A Boussinesq Model for
Wave Transformation in Coastal Regions,
with Application to a Submerged Coral Reef

by

Claus Skotner
M.Sc.Eng., Tech. Univ. Denmark

Doctor of Philosophy

Department of Civil Engineering
University of Queensland
Brisbane, Australia
October 1996
Preface

The present thesis is submitted in fulfilment of the requirements for the degree, Doctor of Philosophy (PhD). Unless otherwise stated, the material is original, and the author's own work. The material has not been submitted previously, in whole or in part, for a degree at this or any other university.

The study was carried out in the Department of Civil Engineering at the University of Queensland in Brisbane, Australia. Dr. Michael R. Gourlay acted as a supervisor from October 1, 1993, to September 30, 1996. The author gratefully acknowledges the guidance received during this period of time.

Special thanks are directed to Prof. Colin J. Apelt for many rewarding discussions, valuable guidance, and a genuine encouragement throughout the study.

A minor part of the study was performed in the fluids laboratory of the Civil Engineering Department. The author is indebted to the technicians for their assistance and support.

The costs of the present study were financed partly by the Department of Civil Engineering and the University of Queensland, and partly by the Danish Research Academy. This support is gratefully acknowledged.

Most of all I thank my wife for being a great companion.

Brisbane, October 1996

Claus Skotner

Claus Skotner
Abstract

Based on a set of Boussinesq-type equations with improved linear dispersion characteristics in deeper water the first part of the thesis describes the development of a computational model in a single horizontal dimension (2-D). The model can be used to simulate the evolution of relatively long, weakly nonlinear waves in water of constant or variable depth provided the bed slope is of the same order of magnitude as the ratio of the mean water depth and a typical wave length. The numerical solution method is based on the finite difference method and the computations are advanced in time by using a fourth-order accurate predictor-corrector method. A special technique is employed which allows the incident wave field to be generated inside the computational domain. A Fourier method is used to prescribe a form of the incident regular wave field which satisfies the governing equations on a horizontal bottom. Scattered waves leaving the fluid domain are absorbed in the vicinity of the model boundary by employment of damping terms in the mass and momentum equations. This ensures that the wave reflection from the boundary is insignificant.

The phase and amplitude portraits of the numerical solution are considered, and examples are given illustrating that the model conserves well basic properties such as the total mass and energy within the computational domain. The model is used to study the transformation of waves in water of variable depth. The results compare well with both existing laboratory measurements and analytical theory.

For practical simulations, e.g. wave evolution inside a proposed harbour, a numerical model is often required which covers two horizontal dimensions (3-D). Consequently, the model is extended to include the second horizontal dimension. Since the formulation is very general, waves can be propagated in virtually any geometry. The analytical manipulations required to generate the incident wave field internally become quite substantial in a formulation covering two horizontal dimensions, and the wave generation concept is therefore generalized and implemented in a simple and efficient way. A number of computational examples are given. These serve as a partial verification of the model.

The second part of the thesis considers the effect of spilling wave breaking and the development of waves in the surf zone. The effect of spilling wave breaking is incorporated into the two-dimensional model using the concept of surface rollers. Based on the assumption of a vertical redistribution of the horizontal velocity in a breaking wave a new set of equations is derived. The temporal development of the surface roller thickness is determined heuristically using an existing method. Although the mathematical basis is rather weak and the physical description is very crude the model has the potential to describe a variety of processes such as the fluctuating breaking point caused by random waves breaking on a beach and the important conversion of potential energy to kinetic energy in the outer region of the surf zone. The surf zone model is calibrated using a single set of laboratory data and subsequently verified.
by comparison with two other sets of laboratory data. The results show that the model is capable of predicting relatively accurately the mean water level and the wave height variation caused by regular waves breaking on a plane and gentle slope.

The last part of the thesis describes a laboratory experiment conducted in a closed wave flume. A description is given of both the laboratory facility and the equipment used as well as the procedures involved in the experiment. The bed profile represents a fringing coral reef located on the coast of Guam. Since the profile incorporates local bed slopes of the order 1 : 10, the main purpose of the experiment is to verify the validity of the surf zone model on steeper slopes. Six incident wave conditions are considered. These describe regular waves of small and finite height in intermediate depth water. In each test series the mean water level and the breaking point are recorded in the steady state. The experimental results are compared with those of the computational model. These indicate that the surf zone model generally predicts the mean water level relatively accurately although there is a tendency to underestimate the maximum wave-induced setup.
Contents

Preface i

Abstract iii

Contents v

Symbols xi

1 Introduction 1

2 The Theory of Long Waves 3

2.1 General ........................................ 3

2.2 Assumptions and Derivation Procedures ................. 3

2.3 Application of Boussinesq-type Equations in Deeper Water . 9

3 Numerical Solution Method 13

3.1 General ........................................ 13

3.2 Governing Equations. Boundary and Initial Conditions .... 14

3.3 Temporal Updating of the Computational Domain .......... 17

3.4 Discretization of Spatial Derivatives ..................... 20

3.4.1 Internal Nodes . 20
### CONTENTS

3.4.2 Special Nodes ................................................. 21
3.4.3 Boundary Nodes ............................................. 22
3.5 Absorption of Outgoing Waves ................................. 23
3.6 Internal Wave Generation ...................................... 24
3.7 Analytical Methods for the Determination of the Incident Wave Field ........................................ 28
  3.7.1 Regular Waves ............................................... 28
  3.7.2 Irregular Waves ............................................. 30
3.8 A Fourier Approximation Method for the Determination of the Incident Wave Field ........................................ 31
3.9 Description of Program Code .................................. 36

4 Properties of the Computational Model .......................... 39
  4.1 General ......................................................... 39
  4.2 Stability of the Computations ................................ 40
    4.2.1 Amplitude Portraits of the Numerical Schemes .............. 40
    4.2.2 Phase Portraits of the Numerical Schemes .................. 44
    4.2.3 Phase Error of the Numerical Solution .................... 46
  4.3 Conservation of Volume and Energy ............................ 47
  4.4 Sponge Layer Performance .................................... 50
  4.5 Regular Waves over a Horizontal Bottom ...................... 52
  4.6 Regular Waves Propagating onto a Gently Sloping Beach ......... 54
  4.7 The Solitary Wave Propagating onto a Shelf .................... 55

5 Wave Propagation in Two Horizontal Dimensions ............... 59
  5.1 General ......................................................... 59
## CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.2 Definitions and Governing Equations</td>
<td>59</td>
</tr>
<tr>
<td>5.3 Numerical Solution Procedure in Two Horizontal Dimensions</td>
<td>62</td>
</tr>
<tr>
<td>5.3.1 Preliminary Considerations</td>
<td>62</td>
</tr>
<tr>
<td>5.3.2 Temporal Updating of the Computational Domain</td>
<td>65</td>
</tr>
<tr>
<td>5.3.3 Approximation of Spatial Derivatives</td>
<td>67</td>
</tr>
<tr>
<td>5.3.4 Absorption of the Wave Field at Open Boundaries</td>
<td>67</td>
</tr>
<tr>
<td>5.3.5 Total Reflection of the Scattered Wave Field</td>
<td>68</td>
</tr>
<tr>
<td>5.3.6 Internal Generation of the Incident Wave Field</td>
<td>69</td>
</tr>
<tr>
<td>5.4 Description of the Code and the Required Input</td>
<td>74</td>
</tr>
<tr>
<td>5.5 Partial Verification of the 3-D Model</td>
<td>76</td>
</tr>
<tr>
<td>5.5.1 Ring Test</td>
<td>76</td>
</tr>
<tr>
<td>5.5.2 Diffraction of Deep Water Waves Around a Breakwater</td>
<td>76</td>
</tr>
<tr>
<td>5.5.3 Refraction-Diffraction Test of Regular Waves Propagating over a Semi-circular Shoal</td>
<td>79</td>
</tr>
<tr>
<td>5.5.4 Waves Propagating into a Fictitious Harbour</td>
<td>83</td>
</tr>
<tr>
<td>6 Dissipation of Wave Energy</td>
<td>87</td>
</tr>
<tr>
<td>6.1 General</td>
<td>87</td>
</tr>
<tr>
<td>6.2 Review of Methods</td>
<td>87</td>
</tr>
<tr>
<td>6.2.1 Qualitative Description of the Surf Zone</td>
<td>87</td>
</tr>
<tr>
<td>6.2.2 Analytical and Experimental Methods</td>
<td>89</td>
</tr>
<tr>
<td>6.2.3 Flow Models</td>
<td>93</td>
</tr>
<tr>
<td>6.3 Inclusion of the Simplified Effect of Wave Breaking</td>
<td>95</td>
</tr>
<tr>
<td>6.3.1 Rederivation of Governing Equations</td>
<td>95</td>
</tr>
<tr>
<td>6.3.2 Temporal Development of the Surface Roller</td>
<td>100</td>
</tr>
</tbody>
</table>
6.3.3 Details of the Numerical Solution Procedure ............................. 101
6.4 Inclusion of the Bottom Friction ................................................. 102
6.5 Properties of the Surf Zone Model ............................................. 103
   6.5.1 Computation of the Wave Height Variation and the Mean Water Level 103
   6.5.2 The Influence of the Wave Breaking Parameters on the Results ....... 111
   6.5.3 Computation of the Undertow ............................................. 113

7 Application to a Submerged Coral Reef ........................................ 117
   7.1 General ............................................................................. 117
   7.2 Introduction to Coral Reefs .................................................. 117
   7.3 Description of the Laboratory Experiment .................................. 119
       7.3.1 The Wave Basin Facility .............................................. 119
       7.3.2 Construction of the Bed Profile ..................................... 120
       7.3.3 Data Acquisition ....................................................... 123
       7.3.4 Generation of the Incident Wave Field .............................. 124
       7.3.5 Experimental Procedure ............................................ 125
   7.4 Experimental Results .......................................................... 128
   7.5 Computational Results ........................................................ 132

8 Summary and Conclusion ............................................................ 139

References .................................................................................... 141

A Double-Sweep Method ............................................................... 151

B Code - One Horizontal Dimension ................................................ 153
   B.1 Abm.2D ............................................................................ 153
CONTENTS

B.2 Boundary.2D ................................................................. 154
B.3 Breaking.2D ................................................................. 164
B.4 Fourier.2D ................................................................. 168
B.5 Mathfunc.2D ............................................................... 176
B.6 Solution.2D ................................................................. 180
B.7 Variable.2D ................................................................. 188
B.8 Various.2D ................................................................. 191

C  Code - Two Horizontal Dimensions 199

C.1 Abm.3D ................................................................. 199
C.2 Boundary.3D ............................................................. 200
C.3 Fourier.3D ................................................................. 211
C.4 Grid.3D ................................................................. 218
C.5 Solution.3D ............................................................... 226
C.6 Variable.3D ............................................................... 237
C.7 Various.3D ............................................................... 240
Symbols

General Remarks Regarding the Notation

A symbol superscribed by an asterisk denotes a dimensional variable. All other quantities are dimensionless. A symbol subscribed by an I designates that the incident wave field is considered. Analogously, a symbol subscribed by an r refers to the reflected wave field.

It is believed to be redundant to present both the dimensional and the dimensionless quantities, and hence the lists given below contain almost exclusively the dimensionless variables.

Roman Symbols

\[ a_m \] Function of the still water depth and its spatial derivatives Sec. 3.2
\[ A_1 \] Constant Sec. 5.2
\[ A_2 \] Constant Sec. 5.2
\[ B_1 \] Constant Sec. 5.2
\[ B_2 \] Constant Sec. 5.2
\[ c \] Absolute wave celerity Sec. 3.7.1
\[ c_s \] Mean mass transport velocity Sec. 3.8
\[ c_E \] Eulerian mean velocity below wave trough level Sec. 3.8
\[ C\tau \] Courant number Sec. 4.2
\[ CB \] Computed breaking point Sec. 7.5
\[ CS_{max} \] Maximum computed wave-induced setup Sec. 7.5
\[ C_1 \] -0.531 Sec. 2.3
\[ d \] Surface roller thickness Sec. 6.3.1
\[ D \] Still water depth Sec. 2.2
\[ D_1 \] Lower diagonal of a tridiagonal equation system Sec. 3.2
\[ D_2 \] Diagonal of a tridiagonal equation system Sec. 3.3
\[ D_3 \] Upper diagonal of a tridiagonal equation system Sec. 3.3
\[ E \] Vector App. A
\[ E_{kin} \] Kinetic energy of the fluid Sec. 4.3
\[ E_{pot} \] Potential energy of the fluid Sec. 4.3
\[ E_{tot} \] Total energy of the fluid Sec. 4.3
\[ f_i \] Error vector Sec. 3.8
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f$</td>
<td>General variable</td>
<td>Sec. 3.3</td>
</tr>
<tr>
<td>$f_d$</td>
<td>Parameter which describes the shape of the surface roller</td>
<td>Sec. 6.3.2</td>
</tr>
<tr>
<td>$f_{new}$</td>
<td>Improved estimate of a dependent variable</td>
<td>Sec. 3.3</td>
</tr>
<tr>
<td>$f_{old}$</td>
<td>Current estimate of a dependent variable</td>
<td>Sec. 3.3</td>
</tr>
<tr>
<td>$f_w$</td>
<td>Wave friction factor</td>
<td>Sec. 6.4</td>
</tr>
<tr>
<td>$f_L$</td>
<td>Boundary function</td>
<td>Sec. 3.2</td>
</tr>
<tr>
<td>$f_R$</td>
<td>Boundary function</td>
<td>Sec. 3.2</td>
</tr>
<tr>
<td>$f_1$</td>
<td>Function used to absorb outgoing waves</td>
<td>Sec. 3.5</td>
</tr>
<tr>
<td>$f_2$</td>
<td>Function used to absorb outgoing waves</td>
<td>Sec. 3.5</td>
</tr>
<tr>
<td>$F^i$</td>
<td>Vector</td>
<td>App. A</td>
</tr>
<tr>
<td>$F_1$</td>
<td>Spatial derivatives of the momentum equation in the $x$-direction</td>
<td>Sec. 5.2</td>
</tr>
<tr>
<td>$F_2$</td>
<td>Spatial derivatives of the momentum equation in the $y$-direction</td>
<td>Sec. 5.2</td>
</tr>
<tr>
<td>$F_3$</td>
<td>Spatial derivatives appearing in the continuity equation</td>
<td>Sec. 5.2</td>
</tr>
<tr>
<td>$F_1$</td>
<td>Spatial derivatives appearing in the momentum equation</td>
<td>Sec. 3.3</td>
</tr>
<tr>
<td>$F_2$</td>
<td>Spatial derivatives appearing in the continuity equation</td>
<td>Sec. 3.3</td>
</tr>
<tr>
<td>$g_L$</td>
<td>Boundary function</td>
<td>Sec. 3.2</td>
</tr>
<tr>
<td>$g_R$</td>
<td>Boundary function</td>
<td>Sec. 3.2</td>
</tr>
<tr>
<td>$G_1$</td>
<td>Derivatives appearing in the momentum equation in the $x$-direction</td>
<td>Sec. 5.2</td>
</tr>
<tr>
<td>$G_2$</td>
<td>Derivatives appearing in the momentum equation in the $y$-direction</td>
<td>Sec. 5.2</td>
</tr>
<tr>
<td>$h$</td>
<td>Mean water depth</td>
<td>Sec. 2.2</td>
</tr>
<tr>
<td>$H$</td>
<td>Wave height</td>
<td>Sec. 2.2</td>
</tr>
<tr>
<td>$H_s$</td>
<td>Significant wave height</td>
<td>Sec. 3.7.2</td>
</tr>
<tr>
<td>$H_0$</td>
<td>Incident wave height</td>
<td>Sec. 4.6</td>
</tr>
<tr>
<td>$i$</td>
<td>Integer</td>
<td>Sec. 2.2</td>
</tr>
<tr>
<td>$i_0$</td>
<td>Location of the wave generation line</td>
<td>Sec. 3.6</td>
</tr>
<tr>
<td>$H$</td>
<td>Maximum node number in the $x$-direction</td>
<td>Sec. 3.3</td>
</tr>
<tr>
<td>$ISL$</td>
<td>Number of sponge layer nodes accompanying the left boundary</td>
<td>Sec. 3.5</td>
</tr>
<tr>
<td>$ISR$</td>
<td>Number of sponge layer nodes accompanying the right boundary</td>
<td>Sec. 3.5</td>
</tr>
<tr>
<td>$j$</td>
<td>Integer</td>
<td>Sec. 3.8</td>
</tr>
<tr>
<td>$J_1$</td>
<td>Imaginary unit</td>
<td>Sec. 3.8</td>
</tr>
<tr>
<td>$J_2$</td>
<td>Jacobi matrix</td>
<td>Sec. 3.8</td>
</tr>
<tr>
<td>$k$</td>
<td>Wave number</td>
<td>Sec. 3.8</td>
</tr>
<tr>
<td>$k_i$</td>
<td>Wave number of a wave component</td>
<td>Sec. 3.7.2</td>
</tr>
<tr>
<td>$k_N$</td>
<td>Nikuradse roughness parameter</td>
<td>Sec. 6.4</td>
</tr>
<tr>
<td>$L$</td>
<td>Wave length</td>
<td>Sec. 2.3</td>
</tr>
<tr>
<td>$L_0$</td>
<td>Wave length calculated from Stokes’ linear theory</td>
<td>Sec. 4.6</td>
</tr>
<tr>
<td>$m$</td>
<td>Integer</td>
<td>Sec. 3.2</td>
</tr>
<tr>
<td>$N$</td>
<td>Number of Fourier components</td>
<td>Sec. 3.8</td>
</tr>
<tr>
<td>$MB$</td>
<td>Measured breaking point</td>
<td>Sec. 3.8</td>
</tr>
<tr>
<td>$MB_{mse}$</td>
<td>Root-mean-square-error of the measured breaking point</td>
<td>Sec. 7.3.5</td>
</tr>
<tr>
<td>$MS_{max}$</td>
<td>Maximum measured wave-induced setup</td>
<td>Sec. 7.5</td>
</tr>
<tr>
<td>$n$</td>
<td>Time level</td>
<td>Sec. 3.3</td>
</tr>
<tr>
<td>$N$</td>
<td>Quantity determining the spatial discretization in each node</td>
<td>Sec. 5.3.1</td>
</tr>
<tr>
<td>$NF$</td>
<td>Number of wave components in a given wave spectrum</td>
<td>Sec. 3.7.2</td>
</tr>
<tr>
<td>Symbol</td>
<td>Description</td>
<td>Reference</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>NN</td>
<td>Number of time steps</td>
<td>Sec. 3.3</td>
</tr>
<tr>
<td>N²</td>
<td>Number of nodes per shallow water wave length</td>
<td>Sec. 4.2.3</td>
</tr>
<tr>
<td>p</td>
<td>Pressure</td>
<td>Sec. 6.3.1</td>
</tr>
<tr>
<td>Q</td>
<td>Horizontal mass flux in an absolute reference system</td>
<td>Sec. 3.4.3</td>
</tr>
<tr>
<td>Ò</td>
<td>Eulerian mean mass flux in the x-direction</td>
<td>Sec. 3.8</td>
</tr>
<tr>
<td>Q₁</td>
<td>Horizontal mass flux in a moving frame of reference</td>
<td>Sec. 3.8</td>
</tr>
<tr>
<td>R</td>
<td>Right hand side of tridiagonal equation system</td>
<td>Sec. 3.3</td>
</tr>
<tr>
<td>RE</td>
<td>Linear reflection coefficient</td>
<td>Sec. 4.4</td>
</tr>
<tr>
<td>R₁</td>
<td>Relative root-mean-square-error of a measured wave height</td>
<td>Sec. 7.3.4</td>
</tr>
<tr>
<td>R₂</td>
<td>Right hand side of tridiagonal equation system in the x-direction</td>
<td>Sec. 5.2</td>
</tr>
<tr>
<td>R₁</td>
<td>Right hand side of tridiagonal equation system in the y-direction</td>
<td>Sec. 5.2</td>
</tr>
<tr>
<td>S</td>
<td>Bernoulli constant in a moving frame of reference</td>
<td>Sec. 3.8</td>
</tr>
<tr>
<td>S</td>
<td>Wave spectrum</td>
<td>Sec. 3.7.2</td>
</tr>
<tr>
<td>t</td>
<td>Time</td>
<td>Sec. 2.2</td>
</tr>
<tr>
<td>t₀</td>
<td>Time of incipient wave breaking</td>
<td>Sec. 6.3.2</td>
</tr>
<tr>
<td>t_max</td>
<td>Maximum time</td>
<td>Sec. 5.5.2</td>
</tr>
<tr>
<td>t₂</td>
<td>Time scale for the development of the surface roller</td>
<td>Sec. 6.3.2</td>
</tr>
<tr>
<td>T</td>
<td>Absolute wave period</td>
<td>Sec. 3.8</td>
</tr>
<tr>
<td>Tₚ</td>
<td>Peak period</td>
<td>Sec. 3.7.2</td>
</tr>
<tr>
<td>u</td>
<td>Horizontal velocity vector</td>
<td>Sec. 2.2</td>
</tr>
<tr>
<td>u₀</td>
<td>Horizontal velocity at the still water level</td>
<td>Sec. 2.2</td>
</tr>
<tr>
<td>u₁</td>
<td>Horizontal velocity in a moving frame of reference</td>
<td>Sec. 2.2</td>
</tr>
<tr>
<td>U</td>
<td>Depth-averaged undertow averaged over a typical wave period</td>
<td>Sec. 6.5.3</td>
</tr>
<tr>
<td>U⁺</td>
<td>The Ursell number</td>
<td>Sec. 2.2</td>
</tr>
<tr>
<td>Uₘ</td>
<td>Fourier component</td>
<td>Sec. 2.2</td>
</tr>
<tr>
<td>v</td>
<td>Horizontal velocity in the y-direction</td>
<td>Sec. 2.2</td>
</tr>
<tr>
<td>v₀</td>
<td>Horizontal velocity in the y-direction evaluated at z = z₀</td>
<td>Sec. 5.2</td>
</tr>
<tr>
<td>V</td>
<td>Volume of water in the fluid domain</td>
<td>Sec. 4.3</td>
</tr>
<tr>
<td>w</td>
<td>Vertical velocity component (z-direction)</td>
<td>Sec. 4.3</td>
</tr>
<tr>
<td>W</td>
<td>Quantity used to generate the incident wave field internally</td>
<td>Sec. 5.3.6</td>
</tr>
<tr>
<td>WF₁</td>
<td>Function added to F₁</td>
<td>Sec. 5.3.6</td>
</tr>
<tr>
<td>WF₂</td>
<td>Function added to F₂</td>
<td>Sec. 5.3.6</td>
</tr>
<tr>
<td>WF₃</td>
<td>Function added to F₃</td>
<td>Sec. 5.3.6</td>
</tr>
<tr>
<td>WG₁</td>
<td>Function added to G₁</td>
<td>Sec. 5.3.6</td>
</tr>
<tr>
<td>WG₂</td>
<td>Function added to G₂</td>
<td>Sec. 5.3.6</td>
</tr>
<tr>
<td>WR₁</td>
<td>Quantity added to R</td>
<td>Sec. 3.6</td>
</tr>
<tr>
<td>WR₂</td>
<td>Quantity added to R₁</td>
<td>Sec. 5.3.6</td>
</tr>
<tr>
<td>W₁</td>
<td>Quantity used to generate the incident wave field internally</td>
<td>Sec. 3.6</td>
</tr>
<tr>
<td>W₂</td>
<td>Quantity used to generate the incident wave field internally</td>
<td>Sec. 3.6</td>
</tr>
<tr>
<td>xᵢ</td>
<td>Solution vector</td>
<td>Sec. 3.8</td>
</tr>
</tbody>
</table>
\( x \) Horizontal coordinate

\( x_b \) Particle amplitude at the bottom derived by use of linear theory

\( x_{\text{max}} \) X-axis coordinate at the right boundary

\( x_{\text{min}} \) X-axis coordinate at the left boundary

\( x_s \) The width of the sponge layer in the x-direction

\( x_0 \) X-axis location where the sponge layer starts

\( x_1 \) Horizontal coordinate in a moving frame of reference

\( y \) Horizontal coordinate

\( y_{\text{max}} \) Y-axis coordinate at the boundary of the computational domain

\( y_s \) The width of the sponge layer in the y-direction

\( z \) Vertical coordinate measured from the still water level

\( z_b \) Elevation of the bottom

\( z_a \) Particular level within the water column \((z_a = C_1 D)\)

\( Z \) Mean initial water level in the piezometers

\( Z_{\text{rmse}} \) Root-mean-square-error of the still water level

**Greek symbols**

\( \alpha \) Constant \((\alpha = \frac{1}{2}C_1^2 + C_1)\)

\( \alpha_1 \) Function

\( \alpha_2 \) Function

\( \alpha_3 \) Function

\( \alpha_4 \) Function

\( \beta_1 \) Imaginary function

\( \beta_2 \) Imaginary function

\( \beta_3 \) Imaginary function

\( \beta_4 \) Imaginary function

\( \gamma \) Damping function used to absorb outgoing waves

\( \gamma_{\text{max}} \) Maximum value of \( \gamma \)

\( \Gamma \) Set denoting the nodes located at the fluid boundary

\( \Gamma_L \) Left boundary of the computational domain

\( \Gamma_R \) Right boundary of the computational domain

\( \delta \) A measure of the amplitude dispersion

\( \delta_{i,j} \) Kronecker delta function

\( \Delta_i \) Change in the solution vector

\( \Delta f \) Relative change in a dependent variable

\( \Delta t \) Time step

\( \Delta x \) Grid spacing in the x-direction

\( \Delta y \) Grid spacing in the y-direction

\( \Delta \omega_i \) Angular frequency spacing between consecutive wave components

\( \varepsilon^2 \) A measure of the frequency dispersion

\( \eta \) Surface elevation in an absolute reference system

\( \bar{\eta} \) Eulerian mean surface elevation
SYMBOLS

\(\eta_{\text{max}}\)  Maximum surface elevation  Sec. 4.7
\(\eta_0\)  Eigenvalue  Sec. 4.2
\(\eta_1\)  Surface elevation in a moving frame of reference  Sec. 3.8
\(\theta\)  Argument of a complex quantity  Sec. 4.2.2
\(\lambda\)  Angle which describes the direction of the incident wave field  Sec. 5.3.6
\(\Lambda_1\)  Typical horizontal length scale in the wave motion  Sec. 2.2
\(\Lambda_2\)  Function  Sec. 3.8
\(\xi\)  Amplification factor of the computational scheme  Sec. 4.2
\(\sigma_{i,j}\)  Kronecker delta function  Sec. 5.3.6
\(\tau\)  Bed shear stress  Sec. 6.4
\(\phi\)  Parameter describing the development of the surface roller  Sec. 6.3.2
\(\phi_i\)  Parameter describing the initiation of wave breaking  Sec. 6.3.2
\(\phi_0\)  Parameter describing the cessation of wave breaking  Sec. 6.3.2
\(\Phi\)  Velocity potential  Sec. 2.2
\(\Phi^{(i)}\)  Velocity potential used in a series expansion, \(i = 0, \ldots\)  Sec. 2.2
\(\Psi_i\)  Random phase of a wave component  Sec. 3.7.2
\(\omega\)  Angular frequency  Sec. 3.7.2
\(\omega_i\)  Angular frequency of a wave component  Sec. 3.7.2
\(\Omega_{\text{int}}\)  Set denoting the nodes located inside the computational domain  Sec. 5.3.1
\(\Omega_{\text{tot}}\)  Set denoting all the computational points  Sec. 5.3.1

Dimensional Quantities

\(D_0^*\)  Characteristic still water depth  Sec. 3.2
\(g^*\)  Acceleration of gravity  Sec. 2.2
\(\rho^*\)  Density of water  Sec. 4.3

Operators and Abbreviations

\(f_t\)  \(\frac{\partial f}{\partial t}\)
\(f_x\)  \(\frac{\partial f}{\partial x}\)
\(f_y\)  \(\frac{\partial f}{\partial y}\)
\(f_z\)  \(\frac{\partial f}{\partial z}\)
MWL  Mean water level
2-D  Two-dimensional
3-D  Three-dimensional
\(O(\cdot)\)  Landau symbol: Order of magnitude
\(\partial\)  Partial derivative
\(\nabla\)  \(\left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}\right)\)
\(\Re\)  Real part of a complex quantity
Chapter 1

Introduction

A thorough understanding of wave transformation in the nearshore zone is essential in relation to both sediment transport and the design of marine structures. Wave action is responsible for the composition and evolution of beach profiles, the sedimentation and erosion around marine structures, and the backfilling of dredged channels. Moreover, the design of marine structures such as breakwaters and harbours requires an accurate prediction of extreme loads caused by waves and currents.

In the past several analytical models have been proposed to describe the variation of the mean water level and the wave height caused by wave breaking. Since the models are based on the depth-integrated and time-averaged equations of mass, energy and momentum, these provide no details of the wave motion during a typical wave cycle. The time-averaged fluxes of mass, energy, and momentum in the horizontal direction are often quantified using a wave theory valid for regular waves of permanent form in water of constant depth. Consequently, the equations cannot predict reliably the location of the breaking point or the important conversion of potential energy to kinetic energy in the outer region of the surf zone.

In recent years experimental studies have provided valuable information on the flow structure inside the surf zone. Presently, it is not feasible to solve the equations of motion in three space dimensions, and the qualitative results have therefore been incorporated into depth-integrated numerical flow models operating in the time-domain. Models of this kind have been shown to improve significantly the description of the mean water level, the wave height variation, and the wave-induced flow caused by regular as well as irregular waves breaking as spilling breakers on various gently sloping beach profiles. However, the general impression is still that much remains to be done before real prediction based on the underlying physics is possible.

The main objective of this thesis is to describe the mean water level and the wave height variation caused by regular waves breaking on relatively steep slopes. Wave transformation on the seaward face of coral reefs can be mentioned as an example. In order to address this issue Chapter 2 describes the fundamental theory of long waves. Particular attention is given to a class of equations capable of describing the evolution of relatively long, weakly nonlinear waves in water of constant or variable depth. Equations of this kind are denoted Boussinesq-type equations. A minor part of the chapter is devoted to the description of the computational methods commonly employed for their solution. Additionally, a summary is given of the efforts undertaken to extend the range of application of Boussinesq-type equations to deeper
Based on a set of Boussinesq-type equations with improved linear dispersion characteristics in deeper water, Chapter 3 describes the development of a computational model in a single horizontal dimension. The model is based on the finite difference method and the computations are advanced in time by using a fourth order accurate predictor-corrector method. Since the incident wave field is generated inside the computational domain, the scattered wave field is absorbed in the vicinity of the model boundary by employment of damping terms in the mass and momentum equations. A Fourier approximation method is used to compute an incident regular wave field which satisfies the governing equations accurately in water of constant depth.

A thorough investigation of the computational model is given in Chapter 4. The first part describes the linearized stability properties of the numerical operator. Examples are given which show that the model conserves well fundamental properties such as the total mass and energy in the fluid domain. In the second part wave propagation in water of variable depth is considered. The examples verify that the model is capable of making accurate predictions of the wave height variation caused by shoaling.

In Chapter 5 the computational model is extended to include the second horizontal dimension. In contrast to existing solution procedures the present model allows physical barriers to be located inside the computational domain, hence indicating that the computational model can be used to study a number of practical problems. Moreover, the internal wave generation method described in Chapter 3 is generalized and included in the formulation. Examples are given illustrating the capabilities of the model.

The dissipation of energy caused mainly by wave breaking is described in Chapter 6. The simplified effect of wave breaking is incorporated into the two-dimensional model only. Based on the assumption of a vertical redistribution of the horizontal velocity field inside the surf zone a new set of equations is derived. The initiation, the temporal development, and the cessation of wave breaking are governed by a method which is capable of reproducing accurately several features of a breaking wave. The surf zone model is calibrated and subsequently verified by comparison with existing wave flume measurements which describe the transformation of regular waves on a plane and gentle slope.

Chapter 7 describes a laboratory experiment conducted in a closed wave flume. A summary is given of the laboratory facility and the equipment used. Since the considered bed profile represents a relatively steep submerged coral reef located on the coast of Guam, the main purpose of the experiment is to verify the validity of the surf zone model on steeper slopes. Six incident wave conditions are considered. In each test series the mean water level is measured at 21 locations along the centre of the flume. The surf zone model is calibrated using a single set of laboratory data and subsequently tested against five other test series. It is illustrated that the surf zone model generally predicts the measured mean water levels relatively accurately although there is a tendency to underestimate the maximum wave-induced setup.

Conclusions are given in Chapter 8.
Chapter 2

The Theory of Long Waves

2.1 General

This chapter summarizes part of the theory relating to the propagation of so-called long waves over a constant or a variable bottom. On the basis of some fundamental assumptions a summary is given of the usual derivation procedures employed in developing the theory of long waves. Particular attention is paid to a class of equations attributed to Boussinesq (1872), since these constitute the foundation of the present report.

In recent years a number of attempts have been made to extend the range of application of Boussinesq-type equations to deeper water. These efforts are described in some detail. Although equations of the Boussinesq-type can be solved analytically in a few canonical cases the majority of all practical applications necessitate the use of a numerical solution method. Since this is far from a trivial task, part of the chapter is devoted to the description of numerical methods commonly employed for their solution.

2.2 Assumptions and Derivation Procedures

In intermediate depth water, i.e. when the mean water depth is of the same order of magnitude as the wave length, the Fourier approximation method of Rienecker & Fenton (1981) as well as the Stokes’ theories form effective tools for modelling regular waves of permanent form over a horizontal bottom, cf. Svendsen & Jonsson (1980) and Fenton (1985, 1990). As a wave propagates into shoaling water the wave motion undergoes significant changes caused by the change in the water depth, thus indicating that the wave is no longer of permanent form, and further, that the mean water depth can not be assumed to be nearly as large as the wave length. In this case the theory of long waves can be applied.

Similar to Stokes’ theory, the derivation of long wave theory is based on the assumption of an inviscid and incompressible fluid. Initially, the flow is assumed to be irrotational which by use of Kelvin’s theorem implies that the motion will remain irrotational in time. Hence, a velocity potential can be defined within the fluid domain. Provided the wave length does
not become too large so that the bottom boundary layer becomes too thick the condition of
irrotationality will be satisfied in practice. Thus, it becomes a matter of solving the Laplace
equation with the proper boundary conditions. For further information on this subject refer-
ence is made to Svendsen & Jonsson (1980).

On a horizontal bottom the long wave problem is characterized by the dependence on
two small parameters, $\delta$ and $\epsilon^2$, which are, in principle, independent of each other. These are
termed the long wave assumptions, and they are given by

$$\delta = \frac{H^*}{h^*} \ll 1$$
$$\epsilon^2 = \left(\frac{h^*}{\lambda^*}\right)^2 \ll 1$$

(2.1)

where $H^*$ is the wave height, $h^*$ is the mean water depth, and $\lambda^*$ is a characteristic horizontal
length scale in the wave motion. It should be noted that a symbol superscribed by an asterisk
denotes a dimensional variable. The first parameter is a measure of the nonlinearity of the wave
motion while the second parameter describes the frequency dispersion caused by streamline
curvature. In order to define the problem uniquely the magnitude of one of these parameters
must be related to the magnitude of the other. It has turned out (Ursell, 1953) that three
cases are of importance

$$\begin{cases}
\delta & \ll \epsilon^2 \\
\delta & = O(\epsilon^2) \\
\delta & \gg \epsilon^2
\end{cases}$$

(2.2)

that is, whether the magnitude of the Ursell number, defined by

$$UR \equiv \frac{\delta}{\epsilon^2}$$

(2.3)

is much smaller than, equal to, or much greater than unity.

One way of solving the problem in a single horizontal dimension (2-D) is to follow a
method outlined by Mei & Le Méhauté (1966) and described in more detail by Svendsen
(1974). By introducing a characteristic horizontal length, a characteristic time scale, and a
characteristic pressure, the variables in the continuity equation, the bottom boundary con-
dition and the kinematic and the dynamic conditions at the free water surface are made
dimensionless. Once a relationship between $\delta$ and $\epsilon^2$ is established the magnitude of each
term within these equations can be evaluated. The next step towards a solution for the di-
mensionless velocity potential, $\Phi(x, z, t)$, consists in expanding $\Phi$ in an infinite series. Here,
$x$ is a horizontal coordinate, $z$ is a vertical coordinate, and $t$ denotes time. From the order
of magnitude considerations mentioned above it can then be deduced (Svendsen, 1974) that
a qualified guess for the simplest form of $\Phi$ will be a Taylor-like expansion, given by

$$\Phi(x, z, t) = \sum_{i=0}^{\infty} z^i \Phi^{(i)}(x, t)$$

(2.4)
2.2. Assumptions and Derivation Procedures

It is noted that $\Phi^{(i)}$ does not depend on the vertical coordinate. If, instead, a dimensionless system was used in which a vertical length scale was present, Svendsen noted that Equation (2.4) results directly from the long wave assumptions, Equation (2.1), in connection with a series expansion for the velocity potential. By substitution of Equation (2.4) into the Laplace equation and use of the bottom boundary condition two recurrence relations can be obtained allowing $\Phi^{(i)}$, $i = 1, 2, \ldots$, to be written as a function of $\Phi^{(0)}$. In combination with the boundary conditions at the free water surface this results, to the desired degree of approximation, in two partial differential equations formulated in terms of the surface elevation, $\eta$, and $\Phi^{(0)}$, or in $\eta$ and a horizontal velocity component.

Depending on the magnitude of the Ursell number the solution procedure outlined above gives rise to different differential equations. For waves of small amplitude over a horizontal bottom, that is, for Ursell numbers much smaller than unity, the nonlinear terms become insignificant so that in this case the so-called linear equations appear (Svendsen & Jonsson, 1980). These represent a second approximation to the linear shallow water wave theory accounting for frequency dispersion.

Ursell numbers much greater than unity lead to the nonlinear shallow water equations which are amplitude but not frequency dispersive. As a consequence, the wave profile will steepen even on a horizontal bottom eventually causing the wave to break, hence implying that the equations would fail to predict the breaking point in a more general context.

In the case of an Ursell number of the order of magnitude of unity equations can be derived in which the frequency dispersive and the amplitude dispersive terms counter balance each other in such a way that the waves remain stable. Traditionally, this group of equations is called Boussinesq-type equations, since Boussinesq (1872) appears to be the first to derive equations of this kind. In the following the long wave assumptions as well as an Ursell number of $UR = O(1)$ will be taken for granted.

By following the aforementioned derivation procedure Svendsen (1974) arrived at the (dimensional) equation system

$$u_0^* + u_0^* u_{0,x}^* + g^* \eta_{2,x}^* = 0$$  \hspace{1cm} (2.5)

$$\eta_{*} + [(D^* + \eta^*) u_0^*]_{x^*} + \frac{1}{3} D_{x^*}^3 u_{0,x^*}^* = 0$$  \hspace{1cm} (2.6)

valid for waves in water of constant depth. In Equations (2.5) and (2.6) $x^*$ and $t^*$ are the dimensional equivalents of $x$ and $t$, respectively, while $\eta^*$ is the surface elevation measured from the still water level ($z^* = 0$, $z^*$ being the dimensional equivalent of $z$). The quantity, $D^*$, denotes the still water depth, $u_0^*$ denotes the horizontal velocity component at $z^* = 0$, $g^*$ is the acceleration of gravity, and subscripts $x^*$ and $t^*$ refer to partial differentiation with respect to $x^*$ and $t^*$, respectively. These are the equations favoured by Boussinesq (1872).

By permitting waves to propagate in a single direction only the KdV-equation named after Korteweg & de Vries (1895) can be derived from Equations (2.5) and (2.6). It was shown by Keulegan & Patterson (1940) and Laitone (1961) that the KdV-equation or the equivalent set of equations, Equations (2.5) and (2.6), has analytical solutions of permanent form. These are referred to as cnoidal waves, since their solution depends on the cn-function as well as on some elliptic integrals which can be evaluated by use of handbooks (Abramowitz & Stegun, 1965).
It may be noted that Peregrine (1966) presented a set of Boussinesq equations formulated in terms of the depth-averaged horizontal velocity, which is valid for waves in water of constant depth. The equations can be shown to be identical to Equations (2.5) and (2.6) by shifting the velocity variable. In the case of waves propagating over a sloping bottom the derivation procedure gets more complicated, since the size of the bottom slope now plays a role in evaluating the magnitude of each term in the Laplace equation and in the expressions for the boundary conditions. Consequently, the magnitude of the bottom slope must be related to either $\delta$ or $e^2$ before the derivation procedure summarized above can be performed. Additionally, it is noted that one of the recurrence relations obtained by substitution of the expansion for the velocity potential into the Laplace equation becomes more complicated due to the inclination of the bottom. By the assumption that the spatial derivatives of the still water depth are of the order $O(\epsilon)$ Mei & Le Méhauté (1966) derived a set of Boussinesq equations in a single horizontal dimension valid for long waves propagating over a sloping bottom. In the case of a horizontal bottom these reduce to Equations (2.5) and (2.6).

An extension of the original equations of Boussinesq (1872) was given by Peregrine (1967) who considered the propagation of long waves over a sloping bottom in two horizontal dimensions (3-D). Similarly to Mei & Le Méhauté (1966) the spatial derivatives of the still water depth were assumed to be $O(\epsilon)$. In contrast to the approach of Svendsen (1974) the derivation was based directly on the Euler equations and the continuity equation in dimensionless form. The independent variables appearing in these equations were expanded in power series using a technique originally introduced by Keller (1948). By use of the boundary conditions at the free surface and at the bottom as well as the irrotationality condition, substitution of the expansion series into the depth-integrated continuity equation and the Euler equations allowed Peregrine to quantify the first and second order expansion coefficients. The first order coefficients were determined directly, whereas the second order coefficients were given as a function of the first order terms. It turned out that the horizontal velocity field varied quadratically over the depth while the vertical velocity field increased linearly from a minimum at the sea bed to a maximum at the free water surface. Finally, by introducing a depth-averaged horizontal velocity vector a set of equations appeared in which the only additional variable was the surface elevation. These are given by

\begin{align}
\mathbf{u}^* \cdot \nabla \mathbf{u}^* + g^* \nabla \eta^* + \left( \frac{\partial}{\partial x^*} \right) \left[ \nabla \cdot (D^* \mathbf{u}^*) \right] = 0 \quad \ldots (2.7)
\end{align}

\begin{align}
\eta^* + \nabla \cdot \left( (D^* + \eta^*) \mathbf{u}^* \right) = 0 \quad \ldots (2.8)
\end{align}

where $\nabla \cdot \left( \frac{\partial}{\partial \sigma^*}, \frac{\partial}{\partial y^*} \right)$, and $y^*$ is the second horizontal direction. In contrast to the equations of Mei & Le Méhauté (1966) the equations of Peregrine are valid in two horizontal dimensions as can be seen from the fact that a symbol in Clarendon type denotes a vector, e.g. $\mathbf{u}^* = (u^*, v^*)$, where $u^*$ and $v^*$ are the mean velocity components in the two horizontal dimensions. The first two terms in Equation (2.7) describe the local and the convective acceleration of the fluid, while the third term is caused by gravity. The remaining terms account for the frequency dispersion associated with the effect of the vertical particle accelerations on the pressure distribution. Equation (2.8) is readily seen to be the depth-integrated continuity equation. The equations
given above have become a prototype for many later studies. Additionally, it is noted that Peregrine solved the equations numerically in a single horizontal dimension by employment of a finite difference scheme.

