), p_a the atmospheric pressure, and ρ the fluid density. The material derivative is defined as,

$$\frac{D}{Dt} \equiv \frac{\partial}{\partial t} + \mathbf{u} \cdot \nabla \tag{4}$$

Along the stationary bottom Γ_b, and other fixed boundaries Γ_{r2}, a no-flow condition is prescribed as,

$$\nabla\phi \cdot \mathbf{n} \equiv \overline{\frac{\partial\phi}{\partial n}} = 0 \qquad \text{on } \Gamma_b \text{ and } \Gamma_{r2} \tag{5}$$

in which $\mathbf{n}$ is the unit outward normal vector.

2.2 Boundary conditions for wave generation

Waves are generated in the model by either prescribing a wavemaker motion on the "open sea" boundary $\Gamma_{r1}(t)$ of the computational domain, by prescribing the elevation and potential

on the free surface, of a known "exact" wave solution of flow equations, or by using an internal line of sources.

General boundary conditions for these three types of wave generation are given in the following. Generation of specific waves is detailed in section 3.

2.2.1 Exact wave solution

"Numerically exact" permanent form solutions of the nonlinear "Wave Boundary Value Problem" over constant depth (WBVP, (1)-(5); i.e., solitary or streamfunction waves), are generated by specifying their potential $\phi(x, t_o)$, and elevation $\eta(x, t_o)$, on the free surface $\Gamma_f(t_o)$, at initial time t_o. In this case, normal velocity is also prescribed to $U(t)$ over the fixed vertical lateral boundaries Γ_{r1}, Γ_{r2}. We get,

$$\begin{aligned} &\overline{\phi} = \phi(x, t_o) \ , \ \overline{z} = \eta(x, t_o) && \text{on } \Gamma_f(t_o) \\ &\overline{\frac{\partial \phi}{\partial n}} = U(t) && \text{on } \Gamma_{r1}, \Gamma_{r2} \end{aligned} \tag{6}$$

in which overbars denote prescribed values.

2.2.2 Plane wavemaker

A plane wavemaker motion $\overline{x} = x_p(z, t)$ is specified on the moving boundary $\Gamma_{r1}(t)$, to generate waves as in laboratory experiments. Paddle motion and normal velocity are specified over the surface of the paddle, as,

$$\overline{\frac{\partial \phi}{\partial n}} = \mathbf{u}_p \cdot \mathbf{n} = \frac{\frac{\partial x_p}{\partial t}}{\sqrt{1 + (\frac{\partial x_p}{\partial z})^2}} \qquad \text{on } \Gamma_{r1}(t) \tag{7}$$

in which the right hand side represents the normal paddle velocity. Equation (7) will be developed in section 3, for the case of piston or flap wavemakers.

2.2.3 Internal sources

The traditional way of generating waves by specifying a velocity distribution or movement along a part of the boundary has the disadvantage that this boundary also reflects waves propagating towards the boundary, from inside the computational domain (such as the scattered wave field from a structure). This is a major problem in any physical model.

In a computational model, this can be avoided to a large degree by generating waves by internal sources (an idea first suggested by Brorsen & Larsen [3] (1987) for a linear model). If oscillating sources are distributed along a vertical, say, line placed a short distance inside the fluid domain, waves will be generated and will propagate away from the sources in both directions. The waves moving into the computational domain are the ones we are interested in. On the other hand waves scattered from structures inside the computational domain will essentially pass through the sourceline . Those scattered waves, along with waves generated away from the domain, should be leaving the domain through its open sea boundary. Hence, a radiation condition should be specified. This case will not be detailed here (see Otta *et al.* [57] 1992 for detail).

When sources (or sinks which are negative sources) are introduced in the fluid domain, Laplace's equation (1) becomes the Poisson equation,

$$\nabla^2\phi = b(\mathbf{x},t) \qquad \text{in } \Omega(t) \tag{8}$$

where $b(\mathbf{x},t)$ is the density of a known distribution of sources inside the domain $\Omega(t)$.

Values of $b(\mathbf{x},t)$ will be discussed in section 3, in the case of the generation of specific waves in the model.

2.3 The time integration

Free surface boundary conditions (2) and (3) are integrated at time t, to establish both the new position and the relevant boundary conditions on the free surface, at a subsequent time $t+\Delta t$ (with Δt being a small time step). This is done, following the approach introduced by Dold

& Peregrine [14], using Taylor expansions for both the position $\mathbf{r}(t)$ and the potential $\phi(\mathbf{r}(t))$ on $\Gamma_f(t)$. Series, truncated to Nth-order, are expressed in terms of the material derivative (4), and of time step Δt, as,

$$\overline{\mathbf{r}}(t+\Delta t) = \mathbf{r}(t) + \sum_{k=1}^{N} \frac{(\Delta t)^k}{k!} \frac{D^k \mathbf{r}(t)}{Dt^k} + \mathcal{O}[(\Delta t)^{N+1}] \tag{9}$$

for the free surface position, and,

$$\overline{\phi}(\mathbf{r}(t+\Delta t)) = \phi(\mathbf{r}(t)) + \sum_{k=1}^{N} \frac{(\Delta t)^k}{k!} \frac{D^k \phi(\mathbf{r}(t))}{Dt^k} + \mathcal{O}[(\Delta t)^{N+1}] \tag{10}$$

for the potential. The last terms in (9) and (10) represent truncation errors. The time updating of the free surface geometry described by (9) actually corresponds to following the motion of fluid particles in time. This procedure is often referred to as a "Mixed Eulerian-Lagrangian" formulation.

Second-order series are used in the present case ($N = 2$). Higher-order Taylor series, however, have successfully been used by others, to provide highly accurate solutions for periodic problems (Dold & Peregrine [14] (N=3), and Seo & Dalrymple [63] 1990 (N=4)). First-order coefficients in (9) and (10) are obtained, based on equations (2) and (3), using ϕ and $\frac{\partial \phi}{\partial n}$, as provided by the solution of Laplace's equation (1) at time t. Second-order coefficients are expressed as $\frac{D}{Dt}$ of (2) and (3), and are calculated using the solution of a second Laplace problem of the form (1), for ($\frac{\partial \phi}{\partial t}$, $\frac{\partial^2 \phi}{\partial t \partial n}$). This is because all time derivatives of the potential satisfy Laplace's equation (1). Higher-order series would simply require that more Laplace's equations are solved for higher-order time derivatives of ϕ.

No-flow boundary conditions for a second Laplace problem for $\frac{\partial \phi}{\partial t}$ are readily obtained along solid boundaries, as,

$$\overline{\frac{\partial^2 \phi}{\partial t \partial n}} = 0 \qquad \text{on } \Gamma_b \text{ and } \Gamma_{r2} \tag{11}$$

The boundary condition at the free surface is obtained from equation (3) and (4) as,

$$\overline{\frac{\partial \phi}{\partial t}} = -\frac{1}{2} \nabla \phi \cdot \nabla \phi - \frac{p_a}{\rho} - gz \qquad \text{on } \Gamma_f(t) \tag{12}$$

Hence, $\overline{\frac{\partial \phi}{\partial t}}$ can be specified on the free surface as a function of known geometry and potential at time t.

When $\Gamma_{r1}(t)$ represents a wavemaker boundary moving at velocity $\mathbf{u}_p(\mathbf{x}_p(t), t)$, we have by (7),

$$\overline{\frac{\partial^2 \phi}{\partial t \partial n}} = \frac{\partial}{\partial t}(\mathbf{u}_p \cdot \mathbf{n})$$

or,

$$\overline{\frac{\partial^2 \phi}{\partial t \partial n}} = [\frac{d(\mathbf{u}_p \cdot \mathbf{n})}{dt} - \mathbf{u}_p \cdot \nabla(\mathbf{u}_p \cdot \mathbf{n})] \qquad \text{on } \Gamma_{r1}(t) \tag{13}$$

in which, $\frac{d}{dt} = \frac{\partial}{\partial t} + \mathbf{u}_p \cdot \nabla$, represents the time derivative following the motion of the boundary $\mathbf{x}_p(t)$. This boundary condition is developed in section 3.

When waves are generated by a line of internal sources, the time derivative of the source strength $\frac{\partial b}{\partial t}(\mathbf{x}, t)$ is introduced in a Poisson equation of the form (8), for $\frac{\partial \phi}{\partial t}$.

2.4 Expressions of Taylor series coefficients

Detailed expressions of coefficients of Taylor series (9) and (10) are derived in the following, using a curvilinear coordinate system $(\mathbf{s}, \mathbf{n})$ on the boundary (Fig. 2).

The kinematic free surface boundary condition (2) provides the first-order coefficient in the series (9), for the free surface position vector $\mathbf{r}$, as,

$$\frac{D\mathbf{r}}{Dt} = \frac{\partial \phi}{\partial s}\mathbf{s} + \frac{\partial \phi}{\partial n}\mathbf{n} \tag{14}$$

Applying the material derivative (4) to equation (2), we get the general expression of the second-order coefficient in (9) as,

$$\frac{D^2\mathbf{r}}{Dt^2} = \frac{D\mathbf{u}}{Dt} = \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} \tag{15}$$

By definition, the first term on the right hand side of (15) is,

$$\frac{\partial \mathbf{u}}{\partial t} = \nabla \frac{\partial \phi}{\partial t} = \frac{\partial^2 \phi}{\partial t \partial s}\mathbf{s} + \frac{\partial^2 \phi}{\partial t \partial n}\mathbf{n} \tag{16}$$

where the curvilinear gradient operator,

$$\nabla \equiv \frac{1}{h_s}\frac{\partial}{\partial s}\mathbf{s} + \frac{\partial}{\partial n}\mathbf{n} \tag{17}$$

is used, with definitions,

$$\mathbf{s} = [\cos\beta\ ,\ \sin\beta]\ ,\ \mathbf{n} = [-\sin\beta\ ,\ \cos\beta] \tag{18}$$

and,

$$\cos\beta = \frac{\partial x}{\partial s}\ ,\ \sin\beta = \frac{\partial z}{\partial s} \tag{19}$$

where β denotes the angle between the horizontal axis x and the tangent $\mathbf{s}$ at the free surface. Derivatives of (**s**,**n**) with respect to their directions are obtained from (18) as,

$$\frac{\partial \mathbf{s}}{\partial s} = \frac{\partial \beta}{\partial s}\mathbf{n}\ ,\ \frac{\partial \mathbf{n}}{\partial s} = -\frac{\partial \beta}{\partial s}\mathbf{s} \tag{20}$$

$$\frac{\partial \mathbf{s}}{\partial n} = \frac{\partial \beta}{\partial n}\mathbf{n}\ ,\ \frac{\partial \mathbf{n}}{\partial n} = -\frac{\partial \beta}{\partial n}\mathbf{s} \tag{21}$$

Now, in a family of curves, n = cst, and of straight lines, s = cst, along the free surface, derivative $\frac{\partial \beta}{\partial n}$ vanishes in (21), and the scale factor h_s, associated with curves n = cst, is defined along the free surface as,

$$-\frac{1}{h_s}\frac{\partial h_s}{\partial n} = \frac{1}{R} = \frac{\partial \beta}{\partial s} \quad \text{with,} \quad h_s = 1 \tag{22}$$

where $R(\mathbf{x})$ is the radius of curvature of the free surface. Thus, h_s is independent of s and only depends on n. Using the above definitions, the second term on the right hand side of (15) becomes,

$$\mathbf{u}\cdot\nabla\mathbf{u} = \nabla\phi\cdot[\frac{1}{h_s}\frac{\partial \nabla\phi}{\partial s}\mathbf{s} + \frac{\partial \nabla\phi}{\partial n}\mathbf{n}]$$

which, using orthogonality of **s** and **n**, can be expressed as,

$$\mathbf{u}\cdot\nabla\mathbf{u} = \frac{1}{h_s^2}\frac{\partial\phi}{\partial s}\frac{\partial\nabla\phi}{\partial s} + \frac{\partial\phi}{\partial n}\frac{\partial\nabla\phi}{\partial n}$$

or,

$$\begin{aligned}\mathbf{u}\cdot\nabla\mathbf{u} &= \frac{1}{h_s^2}\frac{\partial\phi}{\partial s}[\frac{1}{h_s}(\frac{\partial^2\phi}{\partial s^2}\mathbf{s}+\frac{\partial\phi}{\partial s}\frac{\partial\mathbf{s}}{\partial s})+\frac{\partial^2\phi}{\partial s\partial n}\mathbf{n}+\frac{\partial\phi}{\partial n}\frac{\partial\mathbf{n}}{\partial s}]\\ &+\frac{\partial\phi}{\partial n}[\frac{1}{h_s}\mathbf{s}(\frac{\partial^2\phi}{\partial n\partial s}-\frac{1}{h_s}\frac{\partial h_s}{\partial n}\frac{\partial\phi}{\partial s})+\frac{\partial^2\phi}{\partial n^2}\mathbf{n}]\end{aligned} \tag{23}$$

in which, $\frac{\partial\beta}{\partial n}=0$ and $\frac{\partial h_s}{\partial s}=0$, were used. Using equations (17)-(22), it can be shown, continuity, $\nabla\cdot\mathbf{u}=0$, and irrotationality, $\nabla\times\mathbf{u}=0$, conditions transform to,

$$\frac{\partial^2\phi}{\partial s^2}+\frac{\partial^2\phi}{\partial n^2}-\frac{\partial\beta}{\partial s}\frac{\partial\phi}{\partial n}=0 \tag{24}$$

$$\frac{\partial^2\phi}{\partial s\partial n}=\frac{\partial^2\phi}{\partial n\partial s} \tag{25}$$

respectively, along the free surface. Using (24) and (25), equation (23) can be expressed as,

$$\begin{aligned}\mathbf{u}\cdot\nabla\mathbf{u} &= \frac{\partial\phi}{\partial s}[(\frac{\partial^2\phi}{\partial s^2}-\frac{\partial\beta}{\partial s}\frac{\partial\phi}{\partial n})\mathbf{s}+(\frac{\partial^2\phi}{\partial n\partial s}+\frac{\partial\beta}{\partial s}\frac{\partial\phi}{\partial s})\mathbf{n}]\\ &+\frac{\partial\phi}{\partial n}[(\frac{\partial^2\phi}{\partial n\partial s}+\frac{\partial\beta}{\partial s}\frac{\partial\phi}{\partial s})\mathbf{s}+(-\frac{\partial^2\phi}{\partial s^2}+\frac{\partial\beta}{\partial s}\frac{\partial\phi}{\partial n})\mathbf{n}]\end{aligned}$$

or,

$$\mathbf{u}\cdot\nabla\mathbf{u}=\{\frac{\partial\phi}{\partial s}\frac{\partial^2\phi}{\partial s^2}+\frac{\partial\phi}{\partial n}\frac{\partial^2\phi}{\partial n\partial s}\}\mathbf{s}+\{\frac{\partial\phi}{\partial s}\frac{\partial^2\phi}{\partial n\partial s}-\frac{\partial\phi}{\partial n}\frac{\partial^2\phi}{\partial s^2}+\frac{\partial\beta}{\partial s}[(\frac{\partial\phi}{\partial s})^2+(\frac{\partial\phi}{\partial n})^2]\}\mathbf{n} \tag{26}$$

Combining (15), (16), and (26), we get the final expression for the second-order coefficient in (9) as,

$$\begin{aligned}\frac{D^2\mathbf{r}}{Dt^2} &= \{\frac{\partial^2\phi}{\partial t\partial s}+\frac{\partial\phi}{\partial s}\frac{\partial^2\phi}{\partial s^2}+\frac{\partial\phi}{\partial n}\frac{\partial^2\phi}{\partial n\partial s}\}\mathbf{s}+\\ &\{\frac{\partial^2\phi}{\partial t\partial n}-\frac{\partial\phi}{\partial n}\frac{\partial^2\phi}{\partial s^2}+\frac{\partial\phi}{\partial s}\frac{\partial^2\phi}{\partial n\partial s}+\frac{\partial\beta}{\partial s}[(\frac{\partial\phi}{\partial s})^2+(\frac{\partial\phi}{\partial n})^2]\}\mathbf{n}\end{aligned} \tag{27}$$

In the same way, the dynamic free surface boundary condition (3) provides the first-order coefficient in the series (10) for the free surface potential ϕ, using (17) and (22), as,

$$\frac{D\phi}{Dt}=-gz+\frac{1}{2}[(\frac{\partial\phi}{\partial s})^2+(\frac{\partial\phi}{\partial n})^2]-\frac{p_a}{\rho} \tag{28}$$

The second-order coefficient in (10) is obtained by material derivation of (3) as,

$$\frac{D^2\phi}{Dt^2} = -g\frac{Dz}{Dt} + \frac{1}{2}\frac{D}{Dt}(\nabla\phi\cdot\nabla\phi) - \frac{D}{Dt}(\frac{p_a}{\rho}) \tag{29}$$

with, by (2), (17) and (18),

$$\frac{Dz}{Dt} = w = \frac{\partial\phi}{\partial n}\cos\beta + \frac{\partial\phi}{\partial s}\sin\beta \tag{30}$$

and, by definition,

$$\frac{1}{2}\frac{D}{Dt}(\nabla\phi\cdot\nabla\phi) = \mathbf{u}\cdot\frac{D\mathbf{u}}{Dt} = \mathbf{u}\cdot\frac{\partial\mathbf{u}}{\partial t} + \mathbf{u}\cdot(\mathbf{u}\cdot\nabla\mathbf{u}) \tag{31}$$

Now, using orthogonality of $\mathbf{s}$ and $\mathbf{n}$, and (2), (14), (16), and (26), we get,

$$\mathbf{u}\cdot\frac{\partial\mathbf{u}}{\partial t} = \frac{\partial\phi}{\partial s}\frac{\partial^2\phi}{\partial t\partial s} + \frac{\partial\phi}{\partial n}\frac{\partial^2\phi}{\partial t\partial n} \tag{32}$$

and,

$$\mathbf{u}\cdot(\mathbf{u}\cdot\nabla\mathbf{u}) = \frac{\partial\phi}{\partial s}[\frac{\partial\phi}{\partial s}\frac{\partial^2\phi}{\partial s^2} + \frac{\partial\phi}{\partial n}\frac{\partial^2\phi}{\partial n\partial s}] + \frac{\partial\phi}{\partial n}[\frac{\partial\phi}{\partial s}\frac{\partial^2\phi}{\partial n\partial s} - \frac{\partial\phi}{\partial n}\frac{\partial^2\phi}{\partial s^2} + \frac{\partial\beta}{\partial s}(\nabla\phi\cdot\nabla\phi)] \tag{33}$$

Finally, by combining (29)-(33) and using (17), we get the final expression for the second-order coefficient in (10) as,

$$\begin{aligned}\frac{D^2\phi}{Dt^2} \;=\; & \frac{\partial\phi}{\partial s}\{\frac{\partial^2\phi}{\partial t\partial s} + \frac{\partial\phi}{\partial s}\frac{\partial^2\phi}{\partial s^2} + \frac{\partial\phi}{\partial n}\frac{\partial^2\phi}{\partial n\partial s}\} + \\ & \frac{\partial\phi}{\partial n}\{\frac{\partial^2\phi}{\partial t\partial n} - \frac{\partial\phi}{\partial n}\frac{\partial^2\phi}{\partial s^2} + \frac{\partial\phi}{\partial s}\frac{\partial^2\phi}{\partial n\partial s} + \frac{\partial\beta}{\partial s}[(\frac{\partial\phi}{\partial s})^2 + (\frac{\partial\phi}{\partial n})^2]\} - \\ & g\{\frac{\partial\phi}{\partial n}\cos\beta + \frac{\partial\phi}{\partial s}\sin\beta\} - \frac{1}{\rho}\frac{Dp_a}{Dt}\end{aligned} \tag{34}$$

where $\frac{Dp_a}{Dt}$ is the total rate of change of the free surface atmospheric pressure in time.

2.5 Discussion of model assumptions and limitations

No approximations beyond potential flow theory have been made in the model. In particular, unlike analytical or numerical expansion wave theories (see, Dean & Dalrymple [13]), no small

parameter, periodicity, or constant shape wave conditions, have been assumed. This makes the model valid from deep to shallow water, and for arbitrary length waves.

The only limitations—inherent to potential flow theory—of this type of model are that bottom friction and flow separation cannot be modelled, and that computations have to be interrupted shortly after breaking of a wave first occurs. These limitations are discussed in the following :

- Long wave theory shows bottom friction should attenuate long waves in shallow water, whereas short waves should be relatively unaffected.

  For solitary waves shoaling over gentle bottom slopes, however, experiments by Camfield & Street [4] (1969) showed, "bottom roughness has no measurable effect". This was later confirmed in other experiments by Grilli *et al.* [39] (1993). The likely reason for this is, bottom friction only becomes significant when wave height is large and this only occurs in a small region over the slope, just before the wave starts breaking.

  For solitary waves running up a steep slope, Svendsen & Grilli [72] (1990) compared their nonlinear computations to experiments and found frictional effects were negligible, In this case, the distance of propagation over steep slopes was likely too small for friction effects to significantly affect the waves.

  Hence, bottom friction is not an important factor when either wave height and/or distance of propagation are small.

- Flow separation over obstacles on the bottom is significant for steep obstacles (like steps or rectangular bars) of large height to depth ratios, and for high waves (Grilli *et al.* [26, 27] 1992).

  Flow separation leads to an energy loss at the obstacle that reduces the wave crest height downstream of the obstacle.

- When a wave starts overturning, a small horizontal jet forms in the highest region of

the wave crest (Fig. 1). The jet curls up on itself and falls towards the free surface. Breaking occurs when the tip of the falling jet impinges on the free surface, leading to a local violation of continuity equation, manifesting itself by strongly unstable numerical results. Hence, computations with the model are in essence limited to prior to the time impact of a wave on the free surface first occurs. Because of potential flow theory hypotheses, however, computationally accurate results may not be physically realistic up to that stage. This is discussed below.

Dommermuth *et al.* [16] (1988) compared wave profiles calculated using a fully nonlinear potential model, to experimental results, for deep water overturning breakers. They concluded that potential theory is valid up to the moment the tip of the breaker hits the free surface (i.e., slightly further in time than in the situation illustrated in Fig. 1).

Skyner *et al.* [68] (1990) confirmed this conclusion, and compared computed and measured velocities inside plunging breakers. The good agreement they found for the velocities further confirmed the validity of potential flow theory.

- For a train of solitary or periodic waves shoaling over a sloping beach, the front wave of the train is also the steepest wave that first breaks in the shallower wate. Hence, the model can be used to calculate detailed shoaling coefficients over the length of the beach, up to the point the front wave breaks (breaker line). In this case, computations are not greatly affected by the limitation of the model to the first breaking wave discussed above (Grilli & Svendsen [36] (1991), Otta *et al.* [56] (1993), Grilli & Subramanya [32] 1993).

  For irregular wave trains or complex bottom geometry, however, breaking may occur almost anywhere in the shoaling region, due to nonlinear interaction between wave components and between waves and bottom geometry. Hence, computations may have to cease, and the above limitation reduces the utility of the model in its present form for these situations.

3 Wave generation in the model

3.1 Exact solitary waves

"Numerically exact" solitary wave solutions of the WBVP in water of constant depth h_o can be obtained using Tanaka's [74] (1986) method— in the following, these solitary waves are referred to as *exact* solitary waves—, and surface elevation and potential for such waves of specified height can directly be prescribed in the model, as in (6) (notice, in all cases, initial waves are being introduced far enough from lateral boundaries, $U(t) = 0$ is assumed).

Dimensionless variables, x', z', t' and c', will be used in the discussion of *exact* (and later on of *first-order*) solitary waves,

$$x' = \frac{x}{h_o} \ , \ z' = \frac{z}{h_o} \ , \ t' = t\sqrt{\frac{g}{h_o}} \ , \ c' = \frac{c}{\sqrt{gh_o}} \tag{35}$$

in which c denotes the (constant) wave celerity. The initial wave height H_o is identical for solitary waves, to the wave elevation above $z = 0$, and we denote, $H' = \frac{H}{h_o}$, the non-dimensional wave height. The wave Froude number is defined as, $F^2 = \frac{c^2}{gh_o}$.

Tanaka's method solves Cauchy's integral theorem in a frame of reference moving with the celerity c. The crest velocity V_c fully defines the wave in this frame, and the dimensionless crest velocity is defined as, $q_c = \frac{V_c}{c}$. The original method by Tanaka was modified by Cooker [9] (1990), so that the wave height H' could be prescribed as a parameter, instead of q_c.

Main steps in the calculation of *exact* solitary waves are as follows (superscripts denote iteration numbers),

- An approximate crest velocity $\tilde{q_c^o}$ is estimated from the specified H', by interpolation in a table of values of (H', q_c) predetermined within the interval (H'=0.833197, q_c=0) for the highest possible wave (like found, e.g., in Tanaka [74] (1986)), to (H'=0,q_c=1) for a flat free surface.

- Free surface velocity is calculated using the original Tanaka's method, with the approximate crest velocity $\tilde{q_c^o}$.

- Wave celerity $\tilde{c}^o$ and Froude number $(\tilde{F}^2)^o$, are calculated using the free surface velocities, and the corresponding wave amplitude $\tilde{H}'^o$ is obtained from Bernoulli's equation as,

$$\tilde{H}'^o = \frac{1}{2}[1 - (\tilde{q}_c{}^2)^o](\tilde{F}^2)^o \tag{36}$$

  Tanaka's method involves an iterative solution of Cauchy's theorem, using the Froude number as the convergence parameter. The convergence criterion selected here is 10^{-10} in relative value of F^2. We found, 70 to 75 iterations were necessary to achieve convergence within this accuracy.

- A better approximation for the crest velocity, $\tilde{q}_c{}^1$, is re-estimated from $(H', \tilde{H}'^o)$, in the table of values (H', q_c).

- And so on, iteratively, until $\Delta H' = |\frac{H' - \tilde{H}'^n}{H'}|$ is found sufficiently small.

  The convergence criterion selected here is $\Delta H' \leq 10^{-5}$. Three to four iterations only are found necessary to achieve convergence within this accuracy.

- When convergence is reached for both F^2 and H', the wave shape is calculated from free surface velocities. Normal velocity $\frac{\partial \phi}{\partial n}(x, t_o)$ is also calculated on the free surface at this stage (for being used as initial data in the first time step of computations with the model), by noting that, for a constant shape wave,

$$\frac{\partial \phi}{\partial n}(x) = F \sin \beta(x) \tag{37}$$

- Wave area above still water level, m, and kinetic and potential energy, (e_k, e_p), are calculated for the resulting solitary wave, based on the following standard integrals $(\rho' = g' = 1)$,

$$m = \rho' \int_{\Gamma_f} z' dx'$$

$$
\begin{aligned}
e_k &= \frac{1}{2}\rho' \int_{\Gamma_f} \phi \frac{\partial \phi}{\partial n} d\Gamma \\
e_p &= \frac{1}{2}\rho' g' \int_{\Gamma_f} z'^2 dx'
\end{aligned}
\tag{38}
$$

- The resulting wave is finally truncated left and right to points at which free surface elevation $\eta' = \varepsilon_z H'$, with $\varepsilon_z \ll 1$, a pre-selected treshold.

The overall method is found to be quite computationally efficient. Convergence on both F^2 and H' is reached, and all the wave data are calculated within less than 0.6s CPU time (IBM-3090/300), using 80 points on the free surface to describe the wave.

3.2 Wave generation by a plane wavemaker

3.2.1 Introduction

A plane wavemaker is simulated on boundary $\Gamma_{r1}(t)$, to generate waves in the model, the same way as in most laboratory experiments. For selected incident waves, the wavemaker motion $\mathbf{x}_p(t)$ and velocity $\mathbf{u}_p(\mathbf{x}_p(t), t)$ are obtained from first-order wave theory (i.e., Boussinesq theory for long waves, and first-order Stokes theory for periodic short waves).

Waves generated this way propagate without change of form in a model based on first-order theory equations. In the present full nonlinear model—or, for this respect, in a laboratory wavetank—, however, such waves are not expected to correspond to permanent form solutions. Goring [20] (1978), for instance, found that solitary waves of small amplitude ($H' < 0.2$) generated by a piston wavemaker in a wave flume, kept their shape constant within a very small margin. For such small waves, the first-order wave profile is quite close to an *exact* solitary wave. For steeper waves ($H' \geq 0.2$), however, Goring found, solitary waves shed a tail of oscillation behind them as they propagated down the flume. Grilli & Svendsen [38] 1990 also observed in their computations with the model, waves of significant height generated by a wavemaker modulate and adjust their shape while propagating.

Since computations solve the full equations, if one assumes friction to be small, the computed wave train should also be expected closely to follow what actually happens in a wave flume after generation of a wave motion by a wavemaker. For long waves, this was in fact confirmed in many comparison of model results with laboratory experiments (Grilli & Svendsen [38] 1990, Grilli *et al.* [39, 26, 27, 28] 1992,1993, Svendsen & Grilli [72] 1990).

3.2.2 General boundary condition

Boundary conditions for $\frac{\partial \phi}{\partial n}$ and $\frac{\partial^2 \phi}{\partial t \partial n}$ can be expressed for any specified wavemaker motion and velocity, based on (7) abd (13). The latter equation, for $\frac{\partial^2 \phi}{\partial t \partial n}$, includes a time derivative with respect to the rigid body motion that needs to be developed with great care. This was done by Cointe [8] (1989), for the motion of a rigid body of arbitrary shape. In the case of a plane rigid body like a wavemaker, Cointe's expression simplifies into,

$$\overline{\frac{\partial^2 \phi}{\partial t \partial n}} = (\ddot{\boldsymbol{\alpha}} \cdot \mathbf{n}) + \dot{\theta}\,[(\dot{\boldsymbol{\alpha}} \cdot \mathbf{s}) - \frac{\partial \phi}{\partial s}] - \frac{\partial^2 \phi}{\partial n \partial s}(\dot{\boldsymbol{\alpha}} \cdot \mathbf{s}) + \frac{\partial^2 \phi}{\partial s^2}(\dot{\boldsymbol{\alpha}} \cdot \mathbf{n}) \tag{39}$$

in which $\boldsymbol{\alpha}$ denotes the position vector of points on the body surface, and θ the angle of rotation around $\mathbf{x}_g$, and dots denote absolute time derivatives with respect to the body motion, $\frac{d}{dt}$, defined as in (13).

Velocity and acceleration of points on the body boundary can be derived for specific cases and used along with (39). If r_g denotes the distance between point, $\boldsymbol{\alpha} = (\alpha, \beta)$, and $\mathbf{x}_g = (x_g, z_g)$, we get,

$$\begin{aligned} \alpha &= x_g + r_g \cos\theta \\ \beta &= z_g + r_g \sin\theta \end{aligned} \tag{40}$$

Since r_g is constant with respect to any rigid body motion, we also have,

$$\begin{aligned} \dot{\alpha} &= \dot{x_g} - r_g \sin\theta\,\dot{\theta} = \dot{x_g} - (\beta - z_g)\,\dot{\theta} \\ \dot{\beta} &= \dot{z_g} + r_g \cos\theta\,\dot{\theta} = \dot{z_g} + (\alpha - x_g)\,\dot{\theta} \end{aligned} \tag{41}$$

And,

$$\begin{aligned}\ddot{\alpha} &= \ddot{x}_g - r_g\cos\theta\,\dot{\theta}^2 - r_g\sin\theta\,\ddot{\theta}\\ \ddot{\beta} &= \ddot{z}_g - r_g\sin\theta\,\dot{\theta}^2 + r_g\cos\theta\,\ddot{\theta}\end{aligned}$$

or,

$$\begin{aligned}\ddot{\alpha} &= \ddot{x}_g - (\alpha - x_g)\,\dot{\theta}^2 - (\beta - z_g)\,\ddot{\theta}\\ \ddot{\beta} &= \ddot{z}_g - (\beta - z_g)\,\dot{\theta}^2 + (\alpha - x_g)\,\ddot{\theta}\end{aligned} \tag{42}$$

Motion and boundary conditions can now be expressed for two standard types of plane wavemakers.

i) Piston wavemaker : This corresponds to a flat vertical plate ($\theta = \frac{\pi}{2}$) moving horizontally in depth h_o, with, $x_p(t)$ and $u_p(x_p(t),t) = \dot{x}_p(t)$, the specified horizontal piston motion (stroke), and velocity, respectively. Along the wavemaker paddle, we have by (40),(41),(42),

$$\mathbf{n} = [-1,0], \qquad \mathbf{s} = [0,1], \qquad \dot{\theta} = \ddot{\theta} = 0$$

$$\boldsymbol{\alpha} = \mathbf{x}_p = [x_p(t), z], \qquad \dot{\boldsymbol{\alpha}} = \mathbf{u}_p = [u_p(t), 0], \qquad \ddot{\boldsymbol{\alpha}} = \dot{\mathbf{u}}_p = [\dot{u}_p(t), 0] \tag{43}$$

Hence,

$$\dot{\boldsymbol{\alpha}}\cdot\mathbf{n} = -u_p, \qquad \ddot{\boldsymbol{\alpha}}\cdot\mathbf{n} = -\dot{u}_p, \qquad \dot{\boldsymbol{\alpha}}\cdot\mathbf{s} = 0$$

and from (7),(39), and (43), boundary conditions on the piston wavemaker boundary read,

$$\begin{aligned}\overline{\frac{\partial\phi}{\partial n}} &= -u_p(t)\\ \overline{\frac{\partial^2\phi}{\partial t\partial n}} &= -\dot{u}_p(t) - u_p(t)\frac{\partial^2\phi}{\partial s^2} \qquad \text{on } \Gamma_{r1}(t)\end{aligned} \tag{44}$$

in which $\frac{\partial^2\phi}{\partial s^2} = \frac{\partial^2\phi}{\partial x^2}$; and $\dot{u}_p = \ddot{x}_p(t)$ denotes the specified wavemaker acceleration.

ii) Flap wavemaker : This corresponds to a flat plate, hinged at $\mathbf{x}_g = (0, -h_o)$ on the bottom, and oscillating with an angle $\theta(t) \in [\frac{\pi}{2}, 0]$ (defined trigonometrically with respect

to the bottom), with, $x_p(t)$ and $u_p(x_p(t),t) = \dot{x}_p(t)$, the specified flap horizontal motion (stroke) and velocity at $z = 0$, respectively. Along the wavemaker paddle, we have by (40),

$$\mathbf{n} = [-\sin\theta(t), \cos\theta(t)], \quad \mathbf{s} = [\cos\theta, \sin\theta], \quad \boldsymbol{\alpha} = \mathbf{x}_g + r_g\mathbf{s} = [\alpha(t), \beta(t)] \tag{45}$$

in which r_g is given by,

$$r_g(t) = \alpha(t)\cos\theta(t) + [\beta(t) + h_o]\sin\theta(t) \tag{46}$$

Now, by (41) and (42), with $\dot{\mathbf{x}}_g = \ddot{\mathbf{x}}_g = 0$, we have,

$$\begin{aligned} \dot{\boldsymbol{\alpha}} &= \mathbf{u}_p(t) = [-\beta(t) - h_o, \alpha(t)]\,\dot{\theta} \\ \ddot{\boldsymbol{\alpha}} &= \dot{\mathbf{u}}_p(t) = [-\beta(t) - h_o, \alpha(t)]\,\ddot{\theta}(t) - [\alpha(t), \beta(t) + h_o]\,\dot{\theta}^2(t) \end{aligned} \tag{47}$$

Hence, by (45),(46),(47),

$$\begin{aligned} \dot{\boldsymbol{\alpha}} \cdot \mathbf{n} &= [\alpha(t)\cos\theta(t) + (\beta(t) + h_o)\sin\theta(t)]\,\dot{\theta}(t) \\ &= r_g(t)\,\dot{\theta}(t) \\ \ddot{\boldsymbol{\alpha}} \cdot \mathbf{n} &= [\alpha(t)\cos\theta(t) + (\beta(t) + h_o)\sin\theta(t)]\,\ddot{\theta}(t) \\ &+ [-(\beta(t) + h_o)\cos\theta(t) + \alpha(t)\sin\theta(t)]\,\dot{\theta}^2(t) \\ &= r_g(t)\ddot{\theta}(t) \\ \dot{\boldsymbol{\alpha}} \cdot \mathbf{s} &= [-(\beta(t) + h_o)\cos\theta(t) + \alpha(t)\sin\theta(t)]\,\dot{\theta}(t) \\ &= 0 \end{aligned} \tag{48}$$

since one can show, by simple geometric considerations, $[-(\beta + h_o)\cos\theta + \alpha\sin\theta] = 0$.