By retaining terms only of the order $O(1)$ in Equations (2.7) and (2.8) the linearized long wave equations appear. These can be written

\begin{align}
\mathbf{u}^* + g^* \nabla \eta^* &= 0 \\
\eta^* + \nabla \cdot (D^* \mathbf{u}^*) &= 0
\end{align}

(2.9) \quad \text{(2.10)}

Long (1964) first showed that the linearized long wave equations can be used to manipulate higher order terms into a preferred form without loss of accuracy. From a computational viewpoint this is useful as it makes it possible to tailor a given set of Boussinesq-type equations to a numerical solution method rather than the opposite.

In contrast to the rigorous approaches adopted by both Mei & Le Méhauté (1966) and Peregrine (1967) it should be noted that a more direct derivation can be performed by integrating the mass and momentum equations without explicit order of magnitude considerations. This technique was illustrated by Abbott (1979) who considered the propagation of long waves over a horizontal bottom. For waves over an uneven bottom Serre (1953) integrated the flow equations in 3-D by the assumption that the vertical velocity varies linearly over the depth. Despite the fact that no additional simplifying assumptions were made and all terms were retained in the derivation, the Serre equations are not more accurate or complete than the more simple forms of Boussinesq equations, since the former rely strictly on the assumption of a linear vertical velocity distribution over the depth, as noted by Madsen et al. (1991).

In two horizontal dimensions (3-D) Abbott et al. (1978) solved numerically a simplified version of the equations by Peregrine (1967) by employment of a finite difference scheme. The basic equations focused on were identical to those of Peregrine, but the frequency dispersive terms appearing in Equation (2.7) were approximated by the assumption of a constant still water depth, indicating that the computations were restricted to a horizontal bottom. Two examples were given demonstrating the ability of the model to simulate the diffraction of regular waves around a breakwater in water of constant depth as well as the propagation of regular waves into a real harbour of slowly varying bathymetry but no results were verified by comparison with experiments.

It was reported by Abbott et al. (1984) that the numerical solution method developed previously (Abbott et al., 1978) gave rise to some spurious oscillations on the trailing part of the wave signal radiating away from the crest. A truncation error analysis revealed that the discretization of first derivative terms is of particular importance, since these, if not approximated accurately, result in a truncation error which contains third derivative terms of the second order. Since any such term can be manipulated into the same form as the third derivative dispersive term appearing in the momentum equation by invoking the linearized long wave equations, it is crucial to eliminate these from the truncation error. As noted by Abbott et al. (1984) this can be done by approximating first derivative terms correct to the third order. Following the correction of the numerical scheme a comparison with a solitary wave exhibited a reasonable, although not a perfect, agreement. The reason for this was believed to be due to the simplifying assumptions used in deriving the governing differential equations.
McCowan (1978) solved numerically the equations of Peregrine (1966) in a single horizontal dimension using a box operator centred in time. The approximation of the governing equations introduced truncation error terms of the second order in both the time step and the spatial step. By employment of the corresponding linearized long wave equations a Taylor series analysis revealed that some of the truncation error terms had the same form as the leading third derivative dispersive term appearing in the momentum equation. In order to obtain a numerical solution representing the governing equations the undesired truncation error terms were eliminated by further discretization. Since the numerical space derivatives were centred between computational points, the resulting linear system of equations was solved by use of a single boundary condition at each end of the fluid domain. Updating of the horizontal velocity and the surface elevation was accomplished by solving a pentadiagonal equation system at each time level. This was done by invoking a Double-Sweep algorithm, which is a special form of Gauss elimination, see Vreugdenhil (1989).

In principle, the application of Boussinesq-type equations is limited to cases where the long wave assumptions, Equation (2.1), as well as an Ursell number of $UR = O(1)$ are fulfilled. However, by use of the aforementioned numerical method (McCowan, 1978) it was emphasized by McCowan (1981) that in practice waves of height well beyond the theoretical limit can be modelled quite accurately provided the shallow water assumption is satisfied. As the still water depth increases the results indicated that rapidly increasing phase errors are introduced, thus restricting the range of application of the equations to shallow water.

Hauguel (1980) employed a more general form of Boussinesq-type equations originally introduced by Serre (1953) to simulate wave propagation in shallow water. Unlike the equations studied by McCowan (1978) the derivation of these equations involved few simplifying assumptions (McCowan, 1981), hence resulting in a set of equations which is complicated and difficult to solve efficiently. In both one and two horizontal dimensions Hauguel (1980) solved numerically the equations of Serre by employment of a higher order three stage implicit finite difference method. In the case of a solitary wave of a height equal to 25% of the water depth the solution method proved very accurate, since both the amplitude and the phase essentially coincided with the analytical solution. It is noted that the numerical solution did not suffer from spurious oscillations trailing behind the primary wave, hence indicating that the third derivative truncation error terms had been eliminated. An initially solitary wave propagating onto a shelf was modelled demonstrating that the wave eventually disintegrates into a series of solitons. A comparison with an identical example of Madsen & Mei (1969) exhibited reasonable agreement.

A study of the phase and amplitude characteristics of various equations of the Boussinesq-type was carried out by McCowan (1985). Following Peregrine (1974) he reported that different assumptions as well as choice of variables and integration procedures lead to Boussinesq-type equations with different phase and amplitude characteristics. McCowan focused on four different sets of equations, the first one being that of Abbott (1979), while the second set was identical to Equations (2.5) and (2.6). The third and the fourth sets of equations, respectively, were consistent with those of Hauguel (1980) and Peregrine (1967). All sets of equations were identical except for the dispersive term appearing in the momentum equation. The dispersive terms were manipulated into the same form using the linearized long wave equations. Equations (2.9) and (2.10). Each set of equations was solved numerically by employment of a technique described by McCowan (1978). Tests with a truncated solitary wave confirmed that the convective and the dispersive terms are relatively insignificant in comparison with the gravitational and the inertial terms, provided the wave height is significantly smaller than the water depth. In the case of a limiting height solitary wave only the equations consistent
2.3 Application of Boussinesq-type Equations in Deeper Water

with those of Peregrine performed well, indicating that the form of the frequency dispersive terms is highly relevant when modelling waves of finite height.

McCowan (1987) studied the range of application of the Boussinesq-type equations considered in his previous work (McCowan, 1985). In comparison with the stream function theory of Chaplin (1980) the best performance in shallow water was obtained using the equations of Hauguel (1980). In intermediate depth water the equations compatible with those of Peregrine (1967) were found to give the best result.

2.3 Application of Boussinesq-type Equations in Deeper Water

From a mathematical viewpoint it is a violation of the theory to employ equations of the Boussinesq-type to simulate waves in water deeper than recommended by the shallow water limit \( \frac{h}{L^*} \leq 0.05 \), \( L^* \) being the wave length. If the long wave assumption is neglected it is evident that amplitude errors, and phase errors in particular, will occur as the deep water limit is approached. Recently, a number of attempts have been made to improve the dispersion characteristics of Boussinesq-type equations in deeper water. Witting (1984) used a version of the exact fully nonlinear depth-integrated momentum equation formulated in terms of the horizontal velocity at the free surface. A Padé approximation technique was used to relate the different velocity variables in the governing equations with coefficients determined to yield the best linear dispersion characteristics. Excellent results were obtained but the method is restricted to water of constant depth. In addition it seems difficult to use the technique in two horizontal dimensions.

Inspired by the idea of Witting (1984), Madsen et al. (1991) extended the Boussinesq equations studied by Abbott et al. (1984) to incorporate improved linear dispersion characteristics in deeper water. This was done by adding an additional third derivative term to the momentum equation which was derived from the linearized long wave equations. The third derivative term makes the equations effectively linear in deeper water and reduces to zero in shallow water. Since the original equations are restricted to a horizontal bottom, waves propagating from deep to shallow water were not modelled.

The problem mentioned above was partly overcome by Madsen & Sorensen (1992). In contrast to Abbott et al. (1984) and Madsen et al. (1991) their starting point was the original equations of Peregrine (1967). Unlike Abbott et al. (1978) first order spatial derivatives of the still water depth were included when approximating the dispersive terms in Equation (2.7), but higher derivatives and products of derivatives were neglected, hence limiting the computations to a gently sloping bed. On the basis of the linearized Boussinesq equations a shoaling coefficient was computed as a function of the relative still water depth. A comparison between this curve and a shoaling curve computed by employment of the improved Boussinesq-type equations showed an excellent agreement in both shallow and deep water. An example was given of regular waves climbing a plane and gentle slope, and the results were compared with linear wave theory demonstrating the ability of the model to produce linear shoaling almost perfectly.

Yoon & Liu (1989) studied the interaction of currents and weakly nonlinear waves, and presented a different integration procedure than that employed by Peregrine (1967). Their method was adopted by Nwogu (1993) who assumed that the spatial derivatives of the still
water depth are $O(\varepsilon)$. Nwogu integrated the continuity equation and the horizontal momentum equations over the instantaneous water depth using the kinematic boundary conditions at the seabed and the free water surface. Integration of the vertical momentum equation gave an expression for the pressure, which was substituted into the depth-integrated horizontal momentum equations. The vertical velocity component was eliminated from these equations by use of the continuity equation integrated with respect to the vertical coordinate. Eventually, a set of integral expressions was obtained in which the only unknown variable was the horizontal velocity field. In order to perform the integration an expansion was carried out of the horizontal velocity field about an arbitrary level within the water column. Employment of the irrotationality condition finally allowed the integration to be carried out, hence yielding a new set of Boussinesq-type equations valid for waves in water of variable depth. In dimensional form the equations can be written

\begin{align}
\mathbf{u}_{\alpha,*}^{\varepsilon} &+ (\mathbf{u}_{\alpha,*}^{\varepsilon} \cdot \nabla)\mathbf{u}_{\alpha,*}^{\varepsilon} + \mathbf{g}^{\varepsilon} \nabla \eta^{\varepsilon} \\
&+ C_1 \mathbf{D}^{\varepsilon} \{ \frac{C_1}{2} \mathbf{D}^{\varepsilon} \nabla (\mathbf{u}_{\alpha,*}^{\varepsilon} \cdot \nabla \cdot \mathbf{u}_{\alpha,*}^{\varepsilon}) + \nabla [ \nabla \cdot (\mathbf{D}^{\varepsilon} \mathbf{u}_{\alpha,*}^{\varepsilon})] \} = 0
\end{align}

(2.11)

\begin{align}
\eta^{\varepsilon} &+ \nabla \cdot [(\mathbf{D}^{\varepsilon} + \mathbf{g}^{\varepsilon}) \mathbf{u}_{\alpha,*}^{\varepsilon}] \\
&+ \nabla \cdot \{ \left( \frac{C_1^2}{2} - \frac{1}{6} \right) \mathbf{D}^{\varepsilon} \nabla (\nabla \cdot \mathbf{u}_{\alpha}^{\varepsilon}) + \left( C_1 + \frac{1}{2} \right) \mathbf{D}^{\varepsilon} \nabla (\nabla \cdot \mathbf{D}^{\varepsilon} \mathbf{u}_{\alpha,*}^{\varepsilon}) \} = 0
\end{align}

(2.12)

In Equations (2.11) and (2.12) $C_1$ is a fraction of the still water depth denoting the distance from the still water level at which the horizontal velocity vector, $\mathbf{u}_{\alpha,*}^{\varepsilon}$, is measured. In comparison with the equations of Peregrine (1967) it is evident that the frequency dispersive term appearing in the momentum equation is different, while the continuity equation contains an additional term of the order $O(\varepsilon^2)$.

By including terms only of the order $O(1, \varepsilon^2)$ Nwogu (1993) investigated the linear dispersion characteristics of the new equations in a single horizontal direction. In the case of a horizontal bottom the linearized equations are given by

\begin{align}
\mathbf{u}_{\alpha,*}^{\varepsilon} + \mathbf{g}^{\varepsilon} \eta_{\varepsilon,*}^{\varepsilon} + \alpha \mathbf{D}^{\varepsilon} \mathbf{u}_{\alpha,*}^{\varepsilon} &+ \mathbf{D}^{\varepsilon} \mathbf{u}_{\alpha,*}^{\varepsilon} = 0 \\
\eta_{\varepsilon,*}^{\varepsilon} &+ \mathbf{D}^{\varepsilon} \mathbf{u}_{\alpha,*}^{\varepsilon} + \left( \alpha + \frac{1}{3} \right) \mathbf{D}^{\varepsilon} \mathbf{u}_{\alpha,*}^{\varepsilon} = 0
\end{align}

(2.13)

(2.14)

in which $\mathbf{u}_{\alpha,*}^{\varepsilon}$ is the horizontal velocity component in the $x^*$-direction, and

$$
\alpha = \frac{1}{2} C_1^2 + C_1
$$

(2.15)

By use of Equations (2.13) and (2.14) it was demonstrated that excellent linear dispersion characteristics can be obtained in deeper water if the velocity vector is taken at a point situated 53.1% of the still water depth below the still water surface, i.e. $C_1 = -0.531$. In
comparison with the extended Boussinesq equations of Madsen & Sorensen (1992) this is a clear improvement, since the new equations additionally are valid for bed slopes as large as \( O(\varepsilon) \).

Nwogu (1993) used an iterative Crank-Nicholson scheme to solve Equations (2.11) and (2.12) numerically in a single horizontal dimension. Third derivative truncation error terms were incorporated as intentional distortions by back substitution into the numerical scheme. In comparison with laboratory experiments the results showed that the new equations are able to propagate both regular and irregular waves reasonably from deep to shallow water.

The iterative Crank-Nicholson scheme presented by Nwogu (1993) was extended to cover the second horizontal dimension by Nwogu & Mansard (1994). They used the equations of Nwogu to study the propagation of directional waves through a breakwater gap into a harbour. Incoming waves were imposed at the boundary of the computational domain while outgoing waves were absorbed by use of a sponge layer method. A comparison with linear diffraction theory exhibited a reasonable agreement, hence giving a first impression of the capabilities of the model.

Wei & Kirby (1994) approximated the equations of Nwogu (1993) in a more direct way than described above. Instead of incorporating third derivative truncation error terms by backsubstitution into the numerical scheme first derivatives were discretized correct to the fourth order in the time step and in the spatial step. Updating of the computational domain was performed by the third order Adams-Bashforth predictor scheme followed by an iterative correction to convergence using the fourth order Adams-Moulton corrector method. For both the predictor stage and the iterative stage the surface elevation was updated explicitly whereas updating of the horizontal velocity vector involved the solution of a tridiagonal system of equations. Numerical simulations were compared to experimental data demonstrating that the model is able to simulate accurately the transformation of an irregular wave train in initially deep water as it propagates into shoaling water. A comprehensive analysis of the properties of the solution method and the associated boundary conditions can be found in Wei & Kirby (1995).

Further improvements of Boussinesq-type equations were carried out by Kirby & Wei (1994) who extended the equations of Nwogu (1993) to include all orders of the nonlinearity parameter, \( \delta \). By proper choice of the horizontal and the vertical length scales as well as a series expansion for the velocity potential satisfying the Laplace equation in the fluid interior, the fully nonlinear boundary value problem was constructed. By following the solution procedure outlined in Section 2.2 a new set of equations was derived which exhibited correct behaviour to \( O(\varepsilon^2) \), at any order of \( \delta \). In comparison with the standard equations of Nwogu the improved equations involve numerous additional derivatives, hence making even a numerical solution elaborate. Despite this, the equations were solved numerically in a single horizontal dimension by Wei et al. (1995) who utilized the predictor-corrector method of Wei & Kirby (1994). By comparison with the results of a numerically exact boundary element method of Grilli et al. (1989) it was shown that the improved equations predict the wave height variation more accurately than the standard equations. Gobbi & Kirby (1996) incorporated additional terms of \( O(\varepsilon^4) \) in the formulation, thus resulting in better frequency dispersion in deeper water.

Wave propagation over a submerged shelf was studied by Ohyama et al. (1995) who solved numerically three different sets of equations in a single horizontal dimension. The first set of equations was identical to that of Nwogu (1993) while the second and the third set, respectively, were based on a second order Stokes-type approximation and the fully nonlinear potential theory. By comparison with wave flume experiments it was shown that the
Boussinesq-type equations of Nwogu predict the wave profile over the shelf very well whereas higher harmonics of transmitted waves are overestimated. Overall, the fully nonlinear model was found to give the best results, as expected.

Beji & Battjes (1994) studied the similar problem of waves propagating over a trapezoidal bar using a Boussinesq-type model with improved linear dispersion characteristics in deeper water. Although the equations were not identical to those developed by Madsen et al. (1991) the extension to deeper water was performed by a similar method on the basis of the equations of Peregrine (1967). The validity of the model was confirmed by comparison with wave flume experiments.

A different approach was performed by Nadaoka et al. (1994) who derived a set of Boussinesq-type equations in two horizontal dimensions. Unlike previous derivation procedures the Galerkin method was used to provide an optimum form of the horizontal velocity profile, thus leading to a set of equations with excellent dispersion characteristics at all water depths. In a single horizontal dimension a comparison of several numerical examples with wave flume experiments demonstrated the ability of the model to reproduce correctly not only the wave profile but also the velocity field. Beji & Nadaoka (1994) extended the computational model to include the second horizontal dimension. At the boundary of the computational domain outgoing waves were filtered out by employment of a higher order radiation condition originally proposed by Engquist & Majda (1977).

Recently, Schäffer & Madsen (1995) derived yet another set of Boussinesq-type equations using the equations of Nwogu (1993) and the ideas of Madsen et al. (1991). The new equations incorporate further improvements of the linear shoaling and dispersion characteristics for wave lengths as small as the water depth. By retaining terms of higher order in both the frequency dispersion and the amplitude dispersion, Madsen et al. (1996) derived a new set of Boussinesq-type equations. In order to enhance the linear dispersion characteristics in deep water the equations were modified using the technique described by Schäffer & Madsen (1995). Since the new set of equations incorporates fifth derivative terms, it is relatively unsuitable for computational purposes. Chen et al. (1996) employed the same technique to enhance the linear frequency dispersion caused by the interaction of waves and currents.

In this chapter a summary has been given of the efforts undertaken to describe the propagation of long waves in water of constant or variable depth. Owing to the fact that equations of the Boussinesq-type are restricted to shallow water several attempts have been made to extend the range of application of the equations to deeper water. This has been done by Nwogu (1993) on a firm and rigorous basis. Consequently, the computational model developed in this report is based on the equations of Nwogu (1993). The next chapter describes the development of a computational model in a single horizontal dimension.
Chapter 3

Numerical Solution Method

3.1 General

From the previous chapter it is apparent that a variety of Boussinesq-type equations can be applied in simulating the propagation of long waves in water of variable depth provided the spatial derivatives of the still water depth are $O(\epsilon)$. Moreover, it is possible to include additional terms in the formulation caused mainly by wave breaking and bottom friction, thus making the type of equations applicable to a range of coastal problems.

The present chapter describes the development of a computational model in a single horizontal dimension. The model is capable of describing the evolution of relatively long, weakly nonlinear waves in water of variable depth.

The governing equations of the problem in hand are presented and the formal boundary conditions are outlined. All variables are made dimensionless. From a computational viewpoint this is done in a more convenient way than often suggested when developing the theory of long waves. Particular attention is paid to the discretization of the governing equations as well as the numerical procedure used to advance the computations in time.

A special technique is described which allows the incident wave field to be generated inside the computational domain. Although the method is presented in 2-D it can be extended to include the second horizontal dimension. By employment of a Fourier approximation method, numerically exact, it is possible to impose an incoming regular wave field which satisfies the governing equations on a horizontal bottom. Since both of these techniques constitute a cornerstone of the present model, relatively detailed information is given.

Absorption of waves propagating out of the computational domain is performed by a relatively simple but very reliable and efficient method. Similarly, two conditions are established allowing the incident wave field to be reflected at bounding walls.

Finally, a brief description is given of the developed code.
CHAPTER 3. NUMERICAL SOLUTION METHOD

3.2 Governing Equations, Boundary and Initial Conditions

The summary of various long wave theories has revealed that the improved Boussinesq-type equations of Nwogu (1993) are derived on a firm and rigorous basis. The equations can be used to describe the propagation of regular as well as irregular waves in water of variable depth provided the magnitude of the spatial derivatives of the still water depth does not exceed $O(\epsilon)$. Since improved linear dispersion characteristics in deeper water are incorporated into the equations by proper choice of a horizontal velocity variable, these provide a sound basis for wave propagation studies in coastal regions.

It has been noted that Kirby & Wei (1994) extended the equations of Nwogu (1993) to include all amplitude dispersive terms. Regardless of the fact that this may lead to an improved description of wave height envelopes in the vicinity of the breaking point, it is believed that the additional effort required to solve the equations can not be justified by the gain in the accuracy.

In this chapter a computational model is described, based on the equations of Nwogu (1993). Although the model is presented in a single horizontal dimension (2-D) it can be extended to include the second one (3-D). This is done in Chapter 5.

In Figure 3.1 a definition sketch of the computational domain is shown. The figure defines a coordinate system with a horizontal axis, $x^*$, and a vertical axis, $z^*$. The origin of the coordinate system is located at the still water level, $z^* = 0$, while the elevation of the bottom is given by $z^* = -D^*(x^*)$. Hence, the still water depth is equal to $D^*(x^*)$. Using the still water level as a datum the instantaneous surface elevation is denoted $\eta^*(x^*, t^*)$, where $t^*$ denotes the time. This implies that the mean water level (MWL) at any point in space can be found quite simply as the average of $\eta^*$ within an integral number of wave periods. The quantity, $C_1$, is a fraction of the still water depth denoting the level at which the horizontal velocity variable, $u^*_d(x^*, t^*)$, is measured, see Figure 3.1. Additionally, the figure defines

![Figure 3.1: The sketch defines relevant dependent and independent variables of the physical problem. The left and right model boundaries, denoted $\Gamma_L$ and $\Gamma_R$, respectively, are also shown.](image-url)
3.2. Governing Equations. Boundary and Initial Conditions

In dimensional form the momentum equation and the continuity equation derived by Nwogu (1993) are given by

\[
\left( u_{\alpha}^{*} + u_{\alpha}^{*}u_{\alpha}^{*} + g^{*} \eta^{*} + \frac{1}{2} z_{\alpha}^{*} \frac{D^{*}}{D^{0}} \eta^{*} \right) \frac{D^{*}}{D^{0}} + z_{\alpha}^{*} \left( D^{*} \eta^{*} \right) \frac{D^{*}}{D^{0}} = 0
\]

(3.1)

and

\[
\eta_{t}^{*} + \left[ u_{\alpha}^{*} \left( D^{*} + \eta^{*} \right) \right]_{x^{*}} + \left[ \left( \frac{1}{2} z_{\alpha}^{*} - \frac{1}{6} D^{*} \right) D^{*} \eta^{*} \right]_{x^{*}} \frac{D^{*}}{D^{0}} = 0
\]

(3.2)

in which \( g^{*} \) is the acceleration due to gravity, and \( z_{\alpha}^{*} = C_{1} D^{*} \), see Figure 3.1.

The first and the second terms appearing in the momentum equation describe the local and the convective accelerations of the fluid particles. These are of the order \( O(1) \) and \( O(\epsilon) \), respectively. The third term is a gravitational term of the order \( O(1) \). The remaining terms are frequency dispersive terms of the order \( O(\epsilon^{2}) \). It may be noted that the momentum equation is nonlinear due to the presence of a convective term.

The first two terms of the continuity equation are readily identified as the temporal change in the surface elevation and the spatial gradient of the (pseudo) volume flux. The first term is of the order \( O(1) \), while the surface elevation in the second term is \( O(\epsilon) \). It appears that the choice of horizontal velocity variable results in additional terms in the depth-integrated continuity equation. The terms are of the order \( O(\epsilon^{2}) \), and these incorporate spatial derivatives up to and including the third order.

In this chapter the equations of Nwogu (1993) are solved numerically by employment of a finite difference method. All variables are made dimensionless. As mentioned previously the development of long wave theories is commonly based partly upon a nondimensionalization factor, \( \lambda^{*} \), which is a characteristic horizontal length scale in the wave motion. For regular waves \( \lambda^{*} \) could be chosen equal to the wave length at one particular depth. Since the bottom slope may be as large as \( O(\epsilon) \), the wave length can not be assumed to be approximately constant throughout the computational domain, indicating that a characteristic depth must be assumed in determining the wave length. In this work the process of nondimensionalization is therefore based directly on a characteristic depth, \( D_{0}^{*} \), chosen as the still water depth at the left boundary, \( \Gamma_{L} \). It may be noted, though, that since one is concerned only with orders of magnitude, it would have been as legitimate to choose any other characteristic depth within the computational domain.

The variables of the problem in hand are scaled in accordance with the equation

\[
\begin{align*}
(x^{*}, z^{*}, D^{*}, \eta^{*}) &= (x, z, D, \eta) D_{0}^{*} \\
t^{*} &= t \sqrt{\frac{D_{0}^{*}}{g^{*}}} \\
u_{\alpha}^{*} &= u_{\alpha} \sqrt{g^{*} D_{0}^{*}}
\end{align*}
\]

(3.3)
CHAPTER 3. NUMERICAL SOLUTION METHOD

hence indicating that the horizontal coordinates, the still water depth, and the surface elevation are normalized by \( D_0 \). Similarly, the horizontal velocity, \( u_0 \), is scaled by the linear shallow water celerity of Stokes. By substitution of Equation (3.3) into Equations (3.1) and (3.2) an equivalent set of dimensionless equations appear. These are written below in a form convenient for computational purposes. The momentum equation is given by

\[
\mathbf{u}_0 \mathbf{u}_x + \eta_x + a_1 u_0 + a_2 u_{xx} + a_3 u_{xxx} = 0
\]  

(3.4)

while the continuity equation can be written

\[
\eta_t + \left[ (D + \eta) u_0 \right]_x + a_4 u_0 + a_5 u_x + a_6 u_{xx} + a_7 u_{xxx} = 0
\]  

(3.5)

The coefficients, \( a_m = a_m(x) \), \( m = 1, 2, \ldots, 7 \), appearing in the equations are functions merely depending on the bathymetry. These are given by

\[
a_1(x) = 1 + C_1 D D_{xx}
\]  

(3.6)

\[
a_2(x) = 2 C_1 D D_x
\]  

(3.7)

\[
a_3(x) = C_1 \left( \frac{C_1}{2} + 1 \right) D^2
\]  

(3.8)

\[
a_4(x) = (C_1 + \frac{1}{2})(2 D_x D_{xx} + D D_{xxx}) D
\]  

(3.9)

\[
a_5(x) = (C_1 + \frac{1}{2})(3 D D_{xx} + 4 D_x D_x) D
\]  

(3.10)

\[
a_6(x) = \left( \frac{3}{2} C_1^2 + 5 C_1 + 2 \right) D^2 D_x
\]  

(3.11)

\[
a_7(x) = \left( \frac{1}{2} C_1^2 + C_1 + \frac{1}{3} \right) D^3
\]  

(3.12)

In order to obtain a closure of the mathematical problem two conditions must be imposed at the left and right boundaries of the fluid domain. These account for the semi-infinite domains left out of the computation. Formally, this is written

\[
\begin{align*}
\mathbf{u}_0 \left( x_{\text{min}}, t \right) &= f_L(t) \\
\eta \left( x_{\text{min}}, t \right) &= g_L(t)
\end{align*}
\]  

(3.13)
3.3. Temporal Updating of the Computational Domain

\[ u_0(x_{\text{max}}, t) = f_R(t) \]
\[ y(x_{\text{max}}, t) = g_R(t) \]

, \( t > 0 \) \hspace{1cm} (3.14)

in which the dimensionless functions, \( f_L(t), g_L(t), f_R(t), \) and \( g_R(t) \), are assumed to be known. The quantities, \( x_{\text{min}} \) and \( x_{\text{max}} \), denote the \( x \)-axis coordinates at the left and right boundaries, respectively. It is not a trivial task to impose adequate boundary conditions, and the problem is therefore described in detail in Sections 3.5 - 3.8.

Before a numerical solution can be sought the quantities, \( u_0(x, t) \) and \( y(x, t) \), must be known at previous time levels. In this thesis an initially quiescent water state is considered. In dimensionless form this is written

\[ u_0(x, t) = 0 \]
\[ y(x, t) = 0 \]

, \( x_{\text{min}} \leq x \leq x_{\text{max}}, t \leq 0 \) \hspace{1cm} (3.15)

hence closing the mathematical problem. Equations (3.4) - (3.15) are the foundation of the computational model described in the present chapter.

3.3 Temporal Updating of the Computational Domain

One of the important findings of Abbott et al. (1984) was the fact that the discretization of first derivative terms is of particular importance since these, if not approximated sufficiently accurately, result in third derivative truncation error terms of the second order. Since any such term can be manipulated into the same form as the original third derivative dispersive term by employment of the linearized long wave equations (see Long, 1964), the spurious truncation error terms must be eliminated. Most existing solution procedures do this by back substitution into the original scheme, thus incorporating the third derivative truncation error terms as intentional distortions to the modelled dispersive effects, see Abbott et al. (1984) and Nwogu (1993). In this way terms influencing the dispersion characteristics of the solution are brought to third order accuracy, while the remaining terms are approximated correct to the second order.

Originally, the idea was to solve the above problem in a more consistent manner by developing a difference scheme in which the truncation error was of the fourth order in both the time step and the spatial step. On the basis of the successful work of McCowan (1978), who used a modified box operator centred between computational nodes at the considered time level, a highly accurate numerical scheme was developed. Both the velocity variable and the surface elevation were defined in each computational node. Unlike most other attempts to simulate long waves, the difference scheme was consistently fourth order accurate, i.e. all terms in the governing equations were discretized correct to the fourth order at all computational points (including points close to the boundaries). This resulted in a sparse...
Figure 3.2: Definition of the computational mesh. The sketch shows that both the horizontal velocity and the surface elevation are defined in each node. The grid size is denoted $\Delta x$ while the time step is denoted $\Delta t$.

linear system of equations composed of a diagonal, a neighbouring lower diagonal, and two adjacent upper diagonals. The system of equations was solved by invoking a Double-Sweep algorithm, see Vreugdenhil (1989). Several test computations have shown that the method is unconditionally unstable and therefore impossible to use in practice. It was hoped that the instability was caused by the highly accurate difference schemes used in the vicinity of the model boundaries which inevitably are asymmetric in space (Roache, 1976). However, replacement of the relevant number of nodes by spatially centred difference schemes of lower order accuracy resulted in no significant improvements. An investigation of the method, the calculations and the implementation has revealed no errors of significance. Since all numerical tests resulted in instability, the method was not pursued further.

Since the predictor-corrector pairs of Adams are known for their good stability properties, the basic numerical scheme developed in the following is based on the method of Wei & Kirby (1994). It may be noted that the discretization of all spatial derivatives is left out for the moment in order to emphasize the time-stepping procedure.

As shown in Figure 3.2 a uniform mesh is adopted in which $u_0$ and $\eta$ are defined at each computational point. The nodes are uniformly distributed over the computational domain, $x \in [x_{\text{min}}, x_{\text{max}}]$, with a grid spacing, $\Delta x$, and a node numbering system ranging from zero to a maximum number, $NN$. In addition the index, $i$, denotes the node number; $n$ is the time level; $\Delta t$ is the time step; and $NN$ is the maximum time level.

Although the methods of Adams are tailored to solving ordinary differential equations they can be applied in solving systems of partial differential equations. By rearranging terms in Equations (3.4) and (3.5) the third order Adams-Bashforth predictor method is used to provide the initial estimate of $u_0$ and $\eta$ at each new time level. The method is given by Gear
3.3. Temporal Updating of the Computational Domain

It reads

\[ R^{i,n+1} = R^{i,n} + \frac{\Delta t}{12} (23F_1^{i,n} - 16F_1^{i,n-1} + 5F_1^{i,n-2}) \]  \hspace{1cm} (3.16)

\[ \eta^{i,n+1} = \eta^{i,n} + \frac{\Delta t}{12} (23F_2^{i,n} - 16F_2^{i,n-1} + 5F_2^{i,n-2}) \]  \hspace{1cm} (3.17)

By manipulation of the governing equations it appears that \( R^{i,n} \) can be written

\[ R^{i,n} = a_1^i u^{i,n}_\alpha + a_2^i u^{i,n}_\alpha + a_3^i u^{i,n}_{\alpha z} \]  \hspace{1cm} (3.18)

while the spatial derivatives equal

\[ F_1^{i,n} = -\eta^{i,n}_z - u^{i,n}_\alpha u^{i,n}_\alpha \]  \hspace{1cm} (3.19)

and

\[ F_2^{i,n} = -[(D^i + \eta^{i,n}) u^{i,n}_\alpha]_z \]  \hspace{1cm} (3.20)

\[ -a_4^i u^{i,n}_\alpha - a_5^i u^{i,n}_\alpha - a_6^i u^{i,n}_{\alpha z} - a_7^i u^{i,n}_{\alpha z z} \]

From the above expressions it is clear that \( \eta \) can be predicted explicitly, whereas \( u_\alpha \) is given implicitly in terms of the quantity, \( R \).

Assuming that both \( u_\alpha \) and \( \eta \) have been predicted the functions, \( F_1 \) and \( F_2 \), can be evaluated at each computational point at the subsequent time level. Hence, by employment of the fourth order Adams-Moulton corrector scheme improved estimates of \( u_\alpha \) and \( \eta \) can be obtained. Details of the fourth order Adams-Moulton corrector method can be found in Gear (1971). It reads

\[ R^{i,n+1} = R^{i,n} + \frac{\Delta t}{24} (9F_1^{i,n+1} + 19F_1^{i,n} - 5F_1^{i,n-1} + F_1^{i,n-2}) \]  \hspace{1cm} (3.21)

\[ \eta^{i,n+1} = \eta^{i,n} + \frac{\Delta t}{24} (9F_2^{i,n+1} + 19F_2^{i,n} - 5F_2^{i,n-1} + F_2^{i,n-2}) \]  \hspace{1cm} (3.22)

Since the corrector stage requires information from the subsequent, the present and the two previous time levels the quantities, \( F_1 \) and \( F_2 \), must be stored at four different time levels in order to advance the computations in time. In contrast, the quantities, \( \eta \) and \( R \), must be stored only at a single time level. Furthermore, it should be noted that employment of the predictor scheme is a necessity, since information from the new time level is included implicitly in the equations through the functions, \( F_1^{i,n+1} \) and \( F_2^{i,n+1} \). By analogy with the predictor
scheme improved estimates of $\eta$ can be obtained explicitly, whereas the correction of $u_\alpha$ is bound to be done implicitly through the quantity $R$.

For both the predictor and the corrector stage determination of $u_\alpha$ is done by employment of Equation (3.18) in conjunction with the difference approximations for the spatial derivatives. Doing this, and collecting terms of the same kind result in a tridiagonal equation system of the form

$$
\begin{align*}
R^{0,n+1} &= u_{\alpha}^{0,n+1} \\
R^{i,n+1} &= D1^i u_{\alpha}^{i-1,n+1} + D2^i u_{\alpha}^{i,n+1} + D3^i u_{\alpha}^{i+1,n+1}, \ i = 1, \ldots, II - 1 \\
R^{II,n+1} &= u_{\alpha}^{II,n+1}
\end{align*}
$$

(3.23)

where $D1^i, D2^i$ and $D3^i, i = 1, \ldots, II - 1$, are coefficients derived from the difference approximations of $u_{\alpha x}$ and $u_{\alpha xx}$. Since the coefficients are constant in time, these are prefactored and stored for use at each time level. Equation (3.23) is solved by an efficient Gaussian elimination process described in Appendix A. Additionally, it may be of interest to note that the boundary conditions for $u_\alpha$ are invoked at this stage, see Equations (3.13) and (3.14).

At each time level the corrector stage, Equations (3.21) and (3.22), is repeated until a predetermined accuracy is obtained for both $u_\alpha$ and $\eta$. The stop criterion is chosen as the point where the absolute relative change in each dependent variable (summed over all computational nodes) does not exceed a predetermined value. Mathematically, this can be written

$$
\Delta f = \frac{\sum_{i=0}^{II} |(f_{new}^{i,n+1} - f_{old}^{i,n+1})|}{\sum_{i=0}^{II} |f_{new}^{i,n+1}|} < 0.001
$$

(3.24)

in which $f = \{\eta, u_\alpha\}$, and the subscripts, old and new, respectively, refer to the current and the improved estimate of the considered variable at a given time level.

Before the computations can be stepped forward in time the spatial derivatives of the governing equations must be approximated at the computational points, $i = 1, \ldots, II - 1$. In this chapter three kinds of computational points are considered. The first type is referred to as boundary points, since these are situated directly at the model boundary. Computational points whose numerical operator is affected by the presence of the model boundary are termed special nodes. All other computational points are termed internal nodes.

### 3.4 Discretization of Spatial Derivatives

#### 3.4.1 Internal Nodes

Clearly, first order spatial derivatives must be discretized correct to the third or the fourth order in the spatial step to ensure a numerical solution representing the differential equations.
A fourth order difference approximation which utilizes information from two adjacent nodes on each side of the centre point is adopted. It can be written

\[ f_{x}^{i,n} = \frac{f^{i-2,n} - 8f^{i-1,n} + 8f^{i+1,n} - f^{i+2,n}}{12\Delta x} + O(\Delta x^4) \]  

(3.25)

in which \( f = \{\eta, u_{o}, (D + \eta)u_{o}\} \), and \( i = 2, \ldots, II - 2 \).

The remaining spatial derivatives in Equations (3.18) - (3.20) as well as the spatial derivatives appearing in the coefficients, \( a_{m}, m = 1, \ldots, 7 \), are all discretized correct to the second order in the spatial step, since this is known to provide sufficient accuracy. For \( i = 1, \ldots, II - 1 \) first derivatives of the still water depth are discretized as

\[ D_{x}^{i} = -\frac{D^{i-1} + D^{i+1}}{2\Delta x} + O(\Delta x^2) \]  

(3.26)

while second derivative terms are approximated by

\[ f_{xx}^{i,n} = \frac{f^{i-1,n} - 2f^{i,n} + f^{i+1,n}}{\Delta x^2} + O(\Delta x^2) \]  

(3.27)

Similarly, correct to the second order third derivatives can be written

\[ f_{xxx}^{i,n} = -\frac{f^{i-2,n} + 2f^{i-1,n} - 2f^{i+1,n} + f^{i+2,n}}{2\Delta x^3} + O(\Delta x^2) \]  

(3.28)

where \( i = 2, \ldots, II - 2 \). In Equations (3.27) and (3.28), \( f = \{D, u_{o}\} \).

From the above difference approximations it is evident that the internal nodes are numbered \( i = 2, \ldots, II - 2 \), while the special nodes correspond to \( i = 1 \) and \( i = II - 1 \).

### 3.4.2 Special Nodes

The difference approximations, Equations (3.25) and (3.28), can not be used at the points \( i = 1 \) and \( i = II - 1 \), since this would require information from points situated outside the computational domain. Consequently, special precautions must be taken in these nodes. As a substitution for Equation (3.25) a fourth order accurate scheme, asymmetric about the centre point, was developed. It has been pointed out by Roache (1976) that highly accurate difference schemes which are not centred can lead to unstable computations. Indeed, this was the case here. Test computations revealed that numerical errors, growing in time, were generated in the special nodes and spread further into the computational domain, eventually causing the computations to terminate. The problem was cured by introduction of a second order accurate difference approximation, hence leading to an overall discretization of first order.
spatial derivatives which is not consistently fourth order accurate. For \( i = 1 \) and \( i = II - 1 \) the equivalent of Equation (3.25) reads

\[
f_{i,n}^{x} = \frac{-f_{i-1,n}^{i-1,n} + f_{i+1,n}^{i+1,n}}{2 \Delta x} + O(\Delta x^2)
\]  

(3.29)

where, as before, \( f = \{\eta, u_\alpha, (D + \eta) u_\alpha\} \).

In the special nodes, \( i = 1 \) and \( i = II - 1 \), there is only a single information point available on one side of the centre point (the boundary node). As a consequence it is impossible to maintain second order accuracy for the third derivative terms in Equation (3.28) without introducing an asymmetric difference scheme based on extensive use of information at internal nodes. It has turned out that the approximation of third derivative terms by use of asymmetric difference schemes has no influence on the stability properties of the computations. The difference schemes approximating \( D \) and \( u_\alpha \) at \( i = 1 \) and \( i = II - 1 \), respectively, are here given by

\[
f_{i,n}^{xx} = \frac{-3 f_{i-1,n}^{i-1,n} + 10 f_{i,n}^{i,n} - 12 f_{i+1,n}^{i+1,n} + 6 f_{i+2,n}^{i+2,n} - f_{i+3,n}^{i+3,n}}{2 \Delta x^3} + O(\Delta x^2)
\]  

(3.30)

\[
f_{i,n}^{xxx} = \frac{f_{i-3,n}^{i-3,n} - 6 f_{i-2,n}^{i-2,n} + 12 f_{i-1,n}^{i-1,n} - 10 f_{i,n}^{i,n} + 3 f_{i+1,n}^{i+1,n}}{2 \Delta x^3} + O(\Delta x^2)
\]  

(3.31)

in which \( f = \{D, u_\alpha\} \). (Note: Difference schemes can be constructed in various ways. Here it was done, to a desired order of accuracy, by expanding a relevant number of information points in a Taylor series about the centre point. Each expansion series was then multiplied by an unknown factor. By requiring Taylor series terms of the same kind - except the term of interest - to add up to zero, a (generally non-quadratic) linear system of equations emerged. The system was normalized and solved, yielding the coefficients of the difference scheme.)

### 3.4.3 Boundary Nodes

At the boundary of the computational domain it is crucial to impose accurate and sufficient conditions, since these influence the behaviour of the solution inside the fluid domain. For clarity it has been assumed that the dependent variables are given explicitly by Equations (3.13) and (3.14). This is far from the case in reality as will become apparent.

For many practical problems it is desirable to be able to impose either a purely reflecting condition or an absorbing condition at the model boundary. The current section describes the construction of a purely reflecting boundary condition, whereas the rather more complex problem of imposing adequate absorbing boundary conditions is given in Section 3.5.

For waves incident to a vertical barrier the mass flux perpendicular to the barrier must be equal to zero. By the assumption of a constant still water depth at the model boundary the mass flux, \( Q \), can be found from the continuity equation. It reads

\[
Q = (D + \eta) u_\alpha + a_\gamma u_{\alpha xx}
\]  

(3.32)
where \( a_7 = (C_1 (C_2 + 1) + \frac{1}{3}) D^3 \) thus indicating that, in addition to the requirement \( u_\alpha = 0 \), it is crucial to impose \( u_{\alpha xx} = 0 \). The latter requirement constitutes a minor problem, since it is possible to impose only a single condition for \( u_\alpha \) in solving the system, Equation (3.23). By maintaining terms of the order \( O(1) \) in the continuity equation, Equation (3.5), and differentiating with respect to \( x \) it can be deduced that the condition \( u_{\alpha xx} = 0 \) can be replaced by requiring \( \eta_x = 0 \) at all times. Since this is an approximation, it is inevitable that there will be a minor insignificant flow of mass through the computational boundary.