From (7),(39), and (48), boundary conditions on the flap wavemaker boundary read,

$$\begin{aligned} \overline{\frac{\partial\phi}{\partial n}} &= r_g(t)\,\dot{\theta}(t) \\ \overline{\frac{\partial^2\phi}{\partial t\partial n}} &= r_g(t)\,\ddot{\theta}(t) + \dot{\theta}(t)\,[r_g(t)\frac{\partial^2\phi}{\partial s^2} - \frac{\partial\phi}{\partial s}] \end{aligned} \tag{49}$$

Time derivatives of $\theta(t)$ can be expressed as a function of wavemaker stroke $x_p(t)$, and of its time derivatives as,

$$\tan\theta(t) = \frac{h_o}{x_p(t)}$$

or,

$$\begin{aligned}
\theta(t) &= \arctan\frac{h_o}{x_p(t)} \\
\dot{\theta}(t) &= -\frac{\dot{x}_p(t)}{1+(\frac{h_o}{x_p(t)})^2}\frac{h_o}{x_p^2(t)} \\
&= -\frac{\tan^2\theta(t)}{1+\tan^2\theta(t)}\frac{u_p(t)}{h_o} \\
&= -\sin^2\theta(t)\frac{u_p(t)}{h_o} \\
\ddot{\theta}(t) &= -2\sin\theta(t)\cos\theta(t)\frac{u_p(t)}{h_o}\dot{\theta}(t) - \sin^2\theta(t)\frac{\dot{u}_p(t)}{h_o} \\
&= -\sin^2\theta(t)[\frac{\dot{u}_p(t)}{h_o} - \sin 2\theta(t)(\frac{u_p(t)}{h_o})^2] \qquad (50)
\end{aligned}$$

Now, with $R(t) = \frac{h_o}{h_o^2+x_p^2(t)}$ we get,

$$\begin{aligned}
\sin\theta(t) &= R(t)\sqrt{h_o^2+x_p^2(t)} \\
\cos\theta(t) &= x_p(t)\sqrt{h_o^2+x_p^2(t)} \\
\sin 2\theta(t) &= 2R(t)x_p(t) \qquad (51)
\end{aligned}$$

and by (46),(50),(51),

$$\begin{aligned}
\dot{\theta}(t) &= -R(t)\,u_p(t) \\
\ddot{\theta}(t) &= -R(t)\,[\dot{u}_p(t) - 2\,u_p^2(t)\frac{x_p(t)}{h_o}] \\
r_g(t) &= R(t)\,\sqrt{h_o^2+x_p^2(t)}\,[\alpha(t)\frac{x_p(t)}{h_o} + \beta(t) + h_o] \qquad (52)
\end{aligned}$$

in which $[\alpha(t), \beta(t)]$ are coordinates of points along the flap wavemaker.

3.2.3 Generation of a long wave by a piston wavemaker

In a long wave of permanent form over constant depth h_o, we have at any instant,

$$\int_{-h_o}^{\eta} u \ \mathrm{d}z = c_a \eta + Q_s + u_c h_o \tag{53}$$

in which c_a is the propagation speed of the wave in a fixed frame of reference, $\eta(x,t)$ is the wave elevation above still water level, Q_s is the nonlinear mass flux averaged over a wave period, and u_c, the speed of the current defined as the averaged particle velocity below wave trough level.

For a first-order long wave, the right hand side of (53) simply reduces to $c\eta$, where c is the speed of the wave relative to the water, so that (53) becomes the simpler expression used, e.g. by Goring [20] (1978), for determining the motion required by a piston wavemaker to generate a specified water surface elevation immediately in front of the wavemaker. Since the piston motion creates a depth uniform horizontal velocity $u_p(x_p(t),t)$, (53) reduces to,

$$u_p(t) = \frac{c\eta}{h_o + \eta} \tag{54}$$

which means, a surface elevation η can be generated by specifying the piston velocity u_p as defined above. In this case, corresponding horizontal piston motion $x_p(t)$ is given by,

$$x_p(t) = \int_0^t \frac{c\,\eta(x,\tau)}{h_o + \eta(x,\tau)} \ \mathrm{d}\tau \tag{55}$$

i) Generation of a "first-order" solitary wave by a piston wavemaker : In water of depth h_o, a *first-order* solitary wave elevation of amplitude H (i.e., a permanent wave solution of Boussinesq equations) reads,

$$\eta'(x',t') = H'\mathrm{sech}^2[\kappa(x' - c't')] \tag{56}$$

where $\kappa = \frac{\sqrt{3H'}}{2}$ and the celerity $c' = \sqrt{1+H'}$. Substituting (56) into (55), while specifying $x' = x'_p(t)$ throughout the integration, gives the corresponding piston motion.

Since the solitary wave profile (56) extends to infinity in both directions, however, it is necessary to truncate the wave at some distance from the origin, before it is used in the model.

Goring [20] introduced the significant horizontal extension $2\lambda'$ of the wave, corresponding to a reduction in wave elevation to, $\eta' = \varepsilon_z H'$. Using this, we get by (56),

$$\begin{aligned} \varepsilon_z H' &= H' \mathrm{sech}^2[\kappa\lambda'] \\ \varepsilon_z^{-\frac{1}{2}} &= \cosh \kappa\lambda' \end{aligned} \tag{57}$$

and,

$$\ell = \mathrm{arcosh}[\varepsilon_z{}^{-\frac{1}{2}}] \qquad \text{with} \qquad \lambda' = \frac{\ell}{\kappa} \tag{58}$$

Now (see Abramowitz & Stegun [1] (1965)),

$$\begin{aligned} \mathrm{arcosh}\, x &= \log\,[x + (x^2 - 1)^{\frac{1}{2}}] \\ \mathrm{arcosh}[\varepsilon_z{}^{-\frac{1}{2}}] &= \log\,\{\varepsilon_z{}^{-\frac{1}{2}}[1 + (1 - \varepsilon_z)^{\frac{1}{2}}]\} \end{aligned} \tag{59}$$

Hence, since $\varepsilon_z \ll 1$,

$$\ell \simeq \log \frac{4 - \varepsilon_z}{2\varepsilon_z^{\frac{1}{2}}} \tag{60}$$

In the numerical applications, we usually use $\varepsilon_z = 0.002$, to which it corresponds $\ell \simeq 3.80$.

Wave generation by the piston wavemaker, hence, starts at $t'_o = 0$, with $x' = x'_p + \lambda'$. Introducing this in the theoretical wave profile (56), and integrating (55) we then get,

$$x'_p(t') = \frac{H'}{\kappa}\,[\tanh \chi(t') + \tanh \kappa\lambda'] \quad \text{with} \quad \chi(t') = \kappa(c't' - x'_p(t') - \lambda') \tag{61}$$

This transcendental equation in x'_p is solved by Newton iterations for any given time t'.

Wavemaker velocity, $u'_p(t')$ is then computed by (54), for $\eta'(x'_p(t'), t')$, and $\dot{u}'_p(t')$ is found by time derivation of it. We get,

$$\begin{aligned} u'_p(t) &= H'\,(1 + H')^{\frac{1}{2}}\,\frac{1}{\cosh^2 \chi(t') + H'} \\ \dot{u}'_p(t) &= \sqrt{3}\,H'^{\frac{3}{2}}\,(1 + H')\,\frac{\cosh^3 \chi(t') \sinh \chi(t')}{(\cosh^2 \chi(t') + H')^3} \end{aligned} \tag{62}$$

These values are introduced into (44), to define the boundary conditions on the wavemaker.

Initial wavemaker velocity and acceleration at $t'_o = 0$ are deduced as a function of H' and ε_z, by introducing (58), (61) into (62). Since we have $x'_p(t'_o) = 0$, $\chi(t'_o) = -\ell$ and,

$$\cosh \chi(t'_o) = \varepsilon_z^{-\frac{1}{2}}, \qquad \sinh \chi(t'_o) = \varepsilon_z^{-\frac{1}{2}}[1 - \varepsilon_z]^{\frac{1}{2}}$$

which, by (62), leads to,

$$\begin{aligned} u'_p(t'_o) &= H'(1+H')^{\frac{1}{2}} \frac{\varepsilon_z}{1+\varepsilon_z H'} \\ \dot{u}'_p(t'_o) &= \sqrt{3}\, H'^{\frac{3}{2}}(1+H')\,\varepsilon_z \frac{(1-\varepsilon_z)^{\frac{1}{2}}}{(1+\varepsilon_z H')^3} \end{aligned} \tag{63}$$

which both are approximately proportional to ε_z, for a given H'.

Hence, the initial wavemaker acceleration, an important parameter that must be kept small in order to avoid initial mathematical singularity of the solution (see section 4.8), is controlled by selecting the truncation parameter ε_z of specified solitary waves, small enough. For $\varepsilon_z = 0.002$ and H'=0.5, for instance, we get $u_p(t_o) \simeq 0.00122\sqrt{gd}$, and $\dot{u}_p(t_o) \simeq 0.00184g$ which is quite small.

ii) Generation of a "first-order" cnoidal wave by a piston wavemaker : First-order cnoidal waves are periodic wave solutions of KdV or Boussinesq's equations. In water of constant depth, $h'_o = 1$, a cnoidal wave elevation of amplitude H', period T', and length $L' = c'T'$ is given by (e.g. Dean and Dalrymple [13] 1984),

$$\eta'(x',t') = H'\{B + \mathrm{cn}^2[\frac{2K}{L'}(x' - c't'), m]\} \tag{64}$$

in which, $L' = 4K(\frac{m}{3H'})^{\frac{1}{2}}$, the celerity $c' = \sqrt{1 + AH'}$ with $A = A(m) = \frac{1}{m}(2 - m - 3\frac{E}{K})$, and the dimensionless trough $B = B(m) = \frac{1}{m}(1 - m - \frac{E}{K})$.

In (64), cn is a Jacobian elliptic function of parameter m, and $K = K(m)$, $E = E(m)$ denote complete elliptic integrals of the 1st and 2nd kind, respectively (for details and definitions, see Abramowitz & Stegun [1]).

Wave generation starts for $x'_p = t' = 0$, at a given initial phase $x' = \lambda'$ of the wave profile (64). Setting $x' = x'_p + \lambda'$ in (64), and integrating (55), we get the following transcendental

expression for $x_p'(t')$, which is solved by Newton iterations,

$$\chi(t) = \frac{2K}{L'}[x_p'(t) + \lambda' - c't'] \tag{65}$$

$$x_p'(\chi(t)) = \frac{L'}{2K}H'\{\frac{E}{mK}(\chi(t) - \chi_o) - \frac{1}{m}[E(\chi(t), m) - E(\chi_o, m)]\} \tag{66}$$

in which $\chi_o = \frac{2K}{L'}\lambda'$, and $E(\chi(t), m)$ is the incomplete elliptic integral of the 1st kind. Finally, $u_p'(t')$ and $\dot{u}_p'(t')$ are obtained by derivation and introduced into (44), which defines the boundary conditions at the wavemaker.

Initial acceleration $\dot{u}_p'(t_o')$ of the wavemaker varies with the initial phase λ', and hence can be made sufficiently small by adjusting this phase. For $\lambda' = 0$, for instance, initial acceleration is zero. For cnoidal waves, however, this also corresponds to maximum crest elevation and velocity. Hence, the origin is shifted to a point with zero water elevation and velocity, by selecting,

$$\lambda' = \frac{L'}{2K}[2K - \mathrm{cn}^{-1}\sqrt{-B}] + x_p'(0) \tag{67}$$

where $x_p'(0)$ is obtained from (66) for $\chi = 0$. Doing so, the initial acceleration is no longer zero but, for long waves, it is still quite small compared to gravity ($\mathcal{O}(4c'K\frac{H'}{L'})$).

For a cnoidal wave of height $H' = 0.2$, and period $T' = 25$, which is close to the upper limit of long wave theory, for instance, we get $L' = 25.99$, $c' = 1.040$ and $K = 5.035$, and Figure 3 shows the free surface elevation and paddle motion calculated with these data. Corresponding initial acceleration is about $0.03g$.

3.2.4 Generation of a sum of periodic sine waves by a flap wavemaker

A sum of sine waves can be generated by a flap wavemaker in water of depth h_o, by specifying boundary conditions, based on *first-order* Stokes theory, as in laboratory experiments. Due to nonlinearities, however, it is well known, free second and higher-order harmonics are created when waves of finite amplitude propagate down a tank (see, e.g., Mei [53] 1983). This will be illustrated in the applications.

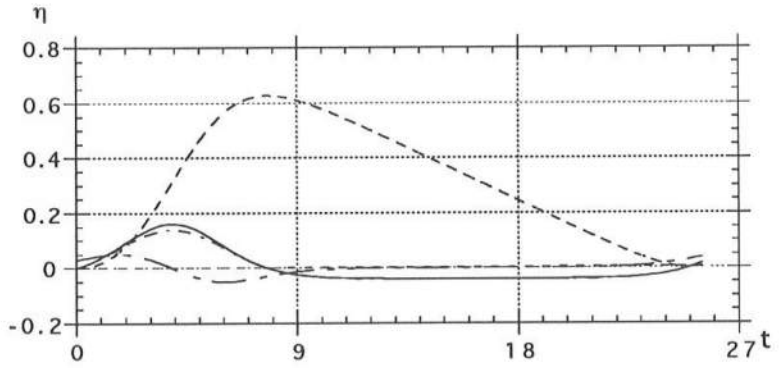


Figure 3: Cnoidal wave elevation and paddle motion as a function of time t', for $H' = 0.2$, $T' = 25$, with η(——), x_p(- - - -), u_p(- — -) and $\dot{u}_p$(. — .).

The paddle stroke $x_p(t)$ is specified as the sum $\mathcal{S}(t)$ of n sine functions of frequency $2\pi\omega_i$, phase φ_i, and amplitudes A_i. Amplitudes A_i are related, in a linear sense, to the corresponding wave component amplitudes to be generated, a_i, by a transfer function $\mathcal{T}(\omega_i, h_o)$ that can be derived from wavemaker theory (see, e.g., Dean & Dalrymple [13] 1984).

Furthermore, a smooth start with small and bounded initial acceleration of the paddle is ensured by multiplying $x_p(t)$, by a damping function $\mathcal{D}(t)$, varying from 0 to $(1 - \varepsilon_z)$ over a given time $2\,t_{\varepsilon_z}$. For $\varepsilon_z \ll 1$, the damping function gives a smooth transition from 0 to $\sim \mathcal{S}(t)$, over a time $2\,t_{\varepsilon_z}$. We get,

$$x_p(t) = \mathcal{S}(t)\mathcal{D}(t) \qquad \text{with} \qquad \mathcal{S}(t) = \sum_{i=1}^{n} \frac{1}{2} A_i \left[1 - \cos\left(\omega_i\, t + \varphi_i\right)\right]$$

$$a_i = A_i\, \mathcal{T}(\omega_i, h_o) \qquad \text{with} \qquad \mathcal{T}(k_i(\omega_i, h_o), h_o) = \frac{4 \sinh^2 k_i h_o}{2k_i d + \sinh 2k_i h_o} \tag{68}$$

with $H_i = 2a_i$, the wave height (predicted by linear wave theory), and $k_i(\omega_i, h_o)$, the wavenumber of a given sine wave component to be generated. By the linear dispersion relation, we have,

$$k_i \tanh k_i h_o = \frac{\omega_i^2}{g} \tag{69}$$

Now, by analogy with the smooth initial paddle motion, obtained above for the generation of solitary waves by a piston wavemaker (61), the damping function is selected as,

$$\mathcal{D}(t) = \frac{1+\varepsilon_z}{2}\left[\tanh \mu(t - t_{\varepsilon_z}) + \frac{1-\varepsilon_z}{1+\varepsilon_z}\right] \tag{70}$$

with μ, a damping coefficient defined from the condition that $\mathcal{D}(0) = 0$ as,

$$\mu = -\frac{1}{2\,t_{\varepsilon_z}} \log \varepsilon_z \tag{71}$$

in which use has been made of the definition (see Abramowitz & Stegun [1]),

$$\operatorname{arctanh} x = \frac{1}{2} \log \frac{1+x}{1-x} \tag{72}$$

One can easily check using (72), that (70)-(71) also satisfy $\mathcal{D}(2\,t_{\varepsilon_z}) = 1 - \varepsilon_z$, which allows to select the rate of damping corresponding to given values of t_{ε_z} and ε_z : for $\varepsilon_z = 0.001$, for instance, we get $\mu \simeq \frac{3.454}{t_{\varepsilon_z}}$.

The time $2t_{\varepsilon_z}$ is selected as an integer multiple N_n of the average wave period of the wave components $\tilde{T}$ defined as,

$$t_{\varepsilon_z} = \frac{N_n \tilde{T}}{2} \qquad \text{and} \qquad \tilde{T} = \frac{1}{n}\sum_{i=1}^{n} \frac{2\pi}{\omega_i} \tag{73}$$

By time derivation, we now get the paddle velocity and acceleration at $z = 0$, from (68)-(70), as,

$$\begin{aligned}
u_p(t) &= \dot{\mathcal{S}}\mathcal{D} + \mathcal{S}\dot{\mathcal{D}} \qquad \text{and} \qquad \ddot{u}_p(t) = \ddot{\mathcal{S}}\mathcal{D} + 2\dot{\mathcal{D}}\dot{\mathcal{S}} + \mathcal{S}\ddot{\mathcal{D}} \\
\dot{\mathcal{S}}(t) &= \sum_{i=1}^{n} \frac{1}{2} A_i \omega_i \sin(\omega_i t + \varphi_i), \qquad \ddot{\mathcal{S}}(t) = \sum_{i=1}^{n} \frac{1}{2} A_i \omega_i^2 \cos(\omega_i t + \varphi_i) \\
\dot{\mathcal{D}}(t) &= \frac{\mu}{2} \frac{1+\varepsilon_z}{\cosh^2 \mu(t - t_{\varepsilon_z})}, \qquad \ddot{\mathcal{D}}(t) = -\mu^2 (1+\varepsilon_z) \frac{\tanh \mu(t - t_{\varepsilon_z})}{\cosh^2 \mu(t - t_{\varepsilon_z})}
\end{aligned} \tag{74}$$

Hence, boundary conditions (49),(52) can be defined on the wavemaker.

The wavemaker velocity and acceleration at initial time $t_o = 0$ can also be obtained from (74), by noting $\mathcal{D}(t_o) = 0$ as,

$$\dot{\mathcal{D}}(t_o) = 2\mu\varepsilon_z \frac{1}{1+\varepsilon_z} \qquad \text{and} \qquad \ddot{\mathcal{D}}(t_o) = 4\mu^2 \varepsilon_z \frac{1-\varepsilon_z}{(1+\varepsilon_z)^2} \tag{75}$$

For $\varepsilon_z = 0.001$, we get by (75), $\dot{\mathcal{D}}(t_o) \simeq 0.0020\mu$ and $\ddot{\mathcal{D}}(t_o) \simeq 2\mu\dot{\mathcal{D}}(t_o)$. This leads by (74) to,

$$u_p(t_o) \simeq \mathcal{S}(t_o)\dot{\mathcal{D}}(t_o) \qquad \text{and} \qquad \dot{u}_p(t_o) \simeq 2\dot{\mathcal{D}}(t_o)(\dot{\mathcal{S}}(t_o) + \mu\mathcal{S}) \tag{76}$$

If we further require $\mathcal{S}(t_o) = 0$, we get by (76), $u_p(t_o) \simeq 0$ and $\dot{u}_p(t_o) \simeq 0.0040\mu\dot{\mathcal{S}}(t_o)$, with $\mu \simeq 1$ for $t_{\varepsilon_z} \simeq 3.454$. The acceleration is thus rather small at $t = t_o$.

3.3 Wave generation by an internal line of sources

Using the Boundary Integral Equation representation introduced in section 4.3, based on free space Green's function, Poisson equation (8) transforms into,

$$\begin{aligned} \alpha(\mathbf{x}_l)\phi(\mathbf{x}_l) &= \int_{\Gamma(\mathbf{x})} [\frac{\partial \phi}{\partial n}(\mathbf{x})G(\mathbf{x},\mathbf{x}_l) - \phi(\mathbf{x})\frac{\partial G(\mathbf{x},\mathbf{x}_l)}{\partial n}]\, d\Gamma(\mathbf{x}) \\ &\quad + \int_{\Omega(\mathbf{x})} b(\mathbf{x},t)G(\mathbf{x},\mathbf{x}_l)\, d\Omega(\mathbf{x}) \end{aligned} \tag{77}$$

which can itself be solved by the Boundary Element Method (BEM) introduced in section 4.4 for Laplace's equation. Domain integrals, however, have to be calculated to account for the source field $b(\mathbf{x},t)$ contribution in (77).

Using a vertical line of sources with linear density for wave generation, $q(s(\mathbf{x}),t)$ ($s(\mathbf{x})$ being measured along the line Γ_ℓ), the source contribution in (77) reduces to,

$$\int_\Omega b(\mathbf{x},t)G(\mathbf{x},\mathbf{x}_l)\, d\Omega = \int_{\Gamma_\ell} q(s(\mathbf{x}),t)G(\mathbf{x},\mathbf{x}_l)\, d\Gamma_\ell \tag{78}$$

In two dimensions, a line of sources with continuously varying strength creates a velocity normal to the line, equal to $\frac{1}{2}q$. Thus, specification of the strength of the source distribution q is straight-forward if particle velocities are known along the line, for the waves to be generated.

In most cases, it is sufficient to specify the source strength only at points along the line Γ_ℓ. Doing so, N_s concentrated sources of strength $B_s(t)$ are specified at these nodes. For

a vertical source line located at $x = x_s$, and divided into N_s segments from bottom to free surface, we thus have,

$$B_s(t) = 2\frac{\eta(x_s,t) + h_o}{N_s}\overline{u_w}(x_s, z_s, t) \qquad s = 1, \ldots, N_s \tag{79}$$

where $\overline{u_w}(x_s, z_s, t)$ represents the mean horizontal velocity of the wave in the s-th segment, and $\eta = \eta(x_s, t)$ the wave elevation above the source line (a stretching has been applied to the line to account for the wave elevation above the line). In this case, also,

$$q(s(\mathbf{x}), t) = \sum_{s=1}^{N_s} B_s(t)\delta(\mathbf{x} - \mathbf{x_s}) \qquad s = 1, \ldots, N_s \tag{80}$$

where $\delta(\mathbf{x} - \mathbf{x_s})$ is the Dirac function at point $\mathbf{x_s}$. Equation (78) simplifies into,

$$\int_\Omega b(\mathbf{x}, t)G(\mathbf{x}, \mathbf{x}_l)\, d\Omega = \sum_{s=1}^{N_s} B_s(t) \int_\Omega G(\mathbf{x}, \mathbf{x}_l)\delta(\mathbf{x} - \mathbf{x_s})\, d\Omega$$

and by the sifting property of the Dirac function,

$$\int_\Omega b(\mathbf{x}, t)G(\mathbf{x}, \mathbf{x}_l)\, d\Omega = \sum_{s=1}^{N_s} B_s(t)G(\mathbf{x}_s, \mathbf{x}_l) \tag{81}$$

Hence, this method of generation makes it possible to model any wave motion for which particle velocity distribution is given along a chosen bottom-to-surface line. Two such cases are detailed in the following.

3.3.1 "Second-order" solitary waves

For a solitary wave whose *first-order* profile is given by (56), the horizontal velocity can be deduced as a function of depth from Boussinesq's theory (see Mei [53]). The horizontal velocity is constant over depth, to the first order in H'. Identical developments can be made up to the 2nd-order accuracy, and we get,

$$u_w(x_s, z, t) = \frac{Hg}{c}\mathrm{sech}^2\chi(t)\,[1 + (\frac{\kappa}{h_o})^2(z + h_o)^2(2\tanh^2\chi(t) + \sinh^2\chi(t))] \tag{82}$$

in which $\chi(t)$ is defined as in (61), and the solitary wave has been limited to its significant part 2λ defined as in (58).

In dimensionless form, (82) reads,

$$u'_w(x'_s, z', t') = \frac{H'}{c'^2}\mathrm{sech}^2\chi(t')\left[1 + (\kappa^2(z'+1)^2(2\tanh^2\chi(t') + \sinh^2\chi(t'))\right] \tag{83}$$

Equation (82) could be extended to higher-order.

Notice, in the implementation of this procedure in the model, source strengths defined based on (81) and (82) correspond to the Poisson equation (8) for ϕ. Corresponding developments have been made for the equation for $\frac{\partial\phi}{\partial t}$, using $\frac{\partial u_w}{\partial t}$ instead of u_w.

3.3.2 "Second-order" periodic waves

For a wave of period T and height H, the horizontal velocity calculated from Stokes theory in water of depth h_o, up to *second-order* in $\frac{H}{L}$, reads (see Dean & Dalrymple [13]),

$$\begin{aligned} u_w(x_s, z, t) &= \frac{Hg}{2c}\frac{\cosh k(h_o+z)}{\cosh kh_o}\cos(kx_s - \omega t) \\ &- \frac{3H^2\omega k}{16}\frac{\cosh 2k(h_o+z)}{\sinh^4 kh_o}\cos 2(kx_s - \omega t) \end{aligned} \tag{84}$$

in which, $\omega = \frac{2\pi}{T}$, is the wave circular frequency, $c = \frac{\omega}{k}$ the wave celerity, and the wavenumber k is given by the linear dispersion relation (69).

To avoid initial singularity during a "cold start", the velocity (83) is multiplied by a damping function of the form,

$$\mathcal{D}(t) = 1 - \exp(-\frac{\omega t}{\mu\pi}) \tag{85}$$

in which μ, is a damping coefficient defined from the condition that the damping reach, $1-\varepsilon_z$, after a specified number N_n of wave periods, i.e., $\mathcal{D}(N_nT) = 1 - \varepsilon_z$ or,

$$\mu = -\frac{2N_n}{\log\varepsilon_z} \tag{86}$$

For $N_n = 3$, for instance, we get $\mu = 1$ for $\varepsilon_z = 0.0025$, and $\mu = 0.65$ for $\varepsilon_z = 0.0001$.

Notice, again, source strengths defined by (81) and (84) correspond to the Poisson equation for ϕ. Corresponding developments have been made for the equation for $\frac{\partial\phi}{\partial t}$, using $\frac{\partial u_w}{\partial t}$ instead of u_w.

4 Numerical Model

4.1 General principle

Taylor series (9) and (10) are truncated to second-order in Δt ($N = 2$), and coefficients (14), (27), (28) and (34) are expressed as function of values of $\{\phi, \frac{\partial \phi}{\partial n}, \frac{\partial \phi}{\partial s}, \frac{\partial^2 \phi}{\partial n \partial s}, \frac{\partial^2 \phi}{\partial s^2}, \frac{\partial \phi}{\partial t}, \frac{\partial^2 \phi}{\partial t \partial n}, \frac{\partial^2 \phi}{\partial t \partial s}, \beta, \frac{\partial \beta}{\partial s}, p_a, \frac{D p_a}{D t}\}$ along the free surface. Potential ϕ, $\frac{\partial \phi}{\partial t}$, and their n-derivatives, are directly obtained from the numerical solution of two Laplace's equations, for ϕ and $\frac{\partial \phi}{\partial t}$, expressed in the same domain geometry at time t.

The s-derivatives of field variables are computed along the free surface, using a 4th-order "sliding" polynomial interpolation on the boundary, and by differentiating inside each polynomial.

At the intersection of the free surface with a moving wavemaker boundary, however, accuracy of the s-derivatives is not in general sufficient, and special relationships have been developed by Grilli & Svendsen [37] ("compatibility conditions") for improving the accuracy at corners on the free surface. These are discussed below.

4.2 Time stepping method

When initial conditions are known on the free surface at a given time, t, i.e., the position $\mathbf{r}(t)$ of the free surface boundary $\Gamma_f(t)$ and the potential ϕ, its normal gradient $\frac{\partial \phi}{\partial n}$ and time derivatives $\frac{\partial \phi}{\partial t}$ and $\frac{\partial^2 \phi}{\partial t \partial n}$ along $\Gamma_f(t)$, the free surface position and potential can be updated to a subsequent time, $t + \Delta t$, using Taylor series (9) and (10). This only requires calculating s-derivatives of the field variables along the free surface (see details about that in a subsection below).

Hence, assuming $\{\phi, \frac{\partial \phi}{\partial n}, \frac{\partial \phi}{\partial s}, \frac{\partial^2 \phi}{\partial n \partial s}, \frac{\partial^2 \phi}{\partial s^2}, \frac{\partial \phi}{\partial t}, \frac{\partial^2 \phi}{\partial t \partial n}, \frac{\partial^2 \phi}{\partial t \partial s}, \beta, \frac{\partial \beta}{\partial s}, p_a, \frac{D p_a}{D t}\}$ are known or calculated at time t on the free surface $\Gamma_f(t)$, position $\mathbf{r}$ and potential ϕ on the free surface

are updated to time $t + \Delta t$, up to second-order accuracy in Δt, as,

$$\overline{\mathbf{r}}(t + \Delta t) = \mathbf{r}(t) + \Delta t \frac{D\,\mathbf{r}}{D\,t}(t) + \frac{(\Delta t)^2}{2}\frac{D^2\mathbf{r}}{Dt^2}(t) + \mathcal{O}[(\Delta t)^3] \tag{87}$$

$$\overline{\phi}(\mathbf{r}(t + \Delta t), t + \Delta t) = \phi(t) + \Delta t \frac{D\,\phi}{D\,t}(t) + \frac{(\Delta t)^2}{2}\frac{D^2\phi}{Dt^2}(t) + \mathcal{O}[(\Delta t)^3] \tag{88}$$

with the coefficients in (87),(88) given by (14), (27), (28) and (34), and $\mathbf{s}$ and $\mathbf{n}$ given by (18),(19) as a function of β.

Values of ϕ or $\frac{\partial \phi}{\partial n}$ and the geometry can be specified on lateral boundaries at time $t + \Delta t$, depending on the specific problem (motion and $\frac{\partial \phi}{\partial n}$ $\frac{\partial^2 \phi}{\partial t \partial n}$ for instance, will be calculated by (44) along a piston wavemaker, and $\frac{\partial \phi}{\partial n}$ is invariably zero along solid boundaries, by (5)).

Hence, well posed boundary values can be specified at $t + \Delta t$, for ϕ or $\frac{\partial \phi}{\partial n}$, along the whole boundary $\Gamma(t + \Delta t)$, and a "first" Laplace problem can be defined with these, and solved to calculate ϕ or $\frac{\partial \phi}{\partial n}$ (whichever is unknown) along Γ (the solution of Laplace's equation by a Boundary Integral equation is discussed later).

Now, $\frac{\partial \phi}{\partial t}$ can be specified on the free surface using Bernoulli's equation (12), as,

$$\overline{\frac{\partial \phi}{\partial t}}(t + \Delta t) = -\frac{1}{2}[(\frac{\partial \phi}{\partial s})^2 + (\frac{\partial \phi}{\partial n})^2] - \frac{1}{\rho}p_a - gz(\mathbf{r}) \qquad \text{on } \Gamma_f(t) \tag{89}$$

in which all right hand side variables and geometry are known at time $t + \Delta t$. Depending on the type of conditions along the rest of the boundary, $\frac{\partial^2 \phi}{\partial t \partial n}$ or $\frac{\partial \phi}{\partial t}$ can now similarly be specified and, hence, well posed boundary values for solving a Laplace's equation for $\frac{\partial \phi}{\partial t}$ can be determined and a "second" Laplace problem be solved.

Notice, since both the above Laplace problems are expressed in the same boundary geometry, $\Gamma(t + \Delta t)$, the extra computational effort required to solve the second problem will be very small. Therefore at this stage, geometry of the boundary and values of ϕ, $\frac{\partial \phi}{\partial n}$, $\frac{\partial \phi}{\partial t}$ and $\frac{\partial^2 \phi}{\partial t \partial n}$ along the boundary are known for time $t + \Delta t$, and the whole procedure can be applied again.

The above operations are globally referred to as "time stepping" and the above procedure corresponds to time t, with the time step being Δt.

4.3 Transformation of Laplace's equations into BIE's

Both Laplace problems for ϕ and $\frac{\partial \phi}{\partial t}$ can be transformed into Boundary Integral Equations (BIE), using third Green's identity, and the free space Green's function G being defined such as to satisfy,

$$\nabla^2 G(\mathbf{x}, \mathbf{x}_l) + \delta(\mathbf{x}, \mathbf{x}_l) = 0 \tag{90}$$

in which $\delta(\mathbf{x}, \mathbf{x}_l)$ represents a Dirac function at point $\mathbf{x}_l$ of domain Ω. With definition (90), third Green's identity for the potential ϕ reads,

$$\phi(\mathbf{x}_l) = \int_{\Gamma(\mathbf{x})} [\, G(\mathbf{x}, \mathbf{x}_l) \frac{\partial \phi}{\partial n}(\mathbf{x}) - \phi(\mathbf{x}) \frac{\partial G}{\partial n}(\mathbf{x}, \mathbf{x}_l) \,] \, d\Gamma(\mathbf{x}) \tag{91}$$

in which the "sifting" property of the Dirac function has been used to eliminate the domain integral.