### 3.5 Absorption of Outgoing Waves

Perfect absorption of outgoing waves can be obtained only in very specialized cases, e.g. in one-dimensional linear wave problems. The Sommerfeld radiation condition used in potential theory which states that the wave field must be purely outgoing at infinity is commonly used for this type of problem (Sommerfeld, 1949). However, since it is accurate only for linear waves incident to the outgoing boundary (Givoli, 1991), it is often inapplicable in practical wave problems. By introduction of pseudo differential operators in conjunction with an expansion technique a hierarchy of higher-order absorbing boundary conditions was derived from the wave equation by Engquist & Majda (1977). Israeli & Orszag (1981) illustrated how a combination of wave damping in the vicinity of the boundary, so-called sponge layers, and higher order absorbing boundary conditions efficiently damps outgoing waves. These ideas were also followed by Romate (1992) and Broeze & Romate (1992) who derived and implemented numerically first and second order absorbing boundary conditions on the basis of the Laplace equation.

In one horizontal dimension a more practical approach was adopted by Beji & Battjes (1994) who studied the propagation of long waves over a bar. The numerical simulation was based on a set of Boussinesq equations compatible with those of Peregrine (1967). At the outgoing boundary the Sommerfeld condition was used to absorb both the surface elevation and the depth-averaged horizontal velocity. In order to avoid excessive reflection of wave energy only waves of small height relative to the still water depth were simulated, thus limiting severely the application of the model.

Because of the rather complex nature of the partial differential equations under consideration it was decided to generate the incident wave field internally, and damp waves travelling out of the computational domain by employment of a sponge layer method. Clearly, the combination of sponge layers and an internal wave generation method allows the scattered wave field to be absorbed accurately and efficiently in both the upstream and the downstream end of the computational domain. In comparison with the majority of existing Boussinesq-type models this is a very useful improvement.

The sponge layer effect can be obtained quite simply by inclusion of damping terms in the momentum equation and in the mass equation. These are rewritten

\[
\begin{align*}
    u_\alpha u_{\alpha x} + \eta_x + a_1 u_\alpha + a_2 u_{\alpha tt} + a_3 u_{\alpha xtt} + \gamma(x)f_1(x, t) &= 0 \\
    \eta_t + [(D + \eta) u_\alpha]_x + a_4 u_\alpha + a_5 u_{\alpha x} + a_6 u_{\alpha xx} + a_7 u_{\alpha xxx} + \gamma(x)f_2(x, t) &= 0
\end{align*}
\]
where $\gamma(x)$ is a predetermined damping coefficient, and $f_1(x, t)$ and $f_2(x, t)$ are functions correlated to $R(x, t)$ and $\eta(x, t)$, respectively. In order to minimize the amount of wave energy reflected from the sponge layer $\gamma(x)$ must increase very gradually from zero furthest into the computational domain to a maximum value at the boundary. In qualitative agreement with Larsen & Dancy (1983) this is done by selecting a quadratic variation of the form

$$\gamma(x) = \gamma_{\text{max}} \frac{|x - x_0|^2}{x_s^2}$$  \hspace{1cm} (3.35)

in which $\gamma_{\text{max}}$ is the largest sponge layer value, $x_0$ denotes the $x$-axis location where the sponge layer starts, and $x_s$ is the width of the sponge layer. The quantity, $\gamma_{\text{max}}$, is typically chosen as $\gamma_{\text{max}} = 0.75$. Additionally, it is mentioned that the number of sponge layer nodes accompanying the left and right boundaries are denoted $\text{ISL}$ and $\text{ISR}$, respectively.

By considering the predictor-corrector method used to advance the computations in time it is evident that a qualified guess on $f_2(x, t)$ would be $f_2(x, t) = \eta(x, t)$, thus implying that the amount of mass in the system would stabilize at a finite level, determined by the amount of setup in the sponge layer. (Note: Although sponge layers constitute both a robust and an efficient tool they are responsible for the dissipation of wave energy, thus causing the radiation stress to decrease, (Longuet-Higgins & Stewart, 1962). This gives rise to a local increase in the mean water level.)

By analogy with the choice of $f_2(x, t)$ the most obvious appearance of $f_1(x, t)$ is $f_1(x, t) = R(x, t)$. Although this implies that the quantity, $R$, tends to zero it does not ensure that the mass flux, $Q$, goes to zero. In the case of a constant still water depth inside the sponge layer (which can be accommodated in most practical problems) the mass flux is given by Equation (3.32), hence suggesting that the physically correct requirement is that both $u_a$ and $u_{\alpha xx}$ must tend to zero in the sponge layer region. Thus, in summarizing, the functions, $f_1(x, t)$ and $f_2(x, t)$, are chosen as

$$f_1(x, t) = u_a(x, t) + u_{\alpha xx}(x, t)$$  \hspace{1cm} (3.36)

and

$$f_2(x, t) = \eta(x, t)$$  \hspace{1cm} (3.37)

By employment of Equations (3.36) and (3.37) preliminary propagation tests over both even and uneven bottoms have suggested a width of the sponge layer of the order of one to two incoming wave lengths, thus yielding virtually no reflection of outgoing waves.

### 3.6 Internal Wave Generation

Larsen & Dancy (1983) reformulated the scattering problem as a radiation problem by adding the incident wave field at a line inside the computational domain. All waves were then treated
as outgoing waves which had to be absorbed at the model boundary. For a wide range of wave frequencies good absorption was achieved by use of sponge layers. The spatial variation of the damping coefficients within each sponge layer was derived by employment of a special technique. Depending on the desired amount of wave reflection these increased gradually from zero to a maximum value at the boundary. Tests in both one and two horizontal dimensions were carried out, successfully demonstrating the method.

The idea of adding the incident wave field internally was described in more detail by Madsen & Larsen (1987) who solved the elliptic mild-slope equation on a space-staggered rectangular grid. The incoming waves were generated internally by use of a source term in the continuity equation. At each time level the source term was represented by a change in the surface elevation of the considered cell. This was expressed in terms of the depth-integrated horizontal mass flux of the incident wave field during that time step. Since the mass added by the source term was motionless, only half of the wave energy entered the area of interest, thus implying that twice the desired mass had to be added at each time level.

Following the idea of Larsen & Dancy (1983) an attempt was made to add mass internally at a single computational point. This resulted in two carrier waves propagating in either direction, each of them superposed by rapid saw-tooth oscillations of approximately the same height as the incident wave field. Although the spurious oscillations did not grow in time they altered the wave profile to such an extent that an efficient elimination was essential for the method to perform well. By considering the numerical space derivatives in the nodes adjacent to the generation node subsequent speculations have revealed that the saw-tooth oscillations resulted from the fact that the present computational model makes use of a uniform grid and a spatially wide numerical operator in the mass equation (caused by the choice of velocity variable). This is in contrast to Larsen & Dancy (1983) and Madsen & Larsen (1987) who both used a space-staggered grid in conjunction with a very narrow numerical scheme.

Moreover, by distributing an equivalent amount of mass over a few nodes on each side of the generation point two perfectly smooth wave trains were generated. Further testing of a triangular, a sinusoidal, and a Gaussian mass distribution revealed no significant differences. For all the tested mass distributions the wave height of the generated wave field decreased in time and space until a stable wave form was achieved. It is possible that the decrease in the wave height resulted from the fact that the potential energy added implicitly by any of the mass distributions was smaller than that corresponding to adding mass at a single point.

A fundamentally different approach was adopted by Ishii et al. (1994) who solved the mild-slope equation in two horizontal dimensions. The incident wave field was generated inside the computational domain along lines, thus enabling scattered waves to be absorbed effectively by employment of sponge layers. Between the physical boundary and the line of wave generation the dependent variables were chosen to be those of the scattered wave field, see Figure 3.3. In the remaining part of the computational domain the total wave field, i.e. the sum of the incident and the scattered wave field, was considered. As a consequence, the numerical operator had to be modified at computational nodes bordering the generation line. Since the wave generation method of Ishii et al. does not rely on the choice of velocity variable (which implies a certain form of the governing equations), it appears to be well suited for the problem under consideration. The technique described below follows the principles of Ishii et al., although the equations and the numerical operator are significantly different.

In Figure 3.3 the problem of generating waves inside the computational domain is illustrated. From the figure it is apparent that the (fictitious) point of wave generation is located in immediate vicinity of the sponge layer, since \( x_0 = x_{\text{min}} + iq_0 \Delta x \), cf. Equation (3.35). By choice, the scattered wave field is considered on the left hand side of the wave...
CHAPTER 3. NUMERICAL SOLUTION METHOD

- Computational node not affected by wave generation
- Modified computational node - scattered side
- Modified computational node - total side

Figure 3.3: Sketch illustrating the computational points affected by the wave generation technique. The computational point, $i_0$, marks the start of the damping region, which is applied on the left hand side of the wave generation point.

generation point, whereas both the scattered and the incident wave field are represented on the right hand side. In effect, this means that the variables to be solved for at each time level are $f_{i_0}^0, \ldots, f_{i_0+1}^0, \ldots, f_{II}$, where $f = \{\eta, u_\alpha\}$, and the subscript, $r$, denotes the reflected (scattered) wave field. For clarity it should be mentioned that the remaining variables denote the total wave field. Although it poses no complications from a physical viewpoint to consider the reflected wave field in part of the computational domain and the total wave field in the other, it does require the numerical solution method to be modified slightly.

From the spatial difference approximations of Section 3.4 it is evident that the numerical operator generally makes use of information from two computational points on each side of the centre point. In constructing the difference approximations used to update $f_{i_0-1}^0$, say, $f = \{\eta, u_\alpha\}$, the numerical operator will, by default, use information from nodes on both sides of the generation point. Since it makes no sense to compute a numerical derivative (in this case on the reflected side) by employment of variables representing both the scattered and the total wave field, an adjustment must be made accounting for the incident wave field. It can be deduced that the terms

$$W_i = \begin{cases} 
0 & , \ i = 0, \ldots, i_0 - 2 \\
-\frac{1}{12\Delta x} (\eta_{i+1}^{i_0+1} + u_{i_0-1}^{i_0} u_{i_0+1}^{i_0}) & , \ i = i_0 - 1 \\
\frac{1}{12\Delta x} (8\eta_{i+1}^{i_0+1} - \eta_{i-2}^{i_0+2} + u_{i_0}^{i_0} (8u_{i_0+1}^{i_0+2} - u_{i_0+2}^{i_0+2})) & , \ i = i_0 \\
\frac{1}{12\Delta x} (-\eta_{i+1}^{i_0+1} + 8\eta_{i-2}^{i_0} + u_{i_0+1}^{i_0+1} (-u_{i_0+1}^{i_0+1} + 8u_{i_0+2}^{i_0+1})) & , \ i = i_0 + 1 \\
-\frac{1}{12\Delta x} (\eta_{i+1}^{i_0} + u_{i_0+2}^{i_0} u_{i_0+1}^{i_0}) & , \ i = i_0 + 2 \\
0 & , \ i = i_0 + 3, \ldots, II 
\end{cases}$$

(3.38)
and

\[
W'_2 = \begin{cases} 
0, & i = 0, \ldots, i_0 - 2 \\
\frac{1}{12 \Delta x} \left( 8 \left( D^{i_0+1} + \eta_f^{i_0+1} \right) u_{\alpha i}^{i_0+1} - (D^{i_0+2} + \eta_f^{i_0+2}) u_{\alpha i}^{i_0+2} \right) & i = i_0 - 1 \\
\frac{1}{12 \Delta x} \left( 8 \left( D^{i_0} + \eta_f^{i_0} \right) u_{\alpha i}^{i_0} - (D^{i_0-1} + \eta_f^{i_0-1}) u_{\alpha i}^{i_0-1} \right) & i = i_0 \\
\frac{1}{12 \Delta x} \left( 8 u_{\alpha i}^{i_0} - u_{\alpha i}^{i_0-1} \right) & i = i_0 + 1 \\
\frac{1}{12 \Delta x} \left( 2 u_{\alpha i}^{i_0} - u_{\alpha i}^{i_0-1} \right) & i = i_0 + 2 \\
0 & i = i_0 + 3, \ldots, II 
\end{cases}
\]  

\[W_R^i = \begin{cases} 
0, & i = 0, \ldots, i_0 - 1 \\
D3^{i_0} v_{\alpha i}^{i_0+1}, & i = i_0 \\
-D1^{i_0+1} u_{\alpha i}^{i_0}, & i = i_0 + 1 \\
0, & i = i_0 + 2, \ldots, II 
\end{cases} \]  

must be added to the spatial derivatives, \( F_1^{i,n} \) and \( F_2^{i,n} \), respectively, cf. Equations (3.19) and (3.20). In the equations given above the subscript, \( i \), symbolizes the incident wave field, i.e. \( f = f_I + f_r \), where \( f = \{ \eta, u_o \} \). As a consequence of the fact that all variables are evaluated at the considered time level the superscript, \( n \), has been omitted for clarity.

By analogy with the alterations of the spatial derivatives it is a necessity to modify the quantity, \( R \), temporarily (cf. Equation (3.18)) in solving Equation (3.23). Since the system of equations has a tridiagonal structure, changes need be made only in nodes bordering the point of wave generation. By considering the form of the equation system it is apparent that the quantity
must be added to the right hand side of the equation system, finally making it possible
to add the incident wave field inside the computational domain. It should be noted that
no presumptions have been made about its form. The method merely serves as a tool for
prescribing a given signal internally. In Sections 3.7 and 3.8 the actual form of the incident
wave field is considered.

3.7 Analytical Methods for the Determination of the Incident
Wave Field

3.7.1 Regular Waves

In quantifying the incoming, regular wave field it is a common procedure to employ either
the linearized equations of the problem in hand or Stokes' linear wave theory, see e.g. Nwogu
(1993), Karambas & Koutitas (1992), or Wei & Kirby (1995). This has proven useful in
studying the propagation of (initially) linear waves. However, since none of these approaches
accounts for the nonlinear nature of the wave motion, more accurate methods must be em-
ployed in prescribing an incident wave field of large height relative to the still water depth.

In addition to a numerically exact Fourier method described in Section 3.8, Stokes' sec-
ond order theory as well as the first order cnoidal theory are considered in the present report.
Relevant details of the analytical expressions may be found in Svendsen & Jonsson (1980).
It should be emphasized that the implementation of Stokes' second order theory is based on
an Eulerian current equal to zero, and further, that the computation of various elliptic func-
tions used in cnoidal theory is carried out by employment of standard methods given in e.g.
Abramowitz & Stegun (1965).

Although the use of either Stokes' second order theory in intermediate depth water or
the first order cnoidal theory in shallow water improves the accuracy of the computations
in comparison with using linear theory, both methods are inadequate in simulating waves of
large height, say $H=0.500$. This could be anticipated since neither of these fulfil the governing
equations considered in this thesis. By the assumption of a constant still water depth an
attempt was therefore made to solve Equations (3.4) and (3.5) in an exact, analytical man-
ner. Since this appeared to be impossible, a series expansion was carried out in a reference
system travelling with the wave celerity. By the assumption that $e^2 = O(\delta) \ll 1$ the absolute
CELERITY OF THE WAVE FIELD, $c$, the horizontal velocity component, $u_0$, and the surface elevation.
$\eta$, were expanded into a series using the nonlinearity, $\delta$, as the expansion parameter. Hence,
the validity of the solution was restricted to shallow water and wave heights much smaller
than the still water depth. In the derivation, terms up to and including the third order were
taken into account. At a given level of approximation the solution was found partly by invok-
ing the solution of the preceding level, partly by the requirement that the so-called secular
terms be equal to zero (Whitham, 1974). The latter requirement allowed the celerity to be
quantified. (Note: Physically, the secular terms represent an unbounded enhancement of the
solution, hence indicating these must be eliminated.) In shallow water the analytical solution
has turned out to provide sufficient accuracy even for large values of $\delta$ but it can not be used
satisfactorily in intermediate depth water, thus implying that an additional wave theory must
be considered in order to cover a practical range of incident water depths and periods. For
this reason a numerically exact Fourier method is described in Section 3.8.
Wei & Kirby (1995) considered the Boussinesq-type equations of Nwogu (1993), and presented an analytical solution for the solitary wave. By the assumption of a constant still water depth the governing equations were transformed into a reference system travelling with the wave celerity. Since the combination of these rendered an expression which incorporates terms of orders inconsistent with the leading order of approximation, that is $O(\delta, \epsilon^2)$, a truncation was made, thus leading to a solution which does not fulfil the governing constant depth equations exactly (although it is a good approximation). Relevant details of the dimensionless expressions are given below. The horizontal velocity component can be written (Wei & Kirby, 1995)

$$u_\alpha(x, t) = \alpha_1 \text{sech}^2(\alpha_2(x - ct))$$ (3.41)

where

$$\alpha_1 = \frac{c^2 - 1}{c}$$ (3.42)

$$\alpha_2 = \sqrt{\frac{c^2 - 1}{4(\alpha + \frac{1}{3} - \alpha c^2)}}$$ (3.43)

and $\alpha = \frac{1}{2}C_1^2 + C_1$. Analogously, the surface elevation equals

$$\eta(x, t) = \alpha_3 \text{sech}^2(\alpha_2(x - ct)) + \alpha_4 \text{sech}^4(\alpha_2(x - ct))$$ (3.44)

in which

$$\alpha_3 = \frac{c^2 - 1}{3(\alpha + \frac{1}{3} - \alpha c^2)}$$ (3.45)

and $\alpha_3 + \alpha_4 = H$. By the requirement that $\eta = H$ at $x = ct$ a dispersion relation can be derived from Equation (3.44), thus making it possible to compute the celerity of the wave form. It reads (Wei & Kirby, 1995)

$$2\alpha c^6 - (3\alpha + \frac{1}{3} + 2\alpha H)c^4 + 2H(\alpha + \frac{1}{3})c^2 + \alpha + \frac{1}{3} = 0$$ (3.46)

In Section 4.7 it will be seen that the analytical solution of Wei & Kirby (1995) significantly improves the simulation of a solitary wave in comparison with existing theory (Svendsen & Jonsson, 1980), as it satisfies the governing constant depth equations almost perfectly.
### 3.7.2 Irregular Waves

For many practical simulations it is of interest to be able to impose an irregular wave field on the basis of a known wave spectrum. In the following this is done by superposition of a number of regular waves computed by use of Stokes’ linear theory. Since irregular waves are not considered in detail in this project, the substantial effort associated with the inclusion of the bound sub- and superharmonics (Sharma & Dean, 1981) was not justified and these are not included in the present formulation, thus restricting the application of the method to wave fields of small significant height and sufficiently large peak periods.

The incoming, irregular wave field is assumed to be of the form

\[ \eta(x,t) = \sum_{i=1}^{NF} \frac{H_i}{2} \cos(\omega_i t - k_i x + \Psi_i) \]  

(3.47)

and

\[ u_\alpha(x,t) = \sum_{i=1}^{NF} \frac{\omega_i H_i}{2} \frac{\cosh(k_i D (1 + C_i))}{\sinh(k_i D)} \cos(\omega_i t - k_i x + \Psi_i) \]  

(3.48)

where \( NF \) denotes the number of frequency components in the specified wave spectrum, and \( \omega_i, k_i, H_i, \) and \( \Psi_i, \) respectively, are the angular frequency, the wave number, the wave height and the random phase of a particular wave component. The wave number, \( k_i, \) is computed by the linear dispersion relation of Stokes.

By choosing a peak period, \( T_p, \) and a significant wave height, \( H_s, \) a given wave spectrum is synthesized by \( NF \) number of frequency components uniformly distributed over the frequency axis. The lower bound of the angular frequency axis is preset to \( \omega_s = \frac{2\pi}{30.0}, \) while the upper limit is computed by the requirement that 99.5\% of the wave energy be contained in the specified wave spectrum. The lower bound ensures that the majority of all practical wave frequencies can be represented.

For each wave component the wave height is computed from the relation

\[ S(\omega_i) \Delta \omega_i = \frac{1}{8} H_i^2 \]  

(3.49)

in which \( S(\omega_i) \) denotes the wave spectrum, and \( \Delta \omega_i \) is the distance between each wave component on the \( \omega \)-axis. It is worth noting that a more sophisticated approach could be adopted by choosing both the frequency and the wave height of each wave component in accordance with their theoretically correct probability density functions. Since this is not within the scope of this thesis, the issue will not be pursued in any more detail. Instead, the interested reader is referred to Longuet-Higgins (1952, 1975, 1980, 1983). The random phases are considered (correctly) to be uniformly distributed in the interval \( \Psi_i \in [-\pi, \pi]. \)

Two different wave spectra can be specified in the computational model. The first one is a general JONSWAP wave spectrum while the second one is a Pierson-Moskowitz spectrum. Details of both of these as well as their parameters can be found in Carter et al. (1986).
3.8 A Fourier Approximation Method for the Determination of the Incident Wave Field

Test computations have shown that regular waves of large height, e.g. \( H = 0.500 \), cannot be generated satisfactorily using either first order cnoidal theory or Stokes' second order theory as the computational model generates free waves of the same magnitude as the higher order forced waves but 180° out of phase in order to fulfill the approximate nature of the wave form. This can be avoided by prescribing a signal satisfying the governing differential equations perfectly.

Based on the assumption of irrotational flow and an incompressible and inviscid fluid Rienecker & Fenton (1981) presented a numerically exact solution method for regular waves over a horizontal bottom. A stream function satisfying the Laplace equation was defined by use of a Fourier series expansion truncated at an order, say \( M \). In a reference system travelling with the wave celerity satisfaction of the fully nonlinear boundary conditions at the free surface and at the bottom resulted in \( 2M + 2 \) nonlinear equations when the symmetry about the crest was taken into account. As the equations involved \( 2M + 5 \) unknown variables (\( M + 1 \) Fourier coefficients, \( M + 1 \) discretization points at the free surface, the wave number, \( k \), a Bernoulli constant, \( R_1 \), and the flow, \( Q_1 \), underneath the steady wave) three additional equations were required to obtain a closure of the problem. The first one stated that the mean water level should be equal to zero while the second one defined the wave height as the difference between the crest and the trough elevation. The third equation introduced the identity \( kcT - 2\pi = 0 \), where \( T \) is the absolute wave period specified initially. Introduction of the celerity, \( c \), implied that an assumption as to the speed at which a wave travels must be stated. In an absolute frame of reference this was done by specification of either the mass transport velocity, \( c_s \), or the Eulerian current, \( c_E \), which finally closed the system of nonlinear equations. The nonlinear system of equations was solved using Newton's iterative method.

In the following a Fourier approximation method similar to that employed by Rienecker & Fenton (1981) is adopted. For the case of a constant water depth the governing equations, Equations (3.4) and (3.5), reduce to

\[
\begin{align*}
\frac{\partial u}{\partial t} + u_{\alpha} u_{\alpha} + \eta_x + a_3 u_{\alpha_{zz}} &= 0 \\
\frac{\partial \eta}{\partial t} + [(D + \eta) u_{\alpha}]_x + a_7 u_{\alpha_{zz}} &= 0
\end{align*}
\] (3.50)

in which \( \eta = \eta(x,t) \) and \( u_{\alpha} = u_{\alpha}(x,t) \). The constants, \( a_3 \) and \( a_7 \), are given by

\[
a_3 = C_1 \left( \frac{C_1}{2} + 1 \right) D^2
\] (3.52)

and

\[
a_7 = \left( C_1 \left( \frac{C_1}{2} + 1 \right) + \frac{1}{3} \right) D^3
\] (3.53)
In order to obtain a solution by means of a Fourier series Equations (3.50) and (3.51) must be transformed into a reference system travelling with the wave celerity. By considering the variable transformation

\[ x_1 = x - ct \]  

(3.54)

where \( x_1 \) is the horizontal coordinate in the relative frame of reference it can be seen that

\[ \frac{\partial}{\partial x} = \frac{\partial}{\partial x_1} \]  

(3.55)

and

\[ \frac{\partial}{\partial t} = -c \frac{\partial}{\partial x_1} \]  

(3.56)

such that Equations (3.50) and (3.51) can be written

\[ u_1 u_{1x_1} + \eta_{1z_1} - a_3 c u_{1x_1} U_1 = 0 \]  

(3.57)

\[ [(D + \eta_1) u_1]_{x_1} + a_7 u_{1x_1} U_1 = 0 \]  

(3.58)

It should be noted that in this frame the surface elevation, \( \eta_1 = \eta_1(x_1) \), and the horizontal velocity, \( u_1 = u_1(x_1) \). These equations are integrated once with respect to \( x_1 \), thus yielding the basis of the Fourier approximation method

\[ \frac{1}{2} u_1^2 + \eta_1 - a_3 c u_{1x_1} x_1 - R_1 = 0 \]  

(3.59)

\[ (D + \eta_1) u_1 + a_7 u_{1x_1} x_1 - Q_1 = 0 \]  

(3.60)

In Equations (3.59) and (3.60) \( Q_1 \) and \( R_1 \) denote a constant flow rate and a Bernoulli-type constant, respectively.

Since the wave is assumed to be regular and of permanent form, the symmetry about the crest can be used to propose a simplified expression consisting solely of cosine terms for the horizontal velocity component. Following Rienecker & Fenton (1981) the Fourier series can be written

\[ u_1(x_1) = U_0 + k \sum_{m=1}^{M} m U_m \frac{\cosh(mkD(1 + C_1))}{\cosh(mkD)} \cos(mx_1) \]  

(3.61)
in which \( U_m, m = 0,1,\ldots,M, \) are the Fourier coefficients to be solved for. In Equation (3.61) the horizontal velocity, \( u_i(x_i) \), is evaluated at the level \( z = C_1 D \).

The \( x_1 \)-axis is discretized by \( M + 1 \) equally spaced computational nodes. \( x_1^j, j = 0,1,\ldots,M, \) such that \( \eta_1^j = \eta_1(x_1^j) \), \( u_1^j = u_1(x_1^j) \), and \( kx_1^j = \frac{j\pi}{M} \). Hence, \( u_1^j \) can be written

\[
  u_1^j = U_0 + \kappa \Lambda_1^j
\]  

(3.62)

where

\[
  \Lambda_1^j = \sum_{m=1}^{M} m U_m \frac{\cosh(mkD(1+C_1))}{\cosh(mkD)} \cos\left(\frac{jm\pi}{M}\right)
\]  

(3.63)

By substitution of Equation (3.62) into Equations (3.59) and (3.60) the discretized momentum equation reads

\[
  \frac{1}{2} (U_0 + \kappa \Lambda_1^j)^2 + \eta_1^j + a_3 c k^3 \Lambda_2^j - R_1 = 0
\]  

(3.64)

in which

\[
  \Lambda_2^j = \sum_{m=1}^{M} m^3 U_m \frac{\cosh(mkD(1+C_1))}{\cosh(mkD)} \cos\left(\frac{jm\pi}{M}\right)
\]  

(3.65)

while the continuity equation becomes

\[
  (D + \eta_1^j)(U_0 + \kappa \Lambda_1^j) - a_7 k^3 \Lambda_2^j - Q_1 = 0
\]  

(3.66)

where \( j = 0,1,\ldots,M \). In Equations (3.64) and (3.66) the unknowns are \( \eta_1^m \) and \( U_m, m = 0,1,\ldots,M, \) \( c, k, Q_1, \) and \( R_1 \). Consequently, four more equations must be sought. These are the same as those given by Rienecker & Fenton (1981). The first one states that the mean water surface must be equal to zero,

\[
  \frac{1}{2M} (\eta_1^0 + \eta_1^M + 2 \sum_{j=1}^{M-1} \eta_1^j) = 0
\]  

(3.67)

while the second one involves the wave height

\[
  \eta_1^0 - \eta_1^M - H = 0
\]  

(3.68)
CHAPTER 3. NUMERICAL SOLUTION METHOD

The third equation can be interpreted as a kind of dispersion equation relating the wave number to the celerity.

\[ kcT - 2\pi = 0 \]  \hspace{1cm} (3.69)

As mentioned previously an additional condition must be imposed in order to obtain a closure of the mathematical problem. Following Rienecker & Fenton (1981) this can be done by specifying either the Eulerian mean current under wave trough level, \( c_E \), or the mass transport velocity, \( c_s \).

For the equations under consideration it is evident that the Eulerian mean current in the absolute frame of reference is related to the mean (in space) velocity at each level within the fluid in the moving frame of reference, \( U_0 \), through the equation

\[ U_0 - c_E + c = 0 \]  \hspace{1cm} (3.70)

thus enabling the nonlinear system of equations to be solved if \( c_E \) is specified.

Since the instantaneous flow rate in the absolute frame of reference, \( Q \), can be written

\[ Q = (D + \eta)u_1 + a_7u_{1x_1} + c(D + \eta) \]  \hspace{1cm} (3.71)

it may be ascertained that the mean flux, \( \overline{Q} \), (calculated as the mean over an integral number of absolute wave periods) reads

\[ \overline{Q} = Q_1 + cD \]  \hspace{1cm} (3.72)

Recalling that \( \overline{Q} = c_sD \).

\[ c_sD - Q_1 - cD = 0 \]  \hspace{1cm} (3.73)

Use of Equation (3.73) requires \( c_s \) to be specified.

It is believed that the distinction between \( c_E \) and \( c_s \) for many practical simulations is merely academic. This, however, is not the case when considering the propagation of waves in a closed wave flume, since any choice of \( c_s \neq 0 \) implies that the mass of the system will not be conserved in time. This may significantly influence the computation of the MWL.

From the equations given above it is apparent that specification of the depth, \( D \), the absolute wave period, \( T \), and the wave height, \( H \), as well as either \( c_E \) or \( c_s \) in combination with Equation (3.70) and Equation (3.73), respectively, results in \( 2M + 6 \) nonlinear equations with \( 2M + 6 \) unknowns. Since all the equations can be differentiated with respect to the variables involved, Newton’s method is used to obtain a solution. Although the majority of the equations above are different from those of Rienecker & Fenton (1981) the procedure is, in principle, identical.
3.8. A Fourier Approximation Method for the Determination of the Incident Wave Field

By writing the equations given above as a vector function of dimension \(2M + 6\) Newton’s method can be written

\[
\begin{align*}
J_i \Delta_i &= -f_i \\
x_{i+1} &= x_i + \Delta_i
\end{align*}
\] (3.74)

where \(i\) denotes the iteration number, \(x_i\) is the solution vector, \(\Delta_i\) is the change in the solution vector, \(f_i\) is the residual vector, and \(J_i\) denotes the Jacobi matrix. The derivation of the derivatives in the Jacobi matrix is straightforward and will be omitted here.

Newton’s method converges quadratically but it requires an accurate initial guess in order to achieve convergence. Test computations have revealed that convergence can not be obtained for waves close to the limiting height. A procedure was therefore developed which successively extrapolates the wave height until the desired height is reached. As the wave height of the first initial guess is only a small fraction of the desired height, it is derived by use of Stokes’ linear theory. For each wave height Newton’s method is used to provide a numerically exact solution. The solution vector is extrapolated in order to obtain a sufficiently accurate initial guess for the next wave height. The procedure outlined above is repeated until the specified wave height is reached. For limiting height waves convergence is obtained by less than five extrapolations, and at each stage the method converges to three decimal places within approximately one or two iterations.

Once a solution vector is obtained it is possible to evaluate \(u_1\) at any point, \(x_1\), by use of Equation (3.61). However, in order to determine the horizontal velocity in the absolute reference system, \(u_\alpha(x,t)\), the solution for \(u_1\) must be transformed through the equation

\[
u_\alpha(x,t) = c + u_1(x_1)
\]

By substitution of Equation (3.59) into Equation (3.60) the second derivative terms are eliminated. Consequently, an analytical solution is obtained for the surface elevation in the fixed or the moving frame of reference. It is given by

\[
\begin{align*}
\eta(x,t) &= \frac{a_3 c (Q_1 - D u_1(x_1)) + a_7 \left[ R_1 - \frac{1}{2} u_1^2(x_1) \right]}{a_3 c u_1(x_1) + a_7}
\end{align*}
\] (3.76)

In Equations (3.75) and (3.76) it must be remembered that \(x_1 = x - ct\).

An example of the newly developed method is given in Figure 3.4. Moreover, an identical example computed by the method of Rienecker & Fenton (1981) is shown. A depth of \(D = 1.00\), a wave height of \(H = 0.500\), an absolute wave period of \(T = 9.89\), and an Eulerian current below wave trough level, \(c_L = 0.00\), is chosen. The computation is carried out using 10 Fourier components, i.e. \(M = 10\). By employment of the method by Rienecker & Fenton it can be seen that the chosen data correspond to \(\frac{H}{L} = 0.0990\) and \(\frac{H}{L} = 0.0495\), i.e. a wave in intermediate depth water of moderate steepness is considered. The figure shows the surface profile, \(\eta_1\), as
Figure 3.4: Surface profiles computed by the newly developed method (dashed), and by the method of Rienecker & Fenton (1981) (solid). Data: $D = 1.00$, $H = 0.500$, $T = 9.89$, $c_E = 0.00$, $M = 10$.

It appears that the profiles are quite similar. Additionally, the computations show that the numerically exact solution of Equations (3.50) and (3.51) gives rise to a wave celerity which is 98.0% of the wave celerity computed by the method of Rienecker & Fenton (1981). The difference is explained by the fact that Equations (3.50) and (3.51) only include the lowest order of nonlinearity, whereas the method of Rienecker & Fenton is based on the exact governing equations and boundary conditions.

### 3.9 Description of Program Code

Based on the numerical solution procedure outlined above a program was written in Borland Pascal 7.0. The code is enclosed in Appendix B.

It appears from the code that the main program, denoted Abm.2D, utilizes a number of units each of which contains various closely related functions and procedures. The units are described briefly below:

- **Boundary.2D** is responsible for the computation of boundary and initial conditions.
- **Breaking.2D** incorporates the simplified effect of wave breaking into the formulation, see Chapter 6.
3.9. Description of Program Code

- Fourier.2D solves the nonlinear system of equations outlined in Section 3.8.

- Mathfunc.2D contains information on mathematical functions not provided by Borland Pascal 7.0, i.e. Jacobian elliptic functions, the Double-Sweep method etc.

- Solution.2D solves the boundary value problem at each time level.

- Variable.2D contains all global constants, types and variables. It should be noted that all floating point computations are performed on the type, extended.

- Various.2D contains mainly functions and procedures used to allocate and deallocate dynamic memory. Additionally, information on the computed output as well as the required input can be found in this unit.

Because of the extensive use of functions and procedures the code is well suited for changes and further improvements.

Since the computational model developed in the present chapter constitutes the foundation of the developments performed in Chapter 6, a thorough verification is given in the next chapter. The analysis illustrates that the model produces sound physical results.
Chapter 4

Properties of the Computational Model

4.1 General

In this chapter the performance of the fundamental computational model is analyzed.

The first part considers the stability properties of the predictor scheme and the corrector scheme. A linearized stability analysis (of Von Neumann type) is carried out by employment of the linearized constant depth equations derived by Nwogu (1993). Using the results of the stability analysis, the phase portraits of the numerical schemes are quantified and compared with analytical results. Based on a typical set of wave data and various grid resolutions in time and space the celerity of the numerical solution is compared with numerically exact values.

Basic properties such as the total mass and energy in the computational domain are studied demonstrating that the model conserves both quantities satisfactorily.

Since the absorption of outgoing waves is based mainly on the employment of sponge layers, the linear reflection from the sponge layer is computed as a function of the wave period and the relative sponge layer width.

In the last part, two computational examples are given which relate to irrotational wave motion in water of variable depth. The computational model is used to study the wave height variation of a regular wave field propagating onto a plane beach of a gentle slope. The computed wave height variation is compared with existing theory as well as a set of laboratory measurements. Finally, the decomposition of a solitary wave as it propagates onto a shelf is studied.
4.2 Stability of the Computations

4.2.1 Amplitude Portraits of the Numerical Schemes

By employment of the linearized constant depth equations of Nwogu (1993) a Von Neumann stability analysis is carried out of the numerical scheme described in Chapter 3. Since higher order and nonlinear terms are not included in the formulation, the analysis is confined to waves of small height relative to the still water depth. Additionally, it is mentioned that the analysis is valid at internal nodes only; no stability analysis is performed in the special nodes located adjacent to the boundary nodes. Consequently, it is emphasized that the analysis provides an indication of the stability properties of the computations rather than a definite limitation.

If the amplitude dispersive terms of the governing equations, Equations (3.4) and (3.5), are not retained the dimensionless equivalents of Equations (2.13) and (2.14) are obtained. These read (Nwogu, 1993)

\[ u_{\alpha t} + \eta x + \alpha u_{\alpha xx} = 0 \]  
\[ \eta t + u_{\alpha x} + (\alpha + \frac{1}{3}) u_{\alpha xx} = 0 \]

in which \( \alpha \) is given by Equation (2.15). The equations given above are valid provided the bottom is horizontal.

A local (linearized) stability analysis of the predictor method is performed by considering a Fourier component of the solution representing Equations (4.1) and (4.2). The Fourier components are written

\[ \eta^{i,n} = \xi^n \eta_0 e^{jik\Delta x} \]  
\[ u^{i,n}_\alpha = \xi^n u_0 e^{jik\Delta x} \]

where \( j \) is the imaginary unit, \( \xi \) is the amplification factor, \( (\eta_0, u_0) \) is an eigenvector of the problem and \( k \) denotes the wave number of a Fourier component. By employment of Equations (4.1) and (4.2) in conjunction with the numerical space derivatives described in Section 3.4, substitution of Equations (4.3) and (4.4) into the third order Adams-Bashforth predictor method results in two equations. By collecting terms of the same kind these can be written in matrix form as

\[ \begin{bmatrix} \xi^2 (\xi - 1) & \beta_1 (23 \xi^2 - 16 \xi + 5) \\ \beta_2 (23 \xi^2 - 16 \xi + 5) & \xi^2 (\xi - 1) \end{bmatrix} \begin{bmatrix} \eta_0 \\ u_0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \]  
\[ \beta_1 = \frac{1}{3} \]  
\[ \beta_2 = \frac{1}{2} \]
4.2. Stability of the Computations

in which $\beta_1$ and $\beta_2$ are given by

$$\beta_1 = -\frac{j \Delta t}{12} (\alpha_1 \sin(2k \Delta x) + \alpha_2 \sin(k \Delta x)) \quad (4.6)$$

$$\beta_2 = \frac{j \Delta t}{72 \Delta x} \left( \frac{8 \sin(k \Delta x) - \sin(2k \Delta x)}{1 + \alpha_3 \cos(k \Delta x) - 1} \right) \quad (4.7)$$

and

$$\alpha_1 = \frac{1}{6 \Delta x} - (\alpha + \frac{1}{3}) \frac{1}{\Delta x^3} \quad (4.8)$$

$$\alpha_2 = -\frac{4}{3 \Delta x} + 2(\alpha + \frac{1}{3}) \frac{1}{\Delta x^3} \quad (4.9)$$

$$\alpha_3 = 2 \alpha - \frac{1}{\Delta x^2} \quad (4.10)$$

In order to obtain a non-trivial solution of Equation (4.5) the determinant of the matrix appearing on the left hand side must vanish. This results in a polynomial of the sixth degree in $\xi$, where $\xi$ is assumed to be a complex quantity. It reads

$$[\xi^2 (\xi - 1)]^2 - \beta_1 \beta_2 (23 \xi^2 - 16 \xi + 5)^2 = 0 \quad (4.11)$$

Due to the fact that Equation (4.11) is a polynomial of the sixth degree which involves imaginary quantities, it is solved numerically. By separating the real and the imaginary parts in the equation, two nonlinear coupled equations emerge which are solved on a computer by Gauss-Newton iteration. Since the solution depends on $k$, $\Delta x$, and the Courant number, $Cr$, defined by

$$Cr = \frac{\Delta t}{\Delta x} \quad (4.12)$$

it is not feasible to solve Equation (4.11) allowing all three quantities to vary. Consequently, a typical grid size of $\Delta x = 0.200$ is chosen. By use of three different Courant numbers ($Cr = \{0.500, 1.00, 2.00\}$) the largest modulus of the amplification factor of the predictor scheme is depicted in Figure 4.1 as a function of the wave number multiplied by the grid size. It is noted that the upper bound of the abscissa corresponds to the smallest wave length, $L = 2 \Delta x$, which can be represented on the computational mesh. It can be ascertained (Abbott, 1979) that larger wave numbers simply cause the signal to be aliased into smaller wave
numbers resolvable on the mesh. Figure 4.1 shows that the predictor scheme is unconditionally stable provided the Courant number is smaller than or equal to unity. Courant numbers slightly greater than unity were not tested. Furthermore, the numerically largest amplification factor is almost indistinguishable from unity if the Courant number is equal to 0.500. This indicates that the amplitude of the numerical solution remains (approximately) constant during a time step. Similarly, Figure 4.1 shows that the predictor scheme reduces the numerical solution by up to 2.71% if the Courant number is equal to unity. In the case of a Courant number, \( Cr = 2.00 \), the predictor scheme is conditionally stable, since the intermediate wave numbers give rise to an amplification of the numerical solution by up to approximately 76.6% from one time level to the next. As a consequence of this the numerical solution would have limited practical use if not corrected.

The stability of the fourth order Adams-Moulton corrector method is investigated by following the procedure outlined above. This results in an equation system similar to Equation (4.5). It reads

\[
\begin{bmatrix}
\xi^2 (\xi - 1) & \beta_3 (9\xi^3 + 19\xi^2 - 5\xi + 1) \\
\beta_4 (9\xi^3 + 19\xi^2 - 5\xi + 1) & \xi^2 (\xi - 1)
\end{bmatrix}
\begin{bmatrix}
\eta_0 \\
u_0
\end{bmatrix}
= \begin{bmatrix} 0 \\
0 \end{bmatrix}
\] (4.13)

The coefficients, \( \beta_3 \) and \( \beta_4 \), are given by
4.2. Stability of the Computations

\[
\beta_3 = -\frac{j \Delta t}{24} (\alpha_1 \sin (2k \Delta x) + \alpha_2 \sin (k \Delta x)) \tag{4.14}
\]

\[
\beta_4 = \frac{j \Delta t}{144 \Delta x} \left( \frac{8 \sin (k \Delta x) - \sin (2k \Delta x)}{1 + \alpha_3 (\cos (k \Delta x) - 1)} \right) \tag{4.15}
\]

while \(\alpha_1, \alpha_2,\) and \(\alpha_3\) remain the same. The stability properties of the corrector scheme are determined by the magnitude of the roots in the polynomial

\[
[\xi^2 (\xi - 1)]^2 - \beta_3 \beta_4 (9 \xi^3 + 19 \xi^2 - 5 \xi + 1)^2 = 0 \tag{4.16}
\]

Using a grid size, \(\Delta x = 0.200,\) and three different Courant numbers, \(C r = \{0.500, 1.00, 2.00\},\) the numerically largest amplification factor of the corrector scheme is depicted in Figure 4.2 as a function of \(k \Delta x.\) The graph illustrates that the corrector scheme is unconditionally unstable, since the largest amplification factor is greater than unity regardless of the values of \(k \Delta x\) and \(Cr.\) This indicates that the consequence of performing an infinite number of corrections is instability. It is noted that the largest Courant number tested \((Cr = 2.00)\) gives rise to an amplification of the intermediate wave number components by up to 2.59% from one time level to the next. Similarly, the graph corresponding to \(Cr = 1.00\) shows that the most pronounced amplification is less than 404 parts per million. In the case of a Courant number, \(Cr = 0.500,\) the numerical solution is amplified by less than 6.43 parts per million, hence indicating that approximately 1546 corrections can be made before the numerical solution is enhanced by 1%.

![Figure 4.2](image)

Figure 4.2: For various Courant numbers the amplification factor of the corrector scheme is shown as a function of the wave number multiplied by the grid size. Data: \(\Delta x = 0.200,\) \(Cr = \{0.500, 1.00, 2.00\}.\)
In the present work updating of the fluid domain from one time level to the next is accomplished by use of a single prediction followed by a number of corrections. Assuming that the Courant number is smaller than unity, the predictor and corrector schemes, respectively, tend to attenuate and amplify the numerical solution, hence resulting in an overall amplification factor very close to unity.

For a wide range of input parameters test computations have shown that satisfactory results are obtained at Courant numbers close to unity if the numerical procedure is allowed to converge at each time step. In this context convergence is referred to as the number of corrections required to obtain a predetermined (finite) accuracy given by the criterion in Equation (3.24). As long as the wave field is relatively linear, i.e. for waves of small height relative to the still water depth, two corrections are normally enough to achieve convergence to three decimal places using a Courant number close to unity (and a given resolution in space, $\Delta x = 0.200$, say). On the other hand, as the wave field becomes increasingly nonlinear, e.g. if limiting height waves are considered, a satisfactory numerical solution requires a smaller Courant number. In the case of a spatial discretization of $\Delta x = 0.200$ tests show that a Courant number of approximately $Cr = 0.500$ is appropriate when modelling waves of finite height.

### 4.2.2 Phase Portraits of the Numerical Schemes

In continuation of the amplitude portraits investigated in the previous section, the phase portraits of the predictor and corrector schemes are analyzed in the following. It is noted that the analysis was suggested by Fenton (1996). This contribution is gratefully acknowledged.

By analogy with Equations (4.3) and (4.4) it is assumed that the exact analytical solution of the linearized equations, Equations (4.1) and (4.2), can be written

$$\eta = \eta_0 e^{j k(x - ct)} \quad (4.17)$$

$$u_\alpha = u_\alpha e^{j k(x - ct)} \quad (4.18)$$

where $c$ is the wave celerity. By substituting these equations into Equations (4.1) and (4.2), respectively, the wave celerity derived by Nwogu (1993) is obtained. It reads

$$c^2 = \frac{1 - (\alpha + \frac{1}{3})k^2}{1 - \alpha k^2} \quad (4.19)$$

Analogously, the exact wave celerity obtained from Stokes’ linear wave theory is given by

$$c^2 = \frac{\tanh k}{k} \quad (4.20)$$

The phase portraits of the numerical schemes are determined, quite simply, by the requirement that Equations (4.3) and (4.4) be equal to Equations (4.17) and (4.18), respectively. Owing to the fact that $x = i \Delta x$ and $t = n \Delta t$, this implies

$$\xi = e^{-j k c \Delta t} \quad (4.21)$$
4.2. Stability of the Computations

thus indicating that the wave celerity is bound to have complex values if $|\xi| \neq 1$. Since the real part of the wave celerity is relevant for practical purposes, the phase portraits associated with the numerical schemes are given by

$$c = \frac{\Re(j \ln \xi)}{Cr k \Delta x} \quad (4.22)$$

where $\ln \xi = \ln|\xi| + j \theta$ and $\theta$ is the argument of $\xi$. By use of the amplification factors computed in the previous section, Equation (4.22) is used to determine the phase portraits of the predictor scheme and the corrector scheme.

In Figures 4.3 and 4.4, respectively, the wave celerity associated with the predictor scheme and the corrector scheme are shown as a function of $k \Delta x$, where $\Delta x = 0.200$. The Courant numbers considered are the same as those used in the previous section. Additionally, the graphs depict the wave celerity of the linearized Boussinesq equations, Equation (4.19), and the celerity of Stokes' linear wave theory, Equation (4.20).

From Figure 4.3 it appears that the wave celerity computed by the predictor scheme using a Courant number smaller than or equal to unity agrees well with Stokes' linear wave celerity provided $k \Delta x$ is smaller than approximately 1.50. It is remarkable that the celerity of the predictor scheme generally follows the celerity of Stokes' linear wave theory more closely than does the celerity of the linearized Boussinesq equations. Additionally, the graph shows that the numerically largest amplification factor of the predictor scheme is not necessarily that with the most appropriate speed. This can be seen from the graph corresponding to $Cr = 2.00$.

Figure 4.3: Wave celerity as a function of $k \Delta x$. The graph shows Equations (4.19) and (4.20) as well as the wave celerity associated with the predictor scheme, Equation (4.22). Data: $\Delta x = 0.200$, $Cr = \{0.500, 1.00, 2.00\}$. 
1.20
1.00
0.80
0.60
0.40
0.20
0.00
0.00
0.50
1.00
1.50
2.00
2.50
3.00
3.50

Figure 4.4: Wave celerity as a function of $k \Delta x$. The graph shows Equations (4.19) and (4.20) as well as the wave celerity associated with the corrector scheme, Equation (4.22). Data: $\Delta x = 0.200$, $Cr = \{0.500, 1.00, 2.00\}$.

The phase properties of the corrector scheme, which are shown in Figure 4.4, are very similar to those of the predictor scheme. On the basis of the Courant numbers considered in the present section the main difference appears to be that the largest amplification factor of the corrector scheme is that with the most appropriate speed.

4.2.3 Phase Error of the Numerical Solution

In Section 4.2.2 a fundamental investigation was made of the wave celerity computed by each of the numerical schemes. The present section considers the phase properties of the combined numerical schemes. Using a typical set of wave data the phase error of the numerical solution is computed as a function of the grid resolution in time and space.

A wave of height $H = 0.100$ and an absolute wave period $T = 9.90$, propagating at a constant depth of $D = 1.00$, is considered. The wave is generated internally by the numerically exact Fourier method outlined in Section 3.8 using 7 Fourier components and an Eulerian current of $c_E = 0.00$. The spatial extent covers the range, $x \in [0, 200]$. Outgoing waves are absorbed by employment of 50 sponge layer nodes in each end of the computational domain. For given values of $Cr$ and $\Delta x$ the absolute celerity of the numerical solution is computed using the wave field depicted at time $t = 198$. Since the absolute celerity is proportional to the wave length, it is computed by determining the distance between consecutive wave crests. The results shown in Figure 4.5 are based on averaging at least 10 wave lengths, hence minimizing inaccuracies caused by the finite spatial resolution.
4.3 Conservation of Volume and Energy

The accuracy of the computational model depends on how well the mass and the energy are conserved in the computational domain as the computations proceed in time. In this section computations are performed showing that the model conserves both quantities well. Since
the fluid is assumed to be incompressible, it is evident that the conservation of volume also implies conservation of mass.

The total dimensionless volume of water in the computational domain, $V(t)$, can be calculated by integrating the instantaneous water depth over the entire computational domain

$$V(t) = \int_{\Gamma_L}^{\Gamma_R} [D(x) + \eta(x, t)] \, dx$$

(4.23)

where $\Gamma_L$ and $\Gamma_R$ refer to the left and the right boundary, respectively.

Analogously, by using the still water level as a datum the dimensionless potential energy in the computational domain, $E_{pot}(t)$, can be found from the equation

$$E_{pot}(t) = \int_{\Gamma_L}^{\Gamma_R} \int_0^\eta z \, dz \, dx$$

(4.24)

while the total kinetic energy, $E_{kin}(t)$, is given by

$$E_{kin}(t) = \int_{\Gamma_L}^{\Gamma_R} \int_{-D}^\eta \frac{1}{2} (u^2 + w^2) \, dz \, dx$$

(4.25)

where $u = u(x, z, t)$ and $w = w(x, z, t)$ are the horizontal and the vertical velocity components, respectively. Since these are available as a function of $u_\alpha$ and its spatial and temporal derivatives of various orders, an assumption must be made as to the order of approximation. The largest terms contributing to Equations (4.24) and (4.25) are $O(\delta^2, \delta^2, \epsilon^4)$ and it is therefore impossible to estimate these on a basis consistent with the terms maintained in the governing equations. By including the aforementioned terms the potential and the kinetic energy are written

$$E_{pot}(t) = \int_{\Gamma_L}^{\Gamma_R} \frac{1}{2} \eta^2 \, dx$$

(4.26)

$$E_{kin}(t) = \int_{\Gamma_L}^{\Gamma_R} \frac{1}{2} u_\alpha^2 D \, dx$$

(4.27)

Both terms are $O(\delta^2)$.

The total energy in the system is readily given by

$$E_{tot}(t) = E_{pot}(t) + E_{kin}(t)$$

(4.28)
4.3. Conservation of Volume and Energy

![Figure 4.6: Dimensionless total energy and volume within the computational domain as a function of time. Data: $D = 1.00$, $H = 0.300$, $T = 9.90$, $c_E = 0.00$, $M = 10$, $x \in [0, 100]$, $\Delta x = 0.250$, $Cr = 0.500$, $ISL = ISR = 50$, $f \in [0, 100]$.]

For completeness it is noted that the physical volume and energy, $V^*(t^*)$ and $E_{tot}^*(t^*)$, respectively, are related to their dimensionless equivalents through the relations

\[
\begin{align*}
V^*(t^*) &= (D_0^*)^2 V(t) \\
E_{tot}^*(t^*) &= \rho^* g^* (D_0^*)^3 E_{tot}(t)
\end{align*}
\]

(4.29)

where $\rho^*$ is the density of water.

In the following the ability of the model to conserve volume and energy is illustrated by a computational example. A wave of height, $H = 0.300$, and an absolute wave period, $T = 9.90$, propagating at a constant depth of $D = 1.00$ is considered. The number of discretization points is $I = 400$, while $\Delta x = 0.250$ and $Cr = 0.500$. In each end of the computational domain 50 nodes are used to absorb outgoing waves. The wave field is generated internally by the Fourier method described in Section 3.8 using $c_E = 0.00$ and $M = 10$.

In Figure 4.6 the total energy and the relative change in the volume are depicted during 100 wave cycles. In the early stages of the computation both the energy and the volume increase in accordance with the fact that the front of the wave field has not yet reached the downstream sponge layer which absorbs volume as well as momentum. A few wave periods after this has happened a stationary balance is found between the small net current generated by the Fourier method and the volume of water being expelled by the sponge layers. As can
be seen from the graph the total volume of water oscillates about a non-zero mean value. This is due to the fact that the employment of sponge layers gives rise to a decrease in the momentum flux, thus causing a setup to build up in part of the computational domain.

Additionally, it may be noted that the total energy in the computational domain during the stationary part of the computation is of the same order as the value obtained from Stokes’ linear theory, \( E_{tot} = 1.13 \). A close agreement could not be expected since the sponge layers cover 25% of the computational domain, hence implying that a better estimate based on Stokes’ theory would be approximately 75% of the value given above, i.e. \( E_{tot} = 0.844 \). This value compares reasonably with that of Figure 4.6.

### 4.4 Sponge Layer Performance

From an engineering viewpoint it is of interest to ensure that the sponge layer method outlined in Section 3.5 does not give rise to excessive amounts of wave reflection, since this may cause unintended interactions of incident and reflected waves. In the worst case the combination of an inappropriate use of sponge layers and insufficient conditions at the model boundary may lead to unstable computations. In this section a physical example is considered and a linear reflection coefficient is computed as a function of the relative width of the sponge layer.

The linear reflection coefficient from the sponge layer, \( R \), can be derived from Stokes’ linear theory (see e.g. Svendsen & Jonsson, 1980)

\[
R = \frac{H_{max} - H_{min}}{H_{max} + H_{min}}
\]  

(4.30)

where \( H_{min} \) and \( H_{max} \) are the minimum and the maximum wave height, respectively. In the computations these are found from the wave height envelope of the numerical solution. Once the computations have become stationary the wave height envelope is determined by depicting the wave field at every time level during a single wave period as shown schematically in Figure 4.7 for a typical set of wave data.

By employment of the procedure described above linear reflection coefficients are computed as a function of the relative sponge layer width and the absolute wave period. The relative width of the sponge layer, \( S \), is defined as \( S = S^*/(T^*(g*D_0)^{1/2}) \) where \( S^* \) is the dimensional equivalent. The sponge layer is used to absorb incident waves, and it is located next to the right hand boundary. Regular waves of height \( H = 0.100 \), propagating at a depth of \( D = 1.00 \) are generated internally by the Fourier approximation method using \( M = 10 \), \( N^x = 19.8 \), and \( Cr = 0.500 \). A sponge layer covering 50 computational nodes borders the left hand boundary, since this is known to absorb reflected waves sufficiently well. This allows both \( u_0 \) and \( \eta_z \) to be set equal to zero at the first computational point. At the last point, i.e. at the right hand boundary, a totally reflecting wall is simulated by the requirement that \( u_0 \) and \( \eta_z \) be equal to zero. The latter requirement enables the surface elevation to be updated correct to the second order in the spatial step using only the end point and two adjacent points.
4.4. Sponge Layer Performance

Figure 4.7: Wave field depicted every fourth time step during almost an entire absolute wave period. Data: $D = 1.00$, $H = 0.100$, $T = 9.90$, $e_E = 0.00$, $M = 10$, $x \in [0, 100]$, $\Delta x = 0.500$, $C_r = 0.500$, $ISL = 50$, $ISR = 10$.

Figure 4.8: Performance of sponge layer method. The graph shows the computed linear reflection coefficient as a function of the relative sponge layer width for various absolute wave periods. Data: $D = 1.00$, $H = 0.100$, $e_E = 0.00$, $M = 10$, $N^2 = 19.8$, $C_r = 0.500$, $ISL = 50$. 

\[ T = 4.95 \]
\[ T = 9.90 \]
\[ T = 14.8 \]
\[ T = 19.8 \]
From Figure 4.8 it is evident that a sponge layer width of approximately two shallow water wave lengths gives rise to an insignificant amount of linear reflection for all the wave periods tested. As the relative sponge layer width decreases the reflection becomes more pronounced. When no sponge layers are present linear reflection coefficients in excess of 0.930, but smaller than unity, are obtained. This is explained by the fact that the prescribed boundary conditions do not reflect the outgoing waves perfectly, thus implying that a minor part of the wave energy will be transmitted numerically through the right hand boundary. Additionally, it is recalled that Equation (4.30) does not account for the finite height of the incident wave field. Consequently, a reflection coefficient of unity could not be expected.

4.5 Regular Waves over a Horizontal Bottom

Since the computation of wave height envelopes, mean water levels and wave induced flows requires that long runs be performed, the computational model must be able to propagate waves correctly during long periods of time. The examples considered in the present section verify this. In addition it is demonstrated that employment of the Fourier method described in Section 3.8 to generate the incident wave field produces significantly better results in comparison with using first order cnoidal theory.

The first example considers the propagation of regular waves over a horizontal bottom in intermediate depth water. The wave height is $H = 0.100$, the absolute wave period is given by $T = 6.93$, and the still water depth is given by $D = 1.00$. By employment of the Fourier method this results in a wave steepness of $\frac{H}{L^*} = 1.67\%$, and a relative water depth of $\frac{D^*}{L^*} = 0.167$, hence permitting the use of Stokes’ theory (Fenton, 1985). The computational domain is discretized by 400 nodes using a spatial step of $\Delta x = 0.125$ and a Courant number of $Cr = 0.500$. Two sponge layers, each 50 nodes wide, are located adjacent to the boundaries of the computational domain. The incident wave field is generated inside the computational domain by employment of either Stokes’ second order theory or the Fourier method described in Section 3.8. In both cases $c_E = 0.00$, while 10 wave components are used for the Fourier computation. The wave profiles of the numerical solutions are depicted in Figure 4.9 at time $\frac{t}{T} = 100$. It appears that these are almost indistinguishable and, further, that the model propagates the specified wave field satisfactorily.

The second example considers a relatively long wave field described by a wave height of $H = 0.200$, and an absolute wave period given by $T = 21.0$. The still water depth is $D = 1.00$. Since the Fourier method yields $UR = 92.5$ and $\frac{P}{L^*} = 0.0465$, first order cnoidal theory may be used to generate the incident wave field. The surface profiles of two numerical solutions are shown in Figure 4.10 at time $\frac{t}{T} = 100$ using as input to the computational model either the numerically exact Fourier method or first order cnoidal theory. Again, the Fourier method is based on $c_E = 0.00$, and $M = 10$. The computational domain is covered by 400 nodes, and 50 nodes are used in each end of the fluid domain to absorb the scattered wave field. The spatial step is $\Delta x = 0.500$, while the Courant number is given by $Cr = 0.500$. It appears from the graph that the Fourier method produces excellent results, whereas the first order cnoidal theory fails to provide a reasonable input to the model. It is speculated that the dispersion characteristics of the cnoidal wave field do not match the governing equations, hence causing a dispersive tail to develop on the trailing part of each wave.

From a computational viewpoint the graph verifies that the model can be run successfully for long periods of time.

From Figure 4.8 it is evident that a sponge layer width of approximately two shallow water wave lengths gives rise to an insignificant amount of linear reflection for all the wave periods tested. As the relative sponge layer width decreases the reflection becomes more pronounced. When no sponge layers are present linear reflection coefficients in excess of 0.930, but smaller than unity, are obtained. This is explained by the fact that the prescribed boundary conditions do not reflect the outgoing waves perfectly, thus implying that a minor part of the wave energy will be transmitted numerically through the right hand boundary. Additionally, it is recalled that Equation (4.30) does not account for the finite height of the incident wave field. Consequently, a reflection coefficient of unity could not be expected.

4.5 Regular Waves over a Horizontal Bottom

Since the computation of wave height envelopes, mean water levels and wave induced flows requires that long runs be performed, the computational model must be able to propagate waves correctly during long periods of time. The examples considered in the present section verify this. In addition it is demonstrated that employment of the Fourier method described in Section 3.8 to generate the incident wave field produces significantly better results in comparison with using first order cnoidal theory.

The first example considers the propagation of regular waves over a horizontal bottom in intermediate depth water. The wave height is $H = 0.100$, the absolute wave period is given by $T = 6.93$, and the still water depth is given by $D = 1.00$. By employment of the Fourier method this results in a wave steepness of $\frac{H}{L^*} = 1.67\%$, and a relative water depth of $\frac{D^*}{L^*} = 0.167$, hence permitting the use of Stokes’ theory (Fenton, 1985). The computational domain is discretized by 400 nodes using a spatial step of $\Delta x = 0.125$ and a Courant number of $Cr = 0.500$. Two sponge layers, each 50 nodes wide, are located adjacent to the boundaries of the computational domain. The incident wave field is generated inside the computational domain by employment of either Stokes’ second order theory or the Fourier method described in Section 3.8. In both cases $c_E = 0.00$, while 10 wave components are used for the Fourier computation. The wave profiles of the numerical solutions are depicted in Figure 4.9 at time $\frac{t}{T} = 100$. It appears that these are almost indistinguishable and, further, that the model propagates the specified wave field satisfactorily.

The second example considers a relatively long wave field described by a wave height of $H = 0.200$, and an absolute wave period given by $T = 21.0$. The still water depth is $D = 1.00$. Since the Fourier method yields $UR = 92.5$ and $\frac{P}{L^*} = 0.0465$, first order cnoidal theory may be used to generate the incident wave field. The surface profiles of two numerical solutions are shown in Figure 4.10 at time $\frac{t}{T} = 100$ using as input to the computational model either the numerically exact Fourier method or first order cnoidal theory. Again, the Fourier method is based on $c_E = 0.00$, and $M = 10$. The computational domain is covered by 400 nodes, and 50 nodes are used in each end of the fluid domain to absorb the scattered wave field. The spatial step is $\Delta x = 0.500$, while the Courant number is given by $Cr = 0.500$. It appears from the graph that the Fourier method produces excellent results, whereas the first order cnoidal theory fails to provide a reasonable input to the model. It is speculated that the dispersion characteristics of the cnoidal wave field do not match the governing equations, hence causing a dispersive tail to develop on the trailing part of each wave.

From a computational viewpoint the graph verifies that the model can be run successfully for long periods of time.
4.5. Regular Waves over a Horizontal Bottom

Figure 4.9: Wave propagation test. The wave fields are generated by either Stokes' second order theory or the Fourier method described in Section 3.8. These are depicted at time $t^* = 100$. Data: $D = 1.00, H = 0.100, T = 6.93, c_E = 0.00, M = 10, II = 400, \Delta x = 0.125, Cr = 0.500, ISL = ISR = 50$.

Figure 4.10: Wave propagation test. The wave fields are generated by either first order cnoidal theory or the Fourier method described in Section 3.8. These are depicted at time $t^* = 100$. Data: $D = 1.00, H = 0.200, T = 21.0, c_E = 0.00, M = 10, II = 400, \Delta x = 0.500, Cr = 0.500, ISL = ISR = 50$. 
4.6 Regular Waves Propagating onto a Gently Sloping Beach

The examples presented in the previous sections all relate to wave propagation in constant depth water. In this section computations are performed of regular waves of finite height climbing a gently sloping beach. It is verified that the model is able to reproduce accurately the effect of wave shoaling almost up to the breaking point.

Hansen & Svendsen (1979) presented detailed measurements of regular waves of initially constant form propagating into shoaling water. The experiments were made in a closed wave flume 0.600 m wide and 32.0 m long. All experiments were performed with a maximum depth of water of 0.360 m followed by a plane slope of 1:34.26 in the downstream part of the flume. The toe of the slope was located 14.8 m from the mean position of the wave paddle. Special care was taken to remove free second harmonics from the incident wave signal following a special technique described by Hansen et al. (1975). In the example given below an incident wave field described by a wave height, $H_0 = 0.0700$ m, and a wave period, $T^* = 1.00$ s, is considered.

Since a run-up condition has not been incorporated into the computational model, the bed configuration of Hansen & Svendsen (1979) cannot be employed without slight modifications. In the computations the sloping part of the beach ends at a depth of $D^* = 0.0348$ m corresponding to $x^* = 25.9$ m. Hence, the still water depth is given by

$$D(x) = \begin{cases} 0.360 \text{ m} & , \quad x^* \in [0, 14.8 \text{ m}] \\ 0.360 \text{ m} - \frac{x^* - 14.8 \text{ m}}{34.26} & , \quad x^* \in [14.8 \text{ m}, 25.9 \text{ m}] \\ 0.0348 \text{ m} & , \quad x^* \in [25.9 \text{ m}, 36.0 \text{ m}] \end{cases}$$

From Equation (4.31) it is noted that the computational domain extends an additional 4.00 m, the reason being that 50 nodes are used in each end of the computational domain to absorb outgoing waves. The incident wave field is generated internally by the Fourier method using 10 wave components and a mass transport velocity of $c^* = 0.00$ m/s (Section 3.8). This is in agreement with the fact that the mean volume flux through any vertical section must be equal to zero in a closed wave flume. In order to ensure that the computations have become stationary the model is run for 150 wave periods before any information is extracted. Additionally, it should be noted that the incident wave height generated in the experiment of Hansen & Svendsen (1979) did not quite reach the specified value of $H_0 = 0.0700$ m. By averaging the measured wave heights at 20 locations very close to the toe of the slope in the part of the wave flume of constant depth (nos. 138 - 157) an actual wave height of $H_0 = 0.0664$ m is determined. Using this value as input to the computations the Fourier method yields $\frac{H_2}{L_2} = 4.63\%$ and $\frac{D}{L_0} = 0.251$, i.e. a relatively steep wave in intermediate depth water is considered. A Courant number of $Cr = 0.500$ and a total of 450 nodes are used.

In Figure 4.11 wave heights normalized by the incident wave height are depicted as a function of $x^*/L_0^*$, where $L_0$ is the dimensionless deep water wave length calculated from linear theory (the dimensional equivalent, $L_0$, is given by $L_0^* = D_0^* L_0$). The toe of the slope is located at $D/L_0 = 0.231$. In addition to the wave height envelope extracted from the computational model during a single wave period in the latter stages of the computation measured mean wave heights of Hansen & Svendsen (1979) are depicted up to the point of wave breaking. Analytical results obtained by employment of Stokes’ linear theory as well as the numerically
4.7. The Solitary Wave Propagating onto a Shelf

Figure 4.11: Relative wave height variation caused by regular waves climbing a plane and gentle slope. The bottom bathymetry is given by Equation (4.31), and the toe of the slope is located at $D/L_0 = 0.231$. Data: $H_0^s = 0.0664$ m, $T^* = 1.00$ s, $c_s^* = 0.00$ m/s and $M = 10$ (Fourier method), $fG = 450$. $Cr = 0.500$, $ISL = ISR = 50$, $\frac{f}{T} \in [0, 150]$.

exact Fourier method of Rienecker & Fenton (1981) are also shown. Since these are derived by the assumption of a locally horizontal bottom, the methods can be applied successfully provided $|D_z|/D_0 \ll 1$. In the current example the Fourier method of Section 3.8 results in $|D_z|/D_0 = 0.116 \ll 1$ at the toe of the slope, hence permitting their use.

From Figure 4.11 it is evident that Stokes' linear wave theory fails to predict the wave height as the waves shoal. The fully nonlinear model of Rienecker & Fenton (1981) follows the measurements reasonably closely but there is a tendency to overestimate the wave height at each depth. It is believed the discrepancy is caused by the fact that the Fourier method does not incorporate the effect of the sloping bottom. In this particular case the present model seems to describe the wave height variation quite well for waves almost up to their breaking point. In particular the initial decrease in the wave height is modelled accurately. Since practically no wave energy is reflected from the downstream sponge layer, the oscillations appearing in the wave height envelope are caused by the reflections from the sloping beach (Madsen & Sørensen, 1992).

4.7 The Solitary Wave Propagating onto a Shelf

It was mentioned in Chapter 2 that Mei & Le Méhauté (1966) were the first to present equations of the Boussinesq-type applicable to waves in water of variable depth. Since the
equations include the lowest order of nonlinearity and frequency dispersion, these are consistent with the equations used in the present report.

Madsen & Mei (1969) solved numerically the equations of Mei & Le Méhauté (1966) by employment of the method of characteristics. They studied the transformation and the subsequent decomposition of an initially solitary wave as it climbs a shelf of a slope of 1:20. In this section an almost identical example is considered. It is shown that the computational model in hand produces results which compare favourably with the results of Madsen & Mei.

A solitary wave described by the approximate analytical solution of Wei & Kirby (1995) is used as input to the computations (Section 3.7.1). In order to perform a comparison with the results of Figure 5 of Madsen & Mei an initial wave height of \( H_0 = 0.120 \) is chosen. The bathymetry of the bottom is defined by

\[
D(x) = \begin{cases} 
1.00 & , x \in [-37.5, 6.0] \\
1.00 - \frac{x-6.00}{20.0} & , x \in [6.00, 16.0] \\
0.500 & , x \in [16.0, 87.5]
\end{cases}
\]

(4.32)

and the computational domain is covered by 500 points. In each end of the fluid domain 50 nodes are used to absorb outgoing waves. The model is run at a Courant number of \( Cr = 0.500 \), and \( t \in [0, 99.0] \).

Figure 4.12: Decomposition of an initially solitary wave. The bathymetry is given by Equation (4.32), and shown vertically distorted in the figure. Data: \( H_0 = 0.120 \), \( II = 500 \), \( Cr = 0.500 \), \( ISL = ISR = 50 \), \( t \in [0, 99.0] \).
4.7. The Solitary Wave Propagating onto a Shelf

Figure 4.13: Decomposition of an initially solitary wave. The surface elevations depicted in the graph were digitized using Figure 5 of Madsen & Mei (1969). The bathymetry is given by Equation (4.32), and shown vertically distorted in the figure. Data: $H_0 = 0.120$.

In Figure 4.12 wave profiles computed by the present model are depicted at various time levels. Additionally, the envelope of the maximum crest is shown as a dashed line. In particular it is noted that the approximate solution of Wei & Kirby (1995) practically eliminates all spurious oscillations at the trailing part of the wave profile. The graph exhibits that a part of the wave energy is reflected as the solitary wave climbs the shelf. The reflected wave has the appearance of a small hump propagating back towards the left hand boundary. As the solitary wave, distorted by climbing the slope, enters the shelf the surface profile gradually becomes more peaked, and the wave splits up into several minor humps trailing behind the original wave.

In order to facilitate the comparison with Madsen & Mei (1969) the surface elevations depicted in Figure 5 of Madsen & Mei were digitized and shown in Figure 4.13. It appears from Figure 4.13 that the crest elevation stabilizes at an approximate value of $\eta_{\text{max}} = 0.195$ which is somewhat smaller than the value computed by the present model ($\eta_{\text{max}} = 0.210$). In general, the models produce very similar results, hence verifying both, since they are based on different equations as well as on different numerical solution procedures.
Chapter 5

Wave Propagation in Two Horizontal Dimensions

5.1 General

Evidently, the computational model described in Chapter 3 constitutes a relatively accurate design tool provided the assumption of a two-dimensional flow field is satisfied. In practice, this assumption is often invalidated by complex wave conditions in conjunction with a bottom bathymetry generated from real topographical maps. Additionally, man made structures may obstruct the flow, thus indicating that a three-dimensional model must be employed in simulating the flow field adequately.

The present chapter is devoted to the description of wave propagation in two horizontal directions. On the basis of the equations of Nwogu (1993) a numerical solution method, marginally different from that of Chapter 3, is given. Since the formulation of the problem is very general, waves can be propagated in virtually any geometry, hence indicating that the model can be used to study a number of practical problems. In comparison with existing solution methods based on the same type of equations this is a definite improvement.

The incident wave field is generated inside the computational domain using the technique of Ishii et al. (1994). The analytical manipulations of Section 3.6 become quite substantial in a formulation covering two horizontal dimensions, and the wave generation concept is therefore generalized and implemented in a simple and efficient way.

Scattered waves propagating out of the computational domain are absorbed in the vicinity of the model boundary using an existing sponge layer method. In addition the reflection of waves interacting with a vertical barrier is addressed.

Finally, examples are given illustrating the capabilities of the computational model.

5.2 Definitions and Governing Equations

An extension of the computational model of Chapter 3 to include the second horizontal dimension requires additional variables to be defined. In accordance with the axis definitions of
Figure 3.1 A right hand coordinate system is adopted. This implies that the second horizontal axis, denoted the \( y^* \)-axis, is positive into the paper as well as perpendicular to both the \( x^* \)-axis and the \( z^* \)-axis. In the \( y^* \)-direction the horizontal velocity component, termed \( u_{\alpha y} \), is measured at a level identical to that of \( u_{\alpha x} \). For clarity, it should be noted that the dependent variables are presumed to be functions of \( x^* \), \( y^* \), and \( t^* \).

The Boussinesq-type equations of Nwogu (1993) are given in dimensional form by Equations (2.11) and (2.12). These are nondimensionalized by use of Equation (3.3) in combination with the relation, \( y^* = y D_0^* \), where \( D_0^* \) is a characteristic depth in the fluid domain. Due to the fact that the boundary of the computational domain becomes a number of line segments in a description covering two horizontal dimensions, no obvious choice of \( D_0^* \) exists. As a consequence, the characteristic depth is chosen as the still water depth used to generate the incident wave field. The dimensionless equivalents of Equations (2.11) and (2.12), respectively, can be written (cf. Nwogu, 1993)

\[
\begin{align*}
\mathbf{u}_{\alpha t} + \mathbf{u}_{\alpha} \cdot \nabla \mathbf{u}_{\alpha} + \nabla \eta &= 0, \\
\eta_{tt} + \nabla \cdot [(D + \eta) \mathbf{u}_{\alpha}] &= 0
\end{align*}
\]

and

\[
\begin{align*}
\mathbf{u}_{\alpha t} &= \mathbf{F} + \mathbf{G}, \\
\mathbf{G} &= -D \left( \frac{C_1}{2} \frac{\partial}{\partial x} \nabla \cdot \mathbf{u}_{\alpha} + \nabla \left( \nabla \cdot (D \mathbf{u}_{\alpha}) \right) \right)
\end{align*}
\]

Since the compact form of the equations is inconvenient for computational purposes, these are expanded and rewritten in a form suitable for the numerical solution method described below. In the \( x \)-direction the momentum equation reads

\[
R_{1t} = F_1 + G_1
\]

where \( R_1 \) takes the form

\[
R_1 = u_{\alpha} + D \left[ A_1 D u_{\alpha xx} + A_2 (D u_{\alpha})_{xx} \right]
\]

By analogy with Chapter 3 the function, \( F_1 \), is given by the expression

\[
F_1 = -\eta_x - u_{\alpha} u_{\alpha x} - v_{\alpha} u_{\alpha y}
\]

while \( G_1 \) contains the so-called cross-derivative Boussinesq terms. It reads

\[
G_1 = -D \left[ A_1 D v_{\alpha xy} + A_2 (D v_{\alpha})_{xy} \right]
\]
5.2. Definitions and Governing Equations

In the second horizontal direction the momentum equation is given by a similar expression of the form

\[ R_{2t} = F_2 + G_{2t} \]  \hfill (5.7)

in which the quantity, \( R_2 \), can be written

\[ R_2 = v_\alpha + D \left( A_1 D v_{\alpha y} + A_2 (D v_\alpha)_{yy} \right) \]  \hfill (5.8)

The functions, \( F_2 \) and \( G_2 \), respectively, are given by the equations

\[ F_2 = - \eta_y - u_\alpha v_{\alpha x} - v_\alpha v_{\alpha y} \]  \hfill (5.9)

and

\[ G_2 = - D \left( A_1 D u_{\alpha xy} + A_2 (D u_\alpha)_{xy} \right) \]  \hfill (5.10)

while the constants, \( A_1 \) and \( A_2 \), appearing in the momentum equations are expressed in terms of \( C_1 \) (Figure 3.1) through the relations \( A_1 = \frac{C_1^2}{2} \) and \( A_2 = C_1 \).

By considering the equations given above, it is evident that \( R_1 \) and \( R_2 \) are the two-dimensional equivalents of the quantity, \( R \), cf. Equation (3.18). Each of the functions, \( F_1 \) and \( F_2 \), appearing on the right hand side of the momentum equations contains a gravitational term of the order \( O(1) \) as well as two convective terms of the order of magnitude \( O(\epsilon^2) \). The functions, \( G_1 \) and \( G_2 \), are frequency dispersive terms of \( O(\epsilon) \). In addition to the convective terms involving both \( u_\alpha \) and \( v_\alpha \), the functions, \( G_1 \) and \( G_2 \), are responsible for the transfer of momentum from one direction to the other.

In dimensionless form the continuity equation of Nwogu (1993) may be written

\[ \eta_t = - \left\{ D_x u_\alpha + D u_{\alpha x} + \eta_x u_\alpha + \eta u_{\alpha x} \right. \]
\[ + D_y v_\alpha + D v_{\alpha y} + \eta_y v_\alpha + \eta v_{\alpha y} \]
\[ + B_1 3 D^2 D_x \left( u_{\alpha xx} + v_{\alpha xy} \right) \]
\[ + B_1 D^3 \left( u_{\alpha xxx} + v_{\alpha xxy} \right) \]
\[ + B_2 2 D D_x \left[ (D u_\alpha)_{xx} + (D v_\alpha)_{xy} \right] \]
\[ + B_2 D^2 \left[ (D u_\alpha)_{xxx} + (D v_\alpha)_{xy} \right] \]
\[ + B_1 3 D^2 D_y \left( u_{\alpha xy} + v_{\alpha yy} \right) \]
\[ + B_1 D^3 \left( u_{\alpha xyy} + v_{\alpha yyy} \right) \]
\[ + B_2 2 D D_y \left[ (D u_\alpha)_{xy} + (D v_\alpha)_{yy} \right] \]
\[ + B_2 D^2 \left[ (D u_\alpha)_{xy} + (D v_\alpha)_{yyy} \right] \} \]  \hfill (5.11)
where the right hand side is denoted $F_3(x, y, t)$ and the constants, $B_1$ and $B_2$, are defined as $B_1 = \frac{C_1^2}{2} - \frac{1}{6}$ and $B_2 = C_1 + \frac{1}{2}$.

The first two lines on the right hand side of the continuity equation are readily recognized as the spatial change of the pseudo volume flux in the two horizontal directions. The terms involving the surface elevation are of the order $O(\delta)$ whereas the remaining four terms are of the order $O(1)$. By analogy with the continuity equation used in Chapter 3 the particular choice of horizontal velocity vector results in additional frequency dispersive terms. These are of the order $O(\epsilon^2)$. It should be kept in mind that the frequency dispersive terms contribute slightly to the volume flux in each horizontal direction.

In the next section a numerical solution method of the governing equations is outlined. The attention is drawn to the techniques used rather than the discretization of the spatial derivatives, since the methods of doing this are well established and therefore straightforward.

5.3 Numerical Solution Procedure in Two Horizontal Dimensions

5.3.1 Preliminary Considerations

Some of the efforts undertaken to model the evolution of long waves in variable as well as constant depth water are summarized in Chapter 2. From the summary it has become apparent that the computational models of Abbott et al. (1978) and Hauguel (1980) are the most comprehensive, since these allow waves to be propagated in (pseudo) arbitrary geometries. As a consequence, they have become the prototype for many later studies of the same kind.

Recently, Nwogu & Mansard (1994) solved the extended Boussinesq-type equations of Nwogu (1993) in two horizontal directions using an iterative Crank-Nicholson scheme. In this approach a rectangular computational domain was considered, thus restricting the application of the model to a relatively few practical cases. Similar efforts were presented by Wei & Kirby (1994) and Beji & Nadaoka (1994) on the basis of different Boussinesq-type equations.

In addition to the fact that the methods just mentioned operate on rectangular fluid domains only, the incident wave field is prescribed at the model boundary. Effectively, this means that scattered waves leaving the computational domain must be filtered out by employment of an appropriate absorbing condition. As mentioned in Section 3.5 waves of finite amplitude propagating at an angle to the grid can not be absorbed satisfactorily using the Sommerfeld radiation condition, hence indicating that higher order boundary conditions must be introduced. Although these can be derived in a systematic way (Engquist & Majda, 1977) the method introduces higher order derivatives in space which are bound to be discretized asymmetrically. Eventually, this may lead to unstable computations. A comprehensive discussion of the matter is given by Givoli (1991). Here, it will suffice to remark that internal wave generation, by far, is the most robust, since outgoing waves can be absorbed artificially by employment of sponge layers.

In order to improve the perception of the following material a number of definitions are made. For computational convenience a rectangular domain is considered throughout. The perimeter of the rectangular domain is denoted the main boundary while other boundaries located inside the rectangular domain or partly at the main boundary are termed internal
5.3. Numerical Solution Procedure in Two Horizontal Dimensions

Boundaries. Similarly, the computational domain denotes the entire computational rectangle including boundary points, whereas the fluid domain only refers to the fluid filled part of the computational domain. Points located inside the computational domain which are not fluid filled are denoted external points. It can be imagined that the combination of external nodes and internal boundaries makes it possible to consider complex fluid domains, e.g. harbours, straits etc.

Definitions relating to the computational domain are given in the following. Figure 5.1 shows that the \( x \)-direction is resolved by \( \Pi + 1 \) computational points uniformly spaced in the interval, \( x \in [0, X_{\text{max}}] \), while \( JJ + 1 \) computational points are distributed (uniformly) in the interval, \( y \in [0, y_{\text{max}}] \). In the \( x \)- and the \( y \)-direction, respectively, the node indices are termed \( i \) and \( j \), where \( i = 0, \ldots, \Pi \) and \( j = 0, \ldots, JJ \), hence indicating that \( x = i \Delta x \) and \( y = j \Delta y \) (\( \Delta y \) being the grid spacing in the \( y \)-direction). The quantities, \( x_{\text{max}} \) and \( y_{\text{max}} \), denote the spatial extent of the computational domain in the \( x \)- and the \( y \)-direction, respectively.

In Chapter 3 a distinction was made between internal nodes, special nodes, and nodes located at the boundary of the computational domain. At internal nodes the computation of the spatial derivatives appearing in the governing equations was unaffected by the presence of the computational boundary. In contrast, off-centred schemes as well as schemes of lower order accuracy were employed in the special nodes located adjacent to the model boundary, since information was available only on the internal side of the boundary. The same concept applies in a description covering two horizontal dimensions but the number of different kinds of discretization increases significantly. As a consequence, it proves useful to introduce a system designating the kind of computation performed in each node. In the present model a node type value is assigned to each node. These are denoted \( N_{i,j} \), where \( i = 0, \ldots, \Pi \) and \( j = 0, \ldots, JJ \).

It appears from Figure 5.1 that the computational domain consists of 30 different kinds of nodes. As the node type definitions merely enable a distinction between different kinds of
discretization, the values chosen are, in principle, irrelevant. However, for easy recognition it may be noted that the node type values shown in the figure resemble the numbering of a matrix. Their values are given by the set, $\Omega_{tot}$, where

$$\Omega_{tot} = \{0, 11, 12, 13, 14, 15, 21, 22, 23, 24, 25, 31, 32, 33, 34, 35, 
41, 42, 43, 44, 45, 51, 52, 53, 54, 55, 111, 115, 151, 155\}$$  (5.12)

In agreement with Figure 5.1 a node type value of $N_{i,j} = 0$ corresponds to an external point, while the internal nodes are contained in the subset, $\Omega_{int}$, given by

$$\Omega_{int} = \{22, 23, 24, 32, 33, 34, 42, 43, 44\}$$  (5.13)

Finally, it may be seen that the remaining points are located at the boundary, $\Gamma$, hence yielding the relationship

$$\Gamma = \Omega_{tot} \setminus \{0, \Omega_{int}\}$$  (5.14)

A closer study of the node type definitions of Figure 5.1 will reveal that the computation of the spatial derivatives is carried out by employment of information from up to five nodes in each direction. Further information on the discretization of the spatial derivatives of the governing equations can be found in Section 5.3.3.

The node type concept can be illustrated by considering the approximation of e.g. $\eta_x$ at a computational point with the corresponding node type, 23, say. By default, first order spatial derivatives are discretized correct to the fourth order to ensure the correct dispersion characteristics of the numerical solution. Under normal circumstances the approximation of $\eta_x$ would require information from nodes located outside the computational domain. Naturally, such a requirement can not be met, and the node type, 23, therefore designates that a scheme of lower order accuracy should be used.

By analogy with Chapter 3 it has particular relevance to impose either an absorbing condition or a purely reflecting condition at the model boundary. The condition stating whether an absorbing condition or a reflecting condition should be used can be programmed, quite simply, in terms of the node types, i.e.

$$N_{i,j} \in \Gamma \Rightarrow \text{Absorption}$$
$$-N_{i,j} \in \Gamma \Rightarrow \text{Reflection}$$  (5.15)

where $i = 0, \ldots, II$ and $j = 0, \ldots, JJ$.

As the node type system provides information on both the spatial discretization of the governing equations and the conditions imposed at the model boundary, it constitutes a cornerstone of the present solution method.
5.3.2 Temporal Updating of the Computational Domain

By the assumption that the dependent variables, $u_\alpha$, $v_\alpha$, and $\eta$ are given at the boundary of the computational domain at the subsequent time level as well as globally at the present and previous time levels, the computations can be advanced in time by employment of a modified version of the predictor-corrector method given in Chapter 3.