In two-dimensions, solution of (90) yields (e.g., Brebbia [2])

$$G(\mathbf{x}, \mathbf{x}_l) = -\frac{1}{2\pi} \log | \, \mathbf{x} - \mathbf{x}_l \, | \tag{92}$$

The function $G(\mathbf{x}, \mathbf{x}_l)$, also called the *fundamental solution* of Laplace's equation, has a logarithmic singularity when point $\mathbf{x}$ approaches point $\mathbf{x}_l$.

To derive a BIE only involving values of the field variables on the boundary, however, it is necessary, in (91), to select points $\mathbf{x}_l$ on boundary Γ over which functions in the right hand side of (91) are integrated. It follows that some of the integrals in (91) become strongly singular, and the limiting process by which $\mathbf{x}_l$ is made to approach the boundary must be carried out with great care.

Such an analysis can be found in many references dealing with BIE problems (e.g., Brebbia [2]), and can be shown to introduce "jumps" in the potential value when point $\mathbf{x}_l$ moves from inside the domain to the boundary and from the boundary to outside the domain. Values of these "jumps" are only function of boundary geometry.

Based on the above discussion, final singular BIE's corresponding to Laplace problems for ϕ and $\frac{\partial \phi}{\partial t}$ read,

$$\alpha(\mathbf{x}_l)\phi(\mathbf{x}_l) = \int_{\Gamma(\mathbf{x})} [\frac{\partial \phi}{\partial n}(\mathbf{x})G(\mathbf{x},\mathbf{x}_l) - \phi(\mathbf{x})\frac{\partial G}{\partial n}(\mathbf{x},\mathbf{x}_l)]\, d\Gamma(\mathbf{x})$$

$$\alpha(\mathbf{x}_l)\frac{\partial \phi}{\partial t}(\mathbf{x}_l) = \int_{\Gamma(\mathbf{x})} [\frac{\partial^2 \phi}{\partial t \partial n}(\mathbf{x})G(\mathbf{x},\mathbf{x}_l) - \frac{\partial \phi}{\partial t}(\mathbf{x})\frac{\partial G}{\partial n}(\mathbf{x},\mathbf{x}_l)]\, d\Gamma(\mathbf{x}) \tag{93}$$

in which $\mathbf{x} = (x, z)$ and $\mathbf{x}_l = (x_l, z_l)$ are points on boundary Γ and $\alpha(\mathbf{x}_l)$ is a geometric coefficient only function of the angle of the boundary at point $\mathbf{x}_l$. This coefficient actually represents the "jumps" in potential mentioned above. Values of $\alpha(\mathbf{x}_l)$ are discussed later.

4.4 Discretization of the Boundary Integral Equations

4.4.1 Principle

The Boundary Element method (BEM) (Brebbia [2]) is used for the discretization required for the numerical solution of the two BIE's (93). Collocation nodes $\mathbf{x}_l$ are distributed along the entire boundary to describe the variation of boundary geometry as well as boundary conditions and the unknown functions of the problem. Between collocation nodes, the variation of all quantities is described by means of shape functions or of splines, and for this purpose, the boundary is divided into elements each of which contains two or more nodes. Details of these procedure are given in a following section.

Each integral in the BEM is transformed into a sum of integrals over each boundary element. Non-singular integrals are calculated by standard Gauss quadrature rules. A kernel transformation is applied to the weakly singular integrals, which are then integrated by a numerical quadrature exact for the logarithmic singularity. An adaptive numerical integration is used for improving the accuracy of regular integrations near corners and other locations, like the overturning jet in breakers or at the upper part of a gentle slope, where elements on different parts of the boundary are close to each other.

A double node technique (Brebbia [2]) is used in combination with specific continuity and compatibility relationships to utilize and make compatible all information given in corners by the boundary conditions. Corner double nodes represent two nodes of identical coordinates with different nodal values of the field variables. Hence, two algebraic BIE's are expressed for each double node, which, however, are not independent. Continuity conditions express continuity of ϕ or $\frac{\partial \phi}{\partial t}$, for both nodes of the double nodes, and compatibility conditions express uniqueness of the velocity vector at corners, based on the values of $\frac{\partial \phi}{\partial s}$ and $\frac{\partial \phi}{\partial n}$ on both intersecting boundaries, i.e., again, for both nodes of the double nodes.

4.4.2 Definition of boundary value problems

N_Γ points $\mathbf{x}_l$ are selected on boundary Γ to define a spatial discretization. On the free surface Γ_f these points represent both actual water particles whose trajectory is followed in time by the Eulerian-Lagrangian time stepping, and collocation nodes at which BIE's (93) are expressed.

With u the unknown field variable (either ϕ or $\frac{\partial \phi}{\partial t}$), boundary conditions prescribe either $\overline{u}$ (Dirichlet) or $\overline{\frac{\partial u}{\partial n}}$ (Neuman) on portions of the boundary. Hence, each BIE (93) reads, for each collocation node $\mathbf{x}_l$ ($l = 1, \ldots, N_\Gamma$),

$$\begin{aligned} \alpha(\mathbf{x}_l)u(\mathbf{x}_l) &= \int_{\Gamma_n(\mathbf{x})} [\overline{\frac{\partial u}{\partial n}}(\mathbf{x})G(\mathbf{x},\mathbf{x}_l) - u(\mathbf{x})\frac{\partial G}{\partial n}(\mathbf{x},\mathbf{x}_l)]\, d\Gamma(\mathbf{x}) \\ &\quad + \int_{\Gamma_d(\mathbf{x})} [\frac{\partial u}{\partial n}(\mathbf{x})G(\mathbf{x},\mathbf{x}_l) - \overline{u}(\mathbf{x})\frac{\partial G}{\partial n}(\mathbf{x},\mathbf{x}_l)]\, d\Gamma(\mathbf{x}) \end{aligned} \tag{94}$$

in which $G(\mathbf{x}, \mathbf{x}_l)$ is the free space Green's function (92) corresponding to Laplace's equation and $\frac{\partial G}{\partial n}$ represents its normal derivative. In two dimensions we get, by (92),

$$\begin{aligned} G(\mathbf{x},\mathbf{x}_l) &= -\frac{1}{2\pi}\log r_l \\ \frac{\partial G}{\partial n}(\mathbf{x},\mathbf{x}_l) &= -\frac{1}{2\pi}\frac{\mathbf{r_l}\cdot\mathbf{n}}{r_l^2} \\ r_l &= |\,\mathbf{r}_l\,|\,,\ \mathbf{r}_l = \mathbf{x}-\mathbf{x}_l \end{aligned} \tag{95}$$

in which r_l is the distance from the "integration point" $\mathbf{x}$ to the collocation point $\mathbf{x}_l$ on the boundary Γ. Γ_n represents all parts of the boundary on which $\overline{\frac{\partial u}{\partial n}}$ is imposed ("Neuman boundary"), and Γ_d all parts on which $\overline{u}$ is imposed ("Dirichlet boundary"). Depending on the case these, for instance read :

For a generation of waves by a wavemaker,

$$\Gamma_n \equiv \Gamma_{r1} \cup \Gamma_{r2} \cup \Gamma_b \qquad \Gamma_d \equiv \Gamma_f \tag{96}$$

For a space periodic wave train with periodicity conditions prescribed on lateral boundaries,

$$\Gamma_n \equiv \Gamma_{r1} \cup \Gamma_b \qquad \Gamma_d \equiv \Gamma_f \cup \Gamma_{r2} \tag{97}$$

Coefficients $\alpha(\mathbf{x}_l)$ can be expressed as functions of the interior angle θ_l of the boundary at $\mathbf{x}_l$ (Brebbia [2]) as,

$$\alpha(\mathbf{x}_l) = \frac{\theta_l}{2\pi} \tag{98}$$

4.4.3 Discretization of BIE's using boundary elements

i) Principles : The Boundary Element Method (BEM) is used to describe the variation of both geometry and field variables along the boundary (interpolation), and hence discretize and solve the BIE's. The interpolation between nodes on the boundary is based on higher-order isoparametric boundary elements, using shape functions (1st to 4th-order) for both geometry and field variables, and on quasi-cubic spline elements for which the interpolation of field variables is based on 1st-order shape functions, and the interpolation of the geometry is based on cubic parametric splines.

Quasi-spline elements have been implemented to ensure continuity of the free surface slope. They are used, when the free surface curvature is large, in combination with isopara-

metric elements on the rest of the boundary [I]. Quasi-spline elements require small extra computational effort, for enforcing the inter-element continuity of the derivatives.

As in the Finite Element Method, both isoparametric and quasi-spline boundary elements are mapped onto reference elements, before integrals are calculated in the BEM, and the Jacobian of the mapping function is determined analytically as a function of the coordinates of the nodes of the discretization, and of the interpolation functions.

ii) Isoparametric boundary elements : To compute integrals in (94), the boundary is divided into M_Γ elements, each of them having m nodes. Within the k-th element corresponding to boundary section Γ_e^k both the boundary geometry and the field variables $(u, \frac{\partial u}{\partial n})$ are discretized using identical sets of higher-order shape functions defined as polynomials of degree $(m-1)$ in $\mathbf{x}$ whose value is 1 at point j of element k, and 0 at all other points $i \neq j$. These shape functions are noted $N_{kj}^{m-1}(\mathbf{x})$, for $j = 1, \ldots, m$, and $k = 1, \ldots, M_\Gamma$.

The use of higher-order shape functions for the discretization increases the rate at which the approximate BEM solution converges to the exact (unknown) solution of the BIE, when the normalized size h (spatial step) of the discretization (i.e., the average distance between two nodes on the boundary is reduced). It can be shown, this rate is roughly proportional to h^m for an m-node element.

For convenience, the set of shape functions is analytically defined on a simple reference element of boundary Γ_ξ, onto which each element of the BEM discretization of boundary Γ_e^k is mapped by a transformation of coordinates. The intrinsic coordinate on this isoparametric reference element is $\xi \in [-1, 1]$. Boundary geometry $\mathbf{x}^k$ and field variables $(u^k, \frac{\partial u^k}{\partial n})$ over the k-th element corresponding to boundary section Γ_e^k are represented as function of their nodal values $(\mathbf{x}_j^k, U_j^k, \frac{\partial U_j^k}{\partial n})$, in which j numbers the nodes within each element [II], and of the

[I] With full spline elements, cubic splines would also be used for the field functions. This, however, would require knowledge of the derivatives of these functions at the extremities of the free surface which may be hard to accurately obtain in some cases.

[II] It is worthwhile pointing out, since all variables— geometry or field functions— are defined by a piecewise interpolation within each element, their nodal values (i.e. values at the nodes of the discretization) can either

shape functions set $N_j^{m-1}(\xi)$, with $j = 1, \ldots, m$ as,

$$\begin{aligned} \mathbf{x}^k(\xi) &= N_j^{m-1}(\xi)\,\mathbf{x}_j^k \\ u^k(\xi) &= N_j^{m-1}(\xi)\,U_j^k \;;\; \frac{\partial u^k}{\partial n}(\xi) = N_j^{m-1}(\xi)\,\frac{\partial U_j^k}{\partial n}; \quad j = 1, \ldots, m \text{ on } \Gamma_\xi \end{aligned} \tag{99}$$

Notice, from now on, the summation convention will be used for repeated subscripts.

Analytic expressions are derived for the shape function coefficients, by expressing that, at nodes $\mathbf{x}_i^k$ $(i = 1, \ldots, m)$, $u^k(\xi)$ in (99) is equal to nodal value U_i^k, or,

$$u^k(\xi(\mathbf{x}_i^k)) = N_j^{m-1}(\xi_i)\,U_j^k = U_i^k$$

or

$$\delta_{ij} U_j^k = U_i^k$$

where δ_{ij} is the Kronecker symbol. Since $N_j^{m-1}(\xi)$ has been defined as a polynomial of degree $m-1$ in ξ, this leads, for the i-th node ξ_i of an m-node reference element, to the equation,

$$N_j^{m-1}(\xi_i) = \delta_{ij}$$

$$\xi_i = (2i - m - 1)/(m-1); \qquad i, j = 1, \ldots, m \text{ on } \Gamma_\xi \tag{100}$$

---

be referred to with indices varying within each element k (local definition), or with indices varying in the global boundary discretization (global definition). For instance, U_j^k represents the nodal value of the potential at node j of element k and also represents the potential at, say, node l of the global discretization. The local numbering will usually be used when dealing with the representation within an element k (i.e., superscript :*element*, subscript: *node number within element*) and the global numbering will be used when dealing with the "assembled" representation, i.e., the final system matrix of the discretized problem (i.e., subscript: *node number in the global discretization*), unless otherwise mentioned. It is implicit that there is a functional correspondence between local and global numbering, depending on the degree $(m-1)$ and position on the boundary of each element k.

Solving (100) for a given m yields the corresponding polynomial coefficients. For example, for a cubic reference element, we get, [III] ($m = 4$),

$$N_1^3(\xi) = \frac{1}{16}(1-\xi)(9\xi^2-1)\ ,\ N_3^3(\xi) = \frac{9}{16}(1-\xi^2)(1+3\xi)$$

$$N_2^3(\xi) = \frac{9}{16}(1-\xi^2)(1-3\xi)\ ,\ N_4^3(\xi) = \frac{1}{16}(1+\xi)(9\xi^2-1) \tag{101}$$

iii) Quasi-spline boundary elements : Cubic spline elements have been used in other studies using the BEM, and have been found accurate, although computationally time consuming. Spline elements, however, require specification of tangential derivatives of both geometry and field variables at extremities of the free surface. For periodic wave problems, this can be avoided using extended periodicity conditions. For non-periodic problems in the physical space, however, tangential derivatives are not directly known and have to be independently calculated.

Based on the observation, for wave problems, higher-order continuity is somewhat more important for the boundary geometry than for the field variables, *quasi-spline elements* have been introduced (see, e.g., Longuet-Higgins & Cokelet [52], Dommermuth & Yue [15], and Grilli & Svendsen [34, 37] (1989,1990)) for which geometry is modeled by 2-node cubic splines and field variables by linear shape functions. Quasi-spline elements turn out to be rather accurate and efficient and provide the additional advantage against full spline elements, that only free surface slopes have to be specified at corners.

To be able to model breaking waves, the spline approximation of the geometry must account for a multi-valued free surface. For that purpose, the point index τ (also used in Longuet-Higgins & Cokelet [52]) is adopted as a parameter, whose value is equal to the index of the free surface nodes, at the position of these nodes (i.e., 1 to N_f where N_f is the total number of free surface nodes). Instead of defining the splines in polar coordinates, however, regular Cartesian coordinates are used to define two single-valued spline approximations of

[III] See, e.g., Brebbia [2], for other orders of shape functions.

the free surface : $x = x(\tau)$ and $z = z(\tau)$ (where $\mathbf{r} = (x, z)$ represents a free surface point). Hence, at the free surface nodes,

$$x_l = x(\tau_l) \quad ; \quad z_l = z(\tau_l) \qquad\qquad l = 1, \ldots, N_f \tag{102}$$

Two standard cubic spline analyses are performed on the free surface, for the points (x_l, τ_l) and (z_l, τ_l). In such analyses, the slope must be specified at each extremity of the approximated curve. In this case, $\frac{dx}{d\tau}$ and $\frac{dz}{d\tau}$ must be specified at both extremities of the free surface, and are estimated based on cubic polynomials fitted to the 4 first and 4 last nodes of the free surface. In some particular cases, however, the slope at one extremity of the free surface can be deduced from the physics of the problem and imposed explicitly as a boundary condition to the geometry.

The spline analyses lead to two tridiagonal matrices, that are solved by LU decomposition (operations of $\mathcal{O}(N_f)$), to provide $(\frac{d^2x_l}{d\tau^2}, \frac{d^2z_l}{d\tau^2})$ at each node of the free surface ($l = 1, ..., N_\Gamma$). Hence, for the k-th quasi-spline element of the free surface, we get,

$$\begin{aligned}
\mathbf{x}^k(\tau) &= (k+1-\tau)\mathbf{x}_1^k + (\tau - k)\mathbf{x}_2^k \\
&\quad + \frac{A(\tau)}{6}(A^2(\tau) - 1)\frac{d^2\mathbf{x}_1^k}{d\tau^2} + \frac{B(\tau)}{6}(B^2(\tau) - 1)\frac{d^2\mathbf{x}_2^k}{d\tau^2} \\
A(\tau) &= k + 1 - \tau \quad ; \quad B(\tau) = \tau - k \\
u^k(\xi) &= N_j^1(\xi)U_j^k \ , \ \frac{\partial u^k}{\partial n}(\xi) = N_j^1(\xi)\frac{\partial U_j^k}{\partial n}; \qquad j = 1,2 \text{ on } \Gamma_\xi
\end{aligned} \tag{103}$$

Notice that for any variable V, V_1^k represents the value of V at the first node of element k, and V_2^k at the second node. Because quasi-spline elements are 2-node elements, however, this is also the same as (V_k, V_{k+1}), in the global numbering of the free surface nodes, defined from 1 to N_Γ.

The reference element for the quasi-spline element, is a 2-node element with the intrinsic coordinate $\xi \in$ [-1,+1] and,

$$\tau(\xi) = k + \frac{\xi + 1}{2} \qquad\qquad k = 1, \ldots, M_f \tag{104}$$

where $M_f = N_f - 1$ is the number of quasi-spline elements on the free surface.

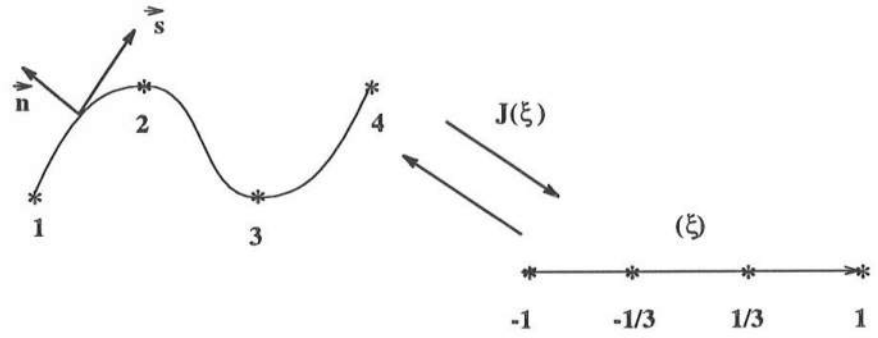


Figure 4: Sketch for the transformation of coordinates on the boundary.

4.4.4 Transformation of coordinates, high order s-derivatives

i) Isoparametric boundary elements :

- *Jacobian and normal vector :*

  All the BEM integrals are computed on a reference element Γ_ξ onto which each element on the actual boundary Γ_e^k is mapped by a transformation of coordinates. The transformation from the k-th boundary element of Γ : Γ_e^k to the reference element $\Gamma_\xi \equiv \xi \in [-1, 1]$ (Figure 4), is described by the Jacobian $J^k(\xi)$ of the transformation given, by definition, by,

$$J^k(\xi) = \frac{\partial s^k}{\partial \xi}(\xi) = [(\frac{\partial x^k}{\partial \xi})^2 + (\frac{\partial z^k}{\partial \xi})^2]^{\frac{1}{2}} \tag{105}$$

  and becomes by (99),

$$J^k(\xi) = [(\frac{dN_j^{m-1}}{d\xi}(\xi)x_j^k)^2 + (\frac{dN_j^{m-1}}{d\xi}(\xi)z_j^k)^2]^{\frac{1}{2}}$$

$$j = 1, \ldots, m \; ; \; k = 1, \ldots, M_\Gamma \qquad \text{on } \Gamma_\xi \tag{106}$$

  In the same way, the outward normal vector is given by,

$$\mathbf{n}^k(\xi) = (-\sin\beta^k(\xi), \cos\beta^k(\xi)) \tag{107}$$

which becomes by (19), the discretization (99) and (106),

$$\mathbf{n}^k(\xi) = \frac{1}{J^k(\xi)}[-\frac{\partial z^k}{\partial \xi}\ ,\ \frac{\partial x^k}{\partial \xi}] \tag{108}$$

$$\mathbf{n}^k(\xi) = \frac{1}{J^k(\xi)}[-\frac{dN_j^{m-1}}{d\xi}(\xi)z_j^k, \frac{dN_j^{m-1}}{d\xi}(\xi)x_j^k] \tag{109}$$

which also provides, together with (107), the expression of $(\cos\beta^k(\xi), \sin\beta^k(\xi))$, within each element.

- *s-derivatives :*

  Using $v^k(\xi)$ to denote either of the variables ϕ, $\frac{\partial\phi}{\partial n}$ or $\frac{\partial\phi}{\partial t}$ over the k-th element, and V_j^k to represent their nodal values, we have,

$$\frac{\partial v^k}{\partial s}(\xi) = \frac{\partial v^k}{\partial \xi}(\xi)\frac{\partial \xi}{\partial s}(\xi) \tag{110}$$

or, by (99), (105),

$$\frac{\partial v^k}{\partial s}(\xi) = \frac{1}{J^k(\xi)}\frac{dN_j^{m-1}}{d\xi}(\xi)\, V_j^k \tag{111}$$

On the same way, we get,

$$\begin{aligned}
\frac{\partial^2 v^k}{\partial s^2}(\xi) &= \frac{\partial}{\partial s}\{\frac{\partial v^k}{\partial s}(\xi)) = \frac{1}{J^k(\xi)}\frac{\partial}{\partial \xi}(\frac{1}{J^k(\xi)}\frac{\partial v^k}{\partial \xi}(\xi)\} \\
&= \frac{1}{J^k(\xi)}\{\frac{-1}{(J^k(\xi))^2}\frac{\partial J^k}{\partial \xi}(\xi) + \frac{1}{J^k(\xi)}\frac{\partial^2 v^k}{\partial \xi^2}(\xi)\} \\
\frac{\partial^2 v^k}{\partial s^2}(\xi) &= \frac{1}{(J^k(\xi))^2}\{\frac{d^2N_j^{m-1}}{d\xi^2}(\xi) - \frac{1}{J^k(\xi)}\frac{dN_j^{m-1}}{d\xi}(\xi) \times \\
&\quad \frac{d^2N_i^{m-1}}{d\xi^2}(\xi)[\cos\beta^k(\xi)x_i^k + \sin\beta^k(\xi)z_i^k]\}V_j^k
\end{aligned} \tag{112}$$

Now, applying (112) to x and z, $\frac{\partial\beta}{\partial s}$ in (22) is easily expressed over the k-th element as,

$$\frac{\partial \beta^k}{\partial s}(\xi) = \frac{1}{(J^k(\xi))^2}\frac{d^2N_i^{m-1}}{d\xi^2}(\xi)[\cos\beta^k(\xi)z_i^k - \sin\beta^k(\xi)x_i^k] \tag{113}$$

Therefore, various s-derivatives in (18)-(34) can be calculated from (111)-(113).

ii) Quasi-spline boundary elements :

- *Jacobian and normal vector :*

The Jacobian of the transformation from the *cartesian* quasi-spline element Γ_e^k, defined on the nodes $(k, k+1)$ of the free surface, to reference element $\Gamma_\xi \equiv \xi \in [-1, 1]$ is, by (103)-(105),

$$J^k(\xi) = \frac{\partial s^k}{\partial \tau}(\xi)\frac{\partial \tau}{\partial \xi}(\xi) \quad \text{with} \quad \frac{\partial \tau}{\partial \xi}(\xi) = \frac{1}{2} \tag{114}$$

Hence by (106),

$$J^k(\xi) = \frac{1}{2}[(\frac{dx_k}{d\tau}(\tau(\xi)))^2 + (\frac{dz_k}{d\tau}(\tau(\xi)))^2]^{\frac{1}{2}} \tag{115}$$

The derivatives with respect to τ of x and z are deduced from the sections of the two spline approximations (103) corresponding to the k-th quasi-spline boundary element of the free surface as,

$$\frac{d\mathbf{x}^k}{d\tau}(\tau(\xi)) = \mathbf{x}_{k+1} - \mathbf{x}_k - \frac{3A^2(\tau(\xi)) - 1}{6}\frac{d^2\mathbf{x}_k}{d\tau^2} + \frac{3B^2(\tau(\xi)) - 1}{6}\frac{d^2\mathbf{x}_{k+1}}{d\tau^2} \tag{116}$$

where A, B are given in (103), and τ as a function of ξ is given in (104). Notice nodal values have been written in their global numbering (subscript k).

The outward normal vector and $(\cos\beta^k, \sin\beta^k)$ inside the k-th element are again defined by (111), (108). Now, with (114), we get,

$$\mathbf{n}^k(\xi) = \frac{1}{2\,J^k(\xi)}[-\frac{dz^k}{d\tau}(\tau(\xi))\,,\ \frac{dx^k}{d\tau}(\tau(\xi))] \tag{117}$$

where $J^k(\xi)$ is given by (115), and the other terms by (116), (104).

- *s-derivatives :*

  Equations (110), (112) are valid which, with $m = 2$ and (114) gives,

$$\begin{aligned}
\frac{\partial v^k}{\partial s}(\xi) &= \frac{1}{J^k(\xi)} \frac{dN_j^1(\xi)}{d\xi} V_j^k \\
\frac{\partial^2 v^k}{\partial s^2}(\xi) &= -\frac{1}{4(J^k(\xi))^3} \frac{dN_j^1(\xi)}{d\xi} [\cos\beta^k(\tau(\xi)) \frac{d^2 x^k}{d\tau^2}(\tau(\xi)) \\
&\quad + \sin\beta^k(\tau(\xi)) \frac{d^2 z^k}{d\tau^2}(\tau(\xi))] V_j^k
\end{aligned} \tag{118}$$

  The derivatives with respect to τ of x and z are deduced from (116) as,

$$\frac{d^2 \mathbf{x}^k}{d\tau^2}(\tau(\xi)) = A(\tau(\xi)) \frac{d^2 \mathbf{x}_k}{d\tau^2} + B(\tau(\xi)) \frac{d^2 \mathbf{x}_{k+1}}{d\tau^2} \tag{119}$$

  where, again, A, B are given in (103), as well as τ as a function of ξ (104).

4.4.5 Discretized system of equations

The BIE (94), discretized element by element, using (99) or (103), becomes a sum of integrals over each Cartesian element, for each collocation node, $l = 1, \ldots, N_\Gamma$ (notice local numbering within elements is used at first). Boundary conditions $\overline{u}$ on Γ_d (Dirichlet boundary : total N_{Γ_d} nodes, M_{Γ_d} elements), and $\overline{\frac{\partial u}{\partial n}}$ on Γ_n (Neuman boundary : total N_{Γ_n} nodes, M_{Γ_n} elements) are also discretized for consistency, the same way as u and $\frac{\partial u}{\partial n}$. We get,

$$\begin{aligned}
\alpha(\mathbf{x}_l) U_l &+ \sum_{k=1}^{M_{\Gamma_n}} \{[\int_{\Gamma_e^k(\mathbf{x})} N_j^{m-1}(\mathbf{x}) \frac{\partial G}{\partial n}(\mathbf{x}, \mathbf{x}_l)\, d\Gamma(\mathbf{x})] U_j^k\} \\
&- \sum_{k=1}^{M_{\Gamma_d}} \{[\int_{\Gamma_e^k(\mathbf{x})} N_j^{m-1}(\mathbf{x}) G(\mathbf{x}, \mathbf{x}_l)\, d\Gamma(\mathbf{x})] \frac{\partial U_j^k}{\partial n}\} \\
&= \sum_{k=1}^{M_{\Gamma_n}} \{[\int_{\Gamma_e^k(\mathbf{x})} N_j^{m-1}(\mathbf{x}) G(\mathbf{x}, \mathbf{x}_l)\, d\Gamma(\mathbf{x})] \overline{\frac{\partial U_j^k}{\partial n}}\} \\
&- \sum_{k=1}^{M_{\Gamma_d}} \{[\int_{\Gamma_e^k(\mathbf{x})} N_j^{m-1}(\mathbf{x}) \frac{\partial G}{\partial n}(\mathbf{x}, \mathbf{x}_l)\, d\Gamma(\mathbf{x})] \overline{U_j^k}\}
\end{aligned}$$

Transformation of coordinates is then performed within each element, for calculating the integrals. Global numbering is adopted for the nodal values, as well as for the shape

functions, with s the global numbering for nodes on boundary section Γ_d, and p for nodes on Γ_n. We get,

$$\begin{aligned}
\alpha(\mathrm{x}_l)U_l \;+\; & \sum_{k=1}^{M_{\Gamma_n}}\{[\int_{\Gamma_\xi} N_p^{m-1}(\xi)\frac{\partial G}{\partial n}(\mathrm{x}^k(\xi),\mathrm{x}_l)J^k(\xi)\,d\xi]U_p\} \\
- & \sum_{k=1}^{M_{\Gamma_d}}\{[\int_{\Gamma_\xi} N_s^{m-1}(\xi)G(\mathrm{x}^k(\xi),\mathrm{x}_l)J^k(\xi)\,d\xi]\frac{\partial U_s}{\partial n}\} \\
= & \sum_{k=1}^{M_{\Gamma_n}}\{[\int_{\Gamma_\xi} N_p^{m-1}(\xi)G(\mathrm{x}^k(\xi),\mathrm{x}_l)J^k(\xi)\,d\xi]\overline{\frac{\partial U_p}{\partial n}}\} \\
- & \sum_{k=1}^{M_{\Gamma_d}}\{[\int_{\Gamma_\xi} N_s^{m-1}(\xi)\frac{\partial G}{\partial n}(\mathrm{x}^k(\xi),\mathrm{x}_l)J^k(\xi)\,d\xi]\overline{U_s}\}
\end{aligned}$$

in which J^k is the Jacobian of the transformation of element Γ_e^k into Γ_ξ defined in the previous section. In matrix form, this leads to a linear algebraic system of equations for the unknown fields u or $\frac{\partial u}{\partial n}$ on the boundary, in which each coefficient of the system matrix is one of the above sums of integrals. We get,

$$[c_{lp}+K_{n_{lp}}]U_p - K_{d_{ls}}\frac{\partial U_s}{\partial n} = K_{d_{lp}}\overline{\frac{\partial U_p}{\partial n}} - [c_{ls}+K_{n_{ls}}]\overline{U_s} \tag{120}$$

with the following definitions,

$$\begin{aligned}
K_{n_{lj}} &= \sum_{k=1}^{M_\Gamma}\{\int_{\Gamma_\xi} N_j^{m-1}(\xi)\frac{\partial G}{\partial n}(\mathrm{x}^k(\xi),\mathrm{x}_l)J^k(\xi)\,d\xi\} = \sum_{k=1}^{M_\Gamma} I_{n_{lj}}^k \\
K_{d_{lj}} &= \sum_{k=1}^{M_\Gamma}\{\int_{\Gamma_\xi} N_j^{m-1}(\xi)G(\mathrm{x}^k(\xi),\mathrm{x}_l)J^k(\xi)\,d\xi\} = \sum_{k=1}^{M_\Gamma} I_{d_{lj}}^k \\
& l,j = 1,\ldots,N_\Gamma\ ;\ s = 1,\ldots,N_{\Gamma_d}\ ;\ p = 1,\ldots,N_{\Gamma_n}\ ;\ N_\Gamma = N_{\Gamma_d}+N_{\Gamma_N}
\end{aligned} \tag{121}$$

in which j is now a global index representing s or p. This can also be written in condensed form as,

$$\mathcal{K}_{lj}\mathcal{X}_j = \mathcal{L}_l \tag{122}$$

in which $\mathcal{K}_{lj}$, $\mathcal{X}_j$, and $\mathcal{L}_l$ represent the system matrix, unknown and load vectors, respectively.

In (120), c_{lj} represents a diagonal matrix whose diagonal coefficients c_{ll} are equivalent to the geometric coefficients $\alpha(\mathbf{x}_l)$ in (94). For a smooth boundary, these diagonal coefficients have the values 1, 1/2 or 0 when $\mathbf{x}_l$ represents points inside the domain Ω ($\theta_l = 2\pi$ in (98)), on the boundary Γ ($\theta_l = \pi$), or outside Ω ($\theta_l = 0$), respectively. When there are discontinuities on the boundary (corners), coefficients c_{ll} must be calculated by a direct numerical evaluation of the interior angles θ_l.

In the present model, however, these coefficients are deduced by the "rigid mode" technique (Brebbia [2]) which does not require calculation of angles, and also leads to a somewhat more accurate solution of the final system. This corresponds to considering a particular Dirichlet problem in which a uniform field u is applied on the whole boundary $\Gamma \equiv \Gamma_d$. In such a case, the normal gradients $\frac{\partial u}{\partial n}$ must vanish at each node. Hence by (120),

$$[c_{lj} + K_{n_{lj}}]\overline{U}_j = 0 \tag{123}$$

or by isolating the diagonal terms of (123), we have for $j, l = 1, \ldots, N_\Gamma$,

$$[c_{ll} + K_{n_{ll}}] = -\sum_{j(\neq l)} K_{n_{lj}} \tag{124}$$

which provides the diagonal term of a row of (123), as minus the sum of its off-diagonal coefficients. The comparison of numerical results in which the c_{ll} were directly computed or deduced from (124) showed, in our case, a decrease of the system matrix condition number of more than one order of magnitude.

4.5 Numerical integration of matrix terms in the discretized BIE's

4.5.1 Principles

The regular integrals in (93), or its discretized form (120),(121), are computed by Gaussian quadrature using up to ten Gauss points per interval between the nodes, and a kernel transformation is applied to the singular integrals which are then computed by a Gauss-like quadrature dealing with the logarithmic singularity.

An adaptive numerical integration method is used for improving the accuracy of the regular integrations for very curved elements and for the elements close to the corners of the fluid domain. This method is based on a binary subdivision of the element to integrate, while keeping the number of integration points constant within each subdivision. Subdivision is performed until the intercept angle from which subdivisions are seen from the considered collocation point outside the element falls under a pre-set value. The same technique is also used when the distance between two boundaries of the fluid domain tends to vanish (e.g. during wave overturning or rundown on a slope). Almost arbitrary accuracy can be achieved in the numerical integrations using this method (i.e., about 15 significant digits in computer double precision).