Temporary estimates of the quantities, $R_1$, $R_2$, and $\eta$ at the next time level are obtained by use of the third order Adams-Bashforth predictor method (Gear, 1971) in combination with a simple backward discretization (in time) of the terms, $G_{1t}$ and $G_{2t}$, as shown by Wei & Kirby (1994). The method reads

\begin{align}
R_{1i,j,n+1} &= R_{1i,j,n} + \frac{\Delta t}{12} \left( 23 F_{1i,j,n} - 16 F_{1i,j,n-1} + 5 F_{1i,j,n-2} \right) \\
&+ 2 G_{1i,j,n} - 3 G_{1i,j,n-1} + G_{1i,j,n-2} \\
R_{2i,j,n+1} &= R_{2i,j,n} + \frac{\Delta t}{12} \left( 23 F_{2i,j,n} - 16 F_{2i,j,n-1} + 5 F_{2i,j,n-2} \right) \\
&+ 2 G_{2i,j,n} - 3 G_{2i,j,n-1} + G_{2i,j,n-2} \\
\eta_{i,j,n+1} &= \eta_{i,j,n} + \frac{\Delta t}{12} \left( 23 F_{3i,j,n} - 16 F_{3i,j,n-1} + 5 F_{3i,j,n-2} \right)
\end{align}

For completeness it may be noted that the Equations (5.16) - (5.18) maintain third order accuracy in time.

As the prediction of the quantities, $R_1$ and $R_2$, allows an implicit determination of the velocity field, the functions on the right hand side of the continuity equation and the momentum equations can be estimated at the subsequent time level. Improved estimates of the dependent variables are then obtained by employment of the fourth order Adams-Moulton corrector method. Following Wei & Kirby (1994) it reads

\begin{align}
R_{1i,j,n+1} &= R_{1i,j,n} \\
&+ \frac{\Delta t}{24} \left( 9 F_{1i,j,n+1} + 19 F_{1i,j,n} - 5 F_{1i,j,n-1} + F_{1i,j,n-2} \right) \\
&+ G_{1i,j,n+1} - G_{1i,j,n} \\
R_{2i,j,n+1} &= R_{2i,j,n} \\
&+ \frac{\Delta t}{24} \left( 9 F_{2i,j,n+1} + 19 F_{2i,j,n} - 5 F_{2i,j,n-1} + F_{2i,j,n-2} \right) \\
&+ G_{2i,j,n+1} - G_{2i,j,n} \\
\eta_{i,j,n+1} &= \eta_{i,j,n} + \frac{\Delta t}{24} \left( 9 F_{3i,j,n+1} + 19 F_{3i,j,n} - 5 F_{3i,j,n-1} + F_{3i,j,n-2} \right)
\end{align}
By analogy with Chapter 3 it should be emphasized that the corrector stage is used iteratively until the relative change of the solution is smaller than a predetermined value. Specifically, the relative change may be written

$$
\Delta f = \frac{\sum_{i=0}^{II} \sum_{j=0}^{JJ} \left( |f_{new}^{i,j,n+1} - f_{old}^{i,j,n+1}| \right)}{\sum_{i=0}^{II} \sum_{j=0}^{JJ} |f_{new}^{i,j,n+1}|} < 0.001
$$

(5.22)

where \( f = \{u_\alpha, v_\alpha, \eta\} \). It may be of computational interest to note that the quantity, \( \Delta f \), can be computed without the reservation of computational storage for both \( f_{old}^{i,j,n+1} \) and \( f_{new}^{i,j,n+1} \), where \( i = 0, \ldots, II, j = 0, \ldots, JJ \), and \( f = \{u_\alpha, v_\alpha, \eta\} \).

By following the same procedure as described in Chapter 3 the surface elevation is advanced explicitly in time whereas \( u_\alpha \) and \( v_\alpha \) are given implicitly in terms of the quantities, \( R1 \) and \( R2 \). For a given \( j, j = 0, \ldots, JJ \), the horizontal velocity in the \( x \)-direction is obtained by solving a tridiagonal equation system, linear in the unknowns at the new time level. It reads

$$
\begin{align*}
R1^{0,j,n+1} &= u_\alpha^{0,j,n+1} \\
R1^{i,j,n+1} &= D1^i u_\alpha^{i-1,j,n+1} + D2^i u_\alpha^{i,j,n+1} + D3^i u_\alpha^{i+1,j,n+1} \\
R1^{II,j,n+1} &= u_\alpha^{II,j,n+1}
\end{align*}
$$

(5.23)

Similarly, for a given value of \( i, i = 0, \ldots, II \), the velocity component in the \( y \)-direction is found as the solution of the system

$$
\begin{align*}
R2^{i,0,n+1} &= v_\alpha^{i,0,n+1} \\
R2^{i,j,n+1} &= D1^j v_\alpha^{i,j-1,n+1} + D2^j v_\alpha^{i,j,n+1} + D3^j v_\alpha^{i,j+1,n+1} \\
R2^{i,JJ,n+1} &= v_\alpha^{i,JJ,n+1}
\end{align*}
$$

(5.24)

The equation systems given above are solved by employment of the Double-Sweep method outlined in Appendix A.

In solving for \( u_\alpha \) along a line in the \( x \)-direction, say, boundary nodes as well as external nodes may be encountered in the interior of the computational domain. The effect of these is incorporated into the formulation by presetting the relevant values of \( D1^i \), \( D2^i \), and \( D3^i \) to either zero or unity as well as storing the appropriate boundary conditions in the quantity, \( R1 \). Similar concepts apply in the \( y \)-direction.

In principle, the coefficients of the tridiagonal matrices of Equations (5.23) and (5.24) may be prefactored, inverted and stored for use at each time level. However, because of severe memory constraints these are evaluated at each time level in the present code, thus giving rise to a prolonged time of program execution.
5.3.3 Approximation of Spatial Derivatives

No particular attention is given to the description of the discretization of the spatial derivatives appearing in the governing equations, since the issue was described in detail in Chapter 3. However, it is noted that first order spatial derivatives are approximated correct to the fourth order, see Equation (3.25), while the remaining spatial derivatives are discretized correct to the second order, as outlined in Section 3.4.1. In the vicinity of the boundary of the fluid domain second order accuracy is maintained as far as possible without off-centering the numerical operator. In agreement with Chapter 3 an exception is made in discretizing the third derivatives terms, \( u_{\alpha xxx} \) and \( u_{\alpha yyy} \), appearing in the continuity equation. For these terms it has proven valid to off-center the numerical operator. Although this implies that the internal spacing between two parallel boundaries must cover at least five grid intervals it ought not impose any restrictions on practical simulations.

It is inevitable that boundaries located in the interior of the computational domain introduce sharp corners. Since corner points from a mathematical viewpoint represent a singularity, special precautions must be taken to minimize their effect on the flow field. Typically, the problem is partly overcome by a refinement of the computational mesh in regions located close to the singularities. For the mesh under consideration this becomes computationally expensive, since a given refinement must be employed globally. Although the problem could be reduced by using a non-uniform grid, it is believed that these are difficult to apply in combination with a fourth order accurate discretization of the first order spatial derivatives. Clearly, first order spatial derivatives must be approximated correct to the fourth order to avoid spurious dispersive effects arising from third derivative truncation error terms of the second order (Abbott et al., 1984).

The problem associated with an abrupt change of the boundary geometry can not be eliminated but it can be significantly reduced by upwinding of the convective terms at the corner points. In the present report a first order upwinding scheme is suggested as it tends to prevent nonphysical oscillations from appearing in the numerical solution. By considering the discretization of e.g. the convective term, \( u_{\alpha} u_{\alpha x} \), of Equation (5.5) the method reads (Press et al., 1992)

\[
\begin{align*}
    u_{\alpha i,j,n}^{i,j,n} &= \begin{cases} 
    u_{\alpha i,j,n}^{i,j,n} - \frac{u_{\alpha i,j,n}^{i,j,n} + u_{\alpha i+1,j,n}^{i+1,j,n}}{\Delta x} + O(\Delta x) & , u_{\alpha i,j,n}^{i,j,n} < 0 \\
    u_{\alpha i,j,n}^{i,j,n} - \frac{u_{\alpha i,j,n}^{i,j,n} + u_{\alpha i-1,j,n}^{i,j,n}}{\Delta x} + O(\Delta x) & , u_{\alpha i,j,n}^{i,j,n} \geq 0
    \end{cases}
\end{align*}
\]

(5.25)

Despite the fact that the method introduces a truncation error of the order \( O(\Delta x) \) it gives a better description of the underlying physics as compared with using a centred scheme of higher order accuracy.

5.3.4 Absorption of the Wave Field at Open Boundaries

The derivation of the spatial derivatives in conjunction with the time-stepping procedure described in Section 5.3.2 allows the surface elevation, \( \eta \), as well as the quantities, \( R1 \) and \( R2 \), to be advanced in time on the basis of the solution at the present and the previous time
levels. However, determination of the velocity field at the subsequent time level requires the tridiagonal equation systems, Equations (5.23) and (5.24), to be solved, thus indicating that proper conditions must be specified at the boundary.

As summarized in previous sections, absorption of outgoing waves is a very delicate matter. Evidently, it is possible to impose adequate conditions at the boundary in a few specialized cases (Givoli, 1991) but generally a satisfactory absorption can not be achieved without the use of sponge layers. In the present report the sponge layer method is employed as it seems to be the potentially most accurate method. By analogy with Chapter 3 the widths of the sponge layer in the x-direction and the y-direction, respectively, are denoted $x_s$ and $y_s$.

In a description covering two horizontal dimensions the computational cost associated with the use of sponge layers increases significantly. Consequently, care should be exercised in specifying the width of the sponge layer region. The analysis carried out in Section 4.4 is a good guide in choosing the width of the sponge layer. On the basis of the wave periods studied it is recommended to use a sponge layer width in excess of three quarters of a shallow water wave length in order to ensure that the amount of reflection is kept below approximately 5%.

An appropriate choice of the sponge layer width allows the dependent variables to be reset to zero at the outgoing boundary, i.e.

\[
\begin{align*}
\eta &= 0 \\
u_o &= 0 \\
v_o &= 0
\end{align*}
\] (5.26)

In the case of an absorbing boundary these conditions make it possible to solve for $u_o$ and $v_o$, see Equations (5.23) and (5.24), and hence the computations can be advanced in time.

In principle, it is possible to impose absorbing conditions at both the internal boundaries and the main boundary. For practical simulations it has little physical interest to absorb mass and momentum at internal boundaries, and it is therefore intended that the present absorption method be applied in connection with the main boundaries. The attenuation of waves inside a harbour could be accomplished by use of damping terms in the momentum equations in conjunction with a very limited number of sponge layer nodes adjacent to the internal boundaries. This would conserve the mass and dissipate the wave energy.

5.3.5 Total Reflection of the Scattered Wave Field

By analogy with Section 3.4.3 total reflection of the scattered wave field is achieved by the requirement that the volume flux normal to an impermeable barrier be equal to zero. Since any boundary must be piecewise parallel to either the x-axis or the y-axis, the problem is closely related to that of Section 3.4.3. In the case of a vertical wall parallel to the y-axis the assumption of a constant water depth leads to the approximate boundary conditions

\[
\begin{align*}
u_o &= 0 \\
\eta_x &= 0
\end{align*}
\] (5.27)
5.3. Numerical Solution Procedure in Two Horizontal Dimensions

as described in Section 3.4.3. Similarly, by considering a vertical wall parallel to the x-axis fronted by a region of constant water depth the boundary conditions are quantified approximately as

\[
\begin{align*}
\nu_\alpha &= 0 \\
\eta_y &= 0
\end{align*}
\]  

(5.28)

In principle, Equations (5.27) and (5.28) enable the computations to be stepped forward in time, since the tangential velocity at a given impermeable wall can be included in the tridiagonal equation systems. cf. Equations (5.23) and (5.24). However, in order to ensure a smooth numerical solution it has proven advantageous to impose a monotonicity constraint on the flow along bounding walls. In agreement with e.g. Hauguel (1980), Rygg (1988), and Wei & Kirby (1995) this is done by additionally imposing the conditions

\[
\begin{align*}
\nu_\alpha y &= 0 \\
\nu_\alpha x &= 0
\end{align*}
\]  

(5.29)

and

\[
\begin{align*}
\eta_y &= 0
\end{align*}
\]  

(5.30)

These correspond to a wall parallel to the x- and the y-direction, respectively. As the conditions essentially impose a no-shear condition along the bounding wall, they are not inconsistent with the inviscid fluid being considered. In effect, the application of the no-shear conditions enables the tangential velocity components to be quantified explicitly.

5.3.6 Internal Generation of the Incident Wave Field

In the preceding sections a method for the temporal updating of the computational domain was outlined but no attention was paid to the generation of the incident wave field. Since the internal generation method of Chapter 3 allows outgoing waves to be absorbed almost perfectly, the method is generalized and incorporated into the present solution procedure.

In a single horizontal dimension the incident wave field can be generated by manipulation of two nodes on each side of the wave generation point. This is not the case in a formulation covering two horizontal dimensions, since the point of wave generation becomes a line segment. In some practical cases it is sufficient to consider a single generation line parallel to either the x-axis or the y-axis, but generally it is useful to be able to impose the incident wave field along several line segments. In order to provide the necessary physical support for the incident wave field the end coordinates of a line segment must be located either at a reflecting boundary or at the end point of another wave generation line. In practice, it is therefore possible that the wave generation lines involve all the different node types defined in Section 5.3.1 (apart from the external nodes). As the node types are closely related to the discretization in space,
a large number of laborious algebraic manipulations must be performed. Although these can be carried out by hand for a given wave generation line (as done by Ishii et al., 1994) it is not feasible to do so in the general case. For that reason a somewhat different approach is adopted in the following.

Figure 5.2 shows an example of a wave generation line and the affected nodes. By choice, the reflected wave field is considered to the left of the wave generation line, while the total wave field is considered on the right hand side. It should be noted that the wave generation line depicted in the figure imposes no restrictions on the direction and the form of the incident wave field. By analogy with the node type definitions of Section 5.3.1 computational points influenced by the generation method must be registered. This can be done by definition of a quantity, $W_{i,j}$, $i = 0, \ldots, II$, $j = 0, \ldots, JJ$, designating whether or not a given node is affected by the generation method. In order to understand the following material it is important to note that a value of $W_{i,j} = \pm 1$ corresponds to an affected node located on the total side or the reflected side, respectively. Similarly, the default value, $W_{i,j} = 0$, designates a node not influenced by the wave generation method.

The key to the understanding of the generation method lies in the treatment of the variables used to compute the spatial derivatives of the incident wave field. As the present code makes extensive use of function calls for the computation of the numerical space derivatives, it is meaningful to express the modifications in terms of these. The argument is further enhanced by the fact that the type of discretization is decided at the stage at which the functions are evaluated. Since the modification of a numerical space derivative at a given node involves information from the opposite side of the wave generation line only, see Equations (3.38) - (3.40), the alterations can be carried out in terms of the functions normally used for their computation. At the expense of a number of two-dimensional arrays, each of the same dimension as the computational domain, the incident wave field is stored at the nodes influenced by the line of wave generation. It is important to point out that the storage of a
5.3. Numerical Solution Procedure in Two Horizontal Dimensions

particular quantity at a given time level, e.g. the incident surface elevation, is based on the employment of two different arrays representing the incident elevation on the reflected and the total side, respectively. As the nodes not influenced by the wave generation line are reset to zero, the alterations of the spatial derivatives can be expressed in terms of the functions normally used to compute the numerical space derivatives.

In contrast to the notation used elsewhere it has proven necessary to adopt a somewhat different notation in the remaining part of this section. As the modification of the governing equations is carried out only in terms of variables denoting the incident wave signal the index, \( I \), is dropped in order to improve the readability of the equations presented. Additionally, it should be emphasized that the equations are valid only if the incident wave signal is stored by the method described previously.

By analogy with the internal wave generation method of Section 3.6 it can be anticipated that the terms, \( F_1, G_1, F_2, \) and \( G_2 \) must be modified in a description covering two horizontal dimensions. In the \( x \)-direction the quantities

\[
WF_{1i,j} = \mp \{ \bar{\eta}_x^\pm u_{\alpha}^x u_{\alpha}^x + \bar{v}_{\alpha}^x u_{\alpha}^y \}_{i,j} \quad (5.31)
\]

and

\[
WG_{1i,j} = \mp \{ D [ A_1 D v_{\alpha}^{x}\bar{y} + A_2 ( D u_{\alpha} )^{\bar{y}}_{xy} ] \}_{i,j} \quad (5.32)
\]

are added to the right hand side of the momentum equation, see Equation (5.3), corresponding to \( W_i,j = \pm 1 \), where \( i = 0, \ldots, II \), and \( j = 0, \ldots, JJ \). The variables, \( \eta, u_{\alpha}, \) and \( v_{\alpha} \) denote the surface elevation and the horizontal velocity field of the incident wave, and the superscripts, + and −, refer to either the total or the reflected side of the wave generation line, respectively. The method is illustrated by considering the contribution of e.g. \( WF_{1} \) at a node where \( W_{i,j} = 1 \). In this case the term to be added to the right hand side of Equation (5.3) equals

\[
- \{ \bar{\eta}_x^+ u_{\alpha}^- u_{\alpha}^- + \bar{v}_{\alpha}^+ u_{\alpha}^- \}_{i,j}
\]

where a negative superscript refers to a variable of the incident wave field, stored on the reflected side of the wave generation line. Similarly, a positive superscript denotes a variable stored on the total side.

In the second horizontal direction the terms

\[
WF_{2i,j} = \mp \{ \bar{\eta}_y^\pm u_{\alpha}^x \bar{v}_{\alpha}^x + \bar{v}_{\alpha}^x u_{\alpha}^y \}_{i,j} \quad (5.33)
\]

and

\[
WG_{2i,j} = \mp \{ D [ A_1 D v_{\alpha}^{x}\bar{y} + A_2 ( D u_{\alpha} )^{\bar{y}}_{xy} ] \}_{i,j} \quad (5.34)
\]

are added to the right hand side of Equation (5.7). The terms, \( ( D v_{\alpha} )^{\bar{y}}_{xy} \) and \( ( D u_{\alpha} )^{\bar{y}}_{xy} \), appearing in Equations (5.32) and (5.34), respectively, are determined from Equations (5.37) and (5.38).

In agreement with the alterations of the momentum equations, the wave generation
method requires the continuity equation to be modified. It can be ascertained that the term given below must be added to the right hand side of the continuity equation. It reads

\[
WF3_{i,j} = \mp \{ Du_{\alpha x}^+ + \eta_{z}^\mp u_{\alpha x}^\pm + \eta_{x}^\mp u_{\alpha z}^+ + \eta_{y}^\mp v_{\alpha y}^+ + B_1 D^2 D_x (u_{\alpha xx}^+ + v_{\alpha xy}^-) + B_1 D^3 (u_{\alpha xzx}^+ + v_{\alpha xzy}^-) + B_2 D D_x [(D u_{\alpha})_{xx}^+ + (D v_{\alpha})_{xy}^-] + B_2 D^2 [(D u_{\alpha})_{xxx}^+ + (D v_{\alpha})_{xxxy}^-] + B_1 3 D^2 D_y (u_{\alpha xy}^- + v_{\alpha yy}^+) + B_1 D^3 (u_{\alpha xyy}^- + v_{\alpha yyyy}^+) + B_2 2 D D_y [(D u_{\alpha})_{xy}^- + (D v_{\alpha})_{yy}^+] + B_2 D^2 [(D u_{\alpha})_{xxy}^- + (D v_{\alpha})_{xyy}^+] \}
\]

where

\[
(D u_{\alpha})_{xx}^\pm = 2 D_{x} u_{\alpha x}^\pm + D u_{\alpha xx}^\pm \\
(D u_{\alpha})_{xy}^\pm = D_{x} u_{\alpha y}^\pm + D_{y} u_{\alpha x}^\pm + D u_{\alpha xy}^\pm \\
(D v_{\alpha})_{xy}^\pm = D_{x} v_{\alpha y}^\pm + D_{y} v_{\alpha x}^\pm + D v_{\alpha xy}^\pm \\
(D v_{\alpha})_{yy}^\pm = 2 D_{y} v_{\alpha y}^\pm + D v_{\alpha yy}^\pm \\
(D u_{\alpha})_{xxx}^\pm = 3 D_{x} u_{\alpha x}^\pm + 3 D_{x} u_{\alpha xx}^\pm + D u_{\alpha xxx}^\pm \\
(D v_{\alpha})_{xxxy}^\pm = D_{xx} v_{\alpha y}^\pm + 2 D_{xy} v_{\alpha x}^\pm + 2 D_{x} v_{\alpha xy}^\pm + D_{x} v_{\alpha xx}^\pm + D v_{\alpha xxy}^\pm \\
(D u_{\alpha})_{xxy}^\pm = D_{y} v_{\alpha x}^\pm + 2 D_{xy} u_{\alpha x}^\pm + 2 D_{y} u_{\alpha xy}^\pm + D_{x} u_{\alpha yy}^\pm + D u_{\alpha xyy}^\pm \\
(D v_{\alpha})_{xyy}^\pm = 3 D_{xy} v_{\alpha y}^\pm + 3 D_{y} v_{\alpha yy}^\pm + D v_{\alpha yyyy}^\pm
\]
5.3. Numerical Solution Procedure in Two Horizontal Dimensions

Equations (5.36) - (5.43) are not particularly interesting in themselves but they all exhibit the same noticeable pattern. Since the equations lack a term proportional to the velocity component considered, they can, at most, be interpreted as a kind of pseudo differential operators. A closer study will reveal that the omitted terms are already included in the numerical differentiation of the terms contained in the functions $G_1$, $G_2$, and $F_3$, see Equations (5.6), (5.10), and (5.11). A similar argument applies with regard to the terms omitted on the right hand side of Equation (5.35).

In addition to the alterations described above it follows from Section 3.6 that changes must be made in solving the tridiagonal equation systems, Equations (5.23) and (5.24). By considering the equation system used to update $u_\alpha$ along a line in the $x$-direction the term

$$WR1_{i,j} = \pm \delta_{i,j}^- D_1_i u_{\alpha i-1,j}^\mp + \delta_{i,j}^+ D_3_i u_{\alpha i+1,j}^\mp$$

is added temporarily to the right hand side of the equation system, see Equation (5.23). The quantity, $\delta_{i,j}^\pm$, is given by the expression

$$\delta_{i,j}^\pm = \begin{cases} 1, & W_{i,j} \neq W_{i\pm1,j} \\ 0, & W_{i,j} = W_{i\pm1,j} \end{cases}$$

(5.45)

corresponding to $W_{i,j} = \pm 1$, and $i = 0, \ldots, II, \ j = 0, \ldots, JJ$.

Analogously, in solving for $v_\alpha$ along a line in the $y$-direction the right hand side of the equation system, see Equation (5.24), must be modified temporarily by addition of the term

$$WR2_{i,j} = \mp \sigma_{i,j}^- D_1_j v_{\alpha i,j-1}^\mp + \sigma_{i,j}^+ D_3_j v_{\alpha i,j+1}^\mp$$

(5.46)

where

$$\sigma_{i,j}^\pm = \begin{cases} 1, & W_{i,j} \neq W_{i,j\pm1} \\ 0, & W_{i,j} = W_{i,j\pm1} \end{cases}$$

(5.47)

Due to the fact that no presumptions are made in generating the incident wave field, various wave theories can be incorporated into the formulation in a straightforward manner. In Chapter 4 it was shown that the Fourier method of Section 3.8 constitutes a very reliable and accurate method for the description of regular waves of finite amplitude in closed as well as open basins of constant depth. In order to allow the incident wave field at the line of wave generation to propagate at an angle to the computational grid the application of the Fourier method is modified slightly. This is done by definition of a variable, $\theta$, denoting the angle between the $x$-axis and a wave orthogonal. The quantity, $\theta$, is assumed to be positive in the anti-clockwise direction. At each time level the horizontal velocity vector of the incident wave field is determined by considering the $x$- and $y$-components of the horizontal velocity computed by the Fourier method.
5.4 Description of the Code and the Required Input

From the description given in the preceding sections it is evident that the computational method relies heavily on the specification of the node types, $N_{i,j}$, as well as the internal wave generation lines, designated by the quantity, $W_{i,j}$. Generally, it is not feasible to provide this information by hand, since practical simulations often require a large number of nodes to be used. For that reason the computational model is accompanied by a module used to determine the values of $N_{i,j}$ and $W_{i,j}$ at each computational point.

An unambiguous determination of the node types in the computational domain can be achieved by specification of the node types at the corner points as well as the end coordinates of each boundary segment. This reduces significantly the amount of input as can be seen from the example given below.

A computational rectangle is considered which covers 60 nodes in the $x$-direction and 100 nodes in the $y$-direction. At the boundary segment connecting the coordinates, $(0,100)$ and $(60,100)$, a reflecting condition is used. The remaining boundaries are absorbing boundaries bolstered with sponge layers, each 10 nodes wide. In short, the information can be written

\begin{verbatim}
BeginMainBoundary....
  11  0  0  60  0  10  A
  51  60  0  60  100  10  A
  55  60  100  0  100  0  R
  15  0  100  0  0  10  A
EndMainBoundary.....
\end{verbatim}

where each row describes a boundary segment. The first column designates the node type of the corner point at which the boundary segment starts. Columns two through five contain the start coordinate and the end coordinate of each boundary segment. The last two columns specify the width of the sponge layer (in nodes) as well as the boundary condition applied at the boundary segment.

In addition to the information associated with the computational domain, information must be provided on the geometry of the internal boundaries (if any). In the present code an internal boundary is defined as a closed contour, hence indicating that the start coordinate of the first boundary segment must be equal to the end coordinate of the last. Information on the closed contours is provided in a manner consistent with the specification of the computational domain.

For the example under consideration an internal reflecting boundary is given. Since the boundary segments connect the coordinates, $(0,0)$, $(20,0)$, $(20,40)$, $(0,40)$, and $(0,0)$, a rectangular region located in the lower, left corner of the computational domain is excluded. The input is provided in the form

\begin{verbatim}
BeginInternalBoundary  4
  155  0  0  20  0  0  R
  115  20  0  20  40  0  R
  111  20  40  0  40  0  R
  151  0  40  0  0  0  R
EndInternalBoundary..
\end{verbatim}

in which the number at the first line specifies that the closed contour consists of four line segments. A virtually unlimited number of line segments can be considered in a given contour
5.4. Description of the Code and the Required Input

and there are no practical restrictions on the number of closed contours.

The quantity, $W_{i,j}$, used to generate the incident wave field along lines is determined by
the specification of a number of line segments on both the reflected and the total side of the
wave generation line. In addition to the boundaries outlined above a single wave generation
line is considered in the current example. The incoming wave field is generated on the right
hand side of the line extending between the coordinates (10, 40) and (10, 100). The wave
generation line is given by

\begin{verbatim}
BeginWaveLine............ 4
  9  40  9  100  R
 10  40 10  100  R
 11  40 11  100  T
 12  40 12  100  T
EndWaveLine............
\end{verbatim}

By considering a given row, i.e. a line segment, the first four columns denote the start and
the end coordinate of the line segment, while the last column specifies whether or not the line
is located on the reflected or the total side of the generation line.

Additionally, it may be noted that the boundary conditions specified at the internal
boundaries as well as at the boundary of the computational domain provide sufficient support
for the incident wave field.

No description is given of the remaining data required for the execution of the computa-
tional model, since the specification of these is straight forward.

By analogy with the computational model described in Chapter 3 the present code
consists of six units in addition to the main program, Abm.3D, see Appendix C. A brief
description of each unit is given below.

- Boundary.3D computes the initial conditions as well as the boundary conditions at each
time step.
- Fourier.3D computes the incident wave field. In contrast to the method given in Section
  3.8 the present code allows the incident wave field to propagate at an angle to the grid
  (at the line of wave generation).
- Grid.3D determines the quantities, $N_{i,j}$ and $W_{i,j}$, at each computational point.
- Solution.3D carries out the modifications caused by the internal generation of the inci-
dent wave field. In addition the unit consists of procedures used to advance the compu-
tations in time.
- Variable.3D contains all global variables, types and constants. The computations are
  performed on floating point numbers of the type, extended.
- Various.3D computes the spatial discretization of a given term at each computational
  point.

In the next section examples are given showing that the computational model produces
physically plausible results. Despite the fact that the results are promising it is believed that
further tests are required to verify the model quantitatively. Consequently, the examples given
should be interpreted only as a partial verification of the model.
5.5 Partial Verification of the 3-D Model

5.5.1 Ring Test

As mentioned in Section 5.2 the cross-derivative terms and the convective terms involving both \( u_a \) and \( v_a \) are responsible for the transfer of momentum from one direction to the other. An accurate simulation of the evolution of waves in two horizontal dimensions requires that these be approximated correctly. In the following a simple test is carried out demonstrating that the terms have been incorporated correctly into the formulation.

A rectangular computational domain is considered. Each horizontal direction is covered by 61 nodes, i.e. \( i \in [0,60] \) and \( j \in [0,60] \), and the grid spacing in each direction is \( \Delta x = \Delta y = 0.333 \). For simplicity, the still water depth is presumed to be constant, and equal to \( D = 1.00 \). In accordance with the examples of Chapter 4 the computational model is run at a Courant number of \( Cr = 0.500 \). In the initial, quiescent water state the surface elevation is given by a Gaussian distribution of the form

\[
\eta(x,y,0) = 0.2 \exp\left\{-\frac{1}{2} \left[\left(\frac{x - 10}{5 \Delta x}\right)^2 + \left(\frac{y - 10}{5 \Delta y}\right)^2\right]\right\}
\]  

(5.48)

thus indicating that there is rotational symmetry about a vertical axis intersecting the \( xy \)-plane in the centre of the computational domain. Absorption of outgoing waves is carried out by employment of sponge layers located next to the main boundary. These are 10 nodes wide.

The evolution of the waveform is depicted in Figures 5.3 and 5.4 as a function of the node indices, \( i \) and \( j \), at times, \( t = 5.94 \) and \( t = 8.91 \), respectively. In addition to a contour plot of the surface elevation a perspective view is included in each figure. The perspective plots aid the visualization of the wave motion, while the contour plots show that perfectly circular patterns are obtained. Since an incorrect approximation of the terms responsible for the transfer of momentum from one direction to the other would cause the contour lines to resemble squares, the circular patterns obtained demonstrate that the model reproduces their effect correctly.

In Figure 5.4 it should be noticed that the contour line located closest to the computational boundary is far from circular. This can be explained by the fact that the wave motion is influenced by the sponge layers in this region.

5.5.2 Diffraction of Deep Water Waves Around a Breakwater

In deep water the diffraction of regular waves of small steepness around a reflecting breakwater can be calculated analytically by employment of the Helmholtz equation (Dean & Dalrymple, 1994). Although Boussinesq-type equations are shallow water equations, the present formulation allows them to be extended into deeper water. As their solution becomes effectively linear in deep water, a comparison can be made with the so-called Sommerfeld diffraction solution (Sommerfeld, 1896).

A computational domain consisting of 61 nodes in the \( x \)-direction and 101 nodes in the \( y \)-direction is considered, i.e. \( II = 60 \) and \( JJ = 100 \). In accordance with the example
5.5. Partial Verification of the 3-D Model

Figure 5.3: The ring test illustrates that the effect of the cross derivative terms as well as the convective terms involving both $u_0$ and $v_0$ is simulated correctly. The wave field is depicted at time, $t = 5.94$. Data: $D = 1.00$, $i \in [0.60]$, $j \in [0, 60]$, $\Delta x = \Delta y = 0.333$, $Cr = 0.500$, $x_s = y_s = 10 \Delta x$. 

(a) Contour plot.

(b) Perspective view.
CHAPTER 5. WAVE PROPAGATION IN TWO HORIZONTAL DIMENSIONS

Figure 5.4: The ring test illustrates that the effect of the cross derivative terms as well as the convective terms involving both $u_o$ and $v_o$ is simulated correctly. The wave field is depicted at time, $t = 8.91$. Data: $D = 1.00$, $i \in [0, 60]$, $j \in [0, 60]$, $\Delta x = \Delta y = 0.333$, $Cr = 0.500$, $x_s = y_s = 10 \Delta x$. 

(a) Contour plot.

(b) Perspective view.
discussed in Section 5.4 a totally reflecting, rectangular breakwater is located in the lower left corner of the computational domain. The coordinates of the corner points are identical to those of Section 5.4. The still water depth is assumed to be constant and equal to $D = 1.00$, while the grid spacing in each horizontal direction is given by $\Delta x = \Delta y = 0.200$. The Courant number is equal to $C_r = 0.500$ and the maximum time of computation is $t_{\text{max}} = 48.7$. In agreement with the example of Section 5.4 outgoing waves are absorbed by use of sponge layers, each 10 nodes wide. For the wave period under consideration this ensures a satisfactory absorption of the outgoing wave field.

By employment of the numerically exact Fourier method (see Section 3.8) a regular wave field is generated along the line described in Section 5.4 using 10 Fourier components and an Eulerian mean current equal to zero (in the direction of a wave orthogonal). At the line of wave generation the incident wave field propagates in the direction of the positive $x$-axis, thus implying that $\theta = 0^\circ$. The wave height is equal to $H_0 = 0.0200$ and the absolute wave period is given by $T = 3.54$. By use of the Fourier method of Section 3.8 it follows that $\frac{L}{L} = 0.496$ and $\frac{H}{T} = 0.993\%$, hence showing that a small amplitude wave in virtually deep water is considered. This allows a comparison to be made with the Sommerfeld diffraction solution.

Figure 5.5 shows the wave height variation as a function of the node indices, $i$ and $j$. The graph denoted a) depicts the computed wave height, extracted during a single wave period in the latter stages of the computation. Similarly, the graph denoted b) shows the Sommerfeld diffraction solution digitized from Figure 4.26 of Dean & Dalrymple (1994). Although details of the computed wave height variation disagree somewhat with the Sommerfeld solution the result exhibits a satisfactory overall agreement. This can be ascertained by considering the distinct wave height extrema in the majority of the remaining part of the fluid domain.

The reasons for the minor discrepancies have not been traced any further, but it is anticipated that the spatial discretization in the vicinity of the corner points can be improved slightly. Additionally, by considering the contour lines in the regions close to the top boundary (i.e. the boundary located at $i = 0, \ldots, 60, j = 100$) there seems to be evidence that wave energy radiating away from the corner point is reflected from the top boundary. Although this is an indication that the time frame of the computation is too long it proves necessary to run the model for the chosen period of time to obtain a fully developed wave field. Clearly, the problem could be eliminated by extending the computational domain in the $y$-direction but this is not an option in the present code due to severe memory constraints.

Figure 5.6 depicts the zero-crossings of the surface elevation at time, $t = 48.7$. From the graph it is apparent that the incident wave field is diffracted around the corner point. The small oscillations of the contour lines are caused by wave energy radiating away from the corner point. Additionally, from the graph simple calculations confirm that the wave length of the incident wave train is almost equal to the linear deep water wave length of Stokes’ as should be expected (since the computation is carried out at a physical depth of $D^* = 50.0$ m).

### 5.5.3 Refraction-Diffraction Test of Regular Waves Propagating over a Semicircular Shoal

In this section the computational model is used to study the diffraction and the refraction of regular waves propagating over a semicircular shoal. By comparison with the experimental data of Whalin (1971) it is shown that the developed model computes accurately the variation in space of the first, second, and third harmonic amplitudes.

Whalin (1971) conducted a series of wave-focusing experiments in a closed wave flume,
CHAPTER 5. WAVE PROPAGATION IN TWO HORIZONTAL DIMENSIONS

(a) Boussinesq model  
(b) Sommerfeld solution (digitized)

Figure 5.5: Wave height variation of deep water waves diffracted around a breakwater. Data: $D = 1.00$, $H_0 = 0.0200$, $T = 3.54$, $c_E = 0.00$, $\theta = 0^\circ$, $M = 10$, $i \in [0, 60]$, $j \in [0, 100]$, $\Delta x = \Delta y = 0.200$, $C_r = 0.500$, $x_s = y_s = 10 \Delta x$, $t_{max} = 48.7$.

Figure 5.6: Zero-crossings of the surface elevation at time, $t = t_{max} = 48.7$. Data: $D = 1.00$, $H_0 = 0.0200$, $T = 3.54$, $c_E = 0.00$, $\theta = 0^\circ$, $M = 10$, $i \in [0, 60]$, $j \in [0, 100]$, $\Delta x = \Delta y = 0.200$, $C_r = 0.500$, $x_s = y_s = 10 \Delta x$. 
25.603 m long and 6.096 m wide. The incident regular wave field was generated in the deeper part of the flume using a wave period of 1.00 s, 2.00 s, or 3.00 s. For each wave period various incident wave heights were considered. The present example considers a wave period of $T^* = 2.00$ s, and an incident wave height given by $H_0^* = 0.0298$ m. The topography is defined as

\[
D^*(x^*) = \begin{cases} 
0.4572 \text{ m} & , 0 \leq x^* < 10.67 \text{ m} - G^* \\
0.4572 \frac{m - G^*}{2} - 0.1524 \text{ m} & , 10.67 \text{ m} - G^* \leq x^* < 18.29 \text{ m} - G^* \\
0.1524 \text{ m} & , 18.29 \text{ m} - G^* \leq x^* \leq 21.34 \text{ m}
\end{cases} \tag{5.49}
\]

in which $G^* = G^*(y^*)$ is given by

\[
G^*(y^*) = \sqrt{y^*(6.096 \text{ m} - y^*)} \tag{5.50}
\]

Figure 5.7 visualizes the topography for the problem under consideration. From the graph it is evident that the computational domain is discretized by 200 nodes in the $x^*$-direction and 40 nodes in the $y^*$-direction using a constant grid spacing of $\Delta x^* = \Delta y^* = 0.1524$ m, i.e. $x^* \in [0.00 \text{ m}, 30.48 \text{ m}]$ and $y^* \in [0.00 \text{ m}, 6.096 \text{ m}]$. The time frame of the computation is 120 s and the Courant number is given by $Cr = 0.300$.

As in the previous examples the incident wave field is generated internally using the Fourier method of Section 3.8 and the wave field is resolved by 10 Fourier components, i.e. $M = 10$. Since the computations relate to wave propagation in a closed wave flume, the mass transport velocity, $c^*$, is set equal to zero. The boundaries perpendicular to the $x^*$-direction are bolstered with absorbing sponge layers, 20 nodes wide. The remaining boundaries are reflective boundaries, thus providing the necessary support for the waves. The incident wave field is imposed on the right hand side of the line extending between the node coordinates $(20,0)$ and $(20,40)$ and the initial direction of the wave orthogonals is parallel to the $x^*$-direction, i.e. $\theta = 0^\circ$.

The surface elevation computed by the model is depicted in Figure 5.8 at time, $t^* = 30.0$ s. The graph shows that the semicircular shoal causes the wave energy to focus in the convergence zone. In the downstream part of the flume of constant still water depth the scattered wave field interacts with the reflecting boundaries, hence giving rise to a local increase in the maximum surface elevation in the downstream corner regions. Additionally, the graph illustrates that the amount of reflection from the absorbing sponge layers is insignificant.

Whalin (1971) presented measurements of the first, second, and third harmonic amplitudes along the centre line of the wave flume. For clarity it is mentioned that the frequencies of these are given by $\frac{1}{T^*}$, $\frac{2}{T^*}$, and $\frac{3}{T^*}$, respectively. In order to predict the harmonic amplitudes accurately and efficiently the surface elevation along the centre line of the flume is extracted from the computations during 2048 time levels prior to program termination. The harmonic amplitudes are computed by performing a fast Fourier transform at each grid point.

Figure 5.9 shows the first, the second, and the third harmonic amplitudes both computed by the model and measured in the experiment of Whalin (1971). It is mentioned that the thick lines to the left and the right designate the last and the first sponge layer node, respectively.
Figure 5.7: Bed elevation. The bathymetry is given by Equation (5.49).

Figure 5.8: Surface elevation depicted at time, $t^* = 30.0$ s. Data: $H_0^* = 0.0298$ m, $T^* = 2.00$ s, $c_0^* = 0.00$ m/s, $\theta = 0^\circ$, $M = 10$, $i \in [0, 200]$, $j \in [0, 40]$, $\Delta x^* = \Delta y^* = 0.1524$ m, $Cr = 0.300$, $x_s^* = 20 \Delta x^*$, $y_s^* = 0$, $t_{max}^* = 120$ s.
5.5. Partial Verification of the 3-D Model

Although the scatter in the measurements is substantial it is evident that the model computes accurately the variation of all three harmonic amplitudes. There seems to be a minor tendency to underestimate the amplitude of the second harmonic in the deeper part of the flume. Similarly, the amplitude of the first harmonic is slightly overestimated in the shallow part of the wave tank. However, the overall agreement is good.

5.5.4 Waves Propagating into a Fictitious Harbour

The last example relates to the propagation of long waves into a fictitious harbour of a constant water depth, \( D = 1.00 \). The corresponding dimensional depth is \( D^* = 5.00 \) m. The computational domain is covered by 80 nodes in the \( x \)-direction and 70 nodes in the \( y \)-direction using a constant grid size of \( \Delta x^* = \Delta y^* = 0.1524 \) m, \( Cr = 0.300, x_s^* = 20 \Delta x^*, y_s^* = 0, t_{max} = 120 \) s.

A regular wave train is generated internally along a single line segment located at the harbour entrance. In contrast to the previous example the incident wave field propagates at an angle of \(-15.0^\circ\) to the grid and the mean mass flux (in the direction of a wave orthog-
nal) is presumed to be equal to zero, hence \( c_s = 0.00 \). Based on an (absolute) wave period, \( T = 16.8 \), and a wave height, \( H_0 = 0.200 \), the Fourier method is invoked using 10 Fourier components, thus yielding that \( \frac{D}{L} = 0.0593 \) (and \( \frac{H}{D} = 0.200 \)). In order to absorb scattered waves travelling out of the harbour basin the boundary located at \( i = 0, j = 0, \ldots, 70 \), is bolstered with a sponge layer, 10 nodes wide. Similarly, a reflecting condition is applied at the remaining internal boundaries.

Figures 5.10 and 5.11 depict the surface elevation at four different time levels after the start of the computation. From the first graph, depicted at time, \( t = 35.0 \), it appears that the initial transient is diffracted in entering the harbour basin. At this stage of the computation a part of the wave front is about to reach the reflecting breakwater located inside the harbour basin.

The second graph, depicted at time, \( t = 70.0 \), shows that the reflected waves propagating out of the harbour are damped by the sponge layers. Additionally, it is apparent that the front of the incident wave field is diffracted in passing the tip of the breakwater inside the harbour.

In the third and the fourth graph, depicted at time, \( t = 105 \) and \( t = 140 \), respectively, the incident wave field has moved far into the harbour. Clearly, the interaction of incident and reflected waves results in a complex pattern of the free water surface.

The present example concludes the discussion of the simulation of waves in two horizontal directions. In the following chapters attention will be given to the inclusion of energy dissipating terms caused mainly by wave breaking. Due to the complexity of the problem in hand the analysis is confined to a single horizontal direction and the computational model developed in Chapter 3 is invoked.
Figure 5.10: Waves entering a fictitious harbour. The graphs are depicted at times (a) $t = 35.0$, and (b) $t = 70.0$. Data: $D = 1.00$, $H_0 = 0.200$, $T = 16.8$, $\theta = -15.0^\circ$, $c_s = 0.00$, $M = 10$, $i \in [0, 80]$, $j \in [0, 70]$, $\Delta x = \Delta y = 1.00$, $Cr = 0.500$, $x_s = 10 \Delta x$. 
Figure 5.11: Waves entering a fictitious harbour. The graphs are depicted at times (c) $t = 105$, and (d) $t = 140$. Data: $D = 1.00$, $H_0 = 0.200$, $T = 16.8$, $\theta = -15.0^\circ$, $c_s = 0.00$, $M = 10$. $i \in [0, 80]$, $j \in [0, 70]$. $\Delta x = \Delta y = 1.00$, $Cr = 0.500$, $x_s = 10 \Delta x$. 
Chapter 6

Dissipation of Wave Energy

6.1 General

An accurate description of the dynamics of the surf zone is of major importance for the prediction of the wave height envelope, wave-induced flows, and the mean water level. On the basis of the computational model of Chapter 3 the present chapter describes the inclusion of energy dissipation caused by wave breaking. This is done in a single horizontal dimension only. Since the bathymetry of coral reefs is generally very rough, a minor part of the chapter is devoted to the inclusion of the bed friction.

In addition to a qualitative description of wave breaking a review is given of the various efforts undertaken to include the most significant effects of the breaking process.

Based on the assumption of a vertical redistribution of the horizontal velocity field in a breaking wave a new set of equations is derived. In the case of a non-breaking wave these simplify to the equations of Nwogu (1993). The initiation, the temporal development, and the cessation of wave breaking are modelled by employment of a method which is capable of reproducing correctly several features of a breaking wave (Schäffer et al., 1993).

Since the wave breaking method incorporates a number of calibration factors, a sensitivity analysis is carried out illustrating that the major characteristics of the breaking process are relatively insensitive to the choice of these although they all affect the details of the solution.

Finally, the results of several computations are compared with laboratory measurements. These verify that the computational model produces sound physical results.

6.2 Review of Methods

6.2.1 Qualitative Description of the Surf Zone

In recent years wave breaking and the development of waves in the surf zone have been studied extensively. Despite the fact that the physical mechanisms are relatively well understood much remains to be done before real prediction based on the underlying physics is possible.
The present section defines the surf zone and outlines the process of spilling and plunging wave breaking on the basis of existing qualitative models.

The breaking point is defined as the location in space where a part of the front face of a wave has become vertical and the wave is about to turn over (at some scale). When the jet penetrates the free surface an irreversible process takes place, and ordered wave energy is transformed into rotational energy which eventually becomes of a random nature, i.e. turbulence. The turbulence gradually dissipates into heat. As a consequence, wave breaking can be defined as a transformation of the flow field from irrotational to rotational (Basco, 1985).

Immediately after a wave has turned over and the jet strikes the free water surface a violent transition takes place over a horizontal distance of several times the water depth at the breaking point. In accordance with Svendsen et al. (1978) this region is termed the outer transition region. Further inside the surf zone the shape of a broken wave changes slowly and the front part resembles a (periodic) bore. This region, denoted the inner region, extends to the shore line where the run-up takes place. Svendsen et al. noted that in the inner region the waves originating from plunging breakers cannot be distinguished visually from those generated by spilling breakers.

Based on visual laboratory observations Peregrine & Svendsen (1978) studied the internal flow pattern of quasi-steady spilling breakers. The observations showed that an aerated surface layer is generated at the front of a spilling breaker. Peregrine & Svendsen proposed that the surface layer, denoted the surface roller, initially acts as a trigger for the production of turbulence. The turbulence originates at the toe of the roller and in the beginning the process is similar to that in a turbulent mixing layer. In the wake behind the breaking region the turbulent fluctuations spread and decay and at some stage the entire flow becomes turbulent.

A review of existing literature was given by Basco (1985) who focused on the outer transition region. Basco emphasized that spilling and plunging wave breaking are similar processes although they happen at far different scales in both time and space. One of the key findings was the fact that the overturning jet penetrates the free water surface and creates a vortex system consisting of a plunger vortex and a surface roller. The plunger vortex emanates from the plunging motion while the surface roller is created by the splash of water as the jet strikes the free surface. In a reference system travelling with the wave celerity the horizontal momentum of the jet causes the plunger vortex to translate forward and form a secondary wave disturbance with the same wave period as the original wave but with far different characteristics. The translation speed of the plunger vortex decreases and the vortex enlarges and drifts downwards. Similarly, the secondary wave disturbance forces the toe of the surface roller to slide down the face of the oncoming wave. The shear stresses acting at the interface between the roller and the surrounding water generate turbulence which causes water to be entrained into the roller, hence indicating that the volume of the surface roller increases as the inner region is approached. Basco defined the start of the inner region as the point where the surface roller has reached an equilibrium position. This is in contrast to Svendsen (1984) who defined the start of the inner region as the location where the gradient of the mean water level starts to increase. The definition of Svendsen will be adopted in the following. For further qualitative details on spilling and plunging wave breaking specific reference is made to Basco (1985).

Svendsen (1984) found that only a very minor part of the ordered wave energy is lost in the outer transition region. The significant decrease in the wave height is mainly associated with a redistribution of potential energy into forward momentum flux concentrating in the surface roller. Consequently, it is important to note that the outer region does not conform to traditional radiation stress theory in conjunction with a wave theory for non-breaking waves.
An accurate prediction of the wave height variation, the mean water level, and the wave-induced flow inside the surf zone requires that the radiation stress, the mean energy flux, and the rate of energy dissipation be known. An important first step towards this is to describe the flow field in the outer transition region.

The next section describes some of the analytical and experimental efforts undertaken to quantify the variation of the wave height and the mean water level inside the surf zone.

### 6.2.2 Analytical and Experimental Methods

By the assumption of a hydrostatic pressure distribution over the water column Dorrestein (1961) considered the horizontal component of the depth-integrated momentum equation. By employment of linear wave theory and some measured wave records the wave setup was quantified inside the surf zone. The theoretical values compared poorly with field measurements.

Horikawa & Kuo (1966) employed the energy equation to derive expressions for the variation of the wave height inside the surf zone. This was done by use of solitary wave theory (Svendsen & Jonsson, 1980) in conjunction with an expression for the variation of the turbulence shorewards of the breaking point. The model proved capable of reproducing the wave height variation accurately in the case of waves breaking on a horizontal bottom but the results agreed poorly with laboratory measurements in the case of a sloping bottom.

A laboratory experiment was carried out by Bowen et al. (1968) who studied the wave setup caused by regular waves breaking on a plane beach. Outside the breaking point the depression of the mean water level, i.e. the set-down, was found to be in good agreement with the radiation stress theory of Longuet-Higgins & Stewart (1962). In the inner region the measurements indicated that the wave height varied approximately linearly with the mean water depth. By employment of the theory of Longuet-Higgins & Stewart this was used to propose a model for the wave-induced setup in the inner region. In addition it appeared from the measurements that a transition region exists between the breaking point and the point where the gradient of the mean water level starts to increase. In this region the set-down was approximately constant, thus implying that the change in the radiation stress was negligible.

Following Le Méhauté (1962) the similarity between a hydraulic jump and a breaking wave was utilized by Divoky et al. (1970). In a frame of reference travelling with the wave celerity the combination of the solitary wave theory (see Svendsen & Jonsson, 1980) and the expression for the energy loss in a hydraulic jump (Lamb, 1945) yielded the (approximate) energy dissipation of a solitary wave breaking on a horizontal or a gently sloping bottom. Measurements of the horizontal velocity component performed under the crest of spilling breakers were found to be in good agreement with those of a limiting height non-breaking wave except in the vicinity of the crest.

A model valid for irregular waves over complex beach topographies was presented by Battjes & Janssen (1978). At a given water depth the wave height distribution was presumed to belong to a Rayleigh distribution. This was formulated in terms of a root-mean-square wave height and a maximum wave height at the considered water depth, thus yielding the upper bound of the wave height distribution. In shallow water the maximum wave height was determined by employment of the breaking criterion of Miche. The mean energy dissipation at a given location was estimated on the basis of the fraction of broken waves as well as the analogy of a hydraulic jump. The height of the jump was assumed to be equal to the maximum wave height at the considered water depth. Based on linear theory and the assumption of a narrow-banded wave spectrum the energy equation and the momentum equation were solved.
simultaneously yielding the variation of both the mean water level and the root-mean-square wave height throughout the surf zone. In order to improve the accuracy of the model two assignable parameters were included in the formulation. These were determined by trial and error. In the case of waves breaking over a bar the results of the model compared very well with laboratory measurements. A detailed calibration of the model was carried out by Battjes & Stive (1985).

Svendsen et al. (1978) performed a combined experimental and theoretical work concentrating on the wave celerity and the energy dissipation of regular waves breaking on a plane and gentle slope. The measurements were conducted in a closed wave flume and the incident wave field was generated correct to the second order in the wave steepness. Inside the surf zone measured values of the wave height and the celerity were compared with analytical values. These were found by integrating the equations of mass, momentum, and energy using a control volume extending over the instantaneous water depth. The energy dissipation in a breaking wave was assumed to be equal to that in a hydraulic jump of the same height. By comparison with the experiments it was found that the calculated bore velocity agrees well with the measured wave celerity, whereas the energy dissipation in a moving bore underestimates the actual dissipation in a breaking wave by up to 20%.

In continuation of the work reported by Svendsen et al. (1978), Svendsen (1984) developed a theoretical model for the variation of the wave height and the setup in the inner surf zone using the analogy of a hydraulic jump. The effect of the surface roller was included in the calculations by the assumption of a redistribution of the horizontal velocity component over the depth. In addition the pressure was assumed to be hydrostatic while the vertical velocity was neglected. As a first approximation the roller was treated as a passive volume of water travelling with the linear shallow water wave celerity. Below the roller a uniform velocity profile was considered. Since the mean mass transport was equal to zero (which is consistent with the fact that the experiments were carried out in a closed wave flume), this allowed the velocity below the roller to be determined. The cross-sectional area of the surface roller was expressed as a function of the wave height on the basis of existing measurements. A comparison with measurements revealed that the inclusion of a surface roller in the basic equations significantly improves the prediction of the mean water level and the wave height decay in the inner surf zone. Although the theoretical model does not, in principle, apply to the outer transition region the theoretically determined wave height variation compared surprisingly well with the measurements in this region. On the other hand, the prediction of the mean water level in the outer region compared poorly with the measurements. This could be anticipated, since the model does not account for the conversion of potential energy to kinetic energy during the initial stages of the breaking process.

Svendsen (1987) used existing wave flume experiments to analyze the variation in time and space of the turbulent kinetic energy under surf zone waves. At a fixed location the analysis indicated that the temporal variation over a wave period of the turbulent kinetic energy is relatively small. In the first stages after the wave front has become vertical the analysis showed that the production of turbulence is much smaller than further shorewards. This is consistent with the fact that the turbulence is generated mainly inside or in the nearest proximity of the surface roller, see Peregrine & Svendsen (1978) and Basco (1985). The experiments also indicated that the variation over the water depth of the turbulent kinetic energy is remarkably weak and strongly governed by vertical mixing. This was attributed to the fact that the turbulence under a breaking wave is spread mainly by convection (due to the presence of large-scale vortices).

Based on existing measurements the variation of the radiation stress, the mean energy
flux and the energy dissipation in the surf zone were analyzed by Svendsen & Putrevu (1993). The results indicated that a relatively insignificant amount of ordered energy is dissipated as the wave front becomes vertical and the jet strikes the free surface. Following Svendsen (1987) they deduced that the initial decrease in the wave height is associated with a conversion of potential energy to kinetic energy in the jet. Consequently, the radiation stress stays approximately constant and the gradient of the mean water level is close to zero. Moreover, Svendsen & Putrevu concluded that the coupling of the radiation stress, the mean energy flux, and the energy dissipation is of great importance in predicting accurately the variation of the wave height, the mean water level, and the wave-induced flow. This is in good agreement with the work by Stive (1984).

An experimental investigation of the velocity field and the pressure field in quasi-steady spilling and plunging breakers was performed by Stive (1980). For waves close to the breaking point the measurements showed that the horizontal velocity field remains quite symmetric about the crest. In addition it appeared that cnoidal wave theory describes well the surface elevation and the vertical distribution of the horizontal particle velocity with the exception of the upper crest region. In the outer transition region the measurements showed that characteristic points located at the water surface in the crest region travel at speeds of the order of 30% in excess of the linear shallow water celerity. This value gradually decreases further inside the surf zone. In the inner region it was noticed that the surface elevation and the horizontal velocity are relatively asymmetric, implying that these can not be predicted by means of a wave theory valid for non-breaking waves. The measurements also indicated that the pressure as a first approximation can be assumed to be hydrostatic. For both spilling and plunging breakers extraction of the turbulent fluctuations of the velocity field showed that in the region behind the crest the turbulence spreads and decays. This confirms, and extends, the qualitative model of Peregrine & Svendsen (1978) which was originally developed for spilling breakers.

By employment of the measurements of Stive (1980) a semi-empirical study of the radiation stress and the mean water level close to the breaking point and inside the surf zone was carried out by Stive & Wind (1982). Since no velocity measurements were available in the crest region, the radiation stress was calculated by linear extrapolation of the (periodic) velocity field. By the assumption that the wave height adjusts in accordance with the maximum energy flux at the considered water depth the shoaling theory of Sakai & Battjes (1980) was used to calculate a theoretical wave height variation before and after breaking. This enabled a theoretical prediction of the mean water level and the radiation stress. A comparison with the experimental values showed a poor agreement, particularly in the outer transition region. This could be expected, since traditional wave theories cannot model the rapid conversion of potential energy to kinetic energy during the initial stages of the breaking process.

Another semi-empirical model for the variation of the wave height and the setup in the inner surf zone was developed by Stive (1984). On the basis of previous experiments it was concluded that the flow field in spilling and plunging breakers on gentle slopes closely resembles that of a hydraulic jump (in the inner surf zone). Consequently, the energy equation was expressed in terms of the dissipation in a hydraulic jump and a correction factor accounting for the effects of turbulent flow. By employment of linear shallow water wave theory the momentum equation and the energy equation were solved simultaneously giving both the wave-induced setup and the wave height. A comparison with experiments showed that the wave height decay was modelled accurately, whereas the setup agreed poorly in both the outer and the inner region of the surf zone. The discrepancies were believed to be caused by the difficulties in predicting accurately both the momentum flux and the energy flux in
the outer region. The analysis of Stive also indicated that the theoretical energy dissipation in a hydraulic jump underestimates the energy loss in a breaking wave by up to 30 – 50% depending mainly on the incident wave conditions.

Dally et al. (1985) developed a surf zone model which does not rely on the analogy of a hydraulic jump. The model is able to describe the initiation and the cessation of regular waves breaking across beaches of arbitrary profile. Breaking initiated when the wave height attained a value greater than a predetermined fraction of the mean water depth. Inside the surf zone the energy dissipation was assumed to be proportional to the difference between the actual energy flux and a stable energy flux. These were expressed in terms of an actual and a stable wave height, respectively. on the basis of linear shallow water wave theory. By examination of the experimental data of Horikawa & Kuo (1966) the stable wave height was assigned a value proportional to the local mean water depth. The energy and momentum equations were solved simultaneously giving the variation of the wave height and the mean water level before and after breaking. Several comparisons with experiments illustrated the ability of the model to reproduce the wave height variation accurately but the model was not able to estimate the start of the inner region, and hence the spatial variation of the wave-induced setup.

In the case of regular waves breaking on a uniform and gently sloping beach Swift (1993) extended the analytical model of Dally et al. (1985). Based on the assumption that the gradient of the mean water level is constant in the inner region (Bowen et al., 1968) linear shallow water wave theory was used to solve the momentum equation in a least squares sense, thus providing an optimum value of the gradient of the mean water level in this region. In contrast to the analytical approach by Dally et al. the wave-induced setup was included in the energy and the momentum equation but the method relies heavily on the empirical determination of both the start of the inner region and the location of the breaking point. Several comparisons with laboratory measurements illustrated that an accurate prediction of the wave height variation in the inner region requires that the wave-induced setup be included in the formulation. For small bed slopes, i.e. slopes of the order 1 : 40, say, the predicted variation of the wave height and the mean water level agreed favourably with the experimental data but the agreement was less pronounced for larger bed slopes.

On the basis of the analytical model of Dally et al. (1985) valid for regular waves, Dally (1990) quantified the wave height distribution of irregular waves transforming on a uniform beach of a gentle slope. By the assumption that the wave heights belong to a Rayleigh distribution seaward of the surf zone the transformation was accomplished by considering the wave height variation of individual wave components due to shoaling and wave breaking. The effect of wave shoaling was based on linear shallow water wave theory, while the wave height decay inside the surf zone was expressed in terms of the analytical result of Dally et al. (1985). By performing a number of variable transformations, this allowed an analytical determination of the wave height distribution as a function of the still water depth. The wave-induced variation of the mean water level was neglected. Several comparisons with field data exhibited the ability of the model to reproduce reasonably the change in the shape of the wave height distribution at various locations inside the surf zone.

Dally & Brown (1996) presented a semi-empirical model for the description of the wave-induced current and the setup across a beach of arbitrary profile. In agreement with Svendsen (1984) the effect of wave breaking was achieved by considering a roller on top of the organized wave motion. By the assumption that the energy is dissipated at the interface between the turbulent roller and the underlying flow, the energy dissipation was described in terms of the cross-sectional area of the roller. In addition the flux of kinetic energy through a vertical section was included in the energy equation and expressed as a function of the roller area.
Similarly, the excess momentum caused by the presence of the roller was included in the momentum equation. In contrast to previous attempts it is noted that the convective acceleration of the Eulerian mean velocity below the roller, i.e. the undertow (considered to be uniform over the depth), was included in the formulation (Longuet-Higgins & Stewart, 1964). Since existing measurements indicate that the undertow is minimum inside the breaking point, and not when the wave height is maximum, Dally & Brown did not relate the cross-sectional area of the roller to the wave height. This stands in strong contrast to the model of Svendsen (1984). Instead, the roller area was considered to be unknown apriori. The radiation stress and the energy flux of the organized wave motion were therefore quantified by use of a wave theory in conjunction with existing measurements of the wave height variation before, during, and after breaking. The model was calibrated using the measurements of Hansen & Svendsen (1984) and verified by comparison with other experimental data. Given the correct wave height variation the results showed that the model is capable of reproducing quite accurately the variation of the mean water level and the undertow inside the surf zone. The best results were produced by employment of a nonlinear wave theory (as opposed to linear theory), hence confirming that the finite steepness of a wave is important in the vicinity of the breaking point.

The above description concludes the summary of analytical surf zone models. The next section reviews the inclusion of wave breaking in computational models based mainly on equations of the Boussinesq-type.

### 6.2.3 Flow Models

Analytical models are capable of predicting accurately the variation of the wave height inside the surf zone but the models are not able to estimate the wave-induced setup and the undertow on a sound theoretical basis. This is primarily due to the fact that the vertical variation of the horizontal velocity field is unknown in the outer transition region and, further, that the start of the inner region is unpredictable in models averaged over a typical wave period. The problem is partly overcome by invoking a time-domain model formulated in terms of the surface elevation and some flow variables. Models of this kind are denoted flow models.

Based on the assumption of a hydrostatic pressure and a uniform vertical distribution of the horizontal velocity profile the nonlinear shallow water equations have previously been used to study the transformation of waves on a beach. This was done by Hibberd & Peregrine (1979), Packwood (1980), and Kobayashi et al. (1989). Since the nonlinear shallow water equations are amplitude dispersive but not frequency dispersive, these are suitable mainly in the inner surf zone. The lack of frequency dispersion implies that the wave profile continues to steepen until a vertical front is formed, and hence the equations can not predict the location of the breaking point.

Equations of the Boussinesq-type incorporate the lowest order of frequency and amplitude dispersion. In accordance with the description in the previous chapters it is evident that extended Boussinesq-type equations constitute a viable tool for the prediction of wave evolution in shallow and deeper water, and a natural step forward is to include the simplified effect of wave breaking.

The inclusion of wave breaking in equations of the Boussinesq-type was performed by Tao (1983) and Abbott et al. (1983) who both considered an eddy viscosity term in the depth-integrated momentum equation (Hamm et al., 1993). This was expressed as a product of the horizontal gradients of the considered flow variable and a local eddy viscosity term related to
the water depth and the turbulent kinetic energy. A transport equation was employed for the description of the turbulent kinetic energy. Karambas et al. (1990, 1991) used the same basic concept but the local eddy viscosity was determined by a simple algebraic closure proportional to the product of the linear shallow water wave celerity and the water depth. Karambas & Koutitas (1992) adopted a more sophisticated approach in which the eddy viscosity was determined by the assumption that the turbulence is produced in the front part of a breaking wave and in the wake of the preceding wave. The location of the breaking point and the width of the outer surf zone were determined on the basis of empirical relationships. The model was used for the simulation of regular waves breaking on a beach of a constant slope. By comparison with a variety of experimental data it was shown that the model predicts very accurately the wave height variation before, during, and after incipient breaking. This is surprising, since Boussinesq-type equations are only weakly nonlinear. In a single case the computed mean water level was compared with experimental results. It appeared that the model significantly underestimates the wave-induced setup. In addition there is evidence that the model does not conserve mass, since the computed mean water level was located exclusively above the still water level. Schäffer et al. (1993) noted that the main disadvantage of the above methods is the presumed relationship between the energy dissipation and the horizontal gradients of the horizontal flow variable. The energy dissipation generally depends on vertical gradients of the horizontal velocity profile (Madsen, 1981) and hence the methods are only marginally different to using a dissipative interface.

Engelund (1981) described a weak hydraulic jump by including an additional pressure term in the depth-integrated momentum equation caused by the presence of the surface roller. Using the analogy of a separated diffusor flow the inclination of the interface between the roller and the underlying organized flow was estimated to be equal to approximately 10°. Deigaard (1989) pursued the ideas of Engelund and introduced the surface roller concept in a Boussinesq model originally based on the equations of Abbott et al. (1978). The surface roller was assumed to be a passive lump of water travelling with the wave celerity. Wave breaking initiated when the local slope of the wave front exceeded the value determined by Engelund. Similarly, breaking ceased when the maximum slope of the wave front attained a value smaller than tan10°. Preliminary examples of regular waves breaking over a barred bed profile demonstrated the potential of the model.

Brocchini et al. (1991,1992) quantified the shear stress at the interface of the surface roller and the underlying flow by the assumption that the pressure inside the roller is hydrostatic. This was incorporated into equations of the Boussinesq-type and combined with an empirical relationship for the detection and the growth of a surface roller. Examples were given showing a reasonable agreement with experimental data.

Madsen & Svendsen (1983) and Svendsen & Madsen (1984), respectively, developed a theoretical model for the description of the front of a turbulent bore moving over a horizontal or a sloping bottom. In this approach a highly turbulent shear flow was considered in the upper part of the water column while a practically irrotational flow was considered in the lower part of the water column. In addition to a momentum equation integrated over the turbulent region only, the depth-integrated versions of the continuity, momentum, and energy equations were solved simultaneously. Without turbulence the system of equations reduced to the nonlinear shallow water equations. An important conclusion made from this study was the fact that the simplified effect of wave breaking can be incorporated into the momentum equations by the assumption of a redistribution of the horizontal velocity over the vertical. This results in additional convective terms in the depth-integrated momentum equations.

The ideas outlined above were followed by Schäffer et al. (1993) who included the ef-
fect of spilling wave breaking in a set of Boussinesq-type equations consistent with those of Peregrine (1967). The calculation was carried out in a single horizontal dimension using the concept of surface rollers. Based on the assumption of a uniform vertical distribution of the horizontal velocity in a non-breaking wave, an additional convective momentum term caused by wave breaking was included in the depth-integrated momentum equation by considering the non-uniform velocity profile proposed by Svendsen (1984). In accordance with the experimental results of Stive (1980) the celerity of the surface roller was modelled as 1.3 times the linear shallow water wave celerity. Wave breaking initiated when the maximum slope of the wave front exceeded an assignable threshold value. Similarly, breaking ceased when the maximum slope of a wave front attained a smaller, terminal value. The temporal thickness of the roller was determined geometrically by prescribing a variation in time for the inclination of the interface between the roller and the underlying organized wave motion. In addition to the parameters describing the initiation and the cessation of wave breaking the model incorporates a time-scale for the development of the surface roller as well as a shape parameter accounting for the primitive way of separating the roller from the underlying flow. Simulations of regular as well as irregular waves breaking over a bar were carried out. By comparison with experimental data it was verified that the model predicts reasonably the variation of both the mean water level and the wave height envelope before, during, and after incipient breaking. In particular it is emphasized that the model proved capable of estimating the start of the inner region, i.e. the point where the mean water level starts to increase. This indicates that the model reproduces the effect of a rapid conversion of potential energy to kinetic energy in the outer transition region. Schäffer et al. (1992) extended the surf zone model to include the second horizontal dimension and presented a preliminary example.

Recently, Nwogu (1996) used a fully nonlinear set of Boussinesq-type equations to simulate the transformation of waves caused by wave breaking in two horizontal dimensions. Wave breaking initiated when the horizontal crest velocity exceeded the wave celerity. The simplified effect of wave breaking was incorporated into the momentum equations using a term proportional to the vertical gradient of the horizontal crest velocity. The model was verified by comparison with measured time series of the surface elevation inside the surf zone.

Yu & Svendsen (1996) developed a mathematically consistent surf zone model in which the flow was not assumed to be irrotational. A set of Boussinesq-type equations was derived by splitting the water column into a rotational flow region close to the surface and a lower irrotational core flow. The rotational part of the flow, which was associated with the surface roller, served as a source of vorticity and turbulence, the vorticity being determined by solving an additional vorticity transport equation.

On the basis of the literature review given above, the next section describes the inclusion of the simplified effect of wave breaking in the computational model developed in Chapter 3.

6.3 Inclusion of the Simplified Effect of Wave Breaking

6.3.1 Rederivation of Governing Equations

In Chapter 2 it was mentioned that the derivation of the theory of long waves is based on the assumption of an inviscid fluid and an irrotational flow. Consequently, it is a violation of the theory to employ e.g. Boussinesq-type equations to simulate the transformation of waves in
the surf zone. It is also clear that computational models based on this kind of equations can not simulate the overturning of a plunging breaker and the surf zone model developed in the present chapter is therefore valid only in the spilling regime, i.e. for waves of a relatively large deep water wave steepness propagating over a gently sloping bed profile.

From the previous section it is apparent that the surf zone model described by Schäffer et al. (1993) reproduces the major characteristics of a breaking wave. On the other hand, the calculation is carried out on an engineering basis and it is therefore unknown whether, in fact, the excess momentum term is consistent with the leading order of approximation, i.e. $O(\delta, c^2)$.

The Boussinesq-type equations developed in the present chapter rely on the ideas of Schäffer et al. (1993) but the approach is different. On the basis of a given vertical distribution of the horizontal velocity the flow equations are integrated over the instantaneous water depth following the derivation procedure described by Nwogu (1993). This is done in a single horizontal dimension.

In accordance with Chapter 2 an inviscid and incompressible fluid is considered and the flow is assumed to be irrotational. In order to evaluate the magnitude of the terms in the Euler equations and the boundary conditions the physical variables of the problem in hand are scaled in qualitative agreement with Nwogu (1993). In contrast to the procedure followed in Chapter 3 the horizontal coordinate is scaled by employment of a typical length scale of the wave motion, while the vertical coordinate is scaled by a characteristic water depth. Similarly, the surface elevation and the pressure are made dimensionless on the basis of a typical wave height. This ensures that all the dimensionless variables are $O(1)$. For clarity it is noted that these are different from the dimensionless variables used for computational purposes.

The independent variables are made dimensionless using the relationship

$$
\begin{align*}
x &= \frac{x}{L} \\
z &= \frac{z}{D_0} \\
t &= \frac{t}{\sqrt{g^* D_0^3}}
\end{align*}
$$

while the dependent variables are nondimensionalized using the equation

$$
\begin{align*}
u &= \frac{D_0^2}{H_0^* \sqrt{g^* D_0^3}} u^* \\
w &= \frac{D_0^2}{H_0^* L^* \sqrt{g^* D_0^3}} w^* \\
\eta &= \frac{\eta^*}{H_0^*} \\
D &= \frac{D^*}{D_0^*} \\
p &= \frac{p^*}{\rho^* g^* H_0^*}
\end{align*}
$$

The quantities, $p^*$ and $w^*$, denote the pressure and the vertical velocity, respectively, and their dimensionless equivalents are denoted $p$ and $w$. The density of the fluid is termed $\rho^*$. 


Similarly, $D^*_0$, $H^*_0$, and $L^*$ denote a still water depth, a wave height, and a typical wave length of the problem. For completeness it is mentioned that $u$, $w$, and $p$ depend on $x$, $z$, and $t$, while $\eta = \eta(x,t)$, and $D = D(x)$.

As shown by Nwogu (1993) Equations (6.1) and (6.2) are used to express the continuity equation, the Euler equations, the irrotationality condition, and the dynamic and kinematic boundary conditions in dimensionless form. The continuity equation and the Euler equations are integrated over the instantaneous water depth following a procedure outlined by Mei (1994). The depth-integrated versions of the momentum equation and the continuity equation, respectively, can be written (Nwogu, 1993)

\[
\frac{\partial}{\partial t} \int_{-D}^{\delta \eta} u \, dz + \frac{\delta}{\partial x} \int_{-D}^{\delta \eta} u^2 \, dz + \frac{\partial}{\partial x} \int_{-D}^{\delta \eta} p \, dz - p(x,-D,t) D^* = 0 \tag{6.3}
\]

\[
\eta_t + \frac{\partial}{\partial x} \int_{-D}^{\delta \eta} u \, dz = 0 \tag{6.4}
\]

where $\delta = \frac{H^*_0}{D^*_0} \ll 1$. It may be noted that the first and the second term of the momentum equation represent the local and the convective acceleration of the fluid particles, respectively. The remaining terms represent the integral of the horizontal pressure gradient over the instantaneous water depth. Similarly, it is readily seen that the terms in the depth-integrated continuity equation are the temporal change of the surface elevation and the spatial gradient of the horizontal volume flux.

The pressure field is obtained by integrating the vertical Euler equation in the $z$-direction and applying the boundary conditions at the free water surface, thus yielding the equation (Nwogu, 1993)

\[
p = \eta - \frac{z}{\delta} + \frac{\partial}{\partial t} \int_{z}^{\delta \eta} w \, dz + \frac{\delta}{\partial x} \int_{z}^{\delta \eta} u w \, dz - \frac{\delta}{\epsilon^2} w^2 \tag{6.5}
\]

in which $\epsilon^2 = \frac{D^*_0}{L^*} \ll 1$. In accordance with Chapter 2 it is assumed that $\delta = O(\epsilon^2)$.

The vertical velocity field is determined by integrating the continuity equation with respect to $z$ from the sea bed to an arbitrary level in the fluid domain. It reads (Nwogu, 1993)

\[
w = -\epsilon^2 \frac{\partial}{\partial x} \int_{-D}^{\delta \eta} u \, dz \tag{6.6}
\]

By considering the equations presented above it is evident that a given vertical distribution of the horizontal velocity enables an unambiguous description of the flow field. In qualitative agreement with Svendsen (1984) the horizontal velocity profile is presumed to be composed of a surface roller travelling with the wave celerity and an initially unknown variation of the horizontal velocity below the roller. Formally, this is written

\[
u = \begin{cases} 
  u_0, & -D \leq z < \delta(\eta - d) \\
  \frac{\epsilon}{\delta}, & \delta(\eta - d) \leq z \leq \delta\eta 
\end{cases} \tag{6.7}
\]
in which \( u_0 = u_0(x, z, t) \) is the horizontal velocity below the roller, \( d = d(x, t) \) is the thickness of the surface roller, and \( c = c(x) \) is the dimensionless wave celerity. The quantity, \( u_0^* \), which is the physical equivalent of \( u_0 \), is scaled in accordance with \( u^* \). In contrast, the celerity, \( c^* \), is normalized by the linear shallow water wave celerity \( (c^* = c \sqrt{gD_0}) \), hence indicating that \( c = O(1) \). Based on the assumption that the dimensional roller thickness, \( d^* \), is of the same order of magnitude as the surface elevation, the quantity, \( d^* \), is given by \( d^* = H^*_D d \). Since the wave celerity and the thickness of the surface roller are assumed to be known (Section 6.3.2), the derivation aims to express Equations (6.3) and (6.4) in terms of the surface elevation, \( \eta \), and the horizontal velocity component, \( u_\alpha \), where \( u_\alpha = u_0(x, z_\alpha, t) \) and \( z_\alpha = C_1 D \). It is furthermore assumed that the horizontal gradients of the still water depth are \( O(1) \). The problem is sketched in Figure 6.1.

Initially, the horizontal velocity, \( u_0 \), is expanded in a Taylor series about the sea bed, \( z = -D \). The expansion series reads

\[
    u_0 = \sum_{i=0}^{\infty} \frac{(z + D)^i}{i!} \frac{\partial^i u_0(x, -D, t)}{\partial z^i}
\]

By use of Equation (6.6) in the irrotationality condition \( (w_x = u_z) \) the vertical gradient of the horizontal velocity at the sea bed is evaluated and substituted into Equation (6.8). Substitution of this expression into Equation (6.6) allows the vertical velocity to be quantified as a function of the horizontal velocity at the bottom. In order to obtain a set of equations with improved dispersion characteristics in deeper water, the horizontal bottom velocity is replaced by the variable, \( u_\alpha \). This gives rise to the following expression

\[
    w = \begin{cases} 
    -\epsilon^2[(Du_\alpha)_z + u_\alpha z] + O(\epsilon^4) & -D \leq z < \delta(\eta - d) \\
    -\epsilon^2[(Du_\alpha)_z + \left( \frac{\epsilon}{\delta} \{z - \delta(\eta - d)\} \right)_z] + O(\delta \epsilon^2, \epsilon^4) & \delta(\eta - d) \leq z \leq \delta \eta 
    \end{cases}
\]
6.3. Inclusion of the Simplified Effect of Wave Breaking

in which terms of $O(\delta^2, \delta^2, \epsilon^4)$ are neglected.

As shown by Nwogu (1993) the horizontal velocity field is obtained by integrating the irrotationality condition from the bottom to a level inside the fluid domain using the known relationship, Equation (6.9). Formulated in terms of the horizontal velocity variable, $u_\alpha$, the velocity field is written

$$u_0 = u_\alpha + \epsilon^2 \left( \frac{1}{2} z_\alpha^2 \right) u_{axx} + (z_\alpha - z) (D u_\alpha)_{xx} + O(\epsilon^4) \quad (6.10)$$

The pressure field is quantified in a similar way by substitution of Equations (6.9) and (6.10) into Equation (6.5) and integrating, retaining terms up to $O(\delta, \epsilon^2)$. It reads

$$p = \begin{cases} 
\eta - \frac{z}{\delta} + \epsilon^2 [u_{ax2} \frac{1}{2} z^2 + (D u_{ax})_{zz}] + O(\delta \epsilon^2, \epsilon^4), & -D \leq z < \delta (\eta - d) \\
\eta - \frac{z}{\delta} + O(\delta \epsilon^2, \epsilon^4), & \delta (\eta - d) \leq z \leq \delta \eta 
\end{cases} \quad (6.11)$$

The pressure inside the surface roller contains a number of additional terms which are not shown in the equation but these turn out to be of higher order when evaluated at a level inside the surface roller.

By employment of Equations (6.9) - (6.11) the depth-integrated versions of the momentum equation and the continuity equation are derived. After a number of calculations the momentum equation can be written

$$u_{at} + \delta u_\alpha u_{ax} + \eta_x + \epsilon^2 \left( \frac{1}{2} z_\alpha^2 u_{axxt} + z_\alpha (D u_{ax})_{xx} \right)$$

$$+ \frac{[d(c - \delta u_\alpha)]_t}{D + \delta \eta} + \frac{d c x}{D + \delta \eta} + \frac{(dc)_x (c - \delta u_\alpha)}{D + \delta \eta} = O(\delta^2, \delta^2, \epsilon^4) \quad (6.12)$$

while the depth-integrated continuity equation is given by

$$\eta_t + [u_\alpha (D + \delta \eta) + d (c - \delta u_\alpha)]_x$$

$$+ \epsilon^2 \left( \frac{1}{2} z_\alpha^2 - \frac{1}{6} D^2 \right) D u_{axx} + (z_\alpha + \frac{1}{2} D) D (D u_\alpha)_{xx} = O(\delta^2, \epsilon^4) \quad (6.13)$$

In comparison with the equations of Nwogu (1993) the new equations include additional terms caused by the presence of the surface roller. The first line in the momentum equation describes the terms favoured by Nwogu while the second line contains three additional terms which describe the effect of the surface roller on the momentum balance. The first term is the temporal change of the excess volume flux in the surface roller. The second term is readily recognized as the convective acceleration of the surface roller. In addition to the last term, which can also be perceived as a kind of convective acceleration, the second term is responsible for the dissipation of wave energy caused by wave breaking. Schaffer et al. (1993) employed a relatively similar expression formulated in terms of the horizontal volume flux.

The depth-integrated continuity equation is almost identical to that proposed by Nwogu (1993). It contains an additional term which describes the spatial change of the horizontal excess volume flux in the surface roller.
The new set of equations is solved numerically by extending the model described in Chapter 3. For computational purposes it is inconvenient to nondimensionalize the governing equations by employment of a horizontal length scale in the wave motion. Consequently, Equations (6.12) and (6.13) are written in dimensional form, and subsequently made dimensionless on the basis of Equation (3.3). Using this notation the momentum equation can be written

\[ u_{at} + u_a u_{ax} + \eta_x + \frac{1}{2} z_a^2 u_{axxt} + z_a (D u_{at})_{xx} \]

\[ + \frac{[d(c - u_{a})]_t}{D + \eta} + \frac{d c c_x}{D + \eta} + \frac{(dc)_x (c - u_{a})}{D + \eta} = 0 \]  

(6.14)

while the depth-integrated continuity equation is given by

\[ \eta_t + [u_{a} (D + \eta)]_x + d(c - u_{a}) \]

\[ + \frac{([\frac{1}{2} z_a^2 - \frac{1}{6} D^2] D u_{axx} + (z_a + \frac{1}{2} D) D (D u_{a})_{xx})_x}{\eta + \frac{1}{2} D} = 0 \]  

(6.15)

For clarity it should be remarked that the magnitudes of the neglected terms are omitted. For non-breaking waves the surface roller thickness, \( d \), is equal to zero and in this case the equations become identical to those proposed by Nwogu (1993). This can be ascertained by considering Equations (3.1) and (3.2) which are given in dimensional form. The new equations have the potential to describe the evolution of waves in deeper and shallow water as well as the (relatively) rapid transformation of waves inside the surf zone. The latter part requires the variation in time and space of the surface rollers to be established.

6.3.2 Temporal Development of the Surface Roller

In comparison with the analytical surf zone models described in Section 6.2.2 the combination of a Boussinesq-type model and a method for the determination of the thickness of the surface rollers makes it possible to reproduce a variety of processes such as the fluctuating breaking point caused by random waves breaking on a beach, the important conversion of potential energy to kinetic energy in the outer transition region and the variation of the wave height, the mean water level, and the wave-induced flow throughout the surf zone.

In this thesis the thickness of the surface rollers, \( d(x,t) \), is determined heuristically by employment of the method developed by Schäffer et al. (1993). Figure 6.2a defines the

![Figure 6.2: Definition of the interface between the surface roller and the underlying wave motion - a) without shape parameter, and b) with shape parameter.](image-url)
6.3. Inclusion of the Simplified Effect of Wave Breaking

Interface between the surface roller and the underlying organized flow. It appears from the sketch that the interface is assumed to be a straight line tangent to the free water surface. The inclination of the interface is denoted \( \phi \) where \( \phi = \phi(t) \), and the hatched area designates the volume of water per unit width in the surface roller. Wave breaking initiates when the maximum slope of the wave front exceeds a given value, \( \tan \phi_b \). Although waves propagating over a horizontal bottom can be stable for local slopes as large as approximately \( \tan 27^\circ \) (Schäffer et al.) equations of the Boussinesq-type are only weakly nonlinear, hence suggesting a smaller value of \( \phi_b = O(20^\circ) \), say. Similarly, breaking ceases when the local slope of the wave front attains a smaller terminal value, denoted \( \tan \phi_0 \). In accordance with Deigaard (1989) a value of \( \phi_0 = O(10^\circ) \) has proven adequate.

Following the initiation of wave breaking, the variation in time and space of the roller thickness, \( d(x,t) \), within each surface roller is determined by prescribing an exponential variation in time of the angle, \( \phi \). It reads (Schäffer et al., 1993)

\[
\tan \phi = \tan \phi_0 + (\tan \phi_b - \tan \phi_0) \exp\left[-\ln(2) \frac{t - t_b}{t_2}\right] \tag{6.16}
\]

where \( t_b \) refers to the time of incipient breaking and \( t_2 \) is a characteristic time-scale for the development of the roller, typically chosen as a fraction of the wave period, e.g. \( t_2 = 0.2 T \).

In order to compensate for the primitive way of separating the surface roller from the underlying flow a shape parameter, denoted \( f_d \), is introduced. This is shown schematically in Figure 6.2b. Once the geometrical relationship described above has been used to establish the variation of the the surface roller thickness at a given time level, the roller thickness is multiplied by the shape parameter. It may be noted that a shape parameter of unity results in the surface roller profile displayed in Figure 6.2a.

Inside the surf zone, i.e. for non-trivial values of the surface roller thickness, a solution of Equations (6.14) and (6.15) requires the wave celerity to be determined. The experimental findings of Stive (1980) suggest a wave celerity of 20 - 30\% in excess of the linear shallow water wave celerity. On the other hand, preliminary tests have shown that the computations become unstable using \( c = 1.3\sqrt{D} \), and in the present thesis the wave celerity is therefore modelled as

\[
c = \sqrt{D} \tag{6.17}
\]

This allows the excess momentum terms caused by the presence of the surface rollers to be determined, and hence the governing equations can be solved. The next section gives a brief description of the numerical solution procedure.

6.3.3 Details of the Numerical Solution Procedure

Since Equations (6.14) and (6.15) are very similar in form to those derived by Nwogu (1993), these are solved numerically by employment of a modified version of the solution procedure described in Chapter 3. For clarity it is repeated that the computations are stepped forward in time by use of the fourth order accurate predictor-corrector method of Adams (Gear, 1971), while the spatial derivatives are approximated in accordance with Chapter 3. A single exception is made in discretizing the convective term, \( u_{\alpha}u_{\alpha x} \).
In regions of large horizontal gradients of the surface elevation, e.g. at the front of a breaking wave, the use of a fourth order accurate centred scheme for the discretization of the convective term does not ensure an accurate description of the mean water level. This is particularly pronounced in the vicinity of the breaking point and in the inner region of the surf zone where the waves resemble bores. In qualitative agreement with Sorensen et al. (1994) a second order accurate centred scheme is applied in regions of non-monotonic flow. This introduces a small amount of artificial dissipation which tends to prevent non-physical oscillations from appearing in the solution. Additionally, it has turned out to be important to discretize the convective term by quadratic upwinding in sharply varying regions. Specifically, quadratic upwinding is applied at points where $|\eta_2| > 0.3$. Analogously, fourth order accuracy is maintained in regions where $|\eta_2| < 0.3$.