4.5.2 Element by element numerical integration

Due to the higher-order shape functions and spline interpolation functions, the integrals in (121) cannot be calculated analytically within each element. When the collocation node l doesn't belong to the integrated element k, the integrals are regular and a standard Gauss-Legendre quadrature is used, with up to 10 integration points for each interval between two nodes.

When l does belong to the element k, r_l tends to zero at one of the nodes of the element, which leads to a weak logarithmic singularity of G, in the integrand of $I^k_{d_{lj}}$ (see (95)). Although this integral is not singular, large variations of the integrand occur for $I^k_{n_{lj}}$, when l belongs to an element k with high curvature (like in the crest of a wave approaching breaking). (it can be shown : $\frac{\partial G}{\partial n} \rightarrow \frac{1}{2\pi}\frac{\partial \beta}{\partial s}$ when $r_l \rightarrow 0$). This leads to a loss of accuracy in its regular integration.

Hence special techniques have been developed for computing both $I^k_{d_{lj}}$ and $I^k_{n_{lj}}$.

i) Singular integrals for $I^k_{d_{lj}}$: A kernel transformation developed for higher-order elements is applied to the weakly singular $I^k_{d_{lj}}$ which are then integrated by a numerical quadrature exact for the logarithmic singularity. In the intervals where $r_l \rightarrow 0$, we extract

the singularity by adding and subtracting $\log|\xi - \xi_l|$. For each of the integrals $I^k_{d_{lj}}$ written in short as,

$$I^k_{d_{lj}} = \int_{\Gamma_\epsilon} G(\mathbf{x}^k(\xi), \mathbf{x}_l) f^k_j(\xi)\, d\xi \ , \ f^k_j(\xi) = N^{m-1}_j(\xi) J^k(\xi)$$

we get after some transformations,

$$\begin{aligned} I^k_{d_{lj}} \ = \ & -\frac{1}{2\pi}\int_{-1}^{+1} [\log \frac{2r_l(\xi')}{|\xi' - \xi_l|} f^k_j(\xi') + \xi_{p1} \log \xi_{p1} f^k_j(\xi_{p1}\xi' - \xi_{p2}) \\ & + \xi_{p2} \log \xi_{p2} f^k_j(\xi_{p2}\xi' + \xi_{p1})]\, d\xi' \\ & + \frac{1}{\pi}\int_0^1 [\xi_{p1} f^k_j(\xi_l - 2\xi_{p1}\xi') + \xi_{p2} f^k_j(\xi_l + 2\xi_{p2}\xi')] \log \frac{1}{\xi'}\, d\xi' \end{aligned} \tag{125}$$

in which ξ_l is given by (100), in case of a m-node element, and $\xi_{p1} = \frac{\xi_l + 1}{2}, \xi_{p2} = \frac{\xi_l - 1}{2}$. It can easily be shown, that the first integral in (125) is not singular. Hence a Gauss-Legendre quadrature formula is used. The second integral in (125) is weakly singular and is integrated by the Berthod-Zaborowsky quadrature formula (Brebbia [2]), which provides the same error properties for the weakly singular logarithmic kernel as the Gauss-Legendre formula does for non-singular integrals.

ii) Improved integrals for $I^k_{n_{lj}}$: When $r_l \to 0$, although there is no singularity, integrations of $I^k_{n_{lj}}$ are improved by performing a change of variable and an analytical integration by part, before using the numerical quadrature. These both result in the somewhat smoothing out of the large variations of the integrand over the element. The change of variable is similar to the one performed by Longuet-Higgins & Cokelet [52],

$$\mu^k_l(\xi) = \arctan \frac{z^k(\xi) - z_l}{x^k(\xi) - x_l} \tag{126}$$

and is followed by an analytic integration by parts, which makes it possible to avoid any numerical s-derivation of the integrand (unlike in [52]). We find, for an m-node element, that,

$$I^k_{n_{lj}} = \int_{\Gamma_\epsilon} \frac{\partial G}{\partial n}(\mathbf{x}^k(\xi), \mathbf{x}_l) N^{m-1}_j(\xi) J^k(\xi)\, d\xi$$

becomes,

$$I^k_{n_{lj}} = \frac{1}{2\pi}[\mu^k_l(-1)\delta_{1j} - \mu^k_l(1)\delta_{mj} + \int_{-1}^{1} \frac{dN^{m-1}_j}{d\xi}(\xi)\mu^k_l(\xi)\, d\xi] \tag{127}$$

Since μ_l^k can be singular when $x^k(\xi) = x_l$, the formulae (126) and (127) are only valid for elements in which the "element slope" $\mid \mu_1(1) \mid$ (i.e., slope of a straight line from node 1 to node m) is less than 45°. If this is not the case, numerator and denominator must simply be permuted in the definition (126) of μ_l, and the right hand side of (127) multiplied by -1. The integral in (127) is regular and again calculated by the Gauss-Legendre quadrature formula. Notice, terms such as (127) are zero for straight line elements (since in (95), $\mathbf{r} \cdot \mathbf{n} = 0$ in $\frac{\partial G}{\partial n}$), and very small for gently curved ones.

iii) General procedure : N_{ip} integration points are used in each element, and N_{ip} is chosen to be even in order to avoid having integration points at $\xi = 0$. In the examples shown hereafter, up to 10 integration points are used per interval between 2 nodes. In regions of Γ_f with high curvature and concentration of nodes, however, this may not be sufficient, mainly because of the rapid variation of $\frac{\partial G}{\partial n}$ in the very curved elements and of the Jacobian within the elements when nodes are getting close to each other. The adaptive integration described in the next section is used for these situations.

4.5.3 Adaptive integration

An adaptive numerical integration method is developed and used for improving the accuracy of regular integrations made with respect to a collocation node $\mathbf{x}_l$, not belonging to the considered element k, from which the element is seen with too large an intercept angle (say $\alpha_l > \alpha_{max}$). This indeed leads to large variations of the integral kernels over the element which cannot quite be caught by regularly spaced Gauss points ($\frac{\partial G}{\partial n}$ being, again, the most sensitive term to this). Large intercept angles α_l occur in the discretization for the elements close to the corners and also when the distance between two boundaries tends to vanish (e.g. through time updating of the fluid domain geometry during wave motion on a slope). In general also large angles occur when the discretization mesh varies quite a lot from one part of the boundary to another one (e.g. due to high ratio length over height of the fluid domain, or due to the concentration of fluid particles-collocation nodes in some region of the flow).

The adaptive integration performs ns binary subdivisions of the element k into segments within which the number of integration points (GP) is kept constant. The subdivision procedure divides the reference element geometry ($\Gamma_\xi \equiv \xi \in [-1,+1]$) into 2^{ns} equal segments of length 2^{1-ns}, until the intercept angle α_{li} of segment i seen from the collocation point $\mathbf{x}_l$ in the actual geometry, becomes smaller than a preset angle α_{max} (i.e. $\alpha_{li} < \alpha_{max}$; $i = 1, ..., 2^{ns}$). Then, each segment is itself mapped onto the interval $[-1,+1]$. Both types of integral over an element k in (121), say I^k for $I^k_{n_{lj}}$ or $I^k_{d_{lj}}$, can be written as,

$$I^k \equiv \int_{\Gamma_\xi} \mathcal{F}_k(\xi)\, d\xi \;=\; \frac{1}{2^{ns}} \sum_{i=1}^{2^{ns}} \int_{-1}^{+1} \mathcal{F}_k(\xi_i(\gamma))\, d\gamma$$

$$\xi_i(\gamma) = \frac{\gamma}{2^{ns}} + \sum_{b=1}^{ns} \frac{(-1)^{1+B(i,b)}}{2^{ns-b+1}} \;; \qquad B(i,b) = \mathrm{INT}[\frac{i-1}{2^{ns-b}}] \tag{128}$$

where $\mathcal{F}_k(\xi)$ represents the product of G (or $\frac{\partial G}{\partial n}$), a shape function and the Jacobian J^k. Integrals in (128) are computed by a regular Gauss quadrature, with respect to the variable γ. Almost arbitrary accuracy can be achieved in the integrations provided ns is chosen large enough (i.e., about 15 significant digits in double precision). To reduce the computation time, however, the number of successive binary subdivisions is limited to $ns = 4$ in the applications (i.e. 16 segments), and, based on our computing experience, α_{max} is selected equal to 40°. Notice, for $ns = 0$, the integration formula (128) reduces to one segment of length 2, which corresponds to the usual regular integral over Γ_ξ.

The adaptive integration (128) is computationally quite efficient over one element, for a given ns, with respect to a given $\mathbf{x}_l$. The selection of the number of subdivisions ns which satisfies the criterion on the intercept angle, for all the elements k, with respect to all the N_Γ collocation points l, however, is computationally expensive. It requires $(2^{ns})!$ computations of angles α_{li} for each couple of values (k,l) (to be compared with α_{max}), which themselves require to perform the change of variable from the reference element Γ_ξ to the actual geometry Γ^k_e.

It is therefore necessary to "a priori" restrict these operations to a number of pairs (k, l) in the computational data. In general, the 8 elements defining the 4 domain corners are selected, and also the elements on parts of the boundary discretization which, one anticipates, will become close to each other in the following time steps (e.g. tip of a plunging breaker, wave running down on a slope,...). These selections can be and are, of course, interactively modified during the computations when the domain geometry changes through the time evolution. Doing so, the extra computational effort of performing adaptive integrations is, in general, reduced to a few percents of the computation time per time step, used otherwise without them.

4.6 Sliding element for s-derivatives

The representations of the field variables inside each element (isoparametric (99) and quasi-spline (103)) only provide interelement continuity of the fields themselves, not of their derivatives. This is also true for the geometry and for the normal vector, in case of isoparametric elements. In case of quasi-splines, both the slope of the free surface and the normal vector are continuous, but not the gradients of the field functions, which are based on a first order shape function only.

The s-derivatives along the free surface are calculated in a special element providing local continuity of at least the 2nd order derivatives. The special element is a 4th-order (5 node) isoparametric element, superimposed on the nodes of the discretization, but independent of the BEM interpolation functions, whose five nodes are mapped, onto the reference element $\xi \in [-1, 1]$ by the Jacobian. Direction cosines and derivatives are then computed at the central node of this element, corresponding on the free surface to, say, node l. The whole element is then moved forward by one node, to have its central node at next node $l + 1$, before calculating new derivatives (hence this process is named "sliding" derivation or "sliding" element).

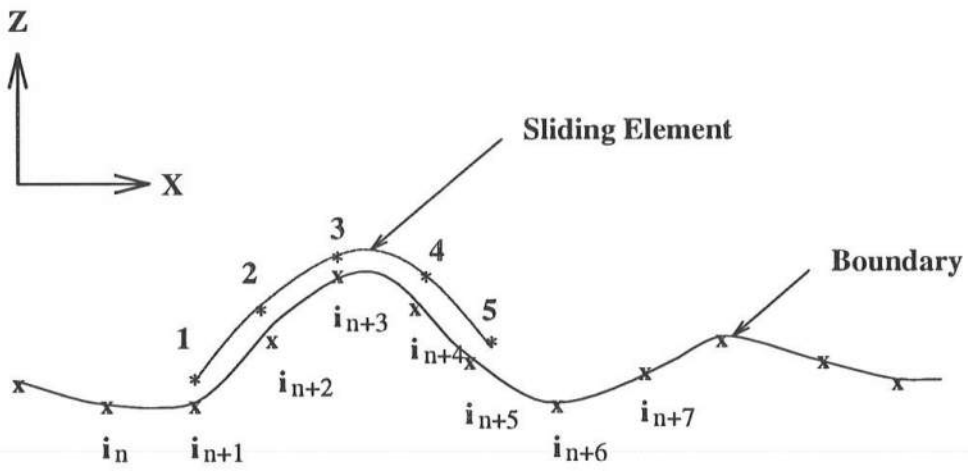


Figure 5: Sketch for the sliding element on the boundary.

Notice that due to corners at both extremities of the free surface, the sliding element remains in the same place, for calculating derivatives at the three first or at the three last nodes of Γ_f. When a space periodic problem is solved, however, periodicity conditions are used to make the sliding process continuous.

4.7 Automatic grid refinement on a slope

It is observed that, during wave motion on a slope (runup-rundown), the size of the last element on the free surface may become much smaller than the size of the first neighboring element on the slope. This leads to somewhat less accurate integrations close to the surface corner, even with the adaptive integration procedure. To improve the accuracy of the integrations, the discretization on the slope is stretched, as suggested by Klopman [47] 1988, according to an exponential law which imposes the length of the upper element on the wall (closest to the free surface) to be the same at all time steps as the length of the last element on the free surface. The other elements on the wall are, accordingly, becoming wider towards the bottom. This automatic grid refinement, associated with the adaptive integration, increase the accuracy of the numerical solution in the corner by several orders of magnitude.

4.8 Corner problems

4.8.1 Mathematical problem

When waves are generated by a wavemaker, there is a corner on the boundary, at the intersection between the wavemaker and the free surface. The same situation also occurs at the intersection of the free surface with other surface piercing structures, like fixed slopes. The flow near the intersection with a moving solid body has given rise to substantial concern in the literature. This was reviewed by Grilli & Svendsen [37], in the particular context of wavemakers starting from a state of rest ("cold start"). The principal conclusion of the review is, provided the initial acceleration of the wavemaker is small with respect to gravity, there will be no strong singularity at the free surface corner.

In the model, the wavemaker cold start is specified in such a way that the acceleration remains small during the first few time steps of the computations. Doing so, no singular behavior or instability of the solution is observed at the corner.

4.8.2 Numerical problem

Well-posedness of governing equations and boundary conditions must be ensured at corners. On the free surface, both ϕ (or $\frac{\partial \phi}{\partial t}$) and $\frac{\partial \phi}{\partial n}$ (or $\frac{\partial^2 \phi}{\partial t \partial n}$) are specified on the "body side" (e.g., wavemaker, slope) of corners, whereas there is a different normal gradient, to be calculated as part of the global solution, on the free surface side of corners. Double nodes are used in the model to represent corners, for which coordinates of both nodes are identical, but normal vectors differ. This makes it possible to express that each node belongs to a different part of the boundary, with different boundary conditions. Hence, $\frac{\partial \phi}{\partial n}$ and $\frac{\partial^2 \phi}{\partial t \partial n}$ are explicitly different for both nodes of a corner, and two algebraic expressions of the BIE's (93) are derived for each corner. As an additional constrain, continuity of ϕ (or $\frac{\partial \phi}{\partial t}$) is imposed explicitly in these algebraic expressions.

In addition, natural kinematic or geometric relationships between ϕ's and $\frac{\partial \phi}{\partial n}$'s (or, $\frac{\partial \phi}{\partial t}$'s

and $\frac{\partial^2 \phi}{\partial t \partial n}$'s) must be satisfied at corner double nodes— these are referred to as compatibility relationships—, like uniqueness of the velocity vector, or a horizontal free surface tangent at a fixed vertical solid boundary. Although these conditions should automatically be satisfied by the numerical solution, it is not in general the case due to numerical errors. Because there is no damping in the model, such errors may add up through time updating, and lead to instability of the corner solution. Hence, errors are reduced by explicitly imposing compatibility relationships to the solution, after each time step.

Details of the numerical treatments of corners can be found in Grilli & Svendsen [37].

4.9 Automatic selection of optimum time step

Accuracy of the computations is checked by computing the change in volume and total energy of the computational domain at each time step. Errors are function of both the resolution and the degree of the elements of the BEM discretization, and of the size of the time step.

Grilli & Svendsen [37] used the model to compute propagation over constant depth h of *exact* solitary waves, in a series of spatio-temporal discretizations. For these waves, volume, total energy and speed of propagation can be calculated as a function of height, with an accuracy of at least eight significant figures, by equations (38).

Results showed, for a given initial distance between nodes on the free surface Δx_o, and for a constant time step Δt_o, numerical errors are proportional to $\Delta t_o'^3$ when the mesh Courant number $\mathcal{C}_o = \frac{\Delta t_o'}{\Delta x_o'} \geq 0.5$ (with $\Delta x_o' = \frac{\Delta x_o}{h}$, and $\Delta t' = \Delta t \sqrt{\frac{g}{h}}$). This is consistent with the second-order accuracy of the Taylor series (9) and (10) used in the time integration. When $\mathcal{C}_o < 0.5$, errors cannot be further reduced by decreasing $\Delta t_o'$, and become proportional to $\Delta x_o'$ only. This corresponds to the effect of the spatial discretization on the accuracy of the BEM solution.

For waves of rapidly changing shape, due to the Lagrangian time updating of free surface nodes, the distance between nodes on the free surface may significantly change at

every time step. In fact, in some configurations of the flow, as in the jet of breaking waves or during wave runup-rundown on a slope, for instance, this distance may decrease considerably. Unless the time step is adjusted accordingly, any pre-set criterion on the value of the Courant number, based on the initial distance between nodes will be violated, and accuracy will rapidly deteriorate.

Hence, a varying time step procedure is introduced in the computations. At initial time t'_o, the initial Courant number C_o is selected equal or around 0.5 and, for any subsequent time t', time step is adjusted as,

$$\Delta t' = \Delta \mid \mathbf{r}' \mid^{min} C_o \tag{129}$$

where $\Delta \mid \mathbf{r}' \mid^{min}$ represents the minimum nondimensional distance between two nodes on the free surface.

In typical applications, maximum relative errors on volume and total energy of a wave propagating without change of form are on the order of 0.01%, even after 1000 time steps of propagation. For strongly unsteady waves and, particularly, for waves close to breaking, errors may be larger. This is because discretization nodes gather at some areas of the boundary where hydrodynamic jets are forming (e.g., crest of an impending breaking wave), and scatter at some other areas (e.g., wave troughs), leading to a less accurate description of the flow. Computations are generally interrupted when errors become greater than 0.50%.

5 Computer Program

5.1 Introduction

A computer software in FORTRAN 77 was written, based on the equations of the mathematical and numerical model presented in sections 2,3, and 4.

This software has three main sub-programs, performing pre-processing, processing (the model itself), and post-processing tasks.

- *Pre-processing* programs are stored in module RADIAP.

  The program GENER helps generating geometric data for simple domain geometry, like a constant depth with a slope.

  The program SOLWAVE generates initial data for exact solitary waves, based on Tanaka's [74] (1986) method (see section 3.1).

- The *processing* program RADIA, the model itself, contains 64 subroutines and functions which, for convenience are split up between four main modules (RADIA0, RADIA1, RADIA2, RADIA3), with each module containing several subroutines.

  Most arrays in these subroutines are declared with variable dimensions or are specified in COMMON statements. A main routine MRADIA declares parameters, performs fixed dimensioning of arrays and COMMON statements, and calls the subroutine RADIA. Hence, changing the maximum size of problems that can be solved with the model just requires modifying parameters in MRADIA and recompiling it.

  All modules of RADIA are to be compiled independently, and linked into one executable file. Subroutines performing time consuming tasks have been written in such a way, automatic vectorization should occur when the code is compiled with a compiler and on a computer featuring such characteristics (e.g., FORTVS2 on the IBM-3090 or -9000; use options VEC and OPT(3)).

- *Post-processing* programs are also stored in module RADIAP.

  The program CURMUL interactively creates data for plotting detailed results on the boundary, and global results, as curves, under format of the commercial graphic software TELEGRAF.

  The program IFIELD interactively creates data for plotting results inside the domain (internal field, velocity and pressure) under TELEGRAF format.

  The program PRESHYD calculates equivalent hydrostatic pressure under a wave running up a slope, and creates data for plotting them as curves, under TELEGRAF format.

  The program DI3000 creates contour levels of pressure inside the domain, based on subroutines from the commercial graphic library DI3000.

  If TELEGRAF or DI3000 are not available on the system used, the generated data file should be modified and used with some other graphic package.

The program code should be portable on any computer featuring a standard FORTRAN 77 compiler, except for one machine dependent subroutine FILEINF (IBM-VM system), which sets up size parameters of direct access files used for storing detailed results of computations, for postprocessing purpose. This routine is used three times in SAVE (subroutine in module RADIA0), two times in CURMUL, and once in IFIELD and in PRESHYD, and should be replaced by the relevant routine for the specific operating system used. The use of FILEINF is illustrated below,

```
      CALL FILEINF(IFRC,'MAXREC',ILMAX*(ISNIN+NOM))
C     ------------
      CFILE=NFILES(1)//MDISK
      OPEN(UNIT=JCURVE,STATUS='NEW',ACCESS='DIRECT',FORM=
     .     'UNFORMATTED',RECL=12*MOT,FILE=CFILE)
```

In this example, the maximum number of records MAXREC of file CFILE (opened immediately after) is specified within the program as ILMAX*(ISNIN+NOM), which varies with the size of the problem. Notice, the format of file name CFILE is also based on the IBM-VM system, with three operands : filename (arbitrary), filetype (e.g., data, exec,...), filemode (i.e., disk code). This format should also be adapted to the specific operating system used.

Definition of the 50 or so sequential input/output files used in the model has also been written for the IBM-VM environment, using .EXEC files. These files should be re-defined for the specific system on which the program will be run. This, however, should not pose problem.

Computer files, with complete source code, user's manual, and applications are available on request through the internet computer network (send Email to : grilli@mistral.oce.uri.edu, for inquiry).

5.2 Overview of the computer model

The wave model is implemented as the main processing program RADIA mentioned above. In this program, the subroutine RADIA (called from the main program) performs the main stages of model computation by calling a series of twelve specific subroutines. The flowchart in Fig. 6 corresponds to these subroutines, as successively called in RADIA, and brief descriptions of the subroutines are presented in subsequent sections. A general overview of the computations in RADIA is given in the following,

- The input of problem data specified by the user is performed in INPUTD. Data are read and checked, from a sequential data file whose *filename* and *filetype* have been supplied to the program.

- Initialization of domain parameters is performed in subroutine INTIAL, which in turn utilizes subroutines SLIDING, FSVELO, BEMK to define initial values for the sliding

derivation on the free surface (section 4.6), the double nodes at the corners (section 4.7), and the boundary element matrices, respectively (section 4.4.5).

At this stage, values for the geometry and for (ϕ, $\frac{\partial \phi}{\partial n}$) have been specified on the free surface, either from initial conditions (i.e., "cold start", or specified wave), or from previous computations.

- Values of u, w, and $\frac{\partial^2 \phi}{\partial s^2}$ are computed in subroutine DUDTPR over the moving boundaries (in general the free surface) as a function of $s-$ and $n-$derivatives of potential ϕ. A prediction of $\frac{D u}{D t}$ and $\frac{D w}{D t}$ is also made on lateral moving boundaries (radiation conditions) (sections 4.2. and 4.4.2).

  This represents the first step in a series of routines used in a loop over time, up to *tmax*, or to a maximum number of iterations (loops) *lomax*.

- Predicted pressure is computed on lateral Dirichlet boundaries in subroutine PREFIX, when specified, and the value of $\frac{\partial \phi}{\partial t}$ is calculated on all Dirichlet boundaries (including the free surface).

- Control is then passed for a first time to RESOL, in which boundary conditions are updated for Laplace's equation for $\frac{\partial \phi}{\partial t}$ (section 3 for wave generation), and the BEM analysis is performed, out of which either $\frac{\partial \phi}{\partial t}$ or $\frac{\partial^2 \phi}{\partial t \partial n}$ are computed on the boundary, whichever is unknown (sections 4.4., 4.5).

- The subroutine D2UTPR computes updated value for $\frac{D u}{D t}$ and $\frac{D w}{D t}$, using the full Taylor series (equation (27)), and estimates the value of $\frac{D^2 u}{D t^2}$ and $\frac{D^2 w}{D t^2}$, when needed for a moving lateral boundary (radiation boundary).

  In this case, predictor-corrector iterations are performed, and this estimate is refined until the error is less than a preset value *tolmax*, or the index *pc* is equal to a maximum preset value *pcmax*.

- SAVE post-processes results of the computation, and stores them for postprocessing. When there are internal field computations, INTERN computes internal field variables.

- DPRFIX computes temporal gradient of pressure on lateral Dirichlet boundaries, when required.

- WRRES prints results of computations when the listing option is selected.

- UPDTDB performs the final updating of the geometry and potential of moving Dirichlet boundaries to time, $t + \Delta t$ (in general the free surface), using the full solution at time t and equations (87) and (88).

- The time increment based on the Courant number is determined by DTSTEP.

- Control is then passed for a second time to RESOL, in which geometry of lateral boundary and boundary conditions are now updated for Laplace's equation for ϕ (section 3 for wave generation), and the BEM analysis is performed, out of which either ϕ or $\frac{\partial \phi}{\partial n}$ are computed on the boundary, whichever is unknown (sections 4.4., 4.5).

  This terminates the time loop, and operations are repeated from this point onward.

The detailed flowchart for RESOL is given in Fig. 7. As mentioned above, this subroutine is called twice, once for solving each of the Laplace's equations for ϕ (*iflag*=2), or for $\frac{\partial \phi}{\partial t}$ (*iflag*=1), whose solutions are needed for performing time and geometry updating. Boundary conditions are set-up on lateral and bottom boundaries, and moving lateral boundary geometry is also updated (only prior to solving for $\frac{\partial \phi}{\partial t}$). Details of the flowchart for RESOL are discussed in the following,

- Lateral boundary conditions are updated in UPLABC.

- If *flagv* is set to 1, the bottom boundary conditions are updated in UPBOTC, and new s-derivatives are computed in SLIDING.

Otherwise, if *flagv* is set to 2, the control is passed directly to the next step.

- FSVELO imposes compatibility conditions at double nodes on the free surface (section 4.8.2).

- If *flags* is set to 1, the geometry has changed, and the BEM analysis is performed in BEMK, out of which, new matrices are created.

- Internal source strengths are computed in SOUSTE, SOURCE, if *sourc* is assigned a value of 1 (section 3.3).

- The final load vector is assembled next in ASSEMP (section 4.4.5).

- Double node compatibility is imposed on the load vector in DBNODP.

- The final system matrix is then solved by Kaletsky's elimination method, in SOLVE. Notice, the system matrix is kept in eliminated form from the solution for $\frac{\partial \phi}{\partial t}$ (*iflag*=1), and used in the solution for ϕ (*iflag*=2), for which only the load vector is eliminated.

- And the solution is sorted by type, variable or its normal gradient, in SORT.

Flowchart for the boundary element analysis in BEMK is given in Fig. 8. Details of the flowchart for BEMK are discussed in the following,

- If *isplin(k)* is set to 1 for boundary section k, a parametric spline analysis of boundary section k is performed in SPLANA, for later use in setting up parameters of two-node quasi spline elements (section 4.4.3).

  The spline analysis is not required for BEM analysis of isoparametric elements.

- Computations then enter a loop over each of *NELEM* elements on the boundary.

- GAUSSP computes regular Gauss points and weights for numerical integration.

- Shape functions and their first and second derivative interpolation functions are then computed for all the Gauss points in FUNF1, DFUNF1, D2FUN1.

- Once the interpolation functions are computed, a decision is made on whether adaptive integration is required, on the basis of the intercept angle on the element. Flag *nsubm* is set to 0 if no adaptive integration is required.

- For *nsubm*=0, BIMAT computes local k_d and k_n matrices (section 4.4.5), with both regular and singular integrations (as in section 4.5.2), for the elements, and saves intrinsic and geometric data for later post-processing of results in SAVE.

- When *nsubm*=1, adaptive integration is required, and BISMAT performs these integrations on the element (as in section 4.5.3), in addition to computations done in BIMAT.

- ASSEML does matrix assembling of local k_d and k_n, into global K_d and K_n.

- Once the above computations are performed on all elements, the rigid mode technique is implemented by INTRCI (section 4.4.5).

- Final assembling of the system matrix K is done in ASSEMK (section 4.4.5).

- Double node compatibility conditions are finally imposed to the system matrix in DBNODK. This completes the BEM analysis.

5.3 Preprocessing and generation of input data

Preprocessing essentially consists in defining,

i) the computational domain geometry and discretization, by generating a set of nodes on the boundary of the physical domain, and specifying boundary elements, to interpolate between the nodes.

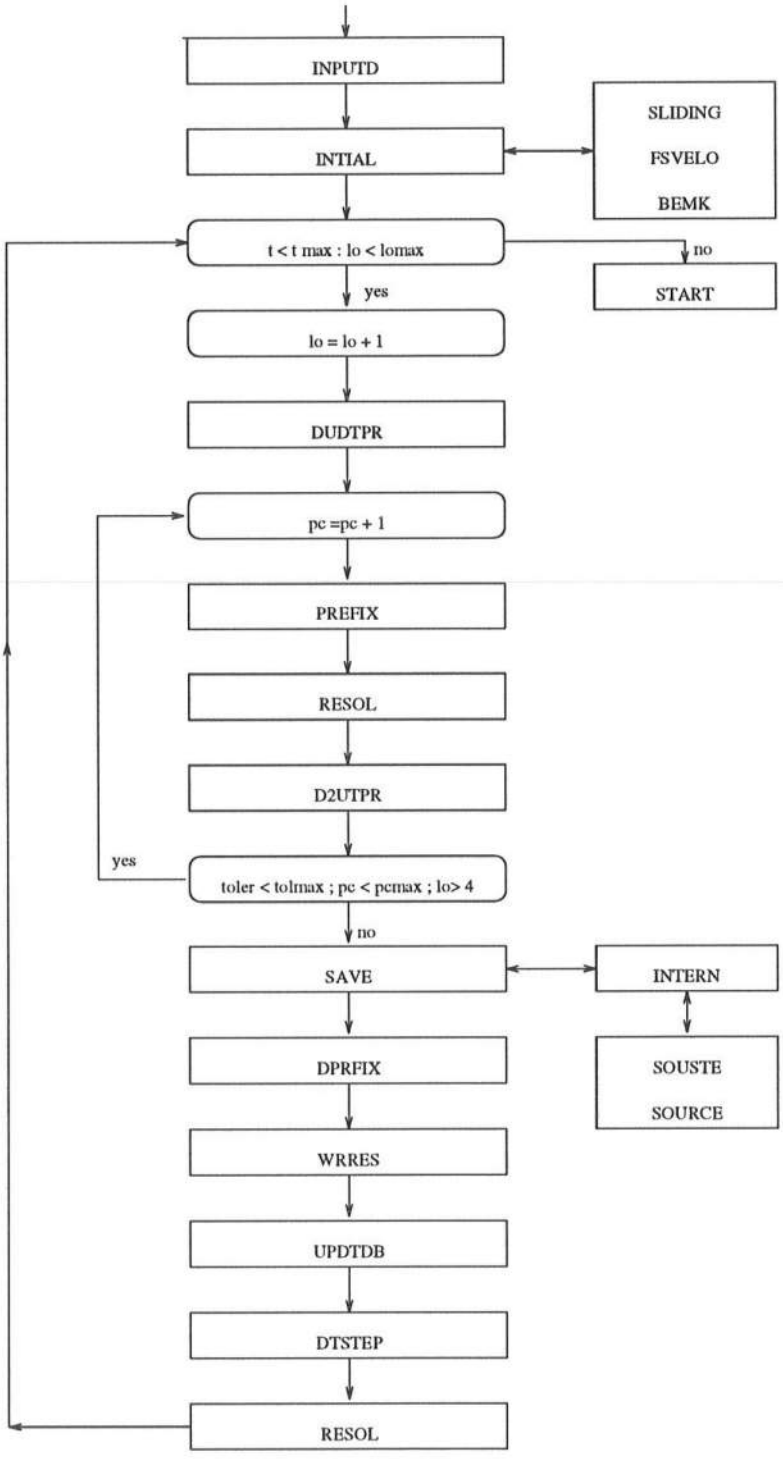


Figure 6: Flow chart for program RADIA

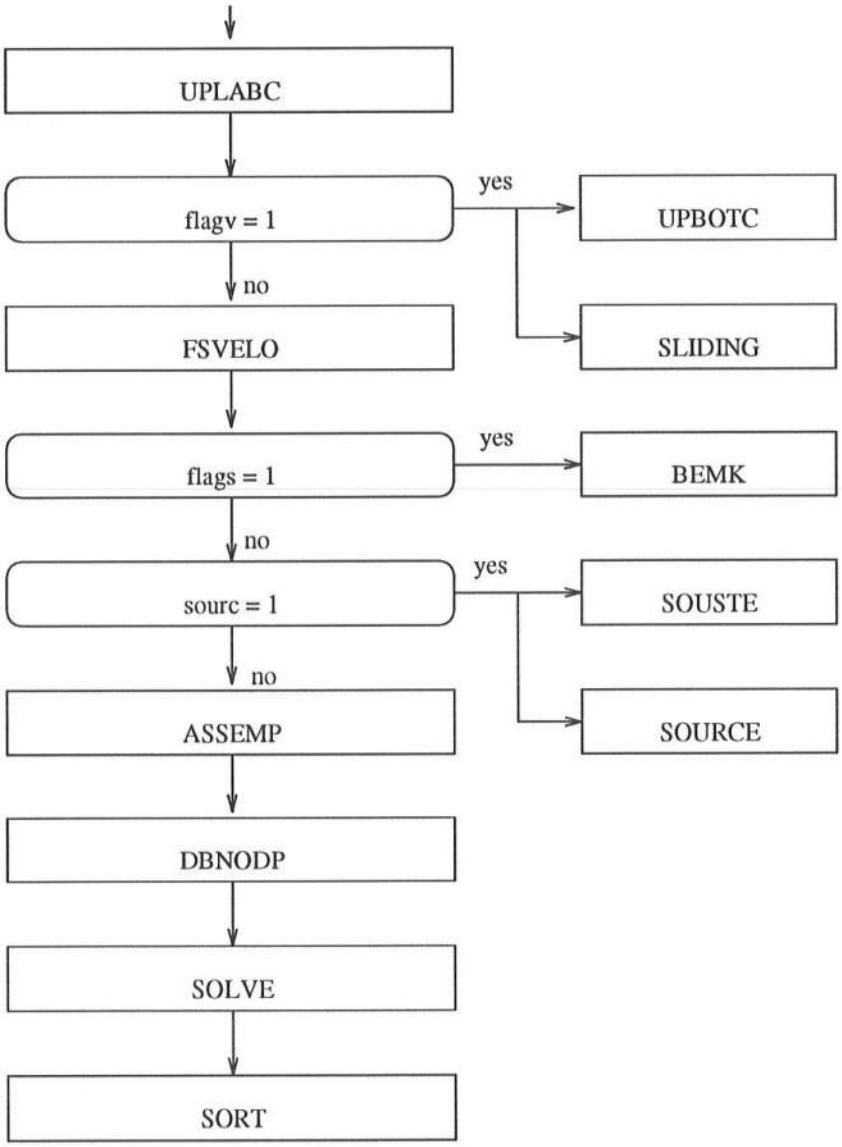


Figure 7: Flow chart for program RESOL

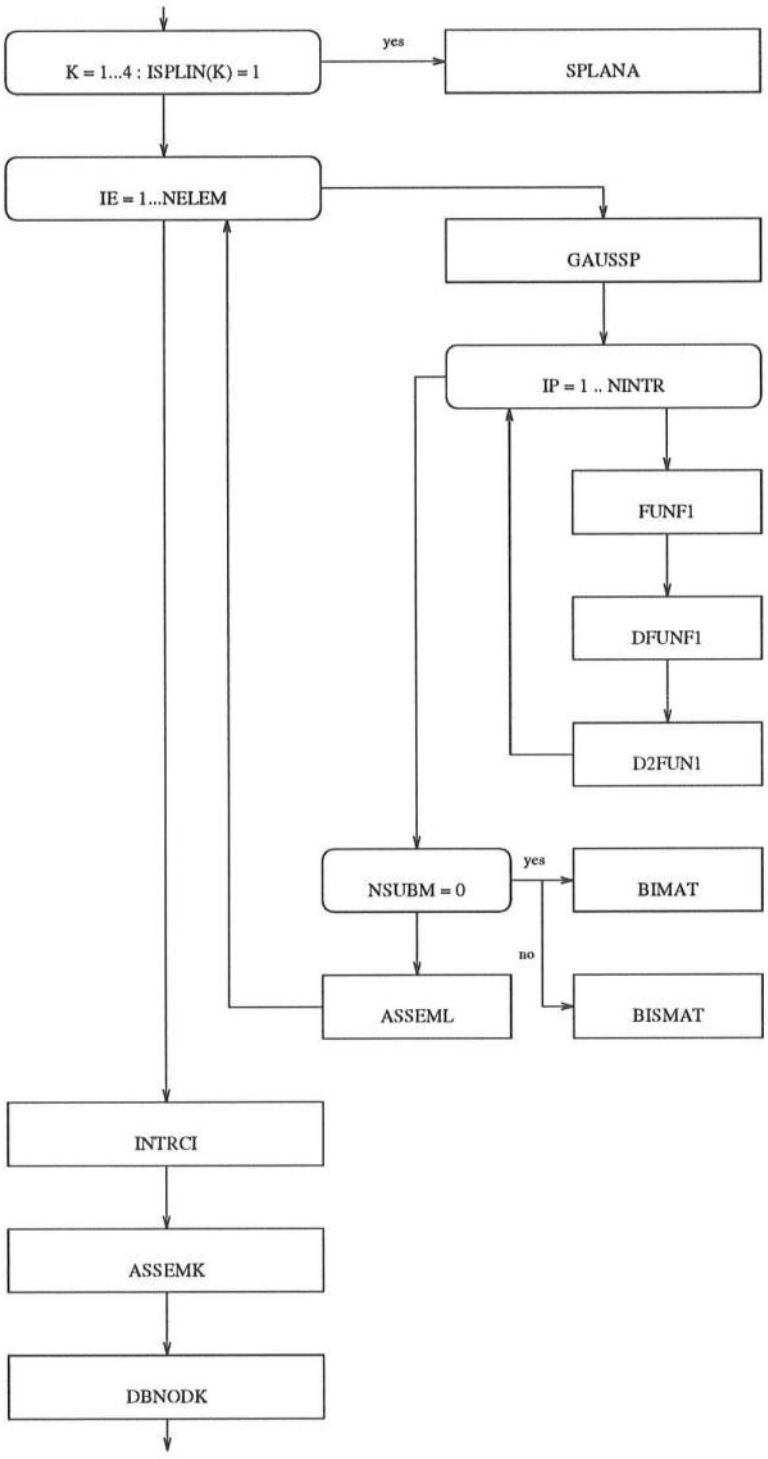


Figure 8: Flow chart for program BEMK

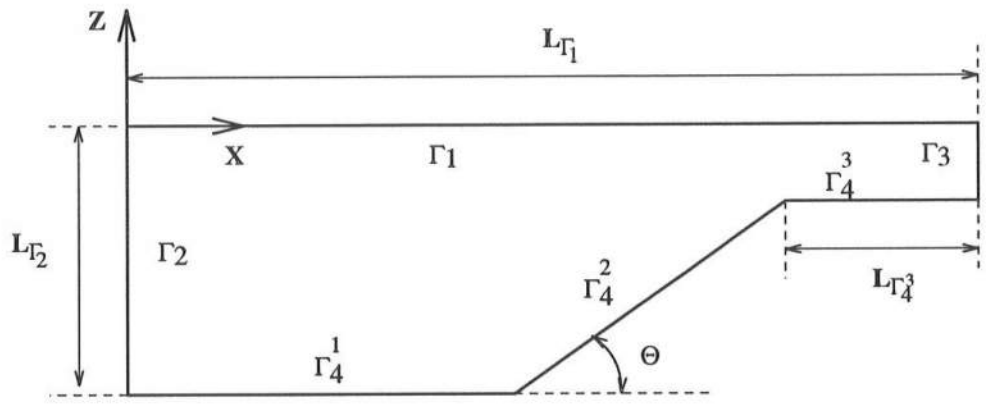


Figure 9: Definition of geometry and parameters for pre-processing program GENER

ii) the type of problem to be solved with the model, by assigning values to various parameters in an input file.