In agreement with Chapter 3 the additional terms caused by wave breaking are discretized correct to the fourth order. At each time level the influence of the surface rollers is computed following the prediction of the surface elevation and the horizontal velocity component. In order to ensure a stable computation and avoid a large number of iterations no corrections are performed of the roller thickness. This is not expected to influence the accuracy of the computations significantly.

It is inevitable that the introduction of surface rollers in the computation results in reflection of wave energy. As the waves start to break a small but finite roller thickness is immediately introduced, hence causing a relatively small amount of wave reflection to occur. In agreement with the fact that the model was originally developed for spilling breakers the problem can be minimized by use of a gradual development of the surface rollers.

The reflections manifest themselves as minor oscillations significantly shorter than the primary wave lengths. Schäffer et al. eliminated the undesired reflections by use of a numerical operator slightly asymmetric in time. Clearly, this is similar to using a dissipative interface. In this report a weak dissipative interface is used to smooth the surface elevation after each time level. It is emphasized that the dissipative interface is applied at a given computational point only if the surface elevation is either a local minimum or a local maximum. As will be seen from the results presented in subsequent sections this ensures that the surf zone model produces sound physical results.

### 6.4 Inclusion of the Bottom Friction

In acknowledgment of the fact that the bathymetry of coral reefs is very rough the bed friction is incorporated into the surf zone model described in the previous sections. Since the computational model is based on the assumption of spilling wave breaking, it is clear that the energy dissipation caused by bottom friction is significantly smaller than the effect of wave breaking.

The dimensionless shear stress acting at the bottom is denoted $\tau$ and the dimensional equivalent is defined by $\tau^* = \rho^* g^* \tilde{D}_0^* \tau$. In qualitative agreement with Madsen & Warren (1984) the bed friction is approximated by

$$\tau = \frac{1}{2} f_w u_\alpha |u_\alpha|$$  \hspace{1cm} (6.18)

where $f_w$ is a wave friction parameter and $u_\alpha$ is the horizontal velocity component defined in Chapter 2. As a consequence of the fact that the present equations are formulated in terms of
6.5. Properties of the Surf Zone Model

a horizontal velocity component (rather than the horizontal volume flux) the dissipative effect
of the bed shear stress is obtained by adding the term, \( \frac{T}{D+\eta} \), to the left hand side of Equation
(6.14). This is consistent with the derivation performed in Section 6.3.1.

For simplicity the wave friction parameter, \( f_w \), is assumed to be constant within the
computational domain. The quantity, \( f_w \), is determined by employment of the relationship
proposed by Jonsson & Carlsen (1976). In dimensionless form the relationship reads

\[
\begin{align*}
\frac{1}{4\sqrt{f_w}} + \log\frac{1}{4\sqrt{f_w}} &= -0.08 + \log \frac{x_b}{k_N}, \quad \frac{x_b}{k_N} < 1.57 \\
&\quad \frac{x_b}{k_N} \geq 1.57
\end{align*}
\]

(6.19)

where \( x_b^* = D_0^* x_b \) denotes the horizontal particle amplitude at the bottom and \( k_N^* = D_0^* k_N \)
denotes the Nikuradse roughness. In accordance with the relatively crude approximations
made already the horizontal particle amplitude at the bottom is calculated by use of linear
shallow water wave theory, hence yielding the expression

\[
x_b = \frac{H_0 T}{4\pi}
\]

(6.20)

where \( H_0 \) and \( T \), respectively, refer to the wave height and the wave period of the incident
wave field. By considering a given incident wave field it is evident that the specification of the
Nikuradse roughness allows the bed friction to be determined, and consequently, the new surf
zone model is complete.

In the following sections computational examples are given. These illustrate that the
new equations produce relatively accurate results inside the surf zone.

6.5 Properties of the Surf Zone Model

6.5.1 Computation of the Wave Height Variation and the Mean Water
Level

The objective of the computations is to quantify the mean water level and the wave height
variation inside the surf zone. Clearly, the mean water level is very susceptible to errors
and inconsistencies in the mass balance, and it is therefore important to ensure that the
mass is conserved. In Section 3.5 a description was given of the sponge layer method used
to absorb the outgoing wave field. The effect of the sponge layers was incorporated into
the depth-integrated continuity and momentum equations by employment of two functions,
\( f_1(x,t) \) and \( f_2(x,t) \), which are responsible for the absorption of momentum and mass, see
Equations (3.36) and (3.37), respectively. Since the damping term appearing in the depth-
integrated continuity equation is given by \( f_2(x,t) = \gamma(x) \eta(x,t) \) where \( \gamma(x) \) is determined by
Equation (3.35), it is evident that the sponge layers tend to eliminate differences between the
instantaneous surface elevation and the still water level. This does not have any implications
on the mass balance provided the difference between the mean water level and the still water
level is insignificant. Inside the surf zone the wave-induced setup is substantial, and in this case the function, \( f_2(x, t) \), cannot be applied directly. It can be speculated that a replacement of \( f_2(x, t) = \gamma(x) \eta(x, t) \) by \( f_2(x, t) = \gamma(x) [\eta(x, t) - \bar{\eta}(x)] \) where \( \bar{\eta}(x) \) is the mean water level at a given location in space, would permit a satisfactory absorption of the outgoing wave field and at the same time conserve the mass (Sorensen, 1996). Although approximate values of the current mean water level can be estimated as the computations proceed in time the method has turned out to be unsuccessful. The problem is overcome here by the requirement that \( f_2(x, t) = 0 \), see Equation (3.37). Consequently, \( f_2(x, t) \) is set equal to zero in the remaining part of the thesis. This ensures that the mass is conserved but it also implies that an increased number of sponge layer nodes must be applied in order to avoid a substantial amount of reflection.

In continuation of the shoaling example given in Section 4.6 the surf zone model is used to study the transformation of regular waves of initially constant form breaking on a plane and gentle slope. Since a comparison is made with the experimental results of Hansen & Svendsen (1979), the bottom topography is defined by

\[
D(x) = \begin{cases} 
1.00, & x \in [0.00, 11.1] \\
1.00 - \frac{x - 11.1}{43.26}, & x \in [11.1, 41.7] \\
0.108, & x \in [41.7, 55.6]
\end{cases}
\] (6.21)

For completeness it is repeated that a still water depth of \( D = 1.00 \) corresponds to a dimensional water depth given by \( D_0 = 0.360 \) m. From Equation (6.21) it appears that the beach slope is equal to 1 : 34.26.

The computations performed in this section are divided into two parts. The first part, denoted test 1, describes the calibration of the surf zone model. On the basis of an existing set of laboratory measurements an optimum set of wave breaking parameters is determined. The experimental data set used to calibrate the model is given by Hansen & Svendsen (1979) and it carries the identification number 101101. The second part describes the verification of the surf zone model. Using the fixed set of wave breaking parameters determined in the calibration the computational results are compared with two different sets of laboratory measurements of Hansen & Svendsen (1979). The first example considers the data set denoted 061091 while the second example is based on the data set denoted 041041. These are referred to as tests 2 and 3, respectively.

<table>
<thead>
<tr>
<th>Test</th>
<th>Purpose</th>
<th>( H_0 )</th>
<th>( T )</th>
<th>( H_0^2 / L^2 )</th>
<th>( D_0^2 / L^2 )</th>
<th>Breaker type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Calibration</td>
<td>0.264</td>
<td>5.22</td>
<td>6.66%</td>
<td>0.252</td>
<td>Spilling</td>
</tr>
<tr>
<td>2</td>
<td>Verification</td>
<td>0.248</td>
<td>8.71</td>
<td>3.08%</td>
<td>0.124</td>
<td>Plunging</td>
</tr>
<tr>
<td>3</td>
<td>Verification</td>
<td>0.108</td>
<td>13.0</td>
<td>0.855%</td>
<td>0.0793</td>
<td>Plunging</td>
</tr>
</tbody>
</table>

Table 6.1: Incident wave conditions calculated from the data of Hansen & Svendsen (1979).
In agreement with Hansen & Svendsen (1979) the incident wave conditions depicted in Table 6.1 are used as input to the computational model. It appears that the surf zone model is calibrated by employment of a data set which describes the transformation of a regular wave field of large steepness in initially intermediate depth water. According to Galvin (1968) the waves break as spilling breakers, hence indicating that the calibration is carried out on a sound theoretical basis. From Table 6.1 it is apparent that the calibrated model is compared with experimental data which represent longer (regular) waves of lower steepness. Since the waves break as plunging breakers (Galvin, 1968), the model can not be expected to produce accurate results in the outer transition region.

The incoming wave field is generated internally by the Fourier method described in Section 3.8 using 10 wave components and a mean mass transport velocity of \( c_s = 0.00 \). This is consistent with the fact that the experiments were carried out in a closed wave flume. As mentioned in Section 4.6 the wave heights generated by Hansen & Svendsen (1979) did not quite reach their specified values. The values depicted in Table 6.1 are therefore obtained by averaging the measured wave heights at 20 locations close to the toe of the slope in the part of the flume of constant water depth. For completeness it is mentioned that the incident still water depth is given by \( D = 1.00 \).

In order to ensure a satisfactory absorption of the outgoing wave field dissipative sponge layers are located adjacent to the boundaries. The left sponge layer covers 50 nodes while the right sponge layer consists of 100 nodes. Both of these are accompanied by a totally reflecting condition which minimizes the flow of mass through the computational boundary (Section 3.4.3). The fluid domain is discretized by 500 nodes using a grid spacing of \( \Delta x = 0.111 \), and the surf zone model is run at a Courant number of \( Cr = 0.500 \). No bottom friction is applied, thus indicating that \( f_w = 0.00 \). Additionally, it is noted that the time frame of the computation is 250 wave periods. As the computation of the mean water level and the wave height variation is carried out during 20 wave periods immediately prior to program termination, this ensures that transient effects have died out.

In the first case, denoted test 1, an optimum set of wave breaking parameters is determined by trial and error. The final values of the parameters are \( \tan \phi_b = 0.45 \), \( f_d = 2.0 \), \( t_2 = T \), and \( \tan \phi_0 = 0.15 \). In comparison with the wave breaking parameters suggested by Schäffer et al. (1993) it appears that the chosen time scale for the development of the surface roller is significantly larger in the present model. This is a consequence of the fact that the model requires a relatively slow development of the surface roller in order to ensure a smooth solution. Moreover, it is noted that the parameter describing the initiation of wave breaking is larger than the value suggested by Schäffer et al. \( (\tan 20^\circ = 0.364) \). The result of the calibration is depicted in Figures 6.3 and 6.4.

In addition to the experimental data of Hansen & Svendsen (1979) Figure 6.3 shows the computed variation of the relative mean wave height as a function of the relative still water depth. The graph illustrates that the surf zone model generally predicts the wave height variation relatively accurately but there is a tendency to underestimate the wave height in the vicinity of the breaking point \( (D \approx 0.4) \). Inside the surf zone the wave height decay agrees well with the experimental data. It may be noted that the vertical line appearing in the lower left corner of the graph is caused by the presence of the dissipative sponge layer located in the most shallow part of the computational domain.

Figure 6.4 displays the computed and the measured variation of the mean water level (MWL) as a function of the relative still water depth. The graph shows that the model predicts very accurately the shape of the MWL before breaking. Inside the surf zone the wave-induced
CHAPTER 6. DISSIPATION OF WAVE ENERGY

Figure 6.3: Variation of the wave height as a function of the relative depth. Computational data: $H_0 = 0.264$, $T = 5.22$, $c_s = 0.00$, $M = 10$, $II = 500$, $\Delta x = 0.111$, $Cr = 0.500$, $ISL = 50$, $ISR = 100$, $\frac{1}{T} \in [0, 250]$, $f_d = 2.0$, $t_2 = T$, $\tan \phi_0 = 0.45$, $\tan \phi_0 = 0.15$, and $f_w = 0.00$. Experimental data: Hansen & Svendsen (1979), test 101101.

Figure 6.4: Variation of the mean water level as a function of the relative depth. Computational data: $H_0 = 0.264$, $T = 5.22$, $c_s = 0.00$, $M = 10$, $II = 500$, $\Delta x = 0.111$, $Cr = 0.500$, $ISL = 50$, $ISR = 100$, $\frac{1}{T} \in [0, 250]$, $f_d = 2.0$, $t_2 = T$, $\tan \phi_0 = 0.45$, $\tan \phi_0 = 0.15$, and $f_w = 0.00$. Experimental data: Hansen & Svendsen (1979), test 101101.
setup compares surprisingly well with the experimental data considering the crude physical
description. By considering both the computed wave height variation and the computed mean
water level it appears that the setup starts at a noticeable distance inshore of the breaking
point. As remarked by Schäffer et al. (1993) this indicates that the model reproduces the
important conversion of potential energy to kinetic energy in the outer transition region. On
the other hand, the graph exhibits a general vertical shift between the measured and the
computed mean water level. This aspect is expected and is physically correct. As the model
conserves mass a local increase in the mean water level must be balanced by a corresponding
depression in the remaining part of the fluid domain. In the physical experiment the horizontal
extent of the regions upstream and downstream of the breaking point are different from those
in the computational example. Consequently, the two curves are expected to exhibit a general
vertical shift. It is important to emphasize that the surf zone model should be judged on its
ability to reproduce the correct shape of the mean water level rather than the exact position
relative to the measurements. Additionally, it is mentioned that the vertical line located at
$D = 0.108$ describes the wave-induced setup inside the sponge layers. Consequently, it is
physically irrelevant.

On the basis of the wave breaking parameters determined above, the surf zone model is
verified by comparison with two different sets of laboratory measurements. In the following,
test 2 is considered, see Table 6.1. Figures 6.5 and 6.6 show the computed and the measured
variation of the wave height and the mean water level, respectively. It appears from Figure
6.5 that the mean wave height is slightly overestimated before breaking. By considering the
measured wave height variation close to the toe of the slope, after its apparent oscillations
have been removed, it seems evident that the experimental wave height is smaller than unity.
This indicates that the incident wave height used as input to the computational model is
somewhat larger than the true mean wave height in the experiment. Clearly, this results in
a minor discrepancy between the measured and the computed wave height variation. Owing
to the fact that the breaking point predicted by the surf zone model is located in relatively
deeper water than dictated by the measurements, the computed wave height is underestimated
in the vicinity of the measured breaking point ($D \approx 0.40$). Further inside the surf zone the
computed wave height decay compares favourably with the experiments. It is speculated that
the slow development of the surface roller immediately after breaking tends to counteract the
effect of the poor prediction of the breaking point.

Figure 6.6 shows the corresponding mean water levels. It appears that the depression
of the mean water level outside the breaking point is modelled very well. By considering
the computed variation of both the wave height and the mean water level it is evident that
the mean water level remains approximately constant in the outer region. Clearly, this is a
major strength of the surf zone model. In the inner region of the surf zone the gradient of the
mean water level is simulated relatively accurately but the wave-induced setup starts further
seawards than indicated by the measurements. This is attributed to the inaccurate prediction
of the breaking point.

As shown in Table 6.1 the second part of the model verification, denoted test 3, considers
the transformation of a relatively long wave field of small initial wave steepness. In accordance
with the previous examples Figure 6.7 shows the computed and the measured relative mean
wave height as a function of the relative still water depth. It appears that the model generally
predicts the wave height variation very well, although there is a tendency to underestimate the
wave height in the vicinity of the breaking point. In this case the breaking point is modelled
accurately, and the wave height decay inside the surf zone compares well with the laboratory
CHAPTER 6. DISSIPATION OF WAVE ENERGY

Figure 6.5: Variation of the wave height as a function of the relative depth. Computational data: $H_0 = 0.248$, $T = 8.71$, $c_s = 0.00$, $M = 10$, $H = 500$, $\Delta x = 0.111$, $Cr = 0.500$, $ISL = 50$, $ISR = 100$, $\frac{L}{h} \in [0, 250]$, $f_d = 2.0$, $t_2 = T$, $\tan \phi_0 = 0.45$, $\tan \phi_0 = 0.15$, and $f_w = 0.00$. Experimental data: Hansen & Svendsen (1979), test 061091.

Figure 6.6: Variation of the mean water level as a function of the relative depth. Computational data: $H_0 = 0.248$, $T = 8.71$, $c_s = 0.00$, $M = 10$, $H = 500$, $\Delta x = 0.111$, $Cr = 0.500$, $ISL = 50$, $ISR = 100$, $\frac{L}{h} \in [0, 250]$, $f_d = 2.0$, $t_2 = T$, $\tan \phi_0 = 0.45$, $\tan \phi_0 = 0.15$, and $f_w = 0.00$. Experimental data: Hansen & Svendsen (1979), test 061091.
Figure 6.7: Variation of the wave height as a function of the relative depth. Computational data: $H_0 = 0.108$, $T = 13.0$, $c_s = 0.00$, $M = 10$, $II = 500$, $\Delta x = 0.111$, $Cr = 0.500$, $ISL = 50$, $ISR = 100$, $\frac{t}{T} \in [0, 0.250]$, $f_d = 2.0$, $t_2 = T$, $\tan \phi_b = 0.45$, $\tan \phi_0 = 0.15$, and $f_w = 0.00$. Experimental data: Hansen & Svendsen (1979), test 041041.

Figure 6.8: Variation of the mean water level as a function of the relative depth. Computational data: $H_0 = 0.108$, $T = 13.0$, $c_s = 0.00$, $M = 10$, $II = 500$, $\Delta x = 0.111$, $Cr = 0.500$, $ISL = 50$, $ISR = 100$, $\frac{t}{T} \in [0, 0.250]$, $f_d = 2.0$, $t_2 = T$, $\tan \phi_b = 0.45$, $\tan \phi_0 = 0.15$, and $f_w = 0.00$. Experimental data: Hansen & Svendsen (1979), test 041041.
Figure 6.9: Trace of surface rollers. Computational data: $H_0 = 0.108$, $T = 13.0$, $c_s = 0.00$, $M = 10$, $II = 500$, $\Delta x = 0.111$, $Cr = 0.500$, $ISL = 50$, $ISR = 100$, $f_d = 2.0$, $t_2 = T$, $\tan \phi_b = 0.45$, $\tan \phi_0 = 0.15$, and $f_w = 0.00$.

Figure 6.10: Surface elevation depicted at times, $t = 291$ (solid) and $t = 312$ (dashed). Computational data: $H_0 = 0.108$, $T = 13.0$, $c_s = 0.00$, $M = 10$, $II = 500$, $\Delta x = 0.111$, $Cr = 0.500$, $ISL = 50$, $ISR = 100$, $f_d = 2.0$, $t_2 = T$, $\tan \phi_b = 0.45$, $\tan \phi_0 = 0.15$, and $f_w = 0.00$. Note: The thick lines to the left and the right, respectively, designate the last and the first sponge layer node.
Figure 6.8 depicts the measured and the computed variation of the mean water level. The graph illustrates that the surf zone model simulates accurately the depression of the mean water level outside the breaking point. Inside the surf zone the gradient of the mean water level is somewhat underestimated but the model produces roughly the same setup as indicated by the measurements although it happens at a somewhat larger horizontal scale. Regardless of the fact that the surf zone model was originally developed for spilling breakers the present example shows a pronounced agreement with the laboratory measurements of Hansen & Svendsen (1979). This is mainly due to the fact that the considered wave field is quite long, indicating that the horizontal gradients of the dependent variables are relatively small (with the exception of the upper crest region).

In addition to the examples described above, Figure 6.9 depicts the trace of a few detected surface rollers relating to test 3. The graph shows the location in space of the surface rollers as a function of time. Because of severe computer memory constraints the graph depicts only a fraction of the actual number of data points in each surface roller. The graph mainly demonstrates that the detection of surface rollers works satisfactorily. The computations have not reached steady state and considering the envelope of the surface rollers it is evident that the initial transient influences the detection of the surface rollers shown in the graph. Since the local speed of the surface rollers is given by \( \frac{\partial x}{\partial t} \), it appears that these slow down as the surf zone is traversed.

Finally, Figure 6.10 shows the computed surface elevation at two different time levels after the start of the computation. The graph provides little quantitative information but it visualizes the process of wave shoaling and breaking. The thick vertical lines to the left and the right designate the last and the first sponge layer node, respectively. By considering the surface elevation inside the left sponge layer it is apparent that a small amount of wave energy is reflected as the waves climb the sloping beach. Since the surface elevation is depicted at times, \( t = 291 \) and \( t = 312 \), the computations have not become stationary, and the mean water level therefore deviates slightly from that shown in Figure 6.8.

### 6.5.2 The Influence of the Wave Breaking Parameters on the Results

The surf zone model incorporates four unknown parameters which determine the initiation and the cessation of wave breaking as well as the characteristics of the breaking process. In this section it is verified that the computed results are relatively insensitive to the choice of these although they all influence the details of the solution.

<table>
<thead>
<tr>
<th>Parameter set</th>
<th>( \tan \phi_b )</th>
<th>( \frac{\partial x}{\partial t} )</th>
<th>( f_d )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.45</td>
<td>1.0</td>
<td>2.0</td>
</tr>
<tr>
<td>2</td>
<td>0.30</td>
<td>1.0</td>
<td>2.0</td>
</tr>
<tr>
<td>3</td>
<td>0.45</td>
<td>0.1</td>
<td>2.0</td>
</tr>
<tr>
<td>4</td>
<td>0.45</td>
<td>1.0</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Table 6.2: Various sets of breaking parameters. In all tests \( \tan \phi_0 = 0.15 \).
CHAPTER 6. DISSIPATION OF WAVE ENERGY

Figure 6.11: Variation of the wave height within the computational domain. Data: $H_0 = 0.108$, $T = 13.0$, $c_s = 0.00$, $M = 10$, $II = 500$, $\Delta x = 0.111$, $Cr = 0.500$, $ISL = 50$, $ISR = 100$, $\frac{f}{f} \in [0, 250]$, and $f_w = 0.00$. The breaking parameters are given in Table 6.2. Experimental data: Hansen & Svendsen (1979), test 041041.

Figure 6.12: Variation of the mean water within the computational domain. Data: $H_0 = 0.108$, $T = 13.0$, $c_s = 0.00$, $M = 10$, $II = 500$, $\Delta x = 0.111$, $Cr = 0.500$, $ISL = 50$, $ISR = 100$, $\frac{f}{f} \in [0, 250]$, and $f_w = 0.00$. The breaking parameters are given in Table 6.2. Experimental data: Hansen & Svendsen (1979), test 041041.
The computations are performed on the basis of the data described in Section 6.5.1. As shown in Table 6.2 four different sets of wave breaking parameters are considered. The first set is identical to that used in test 3 while each of the succeeding parameter sets considers a deviation of a single variable. Clearly, this makes it possible to assess the relative importance of each wave breaking parameter as well as their effect on the computations inside the surf zone. It is noted that a constant value is assigned to the variable governing the cessation of wave breaking, \( \tan \phi_0 \). This is justified by the fact that surf zone waves normally continue to break as they climb a beach of a uniform slope.

By employment of the wave breaking parameters given in Table 6.2 the surf zone model is used to predict the variation of the mean wave height and the mean water level as a function of the horizontal coordinate, \( x \). Figure 6.11 shows the computed and the measured wave height variation while Figure 6.12 displays the corresponding mean water levels. The thick vertical lines to the left and the right, respectively, designate the last and the first sponge layer node. It appears from Figure 6.12 that the right sponge layer causes a relatively large setup in the downstream part of the flume. As mentioned previously this has no physical significance.

In the following the results obtained by use of the second parameter set are compared with those of the default set of wave breaking parameters. It appears from Figure 6.11 that a decrease in the maximum slope of the surface elevation, \( \tan \phi_b \), gives rise to a breaking point located in relatively deeper water and a somewhat slower decay of the wave height inside the surf zone. Somewhat surprisingly, Figure 6.12 shows that the location in space at which the mean water level starts to increase is scarcely influenced by the location of the breaking point. Although the gradient of the mean water level remains practically unchanged immediately after breaking the wave-induced setup is significantly smaller. As explained in Section 6.5.1 the conservation of mass results in a general vertical shift between the two curves.

The third set of wave breaking parameters considers an order of magnitude reduction of the time-scale determining the temporal development of the surface roller (Table 6.2). Figure 6.11 shows that the location of the breaking point is unaffected by the choice of \( t_2 \). Following incipient breaking the wave height decreases more rapidly than indicated by the curve describing the default set of parameters. Similarly, Figure 6.12 illustrates that a reduction of \( t_2 \) results in a decrease in the wave-induced setup. Additionally, it seems evident that a more rapid decrease in the wave height does not change the location at which the mean water level starts to increase.

As shown in Table 6.2 the fourth set of wave breaking parameters incorporates a large reduction of the shape parameter, \( f_d \). Figure 6.11 shows that the computed wave height variation is very similar to the curve describing the default parameters. It appears that a decrease in the shape parameter results in a somewhat larger wave height after breaking. Analogously, Figure 6.12 illustrates that the gradient of the wave-induced setup is somewhat smaller than indicated by the curve which describes the default parameter set. This results in an underestimation of the mean water level inside the surf zone.

### 6.5.3 Computation of the Undertow

It is evident that the surf zone model constitutes a relatively accurate design tool for the prediction of the mean water level and the wave height inside the surf zone. In this section the wave-induced return current is investigated, and a comparison is made with the computational results of Madsen et al. (1994) as well as the laboratory experiments of Hansen & Svendsen
The instantaneous flux of mass through any vertical cross section, $Q(x,t)$, is defined as the integral of the horizontal velocity component, $u(x,z,t)$, from the bottom to the free water surface. By considering the depth-integrated continuity equation, see Equation (6.15), it can be ascertained that the instantaneous mass flux can be written

$$Q(x,t) = [D + \eta - d + D^2 D_x (C_1 + \frac{1}{2})] u_o + 2 D^2 D_x (C_1 + \frac{1}{2}) u_{ox}$$

$$+ D^3 (\frac{1}{2} C_1^2 + C_1 + \frac{1}{3}) u_{ox} + c d$$

In order to compare the results of the computational model with those of Madsen et al. (1994) the wave-induced return current, denoted the undertow, is defined as the mean over a wave period of the instantaneous return current, averaged vertically below the roller. It is given by the expression

$$U = \frac{1}{T} \int_{t}^{t+T} \frac{Q - c d}{D + \eta - d} dt$$

in which $U$ denotes the undertow. For clarity it is mentioned that the return current in practice is a function of the vertical coordinate.

Since the laboratory measurements of Hansen & Svendsen (1984) essentially are a continuation of their earlier work (Hansen & Svendsen, 1979), the computations given below are based on the bathymetry and the data described in Section 6.5.1. The incident regular wave field is characterized by an absolute wave period, $T = 10.4$, and an incident wave height, $H_0 = 0.333$. By employment of the Fourier method described in Section 3.8 it can be shown that $\frac{H_0}{L_0} = 3.31\%$ and $\frac{D_0}{L_0} = 0.0993$, hence indicating that a relatively long wave of finite height is considered. According to Galvin (1968) the waves break as spilling breakers. An optimum set of wave breaking parameters is determined by trial and error. These are given by $f_d = 1.5$, $t_2 = T$, $\tan \phi_b = 0.45$, and $\tan \phi_0 = 0.15$, while the wave friction parameter, $f_w$, is set equal to zero.

Figure 6.13 depicts the computed wave height variation as a function of the horizontal coordinate, $x$. In addition to the experimental data of Hansen & Svendsen (1984) the wave height variation determined by Madsen et al. (1994) is shown. It is noted that these are normalized with the incident wave height, $H_0$. In comparison with the measurements it appears that the present model predicts the wave height variation caused by shoaling quite accurately. Although the prediction of the breaking point is inaccurate, the initially slow development of the surface roller results in a reasonable agreement with the measurements further inside the surf zone. The model of Madsen et al. underestimates the wave height before breaking but it predicts a more realistic variation of the wave height inside the surf zone.

Figure 6.14 shows the variation of the undertow as a function of the horizontal coordinate. The graph depicts the experimental results of Hansen & Svendsen (1984) as well as the undertow computed by both the present model, see Equation (6.23), and the surf zone model of Madsen et al. (1994). In accordance with the measurements both models predict a similar increase in the return flow as the waves shoal. As a consequence of the fact that the present model underestimates the breaking point the return flow is significantly underestimated in the
6.5. Properties of the Surf Zone Model

Figure 6.13: Variation of the wave height inside the computational domain. Data: $H_0 = 0.333$, $T = 10.4$, $c_s = 0.00$, $M = 10$, $H_1 = 500$, $\Delta x = 0.111$, $C_r = 0.500$, $ISL = 50$, $ISR = 100$, $\frac{t}{T} \in [0, 250]$, $f_d = 1.5$, $t_2 = T$, $\tan \phi_0 = 0.45$, $\tan \phi_0 = 0.15$, and $f_w = 0.00$.

Figure 6.14: Variation of the undertow inside the computational domain. Data: $H_0 = 0.333$, $T = 10.4$, $c_s = 0.00$, $M = 10$, $H_1 = 500$, $\Delta x = 0.111$, $C_r = 0.500$, $ISL = 50$, $ISR = 100$, $\frac{t}{T} \in [0, 250]$, $f_d = 1.5$, $t_2 = T$, $\tan \phi_0 = 0.45$, $\tan \phi_0 = 0.15$, and $f_w = 0.00$. 
approximate interval, \( x \in [30.0, 35.0] \). Clearly, the model of Madsen et al. provides a relatively better estimate of the undertow in this region. It is remarkable that both models predict a maximum return flow noticeably far inside the surf zone. This is in excellent agreement with the measurements as well as the ideas of Dally & Brown (1996), see Section 6.2.2. The result indicates that the model predicts the correct variation in space of the surface roller thickness.

The example given above concludes the verification of regular waves breaking on a gentle slope. In the next chapter a laboratory experiment is described. The experimental results verify that the model is capable of simulating reasonably accurately the mean water level of periodic waves breaking on the seaward face of a coral reef. The reef incorporates significantly steeper slopes than that considered in the present chapter.
Chapter 7

Application to a Submerged Coral Reef

7.1 General

It is evident that the surf zone model predicts accurately both the mean water level and the wave height variation of periodic waves breaking on a plane and gentle slope. In this chapter the model is applied to a submerged coral reef which incorporates relatively large bed slopes. The computational results are compared with laboratory measurements obtained in an outdoors wave basin.

In addition to an introduction describing some investigations related to coral reefs, the first part is concerned with the laboratory experiment. A description is given of both the laboratory facility and the equipment used as well as the procedures involved in the experiment. The bed profile represents a fringing coral reef located on the coast of Guam. Six different incident wave conditions were considered. In order to improve the accuracy of the results each experiment was repeated four times. Because of the lack of proper data acquisition software only the mean water level was measured, and hence the experiments can verify the surf zone model only partially.

In the second part a comparison is made between results computed by the surf zone model and those acquired in the experiments. As in the previous chapter the model is calibrated using a single set of experimental data and subsequently tested against five other sets of measurements. It is shown that the surf zone model predicts the mean water level reasonably accurately.

7.2 Introduction to Coral Reefs

Coral reefs are composed of a large number of dead and alive corals. Since corals depend crucially upon the amount of light received from the sun as well as on the consumption of nutrients and larvae, their existence is restricted to the upper part of the water column. The light intensity is largest in the vicinity of the water surface and corals therefore tend to grow
upwards on the formation of dead corals. A relatively horizontal reef top is often formed which is located either below or above the mean water level depending on the tide level and the wave action. At the seaward sloping part of a coral reef the water depth often increases rapidly in comparison with a dominating wave length. It is not uncommon to observe that the bed level close to the reef edge is elevated slightly relative to the adjacent reef top. During low tide this part, denoted the reef rim, serves as an important buffer protecting the corals from excessive damage caused by direct exposure to the sun. A practical range of coral reef bathymetries can be found in Gourlay (1996b).

Depending on the tide level and the characteristics of the incident wave field wave breaking occurs close to the reef edge. Evidently, this results in a setup which allows larger waves to propagate onto the top. Gourlay (1994) reported that up to 95% of the offshore wave energy is dissipated when waves break at the reef face. Moreover, it was noted that the maximum wave height to mean water depth ratio on the reef top is as low as 0.55 (see also Hardy et al., 1990, 1991). This value is significantly lower than that normally used in coastal engineering.

Essentially, there are two kinds of reefs. Both of these are relatively rough. A reef which fronts an island or a continental land mass is denoted a fringing reef while a reef located in the open ocean is termed a platform reef. The conservation of mass implies that the net mass transport across a two-dimensional fringing reef be equal to zero. In contrast, various factors such as the tidal and the wave-induced currents may induce a net mass flow over a submerged platform reef. For further information on coral reefs reference is made to Gourlay (1993, 1996a).

Tait (1972) employed the method of Bowen et al. (1968) to quantify the setup caused by waves breaking at the face of a fringing reef. Since the incoming wave field was presumed to be narrow-banded, the derivation of the setup was based on linear wave theory (valid for monochromatic waves). By the assumption that the wave height depends linearly on the mean water depth the horizontal component of the depth-integrated momentum equation gave an expression for the reef top setup as a function of the still water depth at both the reef top and the breaking point. In the case of a reef top level coinciding with the still water level a comparison was made with measurements of Bowen et al. (1968) showing that the analytical model predicts the wave setup qualitatively correctly.

Gerritsen (1980) developed a semi-empirical model for the variation of the wave height and the wave-induced setup of waves propagating onto a shallow coastal reef. Apart from the fact that the effect of the bottom friction was included in the formulation the model was similar to the analytical model of Battjes & Janssen (1978). By employment of time series of the surface elevation measured offshore the model was used to compute the setup and the root-mean-square wave height at the reef. The results compared reasonably with experiments.

Laboratory measurements of both regular and irregular waves breaking at the face of a fringing coral reef located on the coast of Guam were conducted by Seelig (1983). The considered bed profile incorporates a relatively steep reef face and an adjacent reef crest submerged below the still water level. Various incident wave conditions were considered and the maximum wave-induced setup was measured in a closed lagoon inshore of the reef crest.

In a closed wave flume Nelson & Lesleigher (1985) studied the transformation of waves breaking at the edge of a submerged coral reef. The bathymetry of the reef resembled a vertical step with the reef top located in shallow water. For the wave periods under consideration this gave rise to a substantial amount of undesired reflection, and the incident wave field was therefore generated in short bursts, hence implying that a stationary situation was not reached. The tests showed that approximately 25% of the incident wave energy was dissipated within the first shallow water wave length relative to the reef edge. Furthermore, it appeared
that the broken waves reformed into oscillatory waves at a distance of four to seven shallow water wave lengths from the reef edge.

Massel (1993a) extended the classical mild-slope equation valid for regular waves of small amplitude propagating over a slowly varying bottom topography. Based on the Galerkin eigen-function method an extended refraction-diffraction equation was derived which incorporates the effect of a bed consisting of substantial variations in the water depth within a wave length. In addition to the freely propagating waves the resulting equation involves the non-propagating (evanescent) modes. The extended equation was used to study the wave height variation of regular non-breaking waves of initially small amplitude as they propagate across a submerged coral reef consisting of rapid bed variations.

Massel (1991) included the simplified effect of wave breaking in the formulation (Massel, 1993a) using the analogy of a hydraulic jump. This was done by introduction of a damping factor in the governing equation. A regular wave train, a narrow-banded irregular wave train, and a fully irregular wave field determined by a Rayleigh distribution were used as input to the model. The computed wave height variations were compared with experimental data obtained on various beach profiles, and these showed a good agreement.

Based on the momentum equation and the extended refraction-diffraction equation Massel (1992) studied the variation of the root-mean-square wave height and the wave setup of a narrow-banded irregular wave field breaking on a steep reef slope. The energy dissipation caused by wave breaking was modelled by employment of the Rayleigh distribution and the expression for the energy loss in a hydraulic jump. Similarly, the energy dissipation caused by the bed friction was described in terms of a friction factor and the horizontal velocity amplitude at the bed. The momentum equation and the extended refraction-diffraction equation were solved simultaneously yielding the root-mean-square wave height and the wave-induced setup at each water depth. The model was used to study the transformation of an irregular wave field climbing a steep reef face. At a fixed reef top location the predicted setup was compared with experimental data, showing a reasonable agreement.

Massel (1993b) used the extended refraction-diffraction equation to quantify the wave height variation of a narrow-banded irregular wave field interacting with a three-dimensional conical reef which extends above the mean water level. Energy dissipation caused by wave breaking and bottom friction was modelled using the method described by Massel (1992). A fully reflecting condition was applied at the edge of the conical reef while waves radiating away from the reef were absorbed on the basis of the Sommerfeld radiation condition. A numerical solution of the governing equation showed that at larger distances from the reef the wave height distribution is determined mainly by refraction and diffraction of the incident wave field, whereas in the vicinity of the shoreline wave breaking and bottom friction are the predominant factors. The model was not verified against experiments. Essentially, Massel (1994) applied the model using measured wave data as well as some idealized data for the bathymetry of a real coral reef submerged below the mean water level.

7.3 Description of the Laboratory Experiment

7.3.1 The Wave Basin Facility

A laboratory experiment was conducted in an outdoors wave basin, 32.0 m long and 6.00 m wide, using a still water depth smaller than 0.500 m. The floor of the basin is made of concrete
and the bounding walls are made from concrete blocks. In order to minimize the wall friction as well as the water leakage through the walls these are sealed on the inside using a water resistant paint.

As shown in Figure 7.1 the basin is divided into three similar wave flumes, each 2.00 m wide. Although the wave flumes are generally separated by thin walls in the longitudinal direction they are connected in the upstream region close to the wave generator. The present experiment used the wave flume located in the middle of the basin.

Behind the wave generator and in the downstream region of the middle flume the waves are damped by employment of a gravel wave absorber. This ensures that the wave reflection is kept a minimum in the considered flume. Since all three flumes are connected in the upstream region, wave diffraction about the end points of the internal flume walls causes scattered wave fields to manifest themselves in the middle wave flume. It is therefore noted that the incident wave field is absorbed in the downstream region of both the first and the third wave flume using a gravel wave absorber and a sandy beach, respectively.

Since the wave basin is located outdoors, evaporation from the free surface may alter the mass balance significantly. The wave basin is equipped with an adjustable weir gate which controls the water level during operation. The weir is hinged at the bottom and the water flowing over the weir is supplied by a number of water tappings located in the bounding wall of the third wave flume. The inflow of water to the basin is controlled by a valve located on the side of the first flume. Initially, it was intended to keep the mass inside the basin constant by use of the weir gate and the inflow valve. Since the weir was very corroded, the idea was abandoned and a different approach was adopted. Several tests have revealed that the evaporation and the leakage are insignificant in comparison with the wave-induced variations in the mean water level within the relatively short time frame of each experiment. The argument is further enhanced by the fact that all experiments were carried out either in the early morning or in the late afternoon during which the evaporation was known not to be predominant. Prior to each experiment the wave basin was therefore filled to a fixed level. No alterations were made of the mass balance during each experiment.

The incident regular wave field is generated by a hydraulically actuated piston-type wave generator which is controlled by a personal computer. Figure 7.1 shows that the paddle of the wave generator is a vertical wall. Since the command signal is generated correct to the first order in the wave steepness, the free harmonics of the second order are not suppressed. Hansen & Svendsen (1979) speculated that the interaction between the free second harmonics and the principal wave components exhibits a resonant behaviour - a phenomenon which has not been observed in the present experiment. In order to ensure a sinusoidal motion of the wave paddle the paddle stroke, $S^*$, and the wave period, $T^*$, respectively, must be chosen within the intervals, $S^* \in [20\text{mm}, 100\text{mm}]$ and $T^* \in [1.00\text{s}, 2.00\text{s}]$. Additionally, Figure 7.1 shows that the origin of the horizontal coordinate, $x^*$, is equal to the mean position of the wave paddle.

### 7.3.2 Construction of the Bed Profile

The reef profile described by Seelig (1983) was built in continuation of an existing vertical faced reef model (Figure 7.1). Since the present study is concerned with the transformation of waves as they climb the reef profile, it is clear that the vertical step located in the downstream portion of the flume is of secondary importance. Figure 7.1 shows that the considered bed
Figure 7.1: Experimental setup. The hatched areas located in each end of the flume designate a gravel wave absorber.
CHAPTER 7. APPLICATION TO A SUBMERGED CORAL REEF

Table 7.1: Accuracy of the constructed bed profile. The table also shows the horizontal location of the pressure tappings relative to the mean position of the wave paddle.

<table>
<thead>
<tr>
<th>Pressure tapping ( x^* )</th>
<th>(-z_b^*) Calculated [mm]</th>
<th>(-z_b^*) Measured [mm]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2965</td>
<td>481</td>
</tr>
<tr>
<td>1</td>
<td>5170</td>
<td>290</td>
</tr>
<tr>
<td>2</td>
<td>6170</td>
<td>237</td>
</tr>
<tr>
<td>3</td>
<td>6670</td>
<td>211</td>
</tr>
<tr>
<td>4</td>
<td>7170</td>
<td>184</td>
</tr>
<tr>
<td>5</td>
<td>7319</td>
<td>173</td>
</tr>
<tr>
<td>6</td>
<td>7469</td>
<td>158</td>
</tr>
<tr>
<td>7</td>
<td>7618</td>
<td>144</td>
</tr>
<tr>
<td>8</td>
<td>7767</td>
<td>130</td>
</tr>
<tr>
<td>9</td>
<td>7917</td>
<td>116</td>
</tr>
<tr>
<td>10</td>
<td>8066</td>
<td>102</td>
</tr>
<tr>
<td>11</td>
<td>8215</td>
<td>88.0</td>
</tr>
<tr>
<td>12</td>
<td>8365</td>
<td>74.0</td>
</tr>
<tr>
<td>13</td>
<td>8514</td>
<td>60.0</td>
</tr>
<tr>
<td>14</td>
<td>8663</td>
<td>46.0</td>
</tr>
<tr>
<td>15</td>
<td>8963</td>
<td>64.0</td>
</tr>
<tr>
<td>16</td>
<td>9224</td>
<td>81.0</td>
</tr>
<tr>
<td>17</td>
<td>10524</td>
<td>81.0</td>
</tr>
<tr>
<td>18</td>
<td>11024</td>
<td>81.0</td>
</tr>
<tr>
<td>19</td>
<td>11524</td>
<td>81.0</td>
</tr>
<tr>
<td>20</td>
<td>12024</td>
<td>81.0</td>
</tr>
</tbody>
</table>

Profile incorporates local slopes much larger than that used in Chapter 6, hence indicating that the main purpose of the experiment is to verify the surf zone model on steeper slopes.

In the quiescent water state the still water depth in front of the wave paddle was given by \( D_0^* = 0.481 \) m. This corresponds to a still water depth over the vertical faced reef profile of 0.0810 m, and a depth over the reef crest equal to 0.0410 m. Similarly, it is noted that the toe of the slope is located 3.47 m from the mean position of the wave paddle. Dean & Dalrymple (1994) studied the wavemaker problem described above and presented a solution for the velocity potential (in the fluid interior) on the basis of potential theory. In contrast to linear wave theory the solution incorporates a term which represents a standing wave in front of the wave generator. It can be ascertained that the present experiment was performed at an adequate distance from the wave paddle, since the amplitude of the standing wave decreases exponentially with the distance from the mean paddle position.