5.3.1 Generation of domain geometry

For simple domain geometry, the first task (i) is performed with the help of the pre-processing data generation program GENER, contained in the module RADIAP.

GENER assumes the domain to have the simple bottom geometry depicted in Fig. 2 or 9, with a flat initial free surface (length L_{Γ_1}), a constant depth area (length $L_{\Gamma_1} - L_{\Gamma_4^3} - s\,(L_{\Gamma_2} - L_{\Gamma_1})$; depth L_{Γ_2}), in front of a slope s of angle θ, and a small shelf at the upper part of the slope (length $L_{\Gamma_4^3}$; depth L_{Γ_3}) (see Fig. 9 for definitions).

Input data for GENER are provided as four lines of freely formatted data in a sequential file (*fname.data*),

$l1:$ M_{Γ_1} M_{Γ_2} M_{Γ_3} $M_{\Gamma_4^1}$ $M_{\Gamma_4^2}$ $M_{\Gamma_4^3}$

$l2:$ m_{Γ_1} m_{Γ_2} m_{Γ_3} $m_{\Gamma_4^1}$ $m_{\Gamma_4^2}$ $m_{\Gamma_4^3}$

$l3:$ L_{Γ_1} L_{Γ_2} $L_{\Gamma_4^3}$ Θ

$l4:$ *nintm* *nsub*

in which M_Γ denotes the number of elements on the specified boundary segment (Fig. 9), m_Γ is the type of element (i.e., number of nodes per element, 2,3,...), L_Γ is the length of the

particular boundary segment, θ is the angle of the slope with the bottom in degrees, *nintm* is the maximum number of Gauss points per element (normally 10), and *nsub* is the maximum number of subdivision for the adaptive integration (see example in section 6).

GENER generates the node coordinates, defines the boundary elements connectivity and parameter matrix, and writes this data in a sequential output file : *gener.data* (see example in section 6). Four lines of input parameters must be added at the top of this file, to fully define a problem (see next section). Of these parameters, GENER provides the first two : *nom, nelem.*

For the generation of exact initial solitary waves, the program SOLWAVE, also in RADIAP, can be run to provide initial free surface geometry and kinematics for a solitary wave of specified height H/h, with its crest located at an initial specified location x_o/h. Results from SOLWAVE are provided at specified boundary element node coordinates, under the format used in the input data file for the model, and can thus directly be substituted into the file *gener.data* previously created by GENER, with a flat, non-moving free surface.

For other types of boundary geometry, the user must create its own mesh data file, under the format described in the following section. Notice, in many cases, it is possible to use GENER for generating part of the required mesh, and to edit and modify the generated file for the actual bottom geometry (e.g., an obstacle on the bottom can easily be added this way).

5.3.2 Input parameters

The second task (ii) is performed by defining a sequential input file for the problem, that will be read in subroutine INPUTD. This subroutine reads user defined parameters from a specified sequential input file, in the following format,

The following is a general description of user-defined input parameters listed in the first 4 lines in the following table,

nom : Total number of nodes of the boundary element mesh

INPUTD		
Line	Format	Parameters
1	(10I5,F10.0)	*nom, nelem, isave, lomax, pcmax, iptyp, nbs, iprint, init, ismoth, tolmax*
2	(8I5,2F10.0)	*ibcond(1), isplin(1), ibcond(2), isplin(2), ibcond(3), isplin(3), ibcond(4), isplin(4), almaxb, almaxi*
3	(7F10.0)	*dt, tstart, tmax, rho, cpress, cprest, ge*
4	(5F10.0,5I5)	*tdamp, de1, de2, omega, voh/ulat, isourc, nos, ifield, noi, nht*
5-(4+*nom*) (nodes)	(4F20.0)	*(x(i), z(i), phi(i), phin(i), i=1,...,nom)*
(5+*nom*)-(4+*nom*+*nelem*) (elements)	(5I5, 4I5)	*(node(i,j), j=1,..., nnode(i)), nnode(i), nintr(i), nsubm(i), nside(i)), i=1,..., nelem)*
(5+*nom*+*nelem*)-(4+*nom*+*nelem*+*nos*)	(3F10.0)	$(x_s, z_i, s_i, i = 1, ...,nos)$ (optional)
(5+*nom*+*nelem*+*nos*)-(4+*nom*+*nelem*+*nos*+*noi*)	(2F10.0)	$(x_i, z_i, i = 1, ...,noi)$ (optional)

Table 1: Parameters and formats for input file (*filename.data*), for the model, to be read in subroutine INPUTD

nelem : Total number of boundary elements

isave : Saving flag,

0 : no saving for postprocessing

1 : saving and postprocessing of global simplified results

2 : saving and postprocessing of detailed numerical results and saving of data for plotting curves

lomax : maximum number of time steps in the general time loop (Fig. 6)

pcmax : maximum number of predictor-corrector loops per time step (Fig. 6)

iptyp : type of wave generation,

0 : specification of wave potential on the free surface with possibly lateral current U (e.g., *exact* solitary waves)

1 : generation of a sum of sine waves by a flap wavemaker, with initial damping (see *tdamp*, and file *paddle.data*)

3 : generation of *first-order* solitary or cnoidal waves by a piston wavemaker

5 : generation of *second-order* Stokes waves by internal sources

6 : generation of *second-order* solitary waves by internal sources

nbs : number of segments between nodes on the bottom to which stretching is applied for the geometry updating, when specifying a wavemaker or a moving free boundary, on a lateral boundary (< 150)

iprint : index for selective output listings,

0 : no output listing is created

n : output listing is printed for results generated at every n loops. Result types will depend on the value of *isave* (0,1: simplified results; 2: detailed results), and *ifield* (0 : no interior field results; 1 : print interior field results)

init : initialization of predictor-corrector loops for radiation conditions,

1 : initialization data exist for radiation conditions and will be used

0 : ignored

When *init*=1, initialization data must be provided at the very end of the input data file (see INPUTD for detail).

ismoth : when $K = 2$ or 3 (lateral boundaries), this parameter indicates the precision of numerical computations, 4 to 17, involved in the predictor-corrector loop for the radiation condition. Ignored when there are no radiation condition boundaries.

tolmax : absolute tolerance for predictor-corrector loops, used when, $pcmax > 1$ (Fig. 6). If $toler < tolmax$ predictor-corrector looping stops, in which,

$$toler = |\frac{D\mathbf{u}}{Dt}|^{n+1} - |\frac{D\mathbf{u}^{n}}{Dt}|^{n} \tag{130}$$

Ignored when there are no radiation condition boundaries on lateral boundaries, $K = 2$ or 3.

ibcond(K) : type of boundary condition on boundary segment K ((1) free surface; (2) leftward lateral boundary; (3) rightward lateral boundary; (4) bottom)

0 : Impermeable boundary $\phi_n = 0$ (homogeneous Neuman condition)

1 : Dirichlet boundary condition (i.e., specified ϕ or $\frac{\partial \phi}{\partial t}$)

2 : Wavemaker boundary with geometry updating (non-homogeneous Neuman condition function of the value of *iptyp*)

3 : plane impermeable boundary with exponential stretching of nodes (i.e., slope)

Notice, on the bottom ($K = 4$), a value 0 is selected for *ibcond*, and the geometry is arbitrary. On lateral boundaries, when *ibcond(k)*=0 or 3, the boundary must be plane, and a simple updating with node stretching is performed. When *ibcond(k)*=2 on lateral boundaries, the geometry must also be plane, for a wavemaker boundary. Finally, when *ibcond(k)*=1, the boundary is free (radiation boundary), and the geometry is arbitrary.

isplin(K) : definition of general type of boundary elements for boundary segment K,

0 : no spline analysis, isoparametric elements, 2-5 nodes

1 : spline analysis performed on the geometry, quasi-spline elements, 2 nodes

almaxb : limiting angle for adaptive integration on the boundary

almaxi : limiting angle for adaptive integration for interior field calculations

dt : initial time step, for $t = t_o$, used to calculate initial Courant number by,

$$C_o = \frac{dt * \sqrt{ge * del}}{|\Delta \mathbf{r}|^{min}}$$

with $C_o \leq 0.8$. At later time steps, C_o is assumed constant, and the time step *dt* is modified as,

$$dt = \frac{C_o \, |\Delta \mathbf{r}|^{min}}{\sqrt{ge * del}}$$

tstart : initial time of stepping, i.e., t_o =*tstart*. Notice, computations can be re-started from earlier results calculated with the model, and saved through subroutine START.

tmax : maximum time of stepping, $t \geq tmax$, will stop computations even if, loop $< lomax$ (Fig. 6)

rho : fluid density in arbitrary consistent units, $rho = 1$, implies dimensionless quantities

cpress : atmospheric pressure at the free surface Γ_f ($K = 1$), usually 0

cprest : atmospheric pressure gradient at the free surface Γ_f ($K = 1$), usually 0

ge : acceleration due to gravity in arbitrary consistent units, $ge = 1$, implies dimensionless quantities

tdamp : time over which damping is performed for waves generated by a flap paddle motion (*iptyp*=1; hyperbolic tangent damping, *tdamp*=t_{ϵ_z} in (70),(71)), or damping coefficient for Stokes waves generated by internal sources (*iptyp*=6; exponential damping, *tdamp*=μ in (85),(86))

0 : no damping

$\neq 0$: time over which damping is $\epsilon_z = 1\%$ (*iptyp*=1), or damping coefficient μ (*iptyp*=6).

de1 , *de2* : depth at boundary Γ_2, and Γ_3, respectively. *de1* = 1, means dimensionless quantities will be used, and *de1* will be used as the reference depth everywhere.

omega : incident wave circular frequency with,

iptyp =1; *nht*=0 : single sine wave frequency for flap wavemaker generation

iptyp =3 : 0 for a solitary wave, and > 0 for a cnoidal wave

iptyp =5 : > 0 for a 2nd-order Stokes wave generated by internal sources

voh /ulat : flap paddle stroke velocity (*vo*), wave amplitude (*h*), or current velocity on lateral boundary (*ulat*), with,

iptyp =1; *nht*=0 : single sine wave flap paddle stroke velocity at $z = 0$.

iptyp =0; *ibcond(K)*=0 : current velocity on lateral boundary K (as in (6))

iptyp =3 : solitary or cnoidal wave height

iptyp =5,6 : 2nd-order Stokes or solitary wave height generated by internal sources

isourc : flag for wave generation by internal sources,

0 : no generation by internal sources

1 : generation by internal sources

When *isourc*=1, waves will be generated according to *iptyp*= 5 or 6, and the initial location of internal sources must be provided as input data at the end of the input file (see below)

nos : number of internal sources specified along the vertical source line

ifield : flag for internal field computations,

0 : no internal field computations

1 : internal field computations

When *ifield*=1, the location of internal points must be provided as imput data at the end of the input file (see below)

noi : number of specified internal points at which internal fields will be calculated

nht : index for wave generation by a flap wavemaker,

0 : only one sine component of circular frequency ω=*omega* will be generated, with paddle stroke velocity $\dot{x}_p^{max}$ =*voh*

n : number of sine components to be generated with equations (68)-(76).

When *nht*> 0, a file *paddle.data* must exist on the same disk as the general input data, and contain the following information about the sine components to be generated,

formatted as,

Sine component :	amplitude	frequency	phase
nht lines (3F20.0) :	$ap(ih)$	$op(ih)$	$sp(ih)$

For internal source generation (*isourc*=1), and internal fields calculations (*ifield*=1), locations of sources or internal points must be specified, successively, at the end of the input file,

- for the sources : $(x_s, z_i, s_i, i = 1, ...,nos)$ (3F10.0), in which x_s denotes the constant x-value for the vertical line of sources, and s_i the strength of each sources. Notice, for a cold start, only x_s must be specified in the input data, and (s_i, z_i) will be calculated in the model at every time step, depending on the value of *iptyp*. All the values will be needed only when calculations will re-start from previously calculated results saved through START.

- for the internal field points : $(x_i, z_i, i = 1, ...,noi)$ (2F10.0), must be provided in sequence in the input data file.

Initialization data, if required (*init*=1), will follow these two sets of data.

The following is a general description of input parameters listed after the first 4 lines in the above table,

(x(i), z(i), phi(i), phin(i), i=1,...,nom) : initial nodal coordinates (x, z), and initial value of the potential *phi*, and normal gradient *phin*, for the *nom* elements in the boundary discretization.

Notice, only the free surface values of the potential and of the normal gradient will be used in calculations, and thus have to be specified at initial time $t = t_o$.

(node(i,j), j=1,..., nnode(i)), nnode(i), nintr(i), nsubm(i), nside(i)), i=1,..., nelem) : initial boundary element connectivity matrix, for the *nelem* elements of the BEM discretization, with,

node (i,j) : a vector containing 2 to 5 node numbers for element i

nnode (i) : number of nodes for element i

nintr (i) : number of Gauss points for element i

nsubm (i) : maximum exponent of 2, for subdivisions in the adaptive integration (0 : no need for subdivision is even tested; > 1 : intercept angle is checked from element i to every node, versus *almaxb* and/or *almaxi*, and adaptive integration is carried out, if needed, for element i, up to $2^{nsubm(i)}$ subdivision)

nside(i) : number (K=1 to 4) for boundary segment to which element i belongs

5.4 Subroutines and Functions

59 subroutines and 5 function are used in the program. They are listed in alphabetical order in the following, and a brief description of their tasks is given. Sufficient internal documentation is provided in each subroutine or function source code, to facilitate comprehension.

5.4.1 Subroutines

AGM : Computes arithmetic and geometric mean tables, for elliptic integrals and functions. First and second kind complete integrals are then calculated for the given complementary parameter, $1 - m$

ASSEMK : Assembles global K_d and K_n matrices into the system matrix K, based on specified boundary condition types (Dirichlet or Neuman). When $ibcond(k) = 1$, ϕ (or $\frac{\partial \phi}{\partial t}$) is imposed, and when $ibcond(k) \neq 1$, $\frac{\partial \phi}{\partial n}$(or $\frac{\partial^2 \phi}{\partial t \partial n}$) is specified on boundary k

ASSEML : Assembles local matrices in the global K_d and K_n matrices.

ASSEMP : Assembles the right hand side (load vector) of the general system, based on specified boundary condition types (Dirichlet or Neuman)

BEMK : Boundary element analysis. Computes K_d and K_n, performs rigid mode analysis on K_n, assembles the system matrix K, imposes double node compatibility, and boundary conditions for the 2D-BEM and introduces free coefficients c's into K_n (Fig. 8).

BIMAT : Computes local matrices k_d and k_n for each element *ie*, and saves geometric and intrinsic data for the 2D-BEM. Used only if no adaptive integration is required are required (*nsubm*=0) (Fig. 8)

BISMAT : Performs similar operations as BIMAT, but is called only when adaptive integration is required (*nsubm*> 1) (Fig. 8)

CARAC1 : Computes x_{ip}, z_{ip}, nx_{ip}, nz_{ip} and Ds_{ip} at the integration points *ip*=1,...,*nintr* of the isoparametric element *ie*.

CARAS1 : Computes x_{ip}, z_{ip}, nx_{ip}, nz_{ip} and Ds_{ip} at the integration points *ip*=1,...,*nintr* of the quasi-spline element *ie*. The second derivatives with respect to x and z at the extremities of each element have been previously determined in SPLINE.

CELES : Computes wave celerity from the linear dispersion relation,

$$k\,tanh(kh) = \frac{\omega^2}{g}$$

CHARAC : Calculates the complementary elliptic parameter for cnoidal waves, given the wave height and period

CHSUB : Checks whether adaptive integration is required over each element *ie*. The check is performed by determining whether the angle from which the element is seen from node $i = 1, ..., nom$ is less than $almaxb$. Then, $nsub(ie)$, the exponent of 2 for the number of subdivisions required, is stored in the array *lsub(nom)* for element *ie*

CNIDAL : Implements the generation of cnoidal waves by a piston wavemaker according to Goring's first order KdV solution. The method is accurate for $\frac{L}{h} > 20$, and $\frac{H}{h} < 0.3$

CUBICT : Fits a third-order 4-node polynomial, and computes $\frac{dXZ}{d\tau}$, for the boundary conditions at extremities of the free surface, in the spline analysis in SPLANA

D2FUN1 : Computes second derivatives of shape functions with respect to their intrinsic coordinate η, for one-dimensional isoparametric elements (2-5 nodes)

D2UTPR : Computes the corrected value of accelerations $\frac{Dw}{Dt}$, $\frac{Du}{Dt}$, based on kinematics, as a function of $\frac{\partial\phi}{\partial n}$, $\frac{\partial\phi}{\partial s}$, $\frac{\partial^2\phi}{\partial t\partial s}$, $\frac{\partial^2\phi}{\partial t\partial n}$, $\frac{\partial^2\phi}{\partial s^2}$, $\frac{\partial^2\phi}{\partial n\partial s}$, $cos\beta$, $sin\beta$ and β's. Also predicts $\frac{D^2u}{Dt^2}$ and $\frac{D^2w}{Dt^2}$, based on current and previously calculated values of the acceleration, for later use in $\frac{Dp}{Dt}$ estimation in DPRFIX

DBNODK : Imposes corner double node compatibility in the general system matrix K, according to specified boundary condition types (Dirichlet or Neuman) (see *ibcond(k)*)

DBNODP : Impose corner double node compatibility at domain double nodes, for the right hand side load vector of the global system matrix

DFUNF1 : Computes first derivatives of shape functions with respect to their intrinsic coordinate η, for one-dimensional isoparametric elements (2-5 nodes)

DKE : Computes m-derivatives of elliptic integrals $K(m)$ and $E(m)$

DPRFIX : For lateral radiation boundaries, computes the temporal gradient of pressure on lateral Dirichlet boundaries, based on corrected accelerations ($\frac{Dw}{Dt}$, $\frac{Du}{Dt}$), and on predicted velocity of accelerations, namely $\frac{D^2u}{Dt^2}$ and $\frac{D^2w}{Dt^2}$, by integrating Euler equations in the tangential s direction along the boundary

DTSTEP : Fixes the time step Δt for each time loop, based on a constant Courant number criterion set at initial time step. The maximum value of this Courant number is limited to 0.8 for reasons of stability

DUDTPR : Computes (u, w) values over the boundary as a function of known $\frac{\partial \phi}{\partial n}$ and $\frac{\partial \phi}{\partial s}$. Also computes $\frac{\partial^2 \phi}{\partial s^2}$ for later use. Predicts $\frac{Dw}{Dt}$, $\frac{Du}{Dt}$, based on current and previously calculated values of u & w

ERRORS : Displays appropriate error messages

FCTS : Computes $GHT(m) = 0$ and $\frac{dGHT}{dm}$ at m to find the parameter m of a cnoidal wave

FSVELO : Imposes compatibility condition at double nodes on the free surface. the subroutine determines direction cosines for both nodes, and the potential derivative with respect to the s direction, for both nodes

FUNF1 : Computes shape functions with respect to their intrinsic coordinate η, for one-dimensional isoparametric elements (2-5 nodes)

GAUSSP : Computes values of Gauss points and weights, for the specified number of integration points *nintr*, and calculates values of shape functions and their derivatives at these points

IMPLAT : Updates lateral boundary conditions and geometry (with $iflagv = 1$), for impermeable Neuman boundaries (*ibcond(k)*=0,≥ 3, and $K = 2, 3$)

INTIAL : Performs geometry and potential initialization for all time step arrays. This initialization is performed by calling in turn the subroutines SLIDING, FSVELO, BEMK (Fig. 6). The sliding derivatives on all four boundaries, computed in SLIDING, are stored in *temmbbc*. Double node compatibility is imposed in FSVELO, and BEMK performs the initial boundary element analysis for the specified initial geometry, to set up the integral common *savef*

INPUTD : Reads input data for 2D boundary element method, *i.e.* nodal coordinates, element definitions, flag values to specify problem etc. The data is read from a

specified input file, *filename.data*, containing the data in the format described in the Table above

INTERN : Computes internal fields ϕ, $\frac{\partial \phi}{\partial t}$, $\frac{\partial \phi}{\partial x}$, $\frac{\partial^2 \phi}{\partial x \partial t}$, $\frac{\partial \phi}{\partial z}$, $\frac{\partial^2 \phi}{\partial z \partial t}$ and p at *noi* user-specified points

INTRCI : Introduces free coefficients c into K_n based on the rigid mode technique

NEWTON : Computes zeros of GHT function, using a discrete Newton's method. Part of the m-derivatives are computed using finite differences

PADSIN : Simulation of a flap wavemaker oscillating on a lateral boundary. Boundary conditions, or geometry and boundary conditions are updated, depending on the value of *iflagv*. The paddle motion is damped(hyperbolic tangent function), according to *tdamp*

PADSOL : Simulation of a piston wavemaker motion, based on first-order solitary or cnoidal wave theory, specified on a lateral boundary. *iflagv* determines whether both boundary conditions and geometry are updated or only the boundary conditions are updated

PREDICT : Fits cubic temporal polynomials over $nt - 1$ steps, starting at $nt - iy$. For the component ip of y. Values of y or $\frac{dy}{dt}$ are computed after fitting

PREFIX : Computes pressure on lateral Dirichlet boundaries based on predicted accelerations ($\frac{Dw}{Dt}$, $\frac{Du}{Dt}$), by integrating Euler equations in the tangential s direction along the boundary. If initial data are not provided for prediction, the pressure is assumed hydrostatic, till $iloop = nt - 1$. Computes $\frac{\partial \phi}{\partial t}$ by Bernoulli's equation on all Dirichlet boundaries

PRESPR : Computes pressure or its temporal gradient on boundary K, by integrating the Euler equations in the s direction along K

RADIA : Performs main stages of computations (Fig. 6), by calling the corresponding modules. The general algorithm for the solution of the boundary value problem is implemented in this subroutine. The program is modular, and each step of the solution described in preceding sections is solved in different subroutines

RESOL : Updates boundary conditions and geometry if $iflagv = 1$, otherwise only updates boundary conditions (Fig. 7). If $iflags = 1$ performs BEM analysis in BEMK, double nodes, sources, global assembling and solves for either ϕ or $\frac{\partial \phi}{\partial t}$

SAVE : Performs postprocessing of both boundary and internal fields, saves all results in direct access files. Specifically, SAVE computes the discharge through boundaries, domain (or wave) volume, domain (or wave), kinetic, potential and total energy, forces and moments acting on lateral boundaries (in both dimensional and dimensionless forms), and the internal fields through calling INTERN. It saves the data for generating and plotting result curves in the post-processing

SAVGEO : Saves geometry and shape functions of boundary element for all integration points. These values are stored in *savef* and used in SAVE for post-processing

SGDUDN : Computes local particular integral for $\frac{du}{dn}$ for 2D-BEM, set to 0 if the element is a straight line element.

SLIDING : Computes parameters of s-derivatives $\frac{d}{ds}$, and of $\frac{d^2}{ds^2}$, along each of the four boundary subsections, as a function of domain geometry, and saves them in *tembbc* in the form of shape functions. These computations use a sliding fourth-order isoparametric boundary element, independent from the actual BEM discretization, and the derivatives are being computed at the mid-node of the sliding element, except at the 2 extremities of the free surface, where the element stays unchanged for the 3 first and 3 last nodes

SOLITA : Goring's first-order solution for solitary waves. The equations for the motion of the wavemaker and its kinematics are solved iteratively by Newton's method

SOLVE : Computes the Kaletsky solution for a nonsymmetric, fully populated, linear system of equations. The method provides the solution X_j of the linear system $A_{ij} * X_j = B_i$

SORT : Sorts the unknowns according to type of boundary conditions, after solution in SOLVE

SOURCE : Performs domain integration to specify wave generation by an internal line of sources

SOUSTE : Generates interior source strength according to wave type (2nd-order Stokes or solitary wave), and problem, namely ϕ or $\frac{\partial \phi}{\partial t}$

SPLANA : Analysis of the domain free surface boundary geometry by cubic splines. Since the free surface can be multivalued, two spline analyses are performed, for both x and z, as functions of a parameter chosen to be the point index τ of the nodes

SPLINE : Performs spline analysis and saves $\frac{dy}{dx}$

START : Saves information required for re-starting calculations for the next iteration, after stopping computations with the program. Results are saved in file *start.data*, such as domain geometry, previous time step variables, interior sources and fields etc... and the program is stopped. This procedure, of stopping and re-starting computations, may be required due to memory limitations on the computer used in the calculations

STRBOT : Performs stretching of bottom boundary geometry and nodes, close to the lateral boundary, and over *nbs* intervals between nodes. The bottom is assumed horizontal over the *nbs* intervals

TRAJEC : Computes displacement, velocity and acceleration of a piston wavemaker, for generating a cnoidal wave. The initial stroke is 0, and the stroke varies from 0 to $2 * xpmax$. Velocity and wave elevation are 0 at time $t = 0$

UPBOTC : Updates boundary conditions and geometry on the bottom boundary when *iflagv* $= 1$, according to the value of *iptyp* and *ibcond* (Fig. 7)

UPDTDB : Updates geometry of moving Dirichlet boundaries, based on a second-order Taylor expansion in time using the previously determined kinematics (u, w) and ($\frac{Dw}{Dt}$, $\frac{Du}{Dt}$). The potential is also updated based on a second-order expansion in terms of previously determined parameters (Fig. 6)

UPLABC : Updates boundary conditions or boundary conditions and geometry (*iflagv* $= 2/1$), on lateral Neuman boundaries, depending on the value of *iptyp* and *ibcond* (Fig. 7)

WRRES : Prints results when listing is required

5.4.2 Functions

ACN : Computes the inverse of the Jacobi elliptic cosine of argument u, and parameter m, $acn(u, m)$.

ATANS : Returns the arctangent for subroutine ATANV, and checks for the indeterminate case $\frac{0}{0}$.

ATANV : Returns the arctangent depending on the position in the OXZ coordinate system. It is called in the local particular integration over the element *ie*, for $\frac{du}{dn}$. If the arctangent provides an indeterminate form, *L'Hospital's theorem* is applied up to 2 times.

CN : Computes the Jacobi elliptic cosine of argument u, and parameter m, $cn(u, m)$. If $1 - m$ is less than $5 * 10^{-6}$, a hyperbolic approximation is used to improve numerical

stability.

EINC : Computes the complete elliptic integral of the second kind, $E(u, m)$, with argument u and parameter m.

5.5 Program Execution

To run the program on a large application case, and saving all computed results, a sufficient amount of disk space must be reserved (either physical or virtual). In the IBM-VM operating system, a virtual disk space can be defined, that will exist only for the current logon session.

The reserved disk space is used for storing result data files, for subsequent post-processing computations. Some listing files will also be stored therein, to be printed if required.

A typical IBM-VM session, is included below, for the case of a fully non-linear solitary wave, with wave elevation and potential generated in the program, SOLWAVE. Commands by the user are in **boldface**, computer outputs in `verbatim`, and comments in *italics*.

A command file *radia.exec* has been created for making all the file definitions, and loading and running all the program modules.

A Typical Session

(A virtual disk (type t3380 with 50 cylinders) is defined at first, for storing files containing program results)

def t3380 as 210 cyl 50

```
def t3380 as 210 cyl 50
DASD 0210 DEFINED
Ready; T=0.01/0.01 14:35:06
```

(formatting the disk)

format 210 e

```
DMSFOR603R FORMAT will erase all files on disk E(210). Do you wish to continue?
Enter 1 (YES) or 0 (NO).
```

1

```
1
DMSFOR605R Enter disk label:
```

(any character will suffice)

e

```
e
Formatting disk E
50 cylinders formatted on E(210)
Ready; T=0.03/1.69 14:37:32
```

(disk status)

q disk

```
q disk
LABEL   VDEV M   STAT   CYL TYPE BLKSIZE   FILES   BLKS USED-(%) BLKS LEFT   BLK TOTAL
RAVI    191  A    R/W     18 3380 4096        48       1721-64        979        2700
E       210  E    R/W     50 3380 4096         0          6-00       7494        7500
MNT190  190  S    R/O     70 3380 4096       377       7269-69       3231       10500
MNT19E  19E  Y/S  R/O    120 3380 4096       550      15513-86       2487       18000
ACC19F  19F  Z/S  R/O    150 3380 4096       152      10679-47      11821       22500
Ready; T=0.01/0.01 14:38:40
```

*(**radia** is the executable file* radia.exec *being called,* **step0** *contains the initial model data (input file* step0.data*), regarding the wave and domain geometry and the potentials,* **e** *is the temporary disk on which temporary data files will be stored and* **a** *is the disk on which the input files required for re-starting the program are stored.)*

radia step0 e a

```
radia step0 e a
ERASE FIELD DATA E
File FIELD DATA E not found
+++ R(00028) +++
ERASE CURV1 DATA E
File CURV1 DATA E not found
+++ R(00028) +++
ERASE CURV2 DATA E
File CURV2 DATA E not found
+++ R(00028) +++
FI 4 DISK CURVE DATA E ( BLKSIZE 80 LRECL 80
FI 5 DISK STEP0 DATA A
FI 80 DISK PADDLE DATA A
FI 6 DISK STEP0 LISTING E
FI 7 DISK START DATA A1
FI 10 TERM
FI 13 DISK EPSC DATA E
FI 14 DISK EPSV DATA E
FI 15 DISK ET DATA E
FI 16 DISK OUTF DATA E
FI 18 DISK VOLU DATA A
FI 19 DISK QNF DATA E
FI 20 DISK QNR1 DATA E
FI 21 DISK QNR2 DATA E
FI 22 DISK EP DATA E
FI 23 DISK EK DATA E
FI 30 DISK GENDAT DATA E
FI 48 DISK FSLX1 DATA E
FI 49 DISK FSLZ1 DATA E
FI 50 DISK FSLX2 DATA E
FI 51 DISK FSLZ2 DATA E
FI 52 DISK FSLR1 DATA E
```

```
FI 53 DISK FSLP1 DATA E
FI 54 DISK MSLB1 DATA E
FI 55 DISK FSLR2 DATA E
FI 56 DISK FSLP2 DATA E
FI 57 DISK MSLB2 DATA E
FI 58 DISK HMAXT DATA E
FI 59 DISK HMAXX DATA A
FI 60 DISK TXMAX DATA E
FI 61 DISK XZMAX DATA A
FI 62 DISK GREEN DATA A
GLOBAL TXTLIB IMSL11A IMSL11B VSF2FORT CMSLIB UTILITY
File UTILITY TXTLIB * not found
+++ R(00028) +++
GLOBAL LOADLIB VSF2LOAD
LOAD MRADIA RADIA0 RADIA1 RADIA2 RADIA3 ( CLEAR
The following names are undefined:
 STRVEL   STRETA
+++ R(00004) +++
START
Execution begins...
 Input Fm of DISK files ? :
```

(File Mode of temporary disk files.)

e

```
e
 Loop :     1
 Loop :     2
 Loop :     3
 Loop :     4
 Loop :     5
 Loop :     6
 Loop :     7
```

```
 Loop :      8
 Loop :      9
 Loop :     10
 Loop :     11
 Loop :     12
 Loop :     13
 Loop :     14
 Loop :     15
 Loop :     16
 Loop :     17
 Loop :     18
 Loop :     19
 Loop :     20
Ready; T=67.03/67.99 14:43:30
```

(temporary disk files generated)

flist * * e

```
flist * * e
 LVL 0 --- E 210    7500 BLKS 3380 R/W      22 FILES  1%  FILE      1 OF     22
TXMAX    DATA     E1                        F    80     20       1  9/24/92 14:43
HMAXT    DATA     E1                        F    80     20       1  9/24/92 14:43
MSLB2    DATA     E1                        F    80     20       1  9/24/92 14:43
FSLP2    DATA     E1                        F    80     20       1  9/24/92 14:43
FSLR2    DATA     E1                        F    80     20       1  9/24/92 14:43
MSLB1    DATA     E1                        F    80     20       1  9/24/92 14:43
FSLP1    DATA     E1                        F    80     20       1  9/24/92 14:43
FSLR1    DATA     E1                        F    80     20       1  9/24/92 14:43
FSLZ2    DATA     E1                        F    80     20       1  9/24/92 14:43
FSLX2    DATA     E1                        F    80     20       1  9/24/92 14:43
FSLZ1    DATA     E1                        F    80     20       1  9/24/92 14:43
FSLX1    DATA     E1                        F    80     20       1  9/24/92 14:43
EK       DATA     E1                        F    80     20       1  9/24/92 14:43
```

```
EP         DATA      E1          F    80     20      1  9/24/92 14:43
QNR2       DATA      E1          F    80     20      1  9/24/92 14:43
QNR1       DATA      E1          F    80     20      1  9/24/92 14:43
QNF        DATA      E1          F    80     20      1  9/24/92 14:43
OUTF       DATA      E1          F    80     20      1  9/24/92 14:43
ET         DATA      E1          F    80     20      1  9/24/92 14:43
EPSV       DATA      E1          F    80     20      1  9/24/92 14:43
EPSC       DATA      E1          F    80     20      1  9/24/92 14:43
STEP0      LISTING   E1          V   133   1467     31  9/24/92 14:43
Ready; T=0.01/0.02 14:44:58
```

(disk files generated on disk **a.***)*

flist * * a

```
flist * * a
START      DATA      A1          F    80    466     10  9/24/92 14:43
LOAD       MAP       A5          F   100    302      8  9/24/92 14:40
GREEN      DATA      A1          F    80     19      1  9/24/92 14:43
HMAXX      DATA      A1          F    80     20      1  9/24/92 14:43
XZMAX      DATA      A1          F    80     19      1  9/24/92 14:43
VOLU       DATA      A1          F    80     20      1  9/24/92 14:43
FILE       FT99F001  A1          F    80     20      1  9/24/92 14:43
Ready; T=0.01/0.04 14:48:16
```

(End of Session.)

5.6 Output Files

The following *.data* files are generated automatically by the program.

START : Data required for restarting the program, generated after a first set of iterations

EPSC : Dimensionless error on continuity equation at every time step, obtained by adding up flow rates (discharges) through each boundary segment

EPSV : Dimensionless error on total volume of the domain at every time step

ET : Total energy at every time step. If nondimensional quantities are used and an exact solitary wave is generated, the total energy is the one corresponding to the wave only

EK : Kinetic energy at every time step. If nondimensional quantities are used and an exact solitary wave is generated, the kinetic energy is the one corresponding to the wave only

EP : Potential energy at every time step. If nondimensional quantities are used and an exact solitary wave is generated, the potential energy is the one corresponding to the wave only

VOLU : Total volume of computational domain at each time step. If nondimensional quantities are used and an exact solitary wave is generated, the total volume is the one corresponding to the wave only

OUTF : Sum of net fluxes through all four boundary segments at each time step

QNF : Volume flow through the free surface at each time step

QNR1 : Volume flow through lateral boundary 1 ($K = 2$) at each time step

QNR2 : Volume flow through boundary 2 ($K = 3$) at each time step

FSLX1 : Horizontal force on boundary 1 at each time step

FSLZ1 : Vertical force on boundary 1 at each time step

FSLX2 : Horizontal force on boundary 2 at each time step

FSLZ2 : Vertical force on boundary 2 at each time step

FSLR1 : Magnitude of force vector on boundary 1

FSLP1 : Angle of force vector on boundary 1 at each time step

MSLB1 : Moment about the bottom on boundary 1 at each time step

FSLR2 : Magnitude of force vector on boundary 2 at each time step

FSLP2 : Angle of force vector on boundary 2 at each time step

MSLB2 : Moment about the bottom on boundary 2 at each time step

HMAXT : Maximum value of free surface elevation at each time step

HMAXX : Maximum value of free surface elevation, and its position along the x coordinate

TXMAX : The x position of the maximum value of free surface elevation, and its variation with t (time).

XZMAX : Maximum value of free surface elevation divided by the local depth, as a function of x

GREEN : Tests for adherence to *Green's Law*

5.7 Error and Warning Statements

Values of parameters and results are constantly tested within each subroutine, versus exact or pre-specified values.

In case an error occurs, an eight character error code is transferred to the error handling subroutine ERRORS (e.g., INPUTD02), and the program is stopped. The following message is printed by the program, prior to stopping,

```
PROGRAM STOP DUE TO ERROR  : INPUTD02
```

The six first characters in the error code contain the name of the subroutine in which the error was found (e.g., INPUTD), and the last two characters contain the error code (e.g., 02).

Errors codes, with definitions of errors, are listed within the source code of each subroutine. In INPUTD, for instance, we find the following list,

```
CE  ERRORS    01= Element, nodes, sources, or int. fields nb. out of range
CE            02= Logical data are out of range
CE            03= Boundary conditions are out of range
CE            04= End of data in input file
CE            05= Spline analysis is impossible with data defined
CE            06= Element order is messed up
```

Hence, the user can consult each subroutine code header, for the list of relevant error codes.

In case the code definition (e.g., "Logical data are out of range") is not sufficiently clear, the user can look for the location of the call to the error routine in the subroutine in which the error was found, and find out more details. For instance, when searching for the error code INPUTD02, in INPUTD's source code, we find it to be stored in the variable TEXTE2, and when searching for TEXTE2, we get to the following sequence of FORTRAN code,

```
C
      IF((IPTYP.LT.0.OR.IPTYP.GT.6).OR.(ISAVE.LT.0.OR.ISAVE.GT.2).OR.
     .   (ISOURC.NE.0.AND.ISOURC.NE.1).OR.(IFIELD.NE.0.AND.IFIELD.NE.1))
     .   THEN
         CALL ERRORS(TEXTE2)
C        -----------
      END IF
C
```

which gives us the full extent of the error check done in the program at this stage. We for instance see above that the error code 02 in INPUTD is related to a wrong value for IPTYP, ISAVE, ISOURC, or ISOURC in the input data.

6 Applications

6.1 Introduction

Over the past 5 years, many applications have been calculated with the model, for various types of wave propagation, shoaling, and runup, and wave interaction with emerged and submerged coastal structures or obstacles in the bottom.

Main types of applications are listed in the following, along with references to publications in which the reader can find details of both computational and physical aspects of the problems,

- **Wave generation by a moving vertical boundary :** Grilli & Svendsen [37] (1990) studied the generation of breaking waves by horizontally moving vertical boundaries, using the present model. They analyzed the accuracy of results as a function of both discretization and time step, and evaluated the performance of corner compatibility relationships in the very demanding case where both lateral and free surface boundaries take large displacements.

  Grilli [24] (1991) extended this method to the calculation of breaking bow waves, and wave resistance coefficient of forward moving slender ships. This application is also implemented in the present model, but has not been described in this report.

- **Wave runup over and reflection from a steep slope :** Grilli & Svendsen [33, 35, 36, 38] (1989,1990,1991) and Svendsen & Grilli [72] (1990), through careful numerical experiments, extensively studied the runup on, and reflection of solitary waves from steep slopes, and from vertical walls. They compared model results to laboratory experiments, and in general found surprisingly good agreement between both of these.

  Similar cases are presented in the applications in section 6.2, for the runup of a solitary wave of incident height $\frac{H_o}{h_o} = 0.12$, over two slopes of angle $\Theta = 20°$, and $45°$, and in

section 6.3, for the runup of a cnoidal wave of incident height $\frac{H_o}{h_o} = 0.10$, over a slope of angle $\Theta = 20°$.

These applications were selected for sake of comparison with results obtained by Liu *et al.* [50] (1992) with their nonlinear model, as part of the present NSF project.

- **Wave shoaling and breaking over a gentle slope :** Grilli *et al.* [36] (1991), Otta *et al.* [56] (1993), and Grilli & Subramanya [32] (1993) used the model to calculate shoaling of solitary waves over a gentle slope, up to initiation of breaking. Recently, cases with periodic waves have also been calculated, to study the kinematics and integral properties of breaking waves on beaches, very important for surf-zone dynamics.

  Grilli *et al.* [39] (1993), in particular, made a detailed study of shoaling of solitary waves, up to breaking over various slopes, and compared their results to classical Green's and Boussinesq's law, and to recent very careful laboratory experiments. They concluded, whereas none of the theoretical "shoaling laws" could accurately predict observed shoaling and breaking behaviors, the present fully nonlinear model agreed with experiments up to the breaking point.

  Otta *et al.* [56] (1993), in addition, based on their calculations with the model, developed a criterion for breaking of solitary waves over slopes, and analyzed the kinematics of waves at breaking.

  A similar case is presented in the application in section 6.4, for an incident solitary wave of initial height $\frac{H}{h_o} = 0.20$, shoaling and (spilling) breaking over a slope s =1:35.

- **Wave interactions with submerged obstacles :** Accurate prediction of water wave propagation over submerged obstacles is of prime importance in coastal engineering. Submerged breakwaters are becoming increasingly used as both aesthetic and economical means of shoreline protection against extreme storms and tsunamis. Natural reefs and sandbars are frequent coastal features that function as natural submerged break-

waters. In addition, the study of waves close to the shoreline, and in the surf zone, requires that the offshore wave climate is adequately "propagated" over any existing submerged obstacle, man-made or natural.

Propagation of waves has been calculated with the present nonlinear model, over three different types of submerged obstacles of various engineering implications. Cases with both large incident waves or shallow submerged obstacles have been solved that lead to strong nonlinear interactions between incident waves and the obstacles, and to various instabilities and breaking of incident waves on, or downstream of the obstacles. It is worth pointing out, these phenomena cannot be modeled by any of the standard wave theories, and require a fully nonlinear theory to be accurately described,

- *Step in the bottom :* The simplest possible steep obstacle on the bottom is the step discontinuity between two constant depth regions. Numerous studies of the interaction of a long wave with a step have been carried out using various wave theories, from linear to mildly nonlinear, and numerical models. The main motivation for these studies has been to answer the question : How do long waves behave when propagating from deep water into shallow water over the continental shelf ? More specific questions have also been addressed, by assuming the step represents a first approximation for a wide crested obstacle in shallow water, like a bar, a reef, or even a submerged breakwater.

  In this line, Grilli *et al.* [26] (1992) have used the present model to study strong nonlinear interactions—leading to breaking—of large solitary waves with steps in the bottom. They compared numerical results to laboratory experiments, and found fairly good agreement between both of these.

- *Rectangular bar :* After the steps in the bottom, rectangular obstacles have the simplest possible geometry for representing submerged bars or breakwaters. One may expect, in fact, that most of the phenomena observed or computed over

rectangular bars are, at least qualitatively, also occurring for obstacles of more complex geometry.

Driscoll *et al.* [17] (1993) studied the propagation of small amplitude normally incident cnoidal waves, over an infinitely long submerged shallow bar, with a rectangular cross-section. They compared laboratory experiments to first and second-order analytic models, and to the present full nonlinear BEM model. They found the BEM model could accurately predict the generation of higher-order harmonics, observed in laboratory, in the wave train downstream of the obstacle.

- *Submerged trapezoidal breakwaters :* Submerged breakwaters used for shoreline protection are usually built by dropping rocks from barges at selected offshore locations. Breakwaters, hence, take an approximate trapezoidal shape. The protection offered by submerged breakwaters consists in inducing breaking and partial reflection-transmission of large incident waves, while small wave propagation, and, in some cases, local navigation, can still take place over the structure during normal conditions.

  Cooker *et al.* [12] (1990) used an extension of Dold & Peregrine's [14] nonlinear model, to calculate solitary wave interaction with a submerged semicircular cylinder of radius R in water of depth h_o. Results showed, a variety of behavior occurs depending on wave height and cylinder radius. In short, for small cylinders ($\frac{R}{h_o} < 0.5$), waves essentially transmit and exhibit a tail of oscillations. This is a regime of weak interactions. For larger cylinders ($\frac{R}{h_o} \geq 0.5$), interactions are much stronger : small waves partially transmit and reflect (crest exchange); medium waves undergo a stronger crest exchange over the cylinder, and the first oscillation in their tail may break backward onto the cylinder (direction opposite to propagation); and large waves break forward (plunging), slightly after passing

over the cylinder. A limited number of experiments confirmed these theoretical predictions.

Grilli *et al.* [28] (1993) extended the above study to submerged breakwaters with a more realistic trapezoidal cross-section. Computations using the present model were compared to laboratory experiments, for a large number of solitary wave heights H, and for a breakwater geometry defined by : a height $h_1 = 0.8h_o$, a width at the crest $b = h_1$, and two (seaward and landward) 1:2 slopes. Results qualitatively agreed with earlier observations by Cooker *et al.* [12], as far as crest exchange and breaking behaviors are concerned. In all cases, a reflected wave also forms at the breakwater front face, and starts propagating backward into the tank.

- **Wave impact on coastal structures :** Two cases with more realistic coastal structures have been studied in earlier applications with the model, illustrating its ability to predict shoaling of incident waves from deep to shallow water, over a mild slope, and interaction with a structure in the shallow water region.

  In addition, the model was able to accurately predict peak impact pressures from breaking waves, on the vertical wall of mixed breakwater. Such numerical simulations are helpful for designing coastal structures.

  - *Mixed berm breakwaters :* Most classical breakwaters used for shoreline or harbor protection are constituted of a main trapezoidal breakwater, with a small submerged berm at the toe of an emerged structure. Part of the incident wave energy dissipates by breaking over the berm which, hence, offers some protection to the main structure.

    A similar case has been studied with the model by Grilli & Svendsen [36] (1991), for which, unlike with traditional berm breakwaters, a small detached submerged

structure has simply been located slightly in front of the main structure. The combination of the two structures is called a "mixed berm breakwater". This configuration, while offering the same degree of protection, may be more economical and simpler to build than classical berm breakwaters.

- *Mixed vertical breakwaters :* Mixed vertical breakwaters are composed of a vertical concrete caisson, sitting on a wide berm made of rocks. They function as vertical walls during high tide and as mound breakwaters during low tide. Their upper section is designed to be safe against sliding and overturning due to wave impact force. Laboratory and field experiments show, impacts of normally incident breaking waves are the most severe. In this case, the maximum impact force on the wall may rise up to 10 times the hydrostatic force based on wave elevation at the wall.

  Cooker [9] (1990), and Cooker and Peregrine [11] (1991), solving 2D fully nonlinear potential flows, confirmed these observations. Their model, however, although very accurate, was limited to a vertical wall, and used a large incident long wave, with characteristics selected to create a large scale breaker in the model.

  Grilli *et al.* [29, 27] (1992,1993) computed violent impact of breaking waves on mixed vertical breakwaters, using the present nonlinear model. Laboratory experiments were performed and compared to computations.

  As pointed out in [11], peak impact pressures are obtained for waves with large height to depth ratio. This was achieved in [11], by artificially introducing a very high incident wave in the model. The present model works for arbitrary geometry and wave conditions, which permitted using both more realistic incident waves, and a breakwater geometry closely reproducing the experimental set-up.

Following are details of data and results for three specific applications of the model to long wave shoaling, runup, and breaking over a plane slope.

Although the model can address much more general problems, as described above, detailed applications presented here have been limited to these simple, more academic cases, both for sake of simplicity, and because of the focus of the present NSF sponsored research project on long wave runup.

6.2 Solitary wave runup on a steep slope

The computational domain and set-up for this problem are similar to the case sketched in Fig. 2, except, due to the steep slope used in the present case, there is no need for, and therefore there is no shallow shelf at the rightward extremity of the computational domain.

The runup of a solitary wave of incident height $\frac{H_o}{h_o} = 0.12$, is calculated over two different slopes of angle $\Theta = 20°$, and $45°$. The incident wave is generated by simulating a piston wavemaker on the leftward boundary (Γ_1).

Discretization data can automatically be generated for this simple case, using the pre-processing program GENER with the following input data file. The case with a $20°$ slope is first presented (see Fig. 9 and corresponding table for definition),

```
 120    3    7   55    0    0
   2    3    3    3    0    0
30.0  1.0  0.0 20.0
  10    0
```

Values of the above parameters generate a discretization with 120 2-node elements on the free surface (these elements will be later specified as quasi-spline type), three 3-node elements on the leftward boundary, seven 3-node elements on the rightward boundary (i.e., the slope in the present case), and 55 3-node elements on the bottom.

The domain has a length 30.0, a constant depth 1.0, and a slope angle $20°$. There will be 10 Gauss points per element, and no pre-specified subdivisions for adaptive integration.

Following is the (simplified) file *gener.data* generated by the pre-processing program GENER, with the above data (see Table 1 for details of formats),

{\em (Total number of nodes and elements on the boundary; other data in first 4 lines are missing and have to be user-specified)}

254 185

(Nodes for boundary Γ_1; free surface)

0.0000000000	0.0000000000	0.0000000000	0.0000000000
0.2500000000	0.0000000000	0.0000000000	0.0000000000
0.5000000000	0.0000000000	0.0000000000	0.0000000000
0.7500000000	0.0000000000	0.0000000000	0.0000000000
1.0000000000	0.0000000000	0.0000000000	0.0000000000

.....

29.2500000000	0.0000000000	0.0000000000	0.0000000000
29.5000000000	0.0000000000	0.0000000000	0.0000000000
29.7500000000	0.0000000000	0.0000000000	0.0000000000
30.0000000000	0.0000000000	0.0000000000	0.0000000000

(Nodes for boundary Γ_3; slope)

30.0000000000	0.0000000000	0.0000000000	0.0000000000
29.8037516129	-0.0714285714	0.0000000000	0.0000000000
29.6075032258	-0.1428571429	0.0000000000	0.0000000000
29.4112548387	-0.2142857143	0.0000000000	0.0000000000

.....

27.8412677419	-0.7857142857	0.0000000000	0.0000000000
27.6450193548	-0.8571428571	0.0000000000	0.0000000000
27.4487709676	-0.9285714286	0.0000000000	0.0000000000

```
      27.2525225805      -1.0000000000      0.0000000000      0.0000000000
```

(Nodes for boundary Γ_4; bottom)

```
      27.2525225805      -1.0000000000      0.0000000000      0.0000000000
      27.0025225805      -1.0000000000      0.0000000000      0.0000000000
      26.7525225805      -1.0000000000      0.0000000000      0.0000000000
.....
       0.5025225805      -1.0000000000      0.0000000000      0.0000000000
       0.2525225805      -1.0000000000      0.0000000000      0.0000000000
       0.0000000000      -1.0000000000      0.0000000000      0.0000000000
```

(Nodes for boundary Γ_2; piston wavemaker)

```
       0.0000000000      -1.0000000000      0.0000000000      0.0000000000
       0.0000000000      -0.8333333333      0.0000000000      0.0000000000
       0.0000000000      -0.6666666667      0.0000000000      0.0000000000
       0.0000000000      -0.5000000000      0.0000000000      0.0000000000
       0.0000000000      -0.3333333333      0.0000000000      0.0000000000
       0.0000000000      -0.1666666667      0.0000000000      0.0000000000
       0.0000000000       0.0000000000      0.0000000000      0.0000000000
```

(Boundary Element connectivity matrix and parameters)

(Free surface)

```
1    2    0    0    0    2    10    1    1
2    3    0    0    0    2    10    1    1
3    4    0    0    0    2    10    0    1
4    5    0    0    0    2    10    0    1
```

```
    5    6    0    0    0    2   10    0    1
.....

  109  110    0    0    0    2   10    0    1
  110  111    0    0    0    2   10    0    1
  111  112    0    0    0    2   10    1    1
  112  113    0    0    0    2   10    1    1
.....

  120  121    0    0    0    2   10    1    1
```

(Slope)

```
  122  123  124    0    0    3   10    1    3
.....

  134  135  136    0    0    3   10    1    3
```

(Bottom)

```
  137  138  139    0    0    3   10    1    4
  139  140  141    0    0    3   10    1    4
  141  142  143    0    0    3   10    0    4
.....

  241  242  243    0    0    3   10    0    4
  243  244  245    0    0    3   10    1    4
  245  246  247    0    0    3   10    1    4
```

(Wavemaker)

```
248   249   250     0     0     3    10     1     2
250   251   252     0     0     3    10     1     2
252   253   254     0     0     3    10     1     2
```

We see, in the above file, GENER has generated a discretization with 254 nodes, and 185 elements. The initial distance between nodes is $\Delta x_o = 0.25$ on the free surface, 0.167 on boundary Γ_2, 0.20 on boundary Γ_3, 0.25 on boundary Γ_4. Element types are as requested on each boundary segment.

Notice, in the above data, the number of subdivisions for adaptive integration has been set to 1 (i.e., maximum 2 subdivisions if needed), for corner elements, and for slopes elements, and for free surface elements above the slope. This was done using an editor, after generation of the raw data.

The next task is now to specify values for all the parameters in the 4 first lines of the input file, as described in Table 1. For the present application, these values have been set as follows,

```
254  185    1 1200    1    3    4  100    0    0  0.000000
  1    1    2    0    0    0    0    0       0.7       0.7
 0.0900   0.00000   90.0000    1.0000    0.0000    0.0000    1.0000
 0.0000    1.0000    1.0000    0.0000    0.1200    0    1    0    1 0
```

Details of the above parameter values are as follows,

- In the first line, we see, simplified data will be saved (1), we will calculate up to 1200 time loops (1200), a solitary wave will be generated using a piston wavemaker (3), four intervals on the bottom will move with the wavemaker motion (4), and results will be printed every 100 loops.

- In the second line, we see, the free surface is a Dirichlet boundary with quasi-spline elements, the leftward lateral boundary is a wavemaker, and all other boundaries are Neuman's impermeable. Maximum angles for adaptive integration are 0.7 rad.

- In the third line, we see, the initial time step $\Delta t_o = 0.09$. The initial Courant number is thus $C_o = \Delta t_o/(\Delta x_o\sqrt{gh_o}) = 0.36$ (with $g = h_o = 1$). The initial time is $t_o = 0.$, the maximum time for stepping is $t_{max} = 90.0$, the water density $\rho = 1.0$, and the gravity $g = 1.0$.

- In the fourth line, we see, the depth is constant to $h_o = 1.0$ (with the unit value of other parameters, we see, the problem is a nondimensional one), the wave height is $H_o = H_o/h_o = 0.12$.

These lines have to be placed at the top of the above listed input data file, using an editor.

The data file can now be given a name (e.g., *runup.data*), and the problem is ready to be run as described in section 5.5 (command : **radia runup e a**). Similar data can be generated for the case with a 45° slope, by just changing the angle Θ in the data file for GENER. The average CPU time used per time for this run is 3.3s, on an IBM9000, i.e., 66min for the whole run of 1200 time loops.

Results for the free surface elevation at successive times are presented in Fig. 10-12 (20° slope), and 14-16 (45° slope). One sees, waves propagate from left to right, up to about $t = 43$, and 41, respectively. The maximum runup calculated on both slopes is $R_u = 2.351$ (at $t = 43.07$), and $R_u = 2.275$ (at $t = 41.16$), respectively, which agrees quite well with computations by Liu *et al.* [50], and experiments by Hall & Watts [41] (1953). Waves then rundown, reflect on the slopes, and propagate backward into the numerical tank, trailing a (well resolved) tail of oscillations behind them, and one sees, these oscillations are more pronounced for the smaller slope. After time $t = 60$, the leading oscillations in the reflected waves are seen to reflect on the wavemaker.

Fig. 13 and 17 show indicators of global accuracy of computations for each case, respectively. These are the relative errors on total wave energy $\Delta e/e$, and volume $\Delta v/v$, in which $e = 0.06762$ and $v = 0.83227$ for the generated solitary wave, $\Delta v(t)$ is given in file *volu.data*, and $\Delta e(t)$ is given in file *et.data*. We see, both of these errors are very small

quantities for the initial stages of wave propagation ($\mathcal{O}(10^{-7})$), and then gradually increase. Errors both temporarily decrease during runup and rundown of the waves on the slopes, and then increase to stabilize at about $\mathcal{O}(10^{-3})$ or smaller.

Many detailed results can be saved, listed, plotted, and discussed for these cases, as detailed in section 5. These, however, can not be listed in the present report due to lack of space.

6.3 Cnoidal wave runup on a steep slope

The runup of a cnoidal wave of incident height $\frac{H_o}{h_o} = 0.10$, and period $T\sqrt{g/h_o} = 20$ (i.e., $\omega = 0.31416$, for which $L/h_o \simeq 20$) is calculated over a slope of angle $\Theta = 20°$, using the same discretization and initial data as for the first case with a solitary wave in the previous section.

More specifically, the 4 lines of parameters in the input file *runup.data* now look as follows,

```
254  185    1 1200    1    3    4  100    0    0  0.000000
  1    1    2    0    0    0    0    0      0.7       0.7
 0.0900   0.00000   90.0000    1.0000    0.0000    0.0000    1.0000
 0.0000    1.0000    1.0000   0.31416    0.1000    0    1    0    1 0
```

in which parameters *omega* and *voh*, have been set to the right value, and the rest of the data is unchanged.

Results for the free surface elevation at successive times are presented in Fig. 18-20. One sees, waves propagate from left to right, and a first crest runs-up the slope at about $t = 36$, reflects and propagates back into the tank, interacts with the second crest to produce a slightly higher runup for the second crest at about $t = 56$, and so forth.

Fig. 21 shows the relative error, $\Delta V/V$, on total volume of the computational domain ($V = 28.626$), as a function of time. One sees, this error is very small during all computations. Fig. 21 also shows the paddle trajectory $x_p(t)$, as calculated in the program for generating

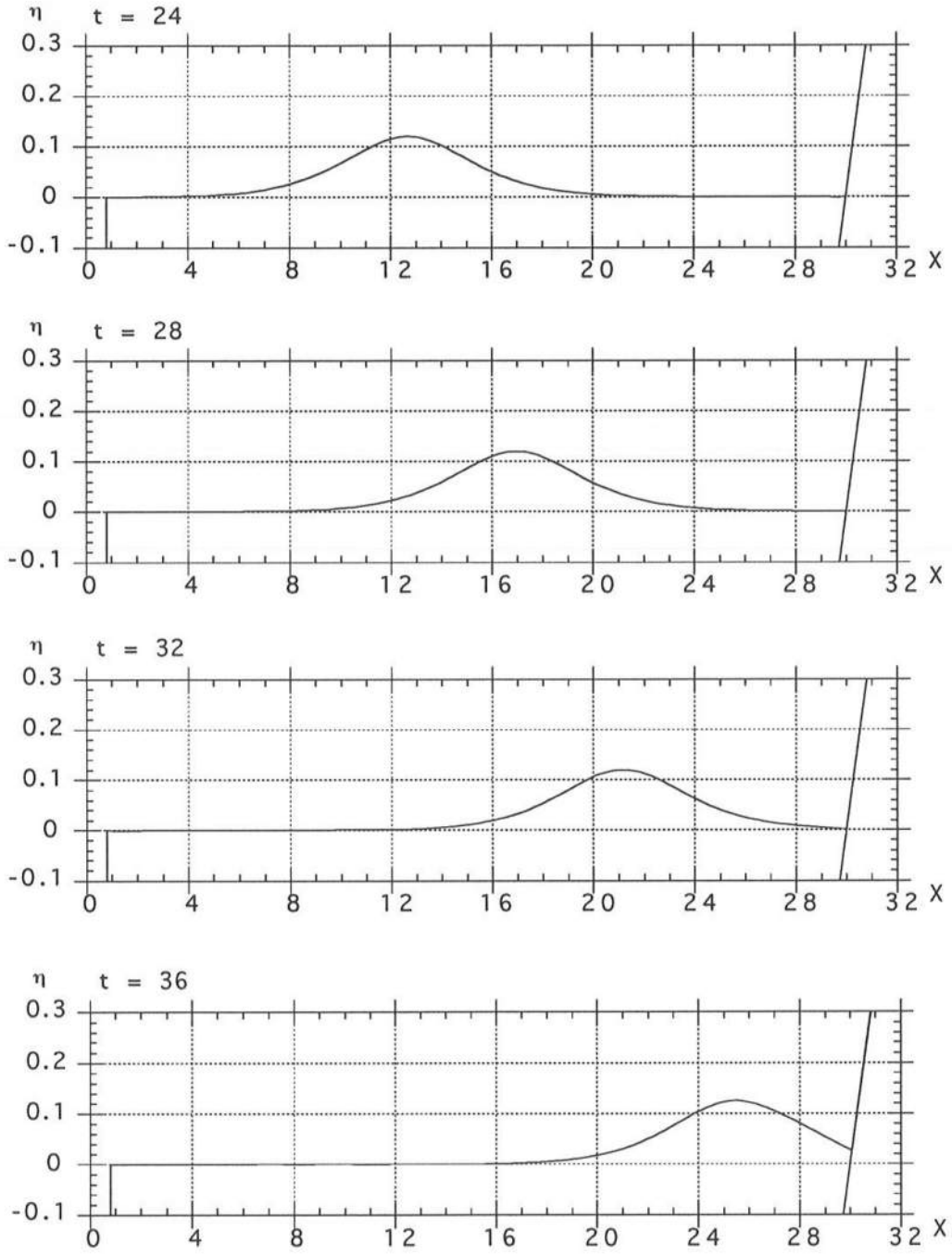


Figure 10: Runup of a solitary wave of height $H_o/h_o = 0.12$, on a 20° slope. Axes are non-dimensional with respect to depth h_o, and figures correspond to successive dimensionless time t

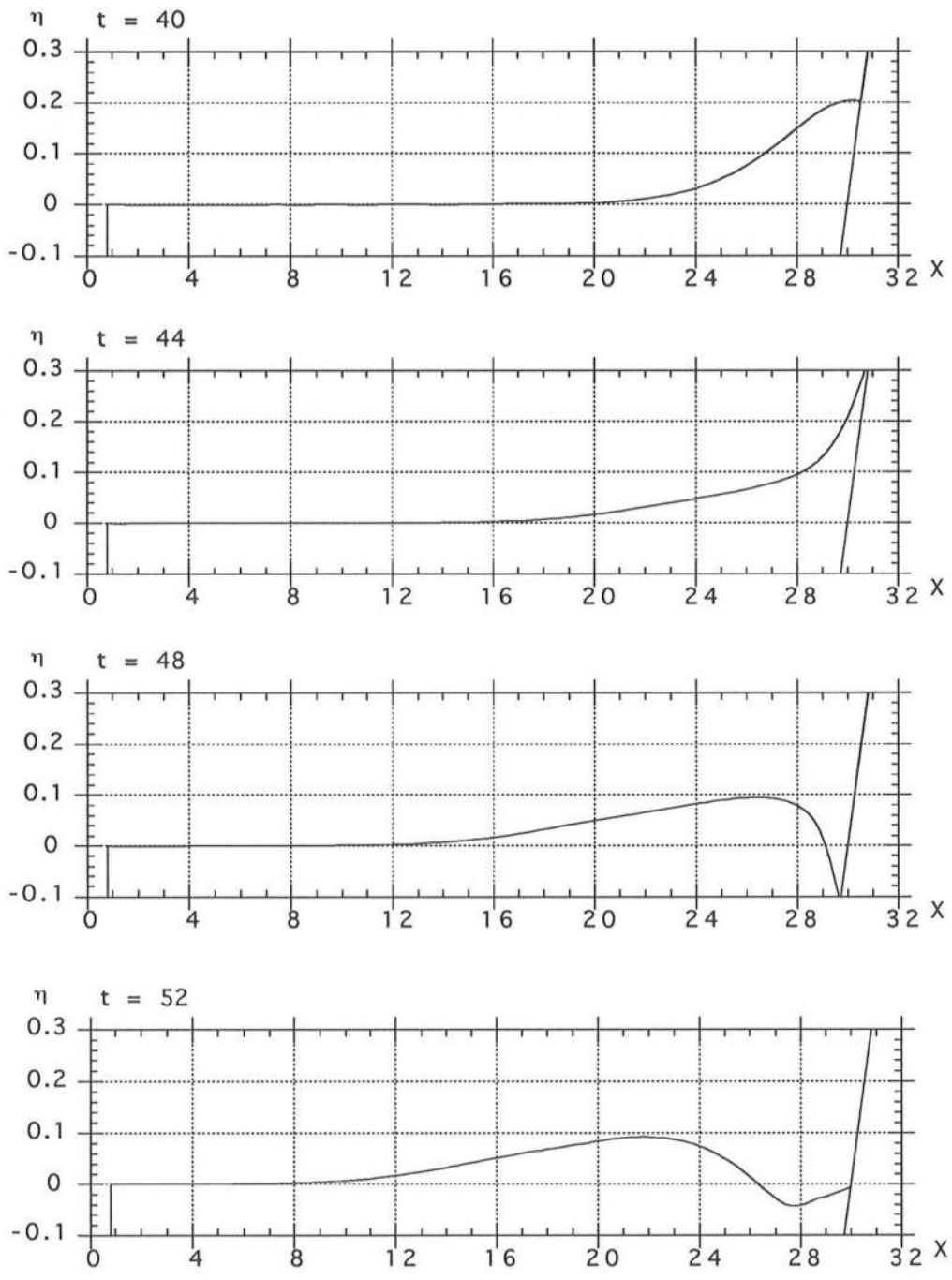


Figure 11: cont. from Fig. 10

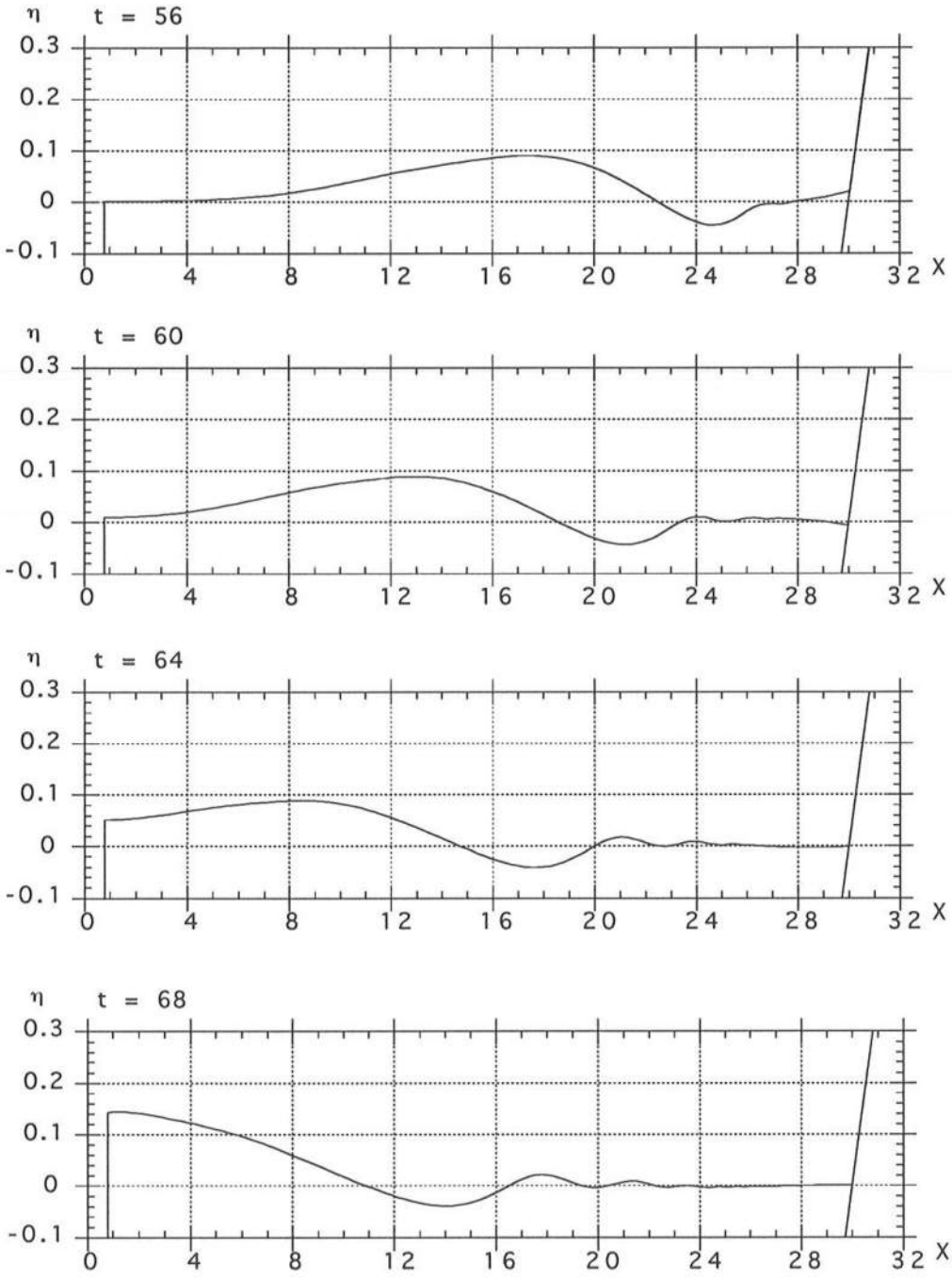
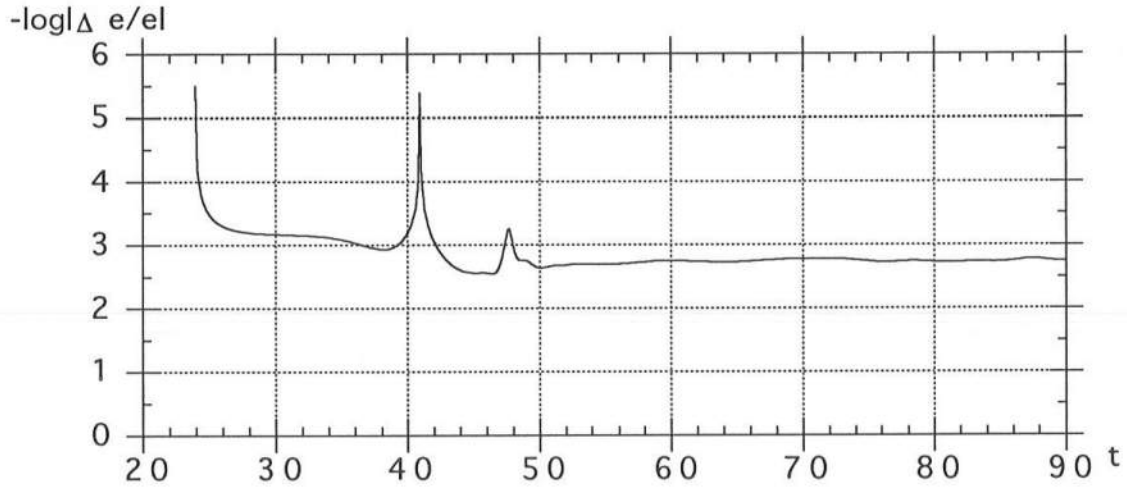


Figure 12: cont. from Fig. 10



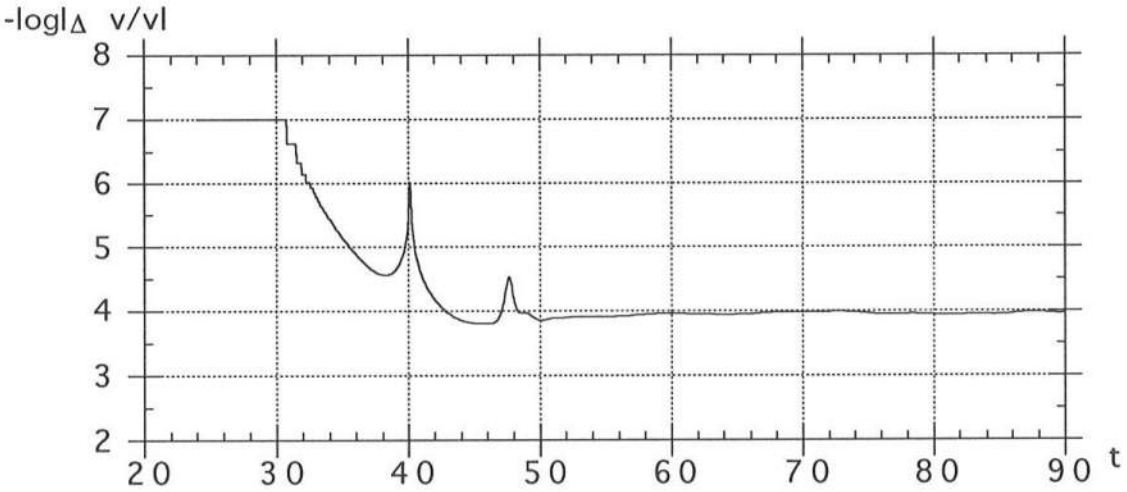


Figure 13: Error on total wave energy, $\Delta e/e$, and volume, $\Delta v/v$, for the computations reported in Fig. 10 to 12

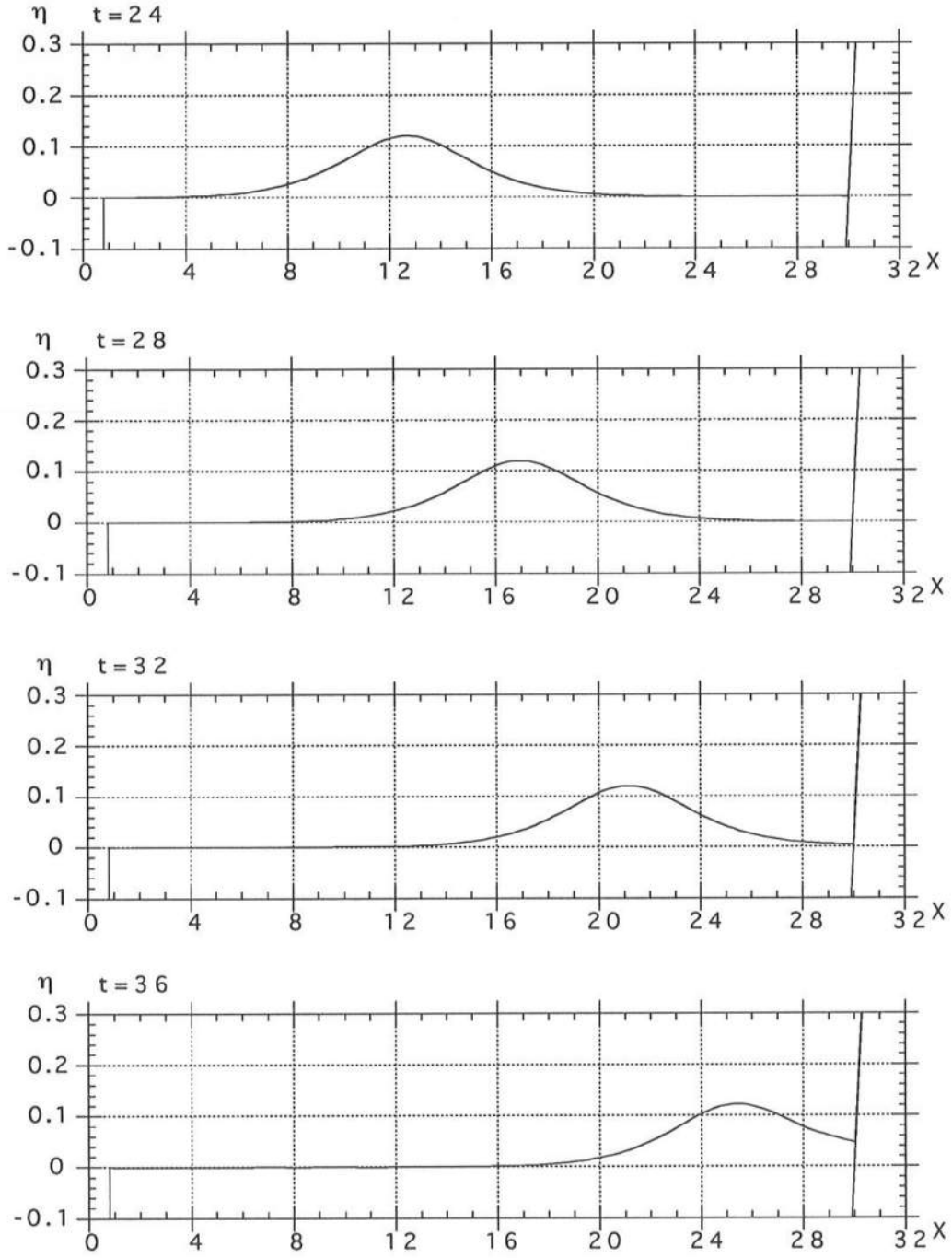


Figure 14: Runup of a solitary wave of height $H_o/h_o = 0.12$, on a 45° slope. Axes are non-dimensional with respect to depth h_o, and figures correspond to successive dimensionless time t

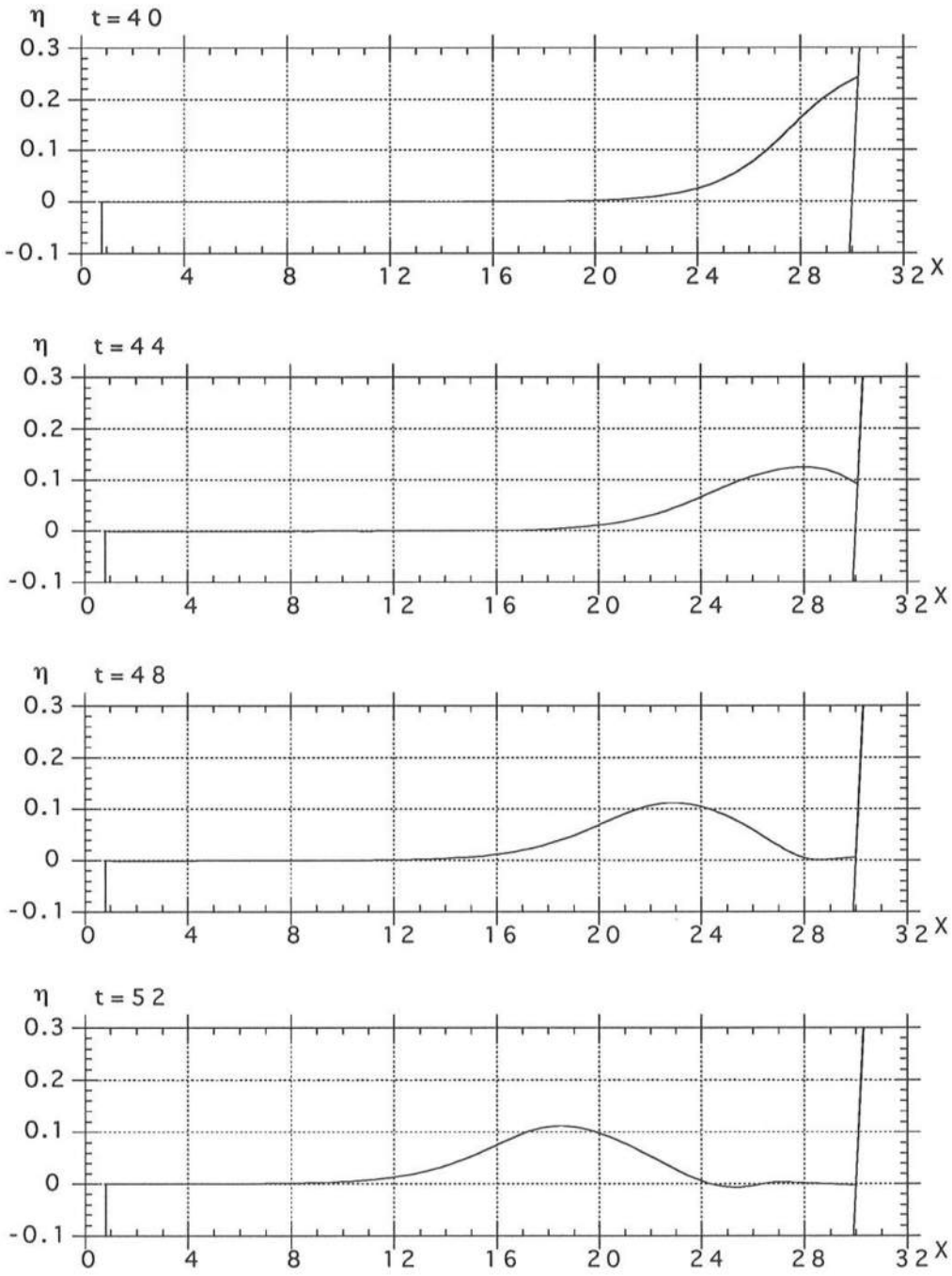


Figure 15: cont. from Fig. 14

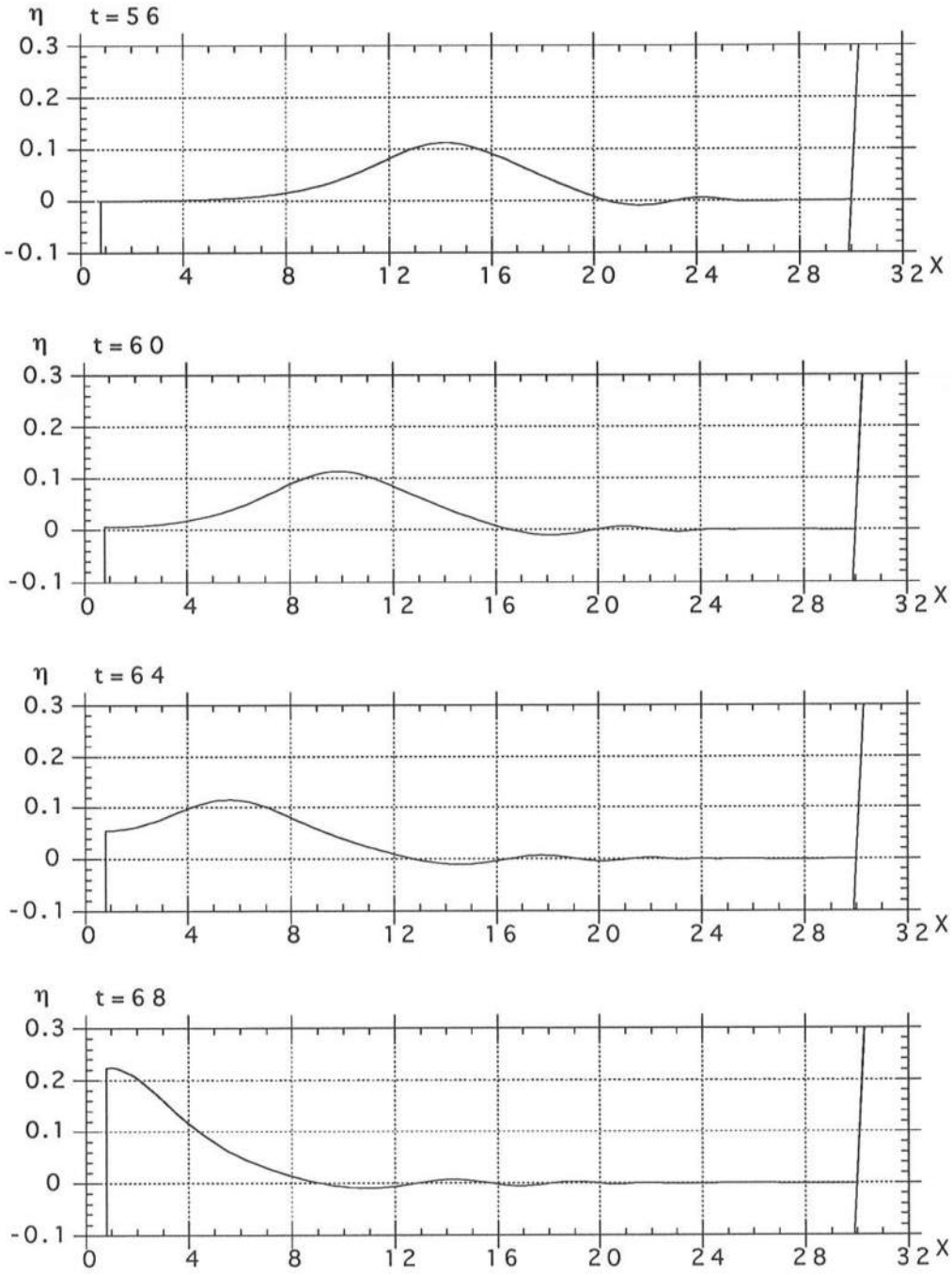


Figure 16: cont. from Fig. 14

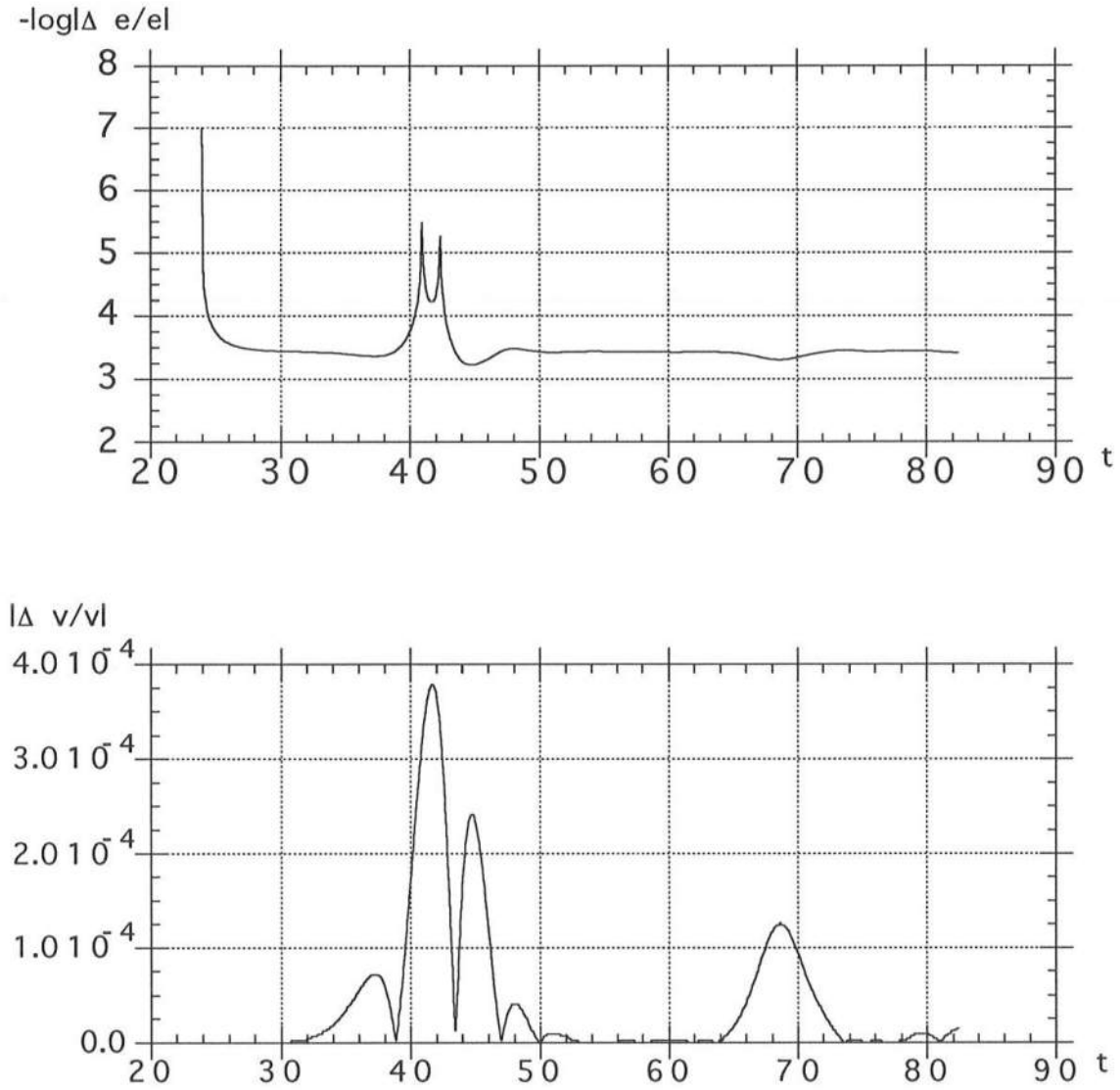


Figure 17: Error on total wave energy, $\Delta e/e$, and volume, $\Delta v/v$, for the computations reported in Fig. 14 to 16

the specified cnoidal wave, and the calculated runup at the shoreline $R(t)$. One sees, the first crest runs-up and down, to about twice its height, and the second and third crests run-up to about 2.4 times the incident wave height, while keeping the same rundown value. These results also fairly well agree with results by Liu *et al.* [50], as far as we can tell from their figure.

Many detailed results can again be saved, listed, plotted, and discussed for this case, but will not be shown in the preent report due to lack of space.

6.4 Solitary wave shoaling and breaking over a gentle slope

A case similar to those calculated by Grilli & Subramanya [32] (1993), and Otta *et al.* [56] is presented in the following, for incident solitary waves of initial height $\frac{H_o}{h_o} = 0.10, 0.15, 0.20$, shoaling and breaking over a 1:35 slope.

The computational domain is as sketched in Fig. 2. To improve accuracy of regular integrations in the upper part of the slope where the domain geometry becomes very narrow, a small shelf has been specified to the right of the domain, in depth $h_1 = 0.1h_o$, unlike in computations with steeper slopes reported in the previous sections. This is to avoid elements on different parts of the boundary from getting too close to each other, thus leading to a loss of accuracy of numerical integrations of the Green's function kernels. This change in geometry—as compared to a plain slope—does not affect shoaling and breaking of a solitary wave, provided these occur as observed in the present case, before reaching the shelf, i.e., for $\frac{x}{h_o} \leq 41.5$.

The free surface discretization has 180 two-node quasi-spline elements, with $\Delta x'_o = 0.25$, and there are 100 quadratic elements on the bottom and lateral boundaries. The total number of nodes is 384. The distance between nodes on the bottom is 0.5 in the constant depth region, and reduces to 0.40, 0.25, 0.20, 0.15, and 0.10 on the slope, in order to get increased resolution where depth decreases. The distance between nodes is 0.15 on the shelf

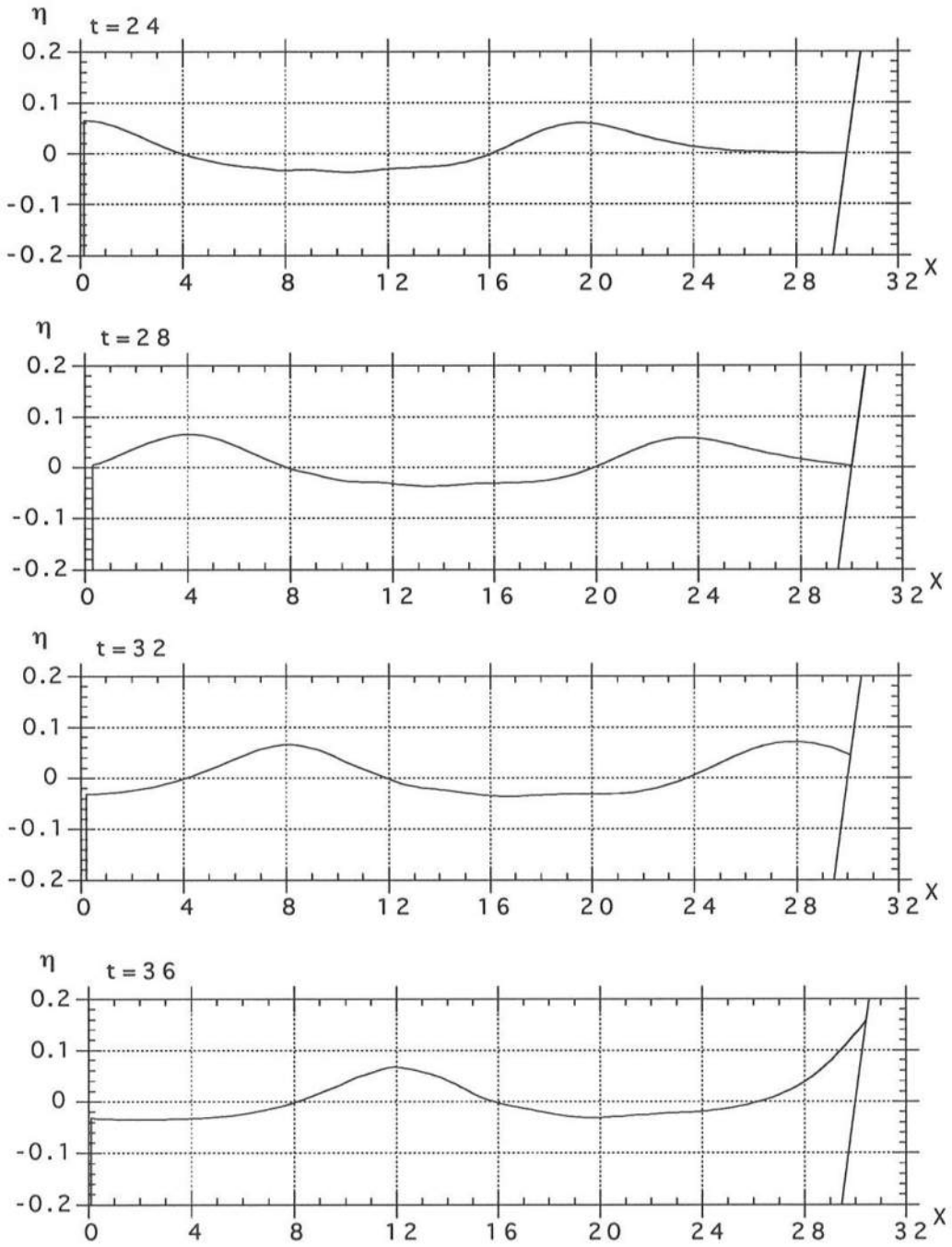


Figure 18: Runup of a cnoidal wave of height $H_o/h_o = 0.10$, and period $T' = T\sqrt{g/h_o} = 20$, on a 20° slope. Axes are non-dimensional with respect to depth h_o, and figures correspond to successive dimensionless time t

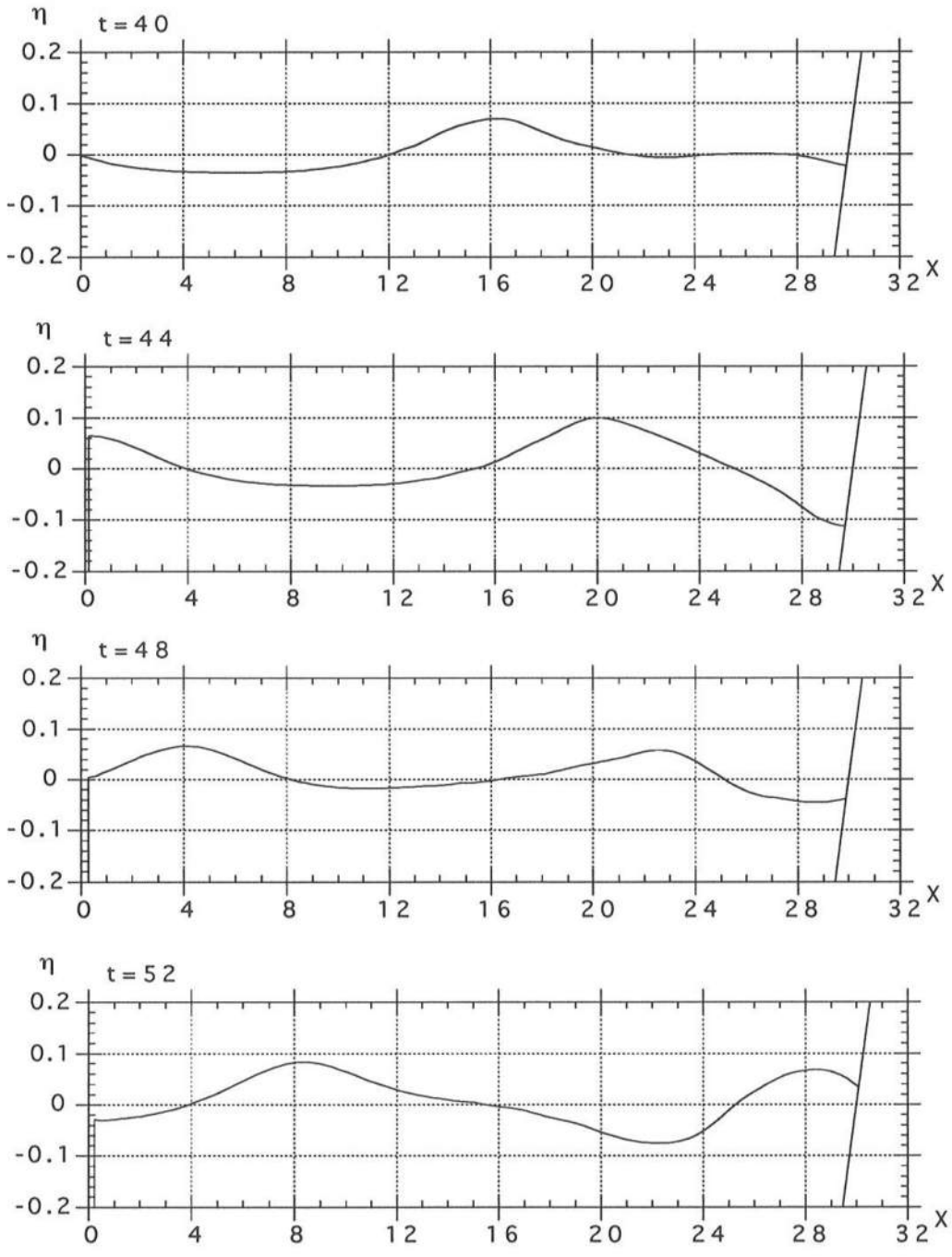


Figure 19: cont. from Fig. 18

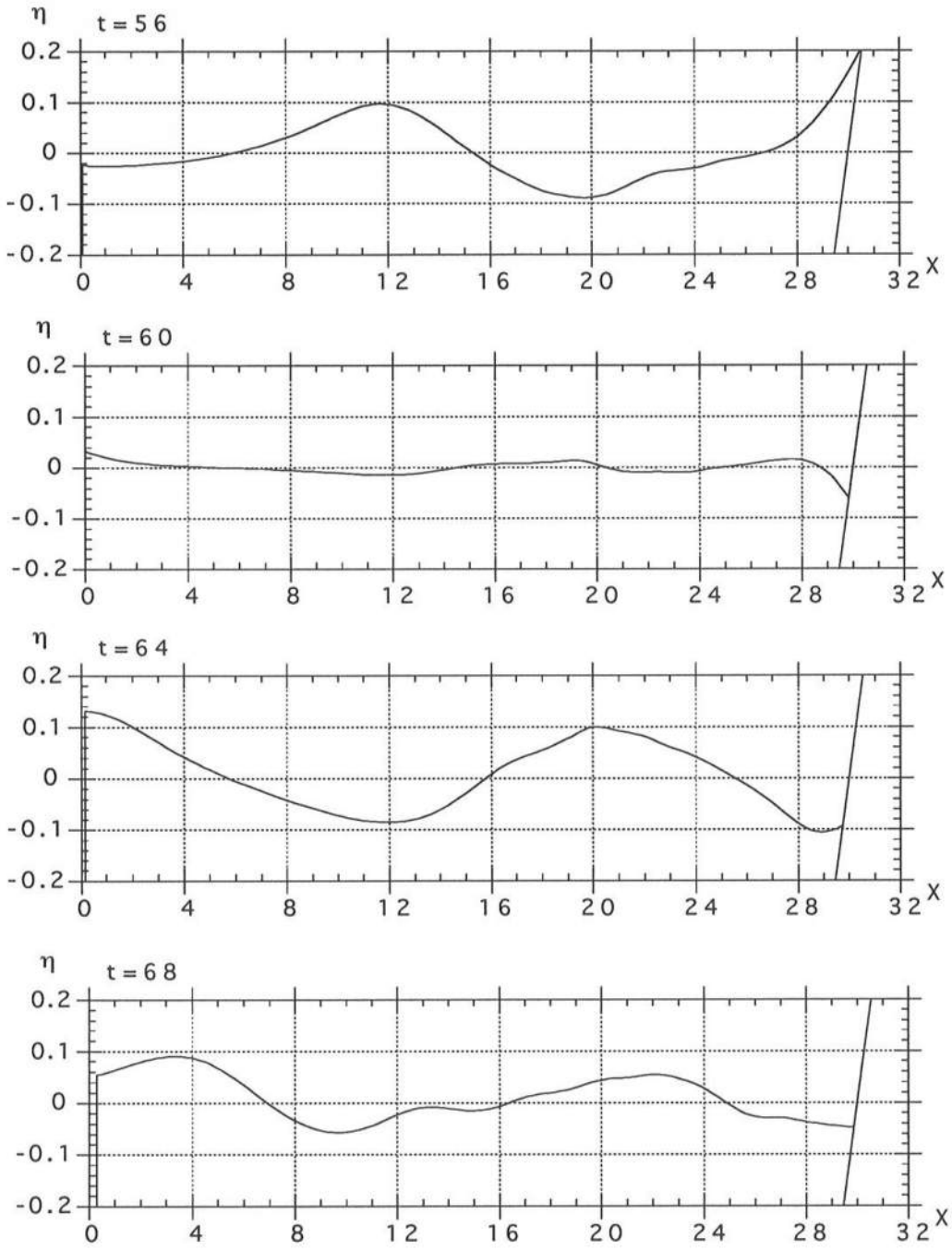


Figure 20: cont. from Fig. 18

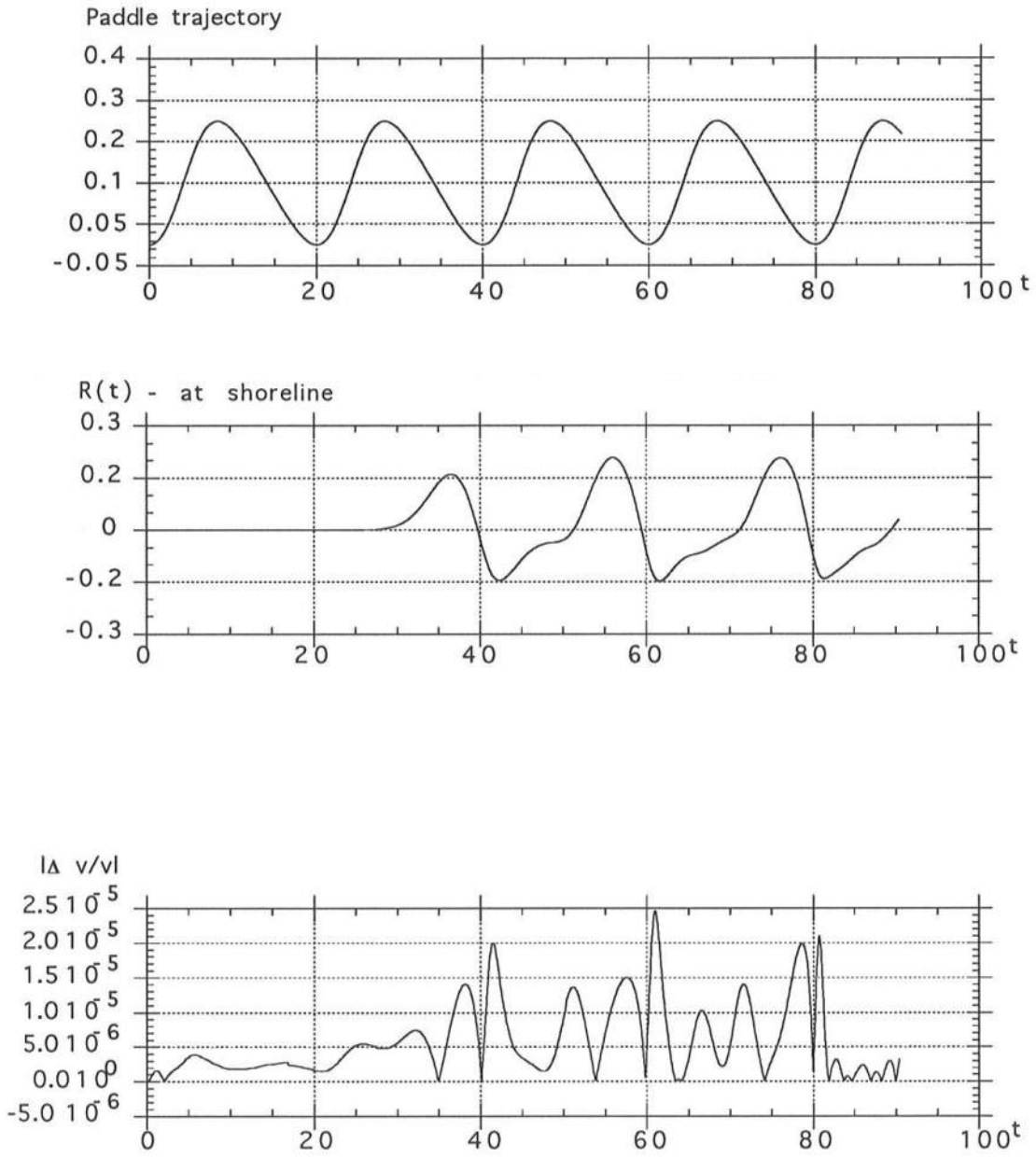


Figure 21: Horizontal motion of piston wavemaker $x_p(t)$, runup at the slope $R(t)$, and error on computational domain volume, $\Delta V/V$, for the computations reported in Fig. 19 to 20

bottom. Adaptive integration with up to 2^{10} subdivisions (as function of the geometry) is specified on the free surface and on the bottom, for the elements located between $x' = 36$ and 45. The mesh Courant number is $C_o = 0.50$ and, hence, $\Delta t'_o = 0.125$. With these data, the CPU time is 10.2sec per time step (IBM9000).

The incident solitary waves are generated on the leftward lateral boundary of the domain, using the numerical piston wavemaker. Fig. 22 shows stages of wave shoaling and breaking calculated for $\frac{H}{h_o} = 0.20$. During propagation, time step reduces down to $\Delta t' = 0.020$ at the time of breaking ($t' = 43.92$). The total number of time steps is 680 and the average time step is 0.065.

In Fig. 22a, free surface profiles are shown at six different times, up to the instant of wave instability by spilling breaking (last profile). Fig. 22a and 22b show blow-ups of the region over the slope where breaking occurs. Fig. 22b shows, breaking occurs at $x'_b = 35.8$, with a wave height $H_b = 0.364$, and a local ratio wave height over depth ("breaking index") $\frac{H_b}{h_b} = 1.38$. This index is much larger than the usual design value for gentle slopes (~ 0.80), and agrees to within 5% with measurements by Grilli *et al.* [39] (1993). Such an agreement can only be obtained when full nonlinearity is used in the equations.

Detailed results of calculations for the three wave heights show, maximum relative errors on volume and total energy are less than 0.01% for $x' \leq 28.$, i.e., more than half the way up the slope (Fig. 23). Notice, for $\frac{H_o}{h_o} = 0.10$, the wave energy is $e = 0.14977$, and the volume is $v = 1.09765$. In the last stages of shoaling over the slope, however, discretization nodes gather in highly curved regions of the boundary where hydrodynamic jets are forming (e.g., crest of an impending breaking wave in Fig. 22c), and scatter at some other areas (e.g., wave troughs), leading to a less accurate description of the flow, and to larger errors. Fig. 22c shows, due to the rather long computational domain, only a few nodes end up approximating the breaking wave jet on the free surface. This, hence, limits the jet resolution or, more exactly, the period of time over which calculations are able to accurately follow the jet further than the overturning point. To improve on this, it would be necessary to either use

regridding, or a finer initial discretization.

In the present application, computations have been interrupted when relative errors become greater than 1.5%. Fig. 23b shows, the energy error first reaches this threshold. The last wave profile shown in all Figs. 22 corresponds to the time of maximum energy error.

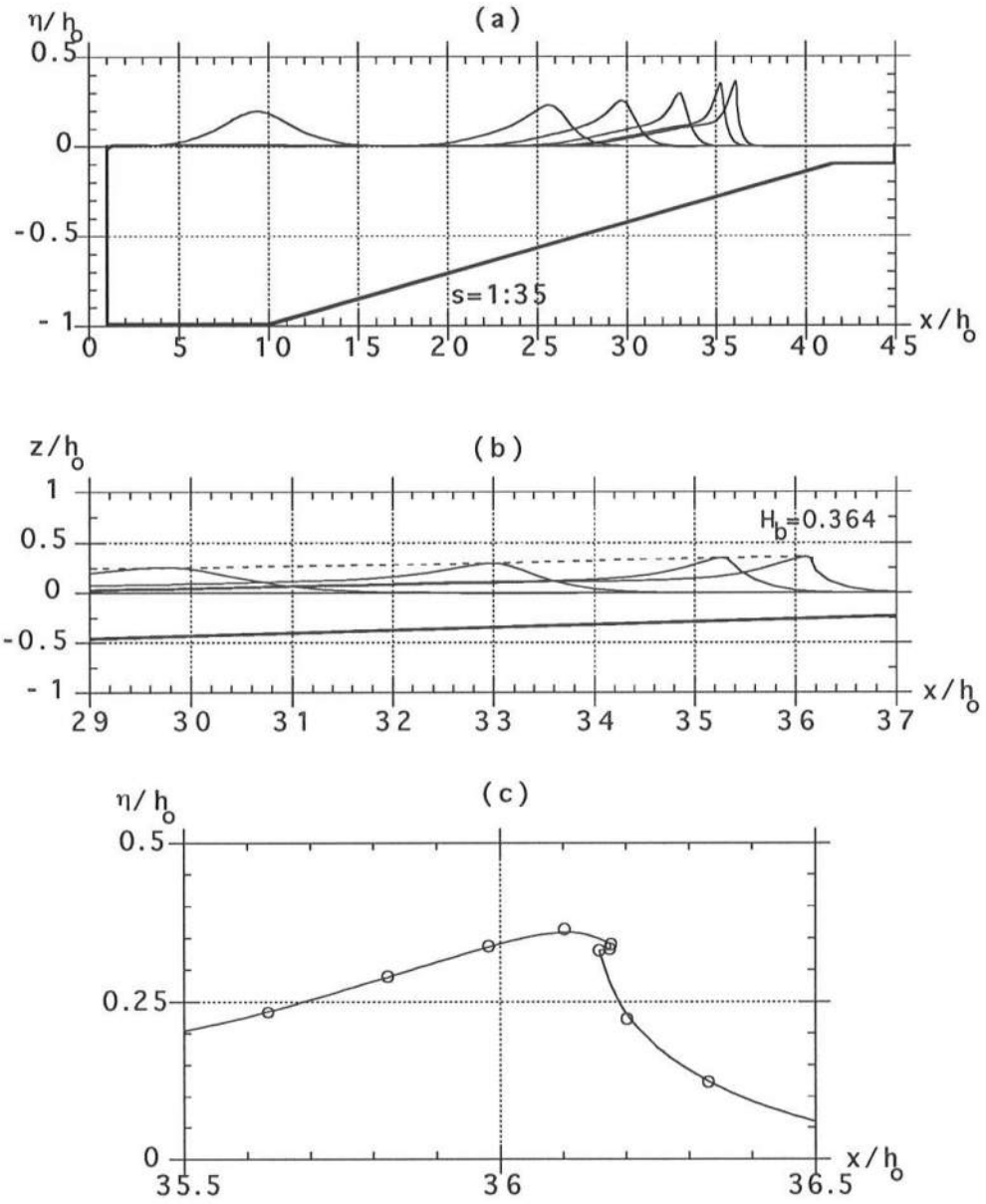


Figure 22: Shoaling of a solitary wave with initial height $H'_o = 0.20$, over a 1:35 slope. PLots correspond to : (a) six profiles at time $t' = 17.46$, 37.21, 39.93, 40.40, 43.27, and 43.92 (left to right); (b) blow-up in full scale of last four profiles in (a), (- - - -) shoaling curve; (c) blow-up of last profile in (a), (o) BEM discretization nodes.

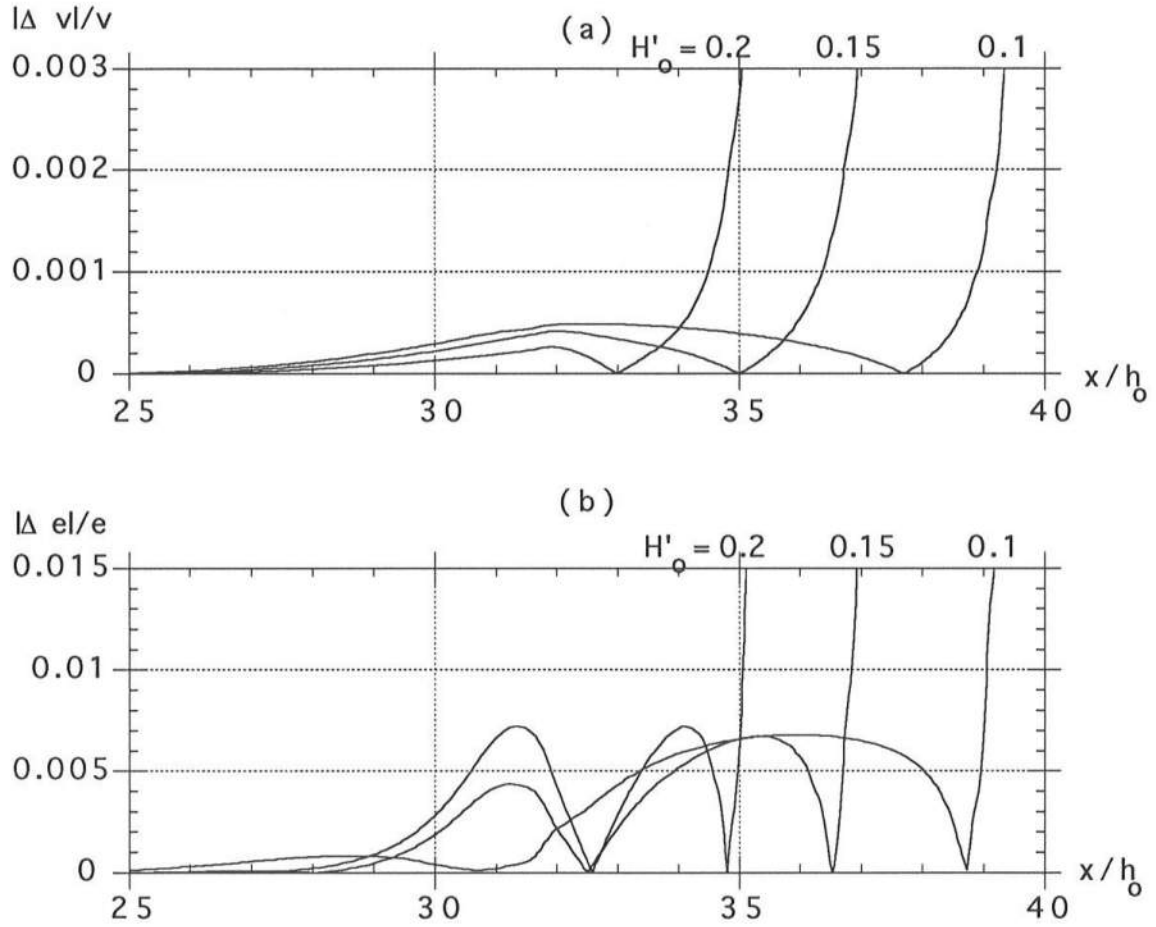


Figure 23: Relative errors on wave volume v (a), and total energy e (b), for the calculations in Fig. 22, with solitary waves of incident height H'_o.

References

[1] Abramowitz, M. & Stegun, I.A. *Handbook of Mathematical Functions.* Dover Pub. Inc. New York, 1965.

[2] Brebbia, C.A. *The Boundary Element Method for Engineers,* John Wiley & Sons, U.K., 1978.

[3] Brorsen, M. & Larsen, J. Source Generation of Nonlinear Gravity Waves with the Boundary Integral Method. *Coastal Engineering* **11,** 93-113, 1987.

[4] Camfield, F.E. & Street, R.L. Shoaling of Solitary Waves on Small Slopes. *ASCE,* **WW95,** 1-22, 1969.

[5] Carrier, G.F. Gravity Waves on Water of Variable Depth. *J. Fluid Mech.* **24** (4), 641-659, 1966.

[6] Carrier, G.F. & Greenspan, H.P. Water Waves of Finite Amplitude on a Sloping Beach. *J. Fluid Mech.* **4** (1), 97-110, 1958.

[7] Cointe, R. Numerical Simulation of a wave Channel. *Engineering Analysis with Boundary Elements* **7** (4), 167-177, 1990.

[8] Cointe, R. Quelques aspects de la simulation numérique d'un canal à houle. *Thèse de Docteur de l'Ecole Nationale des Ponts et Chaussées,* 284 pps., 1989.

[9] Cooker, M. The Interaction Between Steep Water Waves and Coastal Structures. *Ph.D. Dissertation.* School of Mathematics, University of Bristol, England, 1990.

[10] Cooker, M. A Boundary-integral Method for Water Wave Motion Over Irregular Beds. *Engineering Analysis with Boundary Elements* **7** (4), 205-213, 1990.

[11] Cooker, M. and Peregrine, D.H. Violent Water Motion at Breaking-Wave Impact. In *Proc. 22nd Intl. Conf. on Coastal Engineering* (ICCE22, Delft, The Netherland, July 90). Vol **1** , pps. 164-176. ASCE edition, 1991.

[12] Cooker, M.J., Peregrine, D.H., Vidal, C. & Dold, J.W. The Interaction Between a Solitary Wave and a Submerged Semicircular Cylinder. *J. Fluid Mech.* **215,** 1-22, 1990.

[13] Dean, R.G. & Dalrymple R.A. *Water Wave Mechanics for Engineers and Scientists* Prentice-Hall, 1984.

[14] Dold, J.W. & Peregrine, D.H. An Efficient Boundary Integral Method for Steep Unsteady water Waves. *Numerical methods for Fluid Dynamics II* (ed. K.W. Morton & M.J. Baines), pp. 671-679. Clarendon Press, Oxford, 1986.

[15] Dommermuth, D.G. & Yue, D.K.P. Numerical Simulation of Nonlinear Axisymmetric Flows with a Free Surface. *J. Fluid Mech.* **178,** 195-219, 1987.

[16] Dommermuth, D.G. & Yue, D.K.P., Lin, W.M., Rapp, R.J., Chan, E.S. & Melville, W.K. Deep-Water Plunging Breakers : a Comparison between Potential Theory and Experiments. *J. Fluid Mech.* **189,** 423-442, 1988.

[17] Driscoll, A.M., Dalrymple, R.A. & Grilli Harmonic Generation and Transmission Past a Submerged Rectangular Obstacle. In *Proc. 23rd Intl. Conf. on Coastal Engineering* (ICCE23, Venice, Italy, Oct. 92) Vol. **1**, pps. 1142-1152. ASCE edition, 1993.

[18] Fenton, J.D. & Rienecker, M.M. A Fourier Method for Solving Nonlinear Water-Wave Problems : Application to Solitary-Wave Interactions. *J. Fluid Mech.* **118,** 411-443, 1982.

[19] Freilich, M.H. & Guza, R.T. Nonlinear Effects on Shoaling Surface Gravity Waves *Phil. Trans. R. Soc. Lond.* **A311,** 1-41, 1984.

[20] Goring D.G. Tsunamis - The Propagation of Long Waves onto a Shelf. *W.M. Keck Laboratory of Hydraulics and Water Resources, California Institute of Technology, Report No.* **KH-R-38,** 1978.

[21] Gravert, P. Numerische Simulation Extremer Schwerewellen im Zeitbereich mit Direkter Randelementmethode und Zeitschrittverfahren (Ph.D. Dissertation). *Fortschritt-Berichte VDI Verlag Düsseldorf, Reihe 7 (Strömungs-technik) No.* **132.** 1987.

[22] Greenhow, M. Wedge Entry into Initially Calm Water. *Applied Ocean Res.* **9,** 214-223, 1987.

[23] Griffiths, M.W., Easson, W.J. & Greated, C.A. Measured Internal Kinematics for Shoaling Waves with Theoretical Comparisons *J. Waterways, Port Coastal and Ocean Engng.* **118** (3), 280-299, 1992.

[24] Grilli, S. Wave Overturning Induced by Moving Bodies. Application to Slender Ship Wave Resistance. *Invited paper* in *Proc. 1st Intl. Conf. on Computational Modelling of Free and Moving Boundary Problems* (Southampton, England, July 91)(ed. L.C. Wrobel & C.A.

Brebbia) Vol. **1**, pp. 75-90. Computational Mechanics Publications, de Gruyter, Southampton, 1991.

[25] Grilli, S. Modeling of Nonlinear Wave Motion in Shallow Water. Chapter 3 in *Computational Methods for Free and Moving Boundary Problems in Heat and Fluid Flow* (eds. L.C. Wrobel & C.A. Brebbia), pps. 37-65, Computational Mechanics Publication, Elsevier Applied Sciences, London, UK, 1993.

[26] Grilli, S., Losada, M.A. & Martin, F. Kinematics of Solitary Wave Breaking over Submerged Structures : Comparison between Nonlinear Computations and Experimental Results. In *Proc. 4th Intl. Conf. on Hydraulic Engineering Software* (HYDROSOFT92, Valencia, Spain, July 92) (eds. W.R. Blain and E. Cabrera), Fluid Flow Modelling, pp. 575-586. Computational Mechanics Publications, Elsevier Applied Science. (invited paper), 1992.

[27] Grilli, S., Losada, M.A. & Martin, F. Wave Impact Forces on Mixed Breakwaters. In *Proc. 23rd Intl. Conf. on Coastal Engineering* (ICCE23, Venice, Italy, October 92) Vol. **1**, pps. 1161-1174. ASCE edition, 1993.

[28] Grilli, S., Losada, M.A. & Martin, F. Characteristics of Solitary Wave Breaking Induced by Breakwaters. *J. Waterway, Port, Coastal, and Ocean Engng.* (accepted for publication), 1993.

[29] Grilli, S., Losada, M.A., Martin, F. & Svendsen, I.A. Nonlinear Shoaling and Impact of Waves on Coastal Structures. In *Proc. 9th Engineering Mechanics Conf.* (College Station, Texas, May 92) (eds. L.D. Lutes and J.M. Niedzwecki), pp. 79-82. ASCE edition.(invited paper), 1992.

[30] Grilli, S. Skourup, J. & Svendsen, I.A. The Modelling of Highly Nonlinear Waves : A Step Toward the Numerical Wave Tank. Invited paper in *Proc. 10th Intl. Conf. on Boundary Elements, Southampton, England,* Vol. **1** (ed. C.A. Brebbia), pp. 549-564. Computational Mechanics Publication. Springer Verlag, Berlin, 1988.

[31] Grilli, S. Skourup, J. & Svendsen, I.A. An Efficient Boundary Element Method for Nonlinear Water Waves. *Engineering Analysis with Boundary Elements* **6** (2), 97-107, 1989.

[32] Grilli, S. & R. Subramanya Nonlinear Wave Modeling in very Shallow Water. In *Proc. 15th Intl. Conf. on Boundary Elements in Engineering* (BEM15, Worcester, MA, November 1993), 16pp. Computational Mechanics Publication, Elsevier (in press), 1993.

[33] Grilli, S. & Svendsen, I.A. The Modelling of Nonlinear Water Wave Interaction with Maritime Structures. *Advances in Boundary Elements* (Proc. 11th Intl. Conf. on Boundary Elements, Cambridge, Massachusetts, USA), Vol. **2** (ed. C.A. Brebbia, J.J. Connor), pp. 253-268. Computational Mechanics Publication. Springer Verlag, Berlin, 1989.

[34] Grilli, S. & Svendsen, I.A. The Modelling of Highly Nonlinear Waves : Some Improvements to the Numerical Wave Tank. *Advances in Boundary Elements* (Proc. 11th Intl. Conf. on Boundary Elements, Cambridge, Massachusetts, USA), Vol. **2** (ed. C.A. Brebbia, J.J. Connor), pp. 269-281. Computational Mechanics Publication. Springer Verlag, Berlin, 1989.

[35] Grilli, S. & Svendsen, I.A. Computation of Nonlinear Wave Kinematics during Propagation and Runup on a Slope. *Water Wave Kinematics,* (Proc. NATO-ARW, Molde, Norway, May 89) (ed. A. Torum & O.T. Gudmestad) NATO ASI Series E: Applied Sciences Vol. **178,** 387-412. Klüwer Academic Publishers, 1990.

[36] Grilli, S. & Svendsen, I.A. Long Wave Interaction with Steeply Sloping Structures. In *Proc. 22nd Intl. Conf. on Coastal Engineering* (ICCE22, Delft, The Netherland, July 90) Vol. **2**, pps. 1200-1213. ASCE edition, 1991.

[37] Grilli, S. & Svendsen, I.A. Corner Problems and Global Accuracy in the Boundary Element Solution of Nonlinear Wave Flows. *Engineering Analysis with Boundary Elements,* **7** (4), 178-195, 1990.

[38] Grilli, S. & Svendsen, I.A. The Propagation and Runup of Solitary Waves on Steep Slopes. *Center for Applied Coastal Research, University of Delaware, Research Report No.* **91-4,** 1991.

[39] Grilli, S., Svendsen, I.A., Subramanya, R. & Veeramony J. Shoaling and Breaking of Long Waves on Plane Beaches. *Coastal Engineering* (submitted for publication), 1993.

[40] Grilli, S. Svendsen, I.A. & Otta, A.K. Corner Effects Using BEM for Nonlinear Waves. In *Computational Engineering with Boundary Elements* (Proc. 5th Intl. Conf. on Boundary Element Technology, BETECH90, University of Delaware, USA, July 90) (ed. S. Grilli, C.A. Brebbia & A.H-D. Cheng) Vol. **1**, pp. 101-118. Computational Mechanics Publications, Southampton, 1990.

[41] Hall, J.V. & Watts, J.W. Laboratory Investigation of the Vertical Rise of Solitary Waves on Impermeable Slopes. *Beach Erosion Board, US Army Corps of Engineer, Tech. Memo. No.* **33,** 14 pp, 1953.

[42] Hibberd, S. & Peregrine, D.H. Surf and Run-up on a Beach : a Uniform Bore, *J. Fluid Mech.* **95** (2), 323-345, 1979.

[43] Isaacson, M. de St. Q. Nonlinear Effects on Fixed and Floating Bodies. *J. Fluid Mech.* **120,** 267-281, 1982.

[44] Jansen, P.C.M. A Boundary Element Model for Nonlinear Free Surface Phenomena. *Delft University of Technology, Department of Civil Engineering Report No.* **86-2,** 1986.

[45] Kim, S.K., Liu, P.L.-F. & Liggett, J.A. Boundary integral Equation Solutions for Solitary Wave Generation Propagation and Run-up. *Coastal Engineering* **7,** 299-317, 1983.

[46] Kirby, J.T. Intercomparison of Truncated Series Solutions for Shallow Water Waves, *J. Waterways, Port, Coastal and Ocean Engng.* **117** (2), 143-155, 1991.

[47] Klopman, G. Numerical Simulation of Gravity Wave Motion on Steep slopes. *Delft Hydraulics Report No.* **H195,** 1988.

[48] Kobayashi, N. DeSilva, G.S. & Watson, K.D. Wave Transformation and Swash Oscillation on Gentle and Steep Slope. *J. Geoph. Res.* **94** (C1), 951-966, 1989.

[49] Kravtchenko, J. Remarques sur le calcul des amplitudes de la houle lineaire engendrée par un batteur. In *Proc. 5th Intl. Conf. on Coastal Engineering,* pp. 50-61, 1954.

[50] Liu, P.L., Cho, Y.S. & Kim, S.K. A Computer Program for Transient Wave Run-up *Research Report* School of Civil and Environmental Engng., Cornell University, 1992.

[51] Liu, P.L., Yoon, S.B. & Kirby, J.T. Nonlinear Refraction-diffraction of Waves in Shallow Water, *J. Fluid Mech.* **153,** 185-201, 1985.

[52] Longuet-Higgins, M.S. & Cokelet, E.D. The Deformation of Steep Surface Waves on Water - I. A Numerical Method of Computation. *Proc. R. Soc. Lond.* **A350,** 1-26, 1976.

[53] Mei, C.C *The Applied Dynamics of Ocean Surface Waves (2nd ed.).* World Scientific, New Jersey, 1983.

[54] Nakayama, T. Boundary Element Analysis of Nonlinear Water Wave Problems. *Intl. J. Numer. Meth. Engng.* **19,** 953-970, 1983.

[55] New, A.L., McIver, P. & Peregrine, D.H. Computation of Overturning Waves. *J. Fluid Mech.* **150,** 233-251, 1985.

[56] Otta, A.K., Svendsen, I.A. & Grilli, S.T. The Breaking and Runup of Solitary Waves on Beaches. In *Proc. 23rd Intl. Conf. on Coastal Engineering* (ICCE23, Venice, Italy, October 92) Vol. **2,** pps. 1461-1474. ASCE edition, 1993.

[57] Otta, A.K., Svendsen, I.A. & Grilli, S.T. Unsteady Free Surface Waves in Region of Arbitrary Shape. *CACR, University of Delaware, Research Report* **92-10,** 153pp, 1992.

[58] Pedersen, G. & Gjevik, B. Run-up of Solitary Waves *J. Fluid Mech.* **135,** 283-299, 1983.

[59] Peregrine, D.H. Long Waves on Beaches, *J. Fluid Mech.* **27** (4), 815-827, 1967.

[60] Peregrine, D.H. Breaking Waves on Beaches, *Ann. Rev. Fluid Mech.* **15,** 149-178, 1983.

[61] Roberts, A.J. Transient Free-Surface Flows Generated by a Moving Vertical Plate. *Q.J. Mech. Appl. Math.* **40** (1), 129-158, 1987.

[62] Romate, J.E. The Numerical Simulation of Nonlinear Gravity Waves. *Engineering Analysis with Boundary Elements* **7** (4), 156-166, 1990.

[63] Seo, Seung Nam & Dalrymple, R.A. An Efficient Model for Periodic Overturning Waves. *Engineering Analysis with Boundary Elements* **7** (4), 196-204, 1990.

[64] Skjelbreia, J.E. Observations of Breaking Waves on Sloping Bottoms by Use of Laser Doppler Velocimetry, *W.M. Keck Laboratory of Hydraulics and Water Ressources, California Institute of Technology, Report No.* **KH-R-48.** 1987.

[65] Skourup, J., Grilli, S. & Svendsen, I.A. Modelling of Steep and Breaking Waves by the Boundary Element Method. *Inst. Hydrodyn. and Hydraulic Engng., Technical University of Denmark, Progress Report No.* **68,** 59-71, 1988.

[66] Skourup, J., Jonsson, I.G. Grilli, S. & Svendsen, I.A. Computational Modelling of Velocities and Accelerations in Steep Waves. In *Water Wave Kinematics* (Proc. NATO ARW, Molde, Norway, May 89) (ed. A. Torum & O.T. Gudmestad), NATO ASI Series E: Applied Sciences Vol. **178,** 297-312. Klüwer Academic Publishers, 1990.

[67] Skourup, J., Jonsson, I.G. Svendsen, I.A. Grilli, S. & Larsen, J. Non-Linear Water Wave Modelling by a Boundary Integral Equation Method. In *Proc. 23rd I.A.H.R. Congress, Ottawa, Canada, August 89,* Vol. C, pp. 359-366. National Research Council Publication, Canada, 1989.

[68] Skyner, D.J., Gray, C. & Greated, C.A. A Comparison of Time-stepping Numerical Prediction with Whole-field Flow Measurements in Breaking Waves, Chapter in *Water Wave Kinematics* (ed. A. Torum & O.T. Gudmestad), NATO ASI Series E: Applied Sciences Vol. **178,** 491-508, 1990.

[69] Sobey, R.J. & Bando, K. Variations on Higher-Order Shoaling, *J. Waterways, Port, Coastal and Ocean Engng.* **117** (4), 348-368, 1991.

[70] Stiassnie, M. & Peregrine , D.H. Shoaling of Finite Amplitude Surface Waves on Water of Slowly-varying Depth, *J. Fluid Mech.* **97,** 783-805, 1980.

[71] Su, C.H. & Mirie, R.M. On Head-on Collisions between two Solitary Waves. *J. Fluid Mech.* **98** 509-525, 1980.

[72] Svendsen, I.A. & Grilli, S. Nonlinear Waves on Steep Slopes. *J. Coastal Research* **SI 7,** 185-202, 1990.

[73] Synolakis, C.E. The Runup of Solitary Waves. *J. Fluid Mech.* **185,** 523-545, 1987.

[74] Tanaka, M. The Stability of Solitary Waves. *Phys. Fluids* **29** (3), 650-655, 1986.

[75] Tanaka, M., Dold, J.W., Lewy, M. & Peregrine, D.H. Instability and Breaking of a Solitary Wave. *J. Fluid Mech.* **185,** 235-248, 1987.

[76] Vinje, T. & Brevig, P. Numerical Simulation of Breaking Waves. *Adv. Water Ressources* **4,** 77-82, 1981.

[77] Zelt, J.A. & Raichlen, F. A Lagrangian Model for Wave Induced Harbour Oscillations, *J. Fluid Mech.* **213,** 203-228, 1990.