In this experiment pressure forces as well as the gravitational and the inertial forces are the most significant. The bottom bathymetry was therefore constructed in accordance with
7.3. Description of the Laboratory Experiment

the Froude scale using a model scale of 1 : 50, hence implying that the offshore part of the reef profile had to be truncated in order to meet the depth limitation of the wave basin. This does not impose any constraints on the present study, since the experimental results are compared only with those of the computational model.

The bed profile was constructed of marine plywood and screwed to the side walls and the floor of the flume. Since coral reefs are relatively rough the bed was covered by 2 – 3 mm of brushed sand cement mortar. In order to avoid excessive lift forces caused by air trapped underneath the hollow bed a number of holes were drilled in the top part and in the lowest part of the profile, hence permitting water as well as air to escape should this be necessary. It is noted that the holes were sealed during operation.

Figure 7.1 illustrates that 21 pressure tappings are located in the bed. The bed level at each pressure tapping was measured using a levelling instrument. The levels are shown in Table 7.1. For clarity it is noted that the quantity, \( z^* \), denotes the vertical bed coordinate measured from the still water level in the quiescent water state. The first column designates the number of each pressure tapping while the second column describes the horizontal locations of the pressure tappings relative to the mean position of the wave paddle. Although two significant digits are normally used in this experiment the horizontal coordinates of the pressure tappings are presented with up to four significant digits. This is done in order to enable the experiment to be reproduced. The third and the fourth columns in the table describe the calculated and the measured bed level, respectively, at each location. Table 7.1 verifies that the bed was installed with a relatively high degree of accuracy, since the corresponding root-mean-square-error is equal to 2.63 mm.

7.3.3 Data Acquisition

The wave basin facility is equipped with a number of capacitance wave gauges each of which provides an analog output signal. The signals are amplified, converted to their digital form, and stored in a personal computer of the type 486-DX2 66 MHz using a commercial software package named ViewDac. By employment of a data acquisition board of the type DAS-1600 the ViewDac program is also used to generate an analog output signal which controls the wave generator. The wave period is resolved by 100 points.

It is evident that a comparison of time records of the computed and the measured surface elevation would provide a strong measure of the quality of the developed surf zone model, and it was therefore intended to measure the surface elevation at various locations along the centre line of the wave flume. An existing data acquisition program, written in the ViewDac language, was tailored to this particular application and tested in a series of trial runs. These revealed that the ViewDac package is unsuitable for the present application, since the package does not ensure that the time interval between consecutive readings is correct! Despite the fact that the developers of the ViewDac package conducted a thorough investigation they were unable to resolve the problem. One conclusion made from the study was the fact that the code developed for the present application is free of errors. Additionally, it is emphasized that the problem does not affect the generation of the command signal which determines the paddle motion.

Because of the lack of proper electronic data acquisition software it was decided to reduce the size of each experiment, and measure only the mean water level by employment of pressure tappings located in the bed (Figure 7.1). Consequently, the present study does not
provide information on the wave height variation along the wave flume and the experiment can therefore, at best, only verify the surf zone model partially.

As mentioned previously the pressure tappings are numbered from 0 to 20. The first pressure tapping is located furthest offshore in the part of the flume of constant water depth while the pressure tappings numbered 1 through 16 are placed in the constructed reef profile. The remaining pressure tappings are located permanently in the horizontal surface of the vertical faced reef model, hence explaining their somewhat arbitrary locations. Each pressure tapping is a small circular hole in the bed profile, 1.0 mm in diameter, connected to a piece of flexible tube, which has an internal diameter of 4.00 mm. The tubes from the pressure tappings are connected to a series of transparent stilling wells, denoted piezometers, which are mounted on a vertical board located close to the wave basin. The piezometers are circular with an internal diameter of 2 – 3 cm and surface tension effects are avoided by injection of a detergent. A vernier gauge mounted on a horizontal cross bar above the piezometers is used to measure the water level in each well. The vernier gauge can be read to an accuracy of 0.1 mm. Since the flexible tubes are up to approximately 40 m long, the wall friction ensures that rapid oscillations associated with the wave motion can not be distinguished visually from the mean water level in the piezometers. It is noted that the water level in each piezometer represents the mean water level at the location of the pressure tapping provided the assumption of a hydrostatic pressure distribution over the vertical is valid.

As shown in Figure 7.1 two pointer gauges are located in the upstream and the downstream region of the first wave flume. The gauges, which are made of brass, can be read to an accuracy of approximately 0.1 mm. These were used to determine the initial still water level as well as the incident wave height.

7.3.4 Generation of the Incident Wave Field

Since no time records of the surface elevation could be acquired in the experiment, it was not possible to determine the variation of the wave height as a function of the horizontal coordinate, $x^*$. This indicates that the incident wave height was not measured during each experiment. In order to ensure that the wave paddle generated the specified incident wave height a preliminary test series was conducted, in which the incident wave height was measured as a function of the paddle stroke and the wave period. It is noted that each wave height was determined as an average of eight consecutive readings, hence minimizing experimental inaccuracies. The readings were obtained by employment of the pointer gauge located in the upstream region of the first wave flume. Since the incident wave field was generated in short bursts, the measurements are not influenced by waves reflected from the downstream end of the flume. Each burst was generated after 45 minutes during which the wave generator had not been operating, and the readings are therefore not affected by the initial transient.

Table 7.2 depicts the measured mean wave height, $H_0^*$, as a function of the wave period, $T^*$, and the paddle stroke, $S^*$. Additionally, the table displays the relative error, $RE$, defined as the root-mean-square-error of the incident wave height normalized by the mean wave height. The relative errors are generally of the order 1 – 3% which is acceptable, since the wave basin is located outdoors.
7.3. Description of the Laboratory Experiment

<table>
<thead>
<tr>
<th>$T^*$</th>
<th>$S^*$</th>
<th>$H_0^*$</th>
<th>$RE$</th>
</tr>
</thead>
<tbody>
<tr>
<td>[s]</td>
<td>[mm]</td>
<td>[mm]</td>
<td>[%]</td>
</tr>
<tr>
<td>1.10</td>
<td>20.0</td>
<td>24.4</td>
<td>0.869</td>
</tr>
<tr>
<td>1.10</td>
<td>50.0</td>
<td>66.0</td>
<td>1.27</td>
</tr>
<tr>
<td>1.10</td>
<td>80.0</td>
<td>108</td>
<td>1.81</td>
</tr>
<tr>
<td>1.40</td>
<td>20.0</td>
<td>20.9</td>
<td>2.37</td>
</tr>
<tr>
<td>1.40</td>
<td>50.0</td>
<td>57.8</td>
<td>1.75</td>
</tr>
<tr>
<td>1.40</td>
<td>80.0</td>
<td>89.2</td>
<td>1.66</td>
</tr>
<tr>
<td>1.70</td>
<td>20.0</td>
<td>17.7</td>
<td>2.97</td>
</tr>
<tr>
<td>1.70</td>
<td>50.0</td>
<td>43.7</td>
<td>3.41</td>
</tr>
<tr>
<td>1.70</td>
<td>80.0</td>
<td>67.0</td>
<td>2.63</td>
</tr>
<tr>
<td>2.00</td>
<td>20.0</td>
<td>10.9</td>
<td>2.27</td>
</tr>
<tr>
<td>2.00</td>
<td>50.0</td>
<td>29.0</td>
<td>2.82</td>
</tr>
<tr>
<td>2.00</td>
<td>80.0</td>
<td>43.0</td>
<td>4.91</td>
</tr>
</tbody>
</table>

Table 7.2: Reproduction of incident wave heights as a function of the wave period and the stroke of the wave paddle. Data: $D_0^* = 0.481$ m.

7.3.5 Experimental Procedure

In order to cover a wide range of incident wave conditions six different test series were considered in the experiment (Table. 7.3). For simplicity these were chosen in accordance with the incident wave conditions presented by Seelig (1983). The data shown in the table describe regular waves of finite height in initially intermediate depth water. It is mentioned that the relative length and steepness of the incident wave field are computed on the basis of the Fourier approximation method of Rienecker & Fenton (1981) using 10 Fourier components, a still water depth of $D_0^* = 0.481$ m, and a mean transport velocity of $c^*_0 = 0.00$ m/s. According to Galvin (1968) the fifth wave condition gives rise to plunging breakers while the remaining wave conditions result in spilling breakers. It is noted that none of these breaker types are particularly pronounced.

It can be speculated that a standing wave pattern may form in the upstream part of the wave flume if twice the horizontal distance between the mean position of the wave paddle and some point located on the constructed bed profile is equal to an integral number of incident wave lengths. This may influence the readings of the mean water level, and the wave periods given in Table 7.3 are therefore chosen in order to avoid this. Since the nodal or antinodal point located on the bed profile is not clearly defined, the calculation is based on an anticipated location close to the reef crest. Similarly, it is evident that small disturbances in the transverse direction may result in a partial standing wave if twice the width of the flume is equal to an integral number of wave lengths. Although the wave length is a function of the depth the chosen wave conditions do not seem to result in a standing wave pattern.

The paddle strokes corresponding to the wave heights shown in Table 7.3 were obtained
CHAPTER 7. APPLICATION TO A SUBMERGED CORAL REEF

Table 7.3: Breaker type (from Galvin, 1968) as well as the relative length and steepness of the incident wave fields under consideration. The computation of the wave lengths is based on the Fourier method of Rienecker & Fenton (1981). Data: $D_0^* = 0.481$ m, $c_*^* = 0.00$ m/s, and $M = 10$.

by interpolation in Table 7.2. Owing to the fact that the wave basin is located outdoors each experiment was repeated four times. Prior to each experiment it was ascertained that air bubbles were not trapped in the long flexible tubes which connect the pressure tappings and the piezometers. This is a crucial aspect of the experimental procedure since air is compressible. In order to avoid a significant amount of evaporation from the free surface the experiments were performed either in the early morning or in the late afternoon during which the surrounding temperatures were roughly within 20°C to 25°C. No measurements were performed if the wind visibly distorted the surface.

As illustrated in Figure 7.1 the initial still water level in front of the wave paddle was set to $D_0^* = 0.481$ m using the upstream and the downstream pointer gauges located in the first wave flume. In order to ensure that no air bubbles were trapped inside the flexible tubes the still water level was recorded in each piezometer. The mean (in space) and the root-mean-square-error of the still water level, denoted $Z^*$ and $Z_{\text{rmse}}^*$, respectively, are shown in Table 7.4 for each experiment. It is noted that the quantity, $Z^*$, describes the water level measured from a level associated with the cross bar to which the vernier gauge is attached. The levels have no particular physical meaning but they illustrate that the still water level is reproduced from one experiment to the other with an accuracy better than 0.5 mm. Additionally, it is evident that the measured variation of the still water level is of the order 0.1 mm, hence indicating that the piezometers function correctly.

Trial runs have shown that the water level in each piezometer tends to the true mean water level (in the steady state) within approximately 20 – 30 minutes depending on the incident wave conditions. On the other hand, the true mean water level is superposed by an oscillation much longer than a typical wave period, and it is therefore clear that the initial transient influences the wave motion for a relatively longer period of time. It can be ascertained that a time frame of one hour ensures that transient effects have died out. Moreover, it is noted that the mean water level tends to a level constant in time, indicating that the mass inside the basin remains constant within the considered time frame. Consequently, in each experiment the mean water level was acquired after one hour of operation.
Since no time series of the surface elevation could be acquired, the breaking point was determined visually by studying the wave motion through a transparent wall located in the vicinity of the reef crest in the considered wave flume. The waves were assumed to break when part of the wave front became vertical. In addition to the identification numbers of each experiment Table 7.4 depicts the mean position and the root-mean-square-error of the measured breaking point. These are denoted $MB^*$ and $MB_{\text{rms}e}$, respectively. The breaking point is measured relative to the mean position of the wave paddle, and the values shown in the table are based on 10 readings. Within the same test series, the breaking point is quite accurately reproduced from one experiment to the other. Generally, the breaking point is measured to an accuracy of 1 – 2 cm which is satisfactory considering the crude experimental procedure.

<table>
<thead>
<tr>
<th>Test No.</th>
<th>No.</th>
<th>$Z^*$ [mm]</th>
<th>$Z_{\text{rms}e}$ [mm]</th>
<th>$MB^*$ [m]</th>
<th>$MB_{\text{rms}e}$ [mm]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>129.2</td>
<td>0.106</td>
<td>8.45</td>
<td>13.5</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>128.9</td>
<td>0.0944</td>
<td>8.46</td>
<td>14.3</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>128.9</td>
<td>0.0805</td>
<td>8.46</td>
<td>14.5</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>128.9</td>
<td>0.108</td>
<td>8.46</td>
<td>16.6</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>129.0</td>
<td>0.0928</td>
<td>7.83</td>
<td>15.7</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>129.1</td>
<td>0.103</td>
<td>7.83</td>
<td>21.3</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>129.2</td>
<td>0.119</td>
<td>7.83</td>
<td>17.5</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>129.0</td>
<td>0.0805</td>
<td>7.83</td>
<td>20.2</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>129.0</td>
<td>0.140</td>
<td>8.43</td>
<td>21.8</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>129.1</td>
<td>0.121</td>
<td>8.43</td>
<td>16.6</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>128.8</td>
<td>0.157</td>
<td>8.44</td>
<td>15.8</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>129.0</td>
<td>0.0700</td>
<td>8.43</td>
<td>21.0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>129.0</td>
<td>0.108</td>
<td>7.96</td>
<td>23.0</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>129.0</td>
<td>0.123</td>
<td>7.96</td>
<td>20.8</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>128.8</td>
<td>0.0921</td>
<td>7.96</td>
<td>17.7</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>128.9</td>
<td>0.0669</td>
<td>7.96</td>
<td>27.6</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>129.0</td>
<td>0.103</td>
<td>8.14</td>
<td>13.2</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>129.1</td>
<td>0.132</td>
<td>8.15</td>
<td>10.6</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>129.3</td>
<td>0.103</td>
<td>8.15</td>
<td>9.66</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>129.1</td>
<td>0.108</td>
<td>8.15</td>
<td>13.2</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>128.8</td>
<td>0.127</td>
<td>7.85</td>
<td>13.4</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>129.1</td>
<td>0.0995</td>
<td>7.85</td>
<td>13.4</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>128.8</td>
<td>0.138</td>
<td>7.85</td>
<td>11.6</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>129.0</td>
<td>0.162</td>
<td>7.85</td>
<td>12.6</td>
</tr>
</tbody>
</table>

Table 7.4: Accuracy of the measured still water level in the quiescent water state. In addition the mean position and the root-mean-square-error of the measured breaking point are shown for each experiment.
7.4 Experimental Results

In each experiment the type of wave breaking was observed. Although the observations seem to confirm the predictions of Table 7.3 it is noted that the distinction between the wave breaking regimes is relatively ambiguous.

The mean water levels measured in the test series denoted 1 – 6 are depicted in Figures 7.2 - 7.7 as a function of the horizontal coordinate, \( x^* \). Since the graphs are drawn to the same scale, a direct comparison may be performed. In addition to the constructed bed profile each figure depicts the measured breaking point, \( MB^* \), averaged within the test series. The bed profile is shown schematically in each graph, and it provides a realistic perception of the measurements.

By considering each of the Figures 7.2 - 7.7 it is apparent that the mean water level is relatively accurately reproduced from one experiment to the other. This indicates that the mass is conserved within the considered time frame and, further, that the influence of air bubbles trapped inside the flexible tubes is insignificant. It seems evident that the accuracy of the mean water level decreases as both the wave period decreases and the wave height increases.

As shown in Table 7.3 a wave period of \( T^* = 1.13 \) s and an incident wave height of \( H_0^* = 41.0 \) mm are considered in the first test series. Figure 7.2 shows that the mean water level decreases slightly as the wave field climbs the reef profile. In this case the measured breaking point is located close to the reef crest and the wave-induced setup starts at practically the same location. The graph shows that the wave-induced setup gives rise to a corresponding depression in the mean water level in the upstream part of the flume (including the flumes denoted 1 and 3, see Figure 7.1).

In the second test the wave period of the incident wave field is given by \( T^* = 1.13 \) s while the wave height is given by \( H_0^* = 94.0 \) mm. This corresponds to an incident wave field of large steepness in initially intermediate depth water (Table 7.3). In accordance with the first test Figure 7.3 shows that the mean water level decreases as the wave field shoals but the scatter from one experiment to the other is generally more pronounced. It appears from the figure that the setup starts at a noticeable distance shorewards from the measured breaking point, indicating that the radiation stress is approximately constant in this region. In comparison with the first test it is evident that an increase in the wave height results in a larger wave-induced setup as well as a breaking point located in relatively deeper water.

The third test series considers an incident wave field described by a wave period of \( T^* = 1.41 \) s and an initial wave height of \( H_0^* = 47.0 \) mm (Table 7.3). Figure 7.4 shows that the wave-induced decrease in the mean water level close to the breaking point is substantial. All four sets of measurements show the same noticeable trend and it is therefore believed that the readings are correct. In Section 7.3.3 it was mentioned that the water level measured in each piezometer represents the mean water level at the location of the pressure tapping provided the pressure is hydrostatic. Since the flow varies rapidly close to the breaking point, the vertical acceleration of the fluid particles influences the pressure distribution and hence it can be questioned whether, in fact, the water levels shown in Figure 7.4 represent the true mean water level in this region. Furthermore, it is noted that wave breaking occurs relatively close to the reef top, i.e. in very shallow water. In agreement with observation the setup caused by wave breaking induces a significant return current in this region. By considering the horizontal projection of the depth-integrated and time-averaged momentum equation (Longuet-Higgins & Stewart, 1964), it can be ascertained that the convective acceleration of the return current affects the mean water level. Consequently, it is speculated that the return current is respon-
7.4. Experimental Results

Figure 7.2: Measured mean water level. Data of the incident wave field: \( D_0^* = 0.481 \text{ m}, H_0^* = 41.0 \text{ mm}, T^* = 1.13 \text{ s}, \ c_0^* = 0.00 \text{ m/s} \).

Figure 7.3: Measured mean water level. Data of the incident wave field: \( D_0^* = 0.481 \text{ m}, H_0^* = 94.0 \text{ mm}, T^* = 1.13 \text{ s}, \ c_0^* = 0.00 \text{ m/s} \).
Figure 7.4: Measured mean water level. Data of the incident wave field: $D_0 = 0.481$ m, $H_0 = 47.0$ mm, $T^* = 1.41$ s, $c_s^* = 0.00$ m/s.

Figure 7.5: Measured mean water level. Data of the incident wave field: $D_0^* = 0.481$ m, $H_0^* = 80.0$ mm, $T^* = 1.41$ s, $c_s^* = 0.00$ m/s.
7.4. Experimental Results

Figure 7.6: Measured mean water level. Data of the incident wave field: $D_0^* = 0.481$ m, $H_0^* = 48.0$ mm, $T^* = 1.70$ s, $c_s^* = 0.00$ m/s.

Figure 7.7: Measured mean water level. Data of the incident wave field: $D_0^* = 0.481$ m, $H_0^* = 79.0$ mm, $T^* = 1.70$ s, $c_s^* = 0.00$ m/s.
sible for the large decrease in the mean water level close to the reef crest.

In accordance with the previous set of measurements an incident regular wave field described by a wave period, \( T^* = 1.41 \) s, is considered in the fourth test series. The incident wave height is given by \( H_0^* = 80.0 \) mm. As shown in Table 7.3 this corresponds to a wave of finite height in initially intermediate depth water. Figure 7.5 illustrates that the mean water level is approximately constant before breaking. Inside the surf zone a large increase in the mean water level is preceded by a minor decrease in the region close to the breaking point. As mentioned previously the flow varies rapidly in this region, and it is therefore unknown if this is a physical effect. In comparison with the previous set of measurements Figure 7.5 shows that the wave field breaks at a relatively larger distance from the reef crest. This is consistent with the fact that the considered wave height is significantly greater. Additionally, it appears that an increase in the wave height results in a greater wave-induced setup.

The fifth test series considers an incident wave height, \( H_0^* = 48.0 \) mm, and a wave period, \( T^* = 1.70 \) s. As shown in Table 7.3 this corresponds to a relatively long wave of small initial steepness. It appears from Figure 7.6 that the mean water level decreases significantly immediately prior to wave breaking and in the early stages of the breaking process. Moreover, it is noted that the wave-induced setup starts inside the surf zone.

In agreement with the previous test series an incident wave field described by a wave period of \( T^* = 1.70 \) s is considered in test 6. The wave height is given by \( H_0^* = 79.0 \) mm (Table 7.3) and the measured mean water level is shown in Figure 7.7. It appears that the mean water level is qualitatively similar to that displayed in Figure 7.6.

In general, the figures confirm that an increase in both the height and the period of the incident wave field results in a greater wave-induced setup.

### 7.5 Computational Results

In this section a comparison is made between results computed using the surf zone model and those acquired in the experiments. The computational domain is discretized by 450 nodes using a grid spacing of \( \Delta x^* = 0.0400 \) m. The still water depth is given by

\[
D^*(x^*) = \begin{cases} 
0.481 \text{ m}, & x^* \in [-2.00, 3.47] \\
0.481 - \frac{x^*-3.47}{5} \text{ m}, & x^* \in [3.47, 4.17] \\
0.341 - \frac{x^*-4.17}{18.8} \text{ m}, & x^* \in [4.17, 7.18] \\
0.181 - \frac{x^*-7.18}{10.6} \text{ m}, & x^* \in [7.18, 8.66] \\
0.041 + \frac{x^*-8.66}{10.6} \text{ m}, & x^* \in [8.66, 9.09] \\
0.081 \text{ m}, & x^* \in [9.09, 16.0] 
\end{cases} \quad (7.1)
\]

and hence the fluid domain covers the interval, \( x^* \in [-2.00, 16.0] \). Dissipative sponge layers, 50 nodes wide, are located adjacent to the left and the right boundaries. This ensures that outgoing waves are absorbed satisfactorily. In order to conserve mass the sponge layers
7.5. Computational Results

are accompanied by fully reflecting boundary conditions (Section 3.5). The incident (regular) wave field is generated internally using the Fourier approximation method described in Section 3.8 and the wave field is resolved by 10 wave components, i.e. \( M = 10 \). Since the computations relate to wave propagation in a closed wave flume, the Fourier computation is based on \( c^*_T = 0.00 \text{ m/s} \). Similarly, the still water depth is given by \( D_0^* = 0.481 \text{ m} \) while the height and the period of the incident wave field are depicted in Table 7.3. The surf zone model is run for 250 wave periods using a Courant number of \( Cr = 0.500 \), and the mean water level is extracted from the computations during 20 wave periods immediately prior to program termination. This ensures that transient effects are insignificant.

As in the previous chapter the surf zone model is calibrated using the experimental results obtained in the first test series, and subsequently verified by comparison with the experimental test series denoted 2 – 6 (Table 7.3). An optimum set of wave breaking parameters is determined by trial and error, and given by \( f_d = 2.0 \), \( t_2 = 0.7T \), \( \tan \phi_0 = 0.36 \), and \( \tan \phi_0 = 0.12 \). In each test series the wave friction parameter, \( f_w \), is set equal to zero in the part of the flume of constant water depth. Since the constructed bed profile is covered by 2 – 3 mm of sand cement mortar, the Nikuradse roughness height is assumed to be of the order \( k_N^* = 1.00 \text{ mm} \) in this region. The corresponding wave friction parameter, which is shown in Table 7.5 for each test series, is determined by use of Equations (6.19) and (6.20).

Moreover, the table shows the measured and the computed values of both the breaking point and the maximum setup. The measured and the computed breaking point are termed \( MB^* \) and \( CB^* \), respectively, while the corresponding values of the setup are denoted \( MS^*_{\text{max}} \) and \( CS^*_{\text{max}} \), respectively. The maximum setup is defined as the difference between the mean water level at the reef top \( (x^* = 12.0 \text{ m}) \) and the offshore mean water level \( (x^* = 3.00 \text{ m}) \).

The computed and the measured mean water levels are depicted in Figures 7.8 - 7.13 as a function of the horizontal coordinate, \( x^* \), for the test series denoted 1 – 6. It is noted that the experimental curves are obtained as an average of four consecutive data sets within the same test series. The graphs exhibit a general vertical shift between the measured and the computed mean water level. As mentioned in Section 6.5.1 this aspect is expected and is physically correct. In order to conserve mass a given wave-induced setup in the downstream part of the middle wave flume must be balanced by a depression in the remaining part of the fluid domain, including the flumes denoted 1 and 3, see Figure 7.1. In the computational example the horizontal extent of the fluid domains upstream and downstream of the breaking point are significantly different, and the curves are therefore also bound to exhibit a general vertical shift.

Figure 7.8 shows the computed and the measured mean water levels obtained in the first test series. Since the surf zone model is calibrated using the measurements shown in the figure, the computed mean water level is simulated accurately before, during, and after wave breaking. By considering Table 7.5 it is evident that the chosen wave breaking parameters result in an accurate prediction of the breaking point and the maximum wave-induced setup.

As shown in Table 7.3 a steep incident wave field is considered in the second test series. Since the chosen wave condition is beyond the theoretical validity range of any Boussinesq-type model, the surf zone model can not be expected to produce accurate results. Although the breaking point is predicted relatively accurately, Figure 7.9 shows that the model underestimates the wave-induced setup by approximately 30%, see Table 7.5. A part of the discrepancy is caused by the poor prediction of the mean water level close to the breaking point. Since the wave field is strongly nonlinear in this region, the approximation of the convective term is important. Visual observations indicate that the wave field remains relatively composed in the surf zone inshore of the reef crest. On the other hand, the results computed by the surf zone
Figure 7.8: Test 1. Measured and computed MWL. Computational data: \( H_0 = 41.0 \) mm, \( T^* = 1.13 \) s, \( c_s^* = 0.00 \) m/s, \( M = 10 \), \( h = 450 \), \( \Delta x^* = 0.0400 \) m, \( Cr = 0.500 \), \( ISL = ISR = 50 \), \( \psi \in [0, 250] \), \( f_d = 2.0 \), \( t_2 = 0.7T \), \( \tan \phi_0 = 0.36 \), \( \tan \phi_0 = 0.12 \), and \( f_w = 0.0517 \).

Figure 7.9: Test 2. Measured and computed MWL. Computational data: \( H_0 = 94.0 \) mm, \( T^* = 1.13 \) s, \( c_s^* = 0.00 \) m/s, \( M = 10 \), \( h = 450 \), \( \Delta x^* = 0.0400 \) m, \( Cr = 0.500 \), \( ISL = ISR = 50 \), \( \psi \in [0, 250] \), \( f_d = 2.0 \), \( t_2 = 0.7T \), \( \tan \phi_0 = 0.36 \), \( \tan \phi_0 = 0.12 \), and \( f_w = 0.0335 \).
7.5. Computational Results

Figure 7.10: Test 3. Measured and computed MWL. Computational data: $H_0^* = 47.0$ mm, $T^* = 1.41$ s, $c_*^* = 0.00$ m/s, $M = 10$, $II = 450$, $\Delta x^* = 0.0400$ m, $Cr = 0.500$, $ISL = ISR = 50$, $f \in [0, 250]$, $f_d = 2.0$, $t_2 = 0.7T$, $\tan \phi_b = 0.36$, $\tan \phi_0 = 0.12$, and $f_w = 0.0425$.

Figure 7.11: Test 4. Measured and computed MWL. Computational data: $H_0^* = 80.0$ mm, $T^* = 1.41$ s, $c_*^* = 0.00$ m/s, $M = 10$, $II = 450$, $\Delta x^* = 0.0400$ m, $Cr = 0.500$, $ISL = ISR = 50$, $f \in [0, 250]$, $f_d = 2.0$, $t_2 = 0.7T$, $\tan \phi_b = 0.36$, $\tan \phi_0 = 0.12$, and $f_w = 0.0325$. 
CHAPTER 7. APPLICATION TO A SUBMERGED CORAL REEF

Figure 7.12: Test 5. Measured and computed MWL. Computational data: $H_0^* = 48.0$ mm, $T^* = 1.70$ s, $c_s^* = 0.00$ m/s, $M = 10$, $II = 450$, $\Delta x^* = 0.0400$ m, $Cr = 0.500$, $ISL = ISR = 50$, $\frac{f}{f_d} \in [0, 0.250]$, $f_d = 2.0$, $t_2 = 0.7T$, $\tan \phi_b = 0.36$, $\tan \phi_0 = 0.12$, and $f_w = 0.0382$.

Figure 7.13: Test 6. Measured and computed MWL. Computational data: $H_0^* = 79.0$ mm, $T^* = 1.70$ s, $c_s^* = 0.00$ m/s, $M = 10$, $II = 450$, $\Delta x^* = 0.0400$ m, $Cr = 0.500$, $ISL = ISR = 50$, $\frac{f}{f_d} \in [0, 0.250]$, $f_d = 2.0$, $t_2 = 0.7T$, $\tan \phi_b = 0.36$, $\tan \phi_0 = 0.12$, and $f_w = 0.0300$. 
7.5. Computational Results

<table>
<thead>
<tr>
<th>Test</th>
<th>$T^*$ [s]</th>
<th>$H_0^*$ [mm]</th>
<th>$f_w$ [-]</th>
<th>$MB^*$ [m]</th>
<th>$CB^*$ [m]</th>
<th>$MS_{max}^*$ [mm]</th>
<th>$CS_{max}^*$ [mm]</th>
<th>$CS_{max}^* / MS_{max}^*$ [-]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.13</td>
<td>41.0</td>
<td>0.0517</td>
<td>8.46</td>
<td>8.52</td>
<td>5.15</td>
<td>5.44</td>
<td>1.06</td>
</tr>
<tr>
<td>2</td>
<td>1.13</td>
<td>94.0</td>
<td>0.0335</td>
<td>7.83</td>
<td>7.60</td>
<td>19.6</td>
<td>13.9</td>
<td>0.712</td>
</tr>
<tr>
<td>3</td>
<td>1.41</td>
<td>47.0</td>
<td>0.0425</td>
<td>8.43</td>
<td>8.23</td>
<td>6.41</td>
<td>5.56</td>
<td>0.867</td>
</tr>
<tr>
<td>4</td>
<td>1.41</td>
<td>80.0</td>
<td>0.0325</td>
<td>7.96</td>
<td>7.46</td>
<td>17.5</td>
<td>14.7</td>
<td>0.842</td>
</tr>
<tr>
<td>5</td>
<td>1.70</td>
<td>48.0</td>
<td>0.0382</td>
<td>8.14</td>
<td>8.19</td>
<td>9.44</td>
<td>9.36</td>
<td>0.991</td>
</tr>
<tr>
<td>6</td>
<td>1.70</td>
<td>79.0</td>
<td>0.0300</td>
<td>7.85</td>
<td>7.78</td>
<td>22.8</td>
<td>18.6</td>
<td>0.814</td>
</tr>
</tbody>
</table>

Table 7.5: Measured and computed values of the wave-induced setup and the breaking point. Additionally, the wave friction parameter is shown for each test series.

The computed and the measured mean water levels obtained in the third test series are depicted in Figure 7.10. The graph illustrates that the model predicts the shape of the mean water level relatively accurately, although there is a minor tendency to underestimate the setup caused by wave breaking. It appears from Table 7.5 that the computed breaking point is located further offshore than determined by the measurements. Additionally, it is noted that the surf zone model captures the relatively pronounced decrease in the mean water level in the vicinity of the breaking point.

Figure 7.11 shows the computed and the measured mean water levels obtained in the fourth test series. It appears from the graph that the surf zone model overestimates the wave-induced decrease in the mean water level as the wave field climbs the slope. Although the prediction of the breaking point is inaccurate (Table 7.5), the initially slow development of the surface roller results in a satisfactory prediction of the location at which the mean water level starts to increase rapidly ($x^* \approx 8.4$ m). Owing to the fact that the mean water level is underestimated prior to wave breaking, the maximum wave-induced setup is underestimated by approximately 16%.

The fifth wave condition describes a relatively long wave of a small initial height. In agreement with the prediction of Table 7.3 the observations indicate that the waves break as plunging breakers, see Galvin (1968). Figure 7.12 illustrates that the surf zone model predicts accurately the decrease in the mean water level prior to wave breaking. As shown in Table 7.5 the breaking point and the maximum wave-induced setup are simulated accurately. Since the mean water level is underestimated in part of the surf zone, it is evident that the temporal development of the surface roller is too slow.

The results obtained in test 6 are given in Figure 7.13. The graph shows that the computed shape of the mean water level agrees well with the experimental results prior to
wave breaking. From Table 7.5 it is evident that the model estimates the breaking point accurately but the mean water level starts to increase further offshore than dictated by the measurements. In accordance with the majority of the previous examples the maximum wave-induced setup is underestimated.

As mentioned previously the present chapter aims to verify the validity of the surf zone model on steeper slopes. Generally, the results indicate that the surf zone model predicts reasonably accurately the location of the breaking point as well as the variation of the mean water level before, during, and after wave breaking. It is noted that the model captures the pronounced decrease in the mean water level in the vicinity of the breaking point. The maximum setup caused by wave-breaking is predicted accurately provided the incident wave height is significantly smaller than the offshore still water depth. As the incident wave height increases, the surf zone model underestimates the maximum setup. Since the wave-induced setup depends strongly on the determination of the surface roller thickness in time and space, it is crucial to ensure that the breaking waves remain sufficiently composed in passing the reef top. It is speculated that a better approximation of the convective term in the vicinity of the breaking point may achieve this and, thereby, enhance the prediction of the wave-induced setup. Consequently, this part will be studied in future investigations.
Chapter 8

Summary and Conclusion

Wave transformation in water of variable depth and the development of waves in the surf zone have been studied numerically and experimentally.

Based on the assumptions of irrotational flow and an incompressible and inviscid fluid the first part of the thesis describes the development of a computational model in a single horizontal dimension. A set of Boussinesq-type equations with improved linear dispersion characteristics in deeper water is solved numerically using the finite difference method and a highly accurate time-stepping procedure. Since first derivative terms are approximated correct to the fourth order, truncation error terms influencing the dispersion characteristics of the numerical solution are reduced to a level significantly smaller than that maintained in the governing equations. The incident wave field is generated inside the computational domain while the scattered waves are absorbed almost perfectly in the vicinity of open boundaries by employment of damping terms in the mass and momentum equations. A Fourier method is used to impose an incident regular wave field which fulfils the governing equations perfectly on a horizontal bottom. The model can be used to describe the transformation of regular as well as irregular waves in water of variable depth provided the bottom slope is of the same order of magnitude as the ratio of the mean water depth and a typical wave length.

An investigation is made of the phase and amplitude portraits of the numerical solution, hence providing practical information on the time step size and the grid spacing. In general, waves of finite height can be modelled accurately using 20 computational points per shallow water wave length as well as a Courant number of 0.500. It is shown that the model conserves well basic properties such as the total mass and energy inside the fluid domain. The computational model is applied to predict the wave height variation caused by regular waves of finite height in initially intermediate depth water propagating onto a plane and gentle slope. The results compare well with existing wave flume measurements. Additionally, a study is made of the transformation and the subsequent decomposition of a solitary wave as it climbs a shelf. The results agree qualitatively and quantitatively well with what is found in the literature.

For practical simulations a numerical model is often required which covers two horizontal dimensions. Consequently, the numerical solution method is extended to include the second horizontal dimension. The formulation is very general indicating that the model can be used to study e.g. the evolution of waves inside harbours of complex geometries. The analytical manipulations required to generate the incident wave field inside the computational domain become quite substantial in a formulation covering two horizontal dimensions, and the internal
CHAPTER 8. SUMMARY AND CONCLUSION

The wave generation method is therefore generalized and included in the formulation in a simple and efficient way. It is verified that the transfer of momentum from one direction to the other is modelled correctly. The numerical model is used to study both the diffraction of linear deep water waves around a breakwater and the combined refraction and diffraction of regular waves propagating over a semicircular shoal. The results compare well with existing analytical theory and experiments, respectively. The general formulation of the model is illustrated by an example which describes the propagation of regular waves into a fictitious harbour.

The second part of the thesis is concerned with wave breaking and the temporal development of waves in the surf zone. The effect of spilling wave breaking is incorporated into the two-dimensional model using the concept of surface rollers. Based on the assumption of a vertical redistribution of the horizontal velocity in a breaking wave a new set of equations is derived. The temporal development of the surface roller thickness is determined geometrically using an existing method. Despite the fact that the mathematical basis is relatively weak and the physical description is very crude, the model has the potential to predict a variety of processes such as the fluctuating breaking point caused by random waves breaking on a beach and the important conversion of potential energy to kinetic energy in the outer region of the surf zone. The model is calibrated using a single set of laboratory data and subsequently verified by comparison with two other sets of measurements. The results show that the model is capable of predicting relatively accurately the mean water level and the wave height variation caused by regular waves breaking on a plane and gentle slope. Additionally, it is noted that the wave-induced flow computed by the model peaks at a notable distance shorewards of the breaking point. This result agrees well with existing wave flume measurements.

The last part of the thesis describes a laboratory experiment conducted in a closed wave flume. A description is given of both the laboratory facility and the equipment used as well as the procedures involved in the experiment. The considered bed profile, which represents a fringing coral reef located on the coast of Guam, consists of a relatively steep reef face and an adjacent reef crest submerged below the still water level. Consequently, the main purpose of the experiment is to verify the validity of the surf zone model on steeper slopes. The incident wave conditions describe regular waves of small and finite height in initially intermediate depth water. In each test series the mean water level and the breaking point are recorded in the steady state. The experimental results are compared with those of the computational model and these indicate that the surf zone model generally predicts the location of the breaking point accurately. Provided the incident wave height is significantly smaller than the offshore still water depth the mean water level is predicted accurately before, during, and after wave breaking. In particular it is mentioned that the model captures the pronounced decrease in the mean water level observed in the vicinity of the reef crest. Supported by visual observation, it is speculated that the substantial drop in the mean water level is caused by the convective acceleration of the return current. As the incident wave height increases relative to the offshore still water depth, the surf zone model tends to underestimate the maximum setup. It is believed that a better approximation of the space derivatives in the vicinity of the breaking point, e.g. the convective term, is likely to enhance the prediction of the wave-induced setup, and this part will therefore be subjected to future research.
References


Boussinesq, J. (1872), "Theorie des ondes et des remous qui se propagent le long d'un canal rectangulaire horizontal, en communiquant au liquide contenu dans ce canal des vitesses sensiblement pareilles de la surface au fond", *Journal de Mathematiques Pures et Appliquees Deuxieme Serie*, 17, pp. 55 - 108.


REFERENCES


REFERENCES

Fenton, J. D. (1996), "Private communication".


REFERENCES


REFERENCES


Roache, P. J. (1976), Computational fluid dynamics, Hermosa Publishers, Albuquerque, N. M.


Sørensen, O. R. (1996), "Private communication".


REFERENCES


Appendix A

Double-Sweep Method

The Double-Sweep method is used to solve a linear system of equations which has a tridiagonal structure. In accordance with Section 3.3 the equation system is given by

\[
\begin{align*}
R^0 &= u_0^0 \\
R^i &= D1^i u_{i+1}^{i-1} + D2^i u_i^i + D3^i u_{i+1}^{i+1}, \ i = 1, \ldots, II - 1 \\
R^{II} &= u_{II}^{II}
\end{align*}
\] (A.1)

in which the time levels are omitted for brevity and the coefficients, \(D1^i, D2^i, D3^i, i = 1, \ldots, II - 1\), and \(R^i, i = 0, \ldots, II\), are presumed to be known. Since the boundary conditions are invoked at this stage, see Section 3.3, the velocity field is known at the end points, \(i = 0\) and \(i = II\), where \(II\) denotes the maximum node number. Consequently, the unknown variables are given by \(u_0^0, i = 1, \ldots, II - 1\).

The Double-Sweep method is based on the assumption of a linear relationship between adjacent variables, and hence the method can be perceived as a special form of Gauss elimination (Vreugdenhil, 1989). The relationship is given by

\[
\begin{align*}
u_i^i &= E^i + F^i u_{i+1}^{i+1}, \ i = 0, \ldots, II - 1
\end{align*}
\] (A.2)

in which the coefficients, \(E^i\) and \(F^i, i = 0, \ldots, II - 1\), are determined in the forward sweep and the velocity field is quantified in the subsequent return sweep. From the equation given above it is evident that

\[
\begin{align*}
u_{i+1}^{i-1} &= E^{i-1} + F^{i-1} u_i^i, \ i = 1, \ldots, II
\end{align*}
\] (A.3)

By substitution of Equation (A.3) into the original tridiagonal equation system, Equation (A.1), it follows directly from Equation (A.2) that the coefficients, \(E^i\) and \(F^i, i = 1, \ldots, II - 1\),
can be written

\[ E^i = \frac{R^i - D1^i E^{i-1}}{D2^i + F^{i-1} D1^i}, \quad i = 1, \ldots, II - 1 \]  \quad (A.4)

and

\[ F^i = \frac{-D3^i}{D2^i + F^{i-1} D1^i}, \quad i = 1, \ldots, II - 1 \]  \quad (A.5)

The velocity field is given explicitly at the left boundary, \( i = 0 \), and the quantities, \( E^0 \) and \( F^0 \), respectively, are therefore given by \( E^0 = u_0 \) and \( F^0 = 0 \), see Equation (A.2). Equations (A.4) and (A.5) are used successively to determine the remaining coefficients in the forward sweep. Since the quantity, \( u_{II} \), is assumed to be known, Equation (A.3) is used to quantify the velocity field, \( u_{\alpha} \), \( i = 1, \ldots, II - 1 \), in the subsequent return sweep.
Appendix B

Code - One Horizontal Dimension

B.1 Abm.2D

Program ABM;

{$D*,L*,N*,Y*}$

Uses
  Boundary,
  Breeding,
  Fourier,
  MathFunc,
  Solution,
  Variable,
  Various,
  Crt;

Begin
  InitComputation;
  While N<=NN Do
  Begin
    Predictor;
    Corrector;
    ApplyDissipativeInterface;
    UpdateComputation;
  End;
  ComputeMeanSetupAndFlow;
  EndComputation;
  End.
APPENDIX B. CODE - ONE HORIZONTAL DIMENSION

B.2 Boundary.2D

Unit Boundary;

Interface
{$D+,L+,N+,Y+}$

Uses
Fourier,
MathFunc,
Variable,
Various;

Function CnSurf(J,N:Integer):Extended;
Function CnVel(J,N:Integer):Extended;
Function EllipticA(EM,El,E2:Extended):Extended;
Function Fedel:Extended;
Function FourSurf(J,N:Integer):Extended;
Function FourVel(J,N:Integer):Extended;
Function IrregulcirSurface(J,N:Integer):Extended;
Function IrregulcirVelocity(J,N:Integer):Extended;
Function SinSurf(J,N:Integer):Extended;
Function SinVel(J,N:Integer):Extended;
Function SolSurf(J,N:Integer):Extended;
Function SolVel(J,N:Integer):Extended;
Function Surface(J,N:Integer):Extended;
Function Velocity(J,N:Integer):Extended;
Procedure ComputeBathymetry;
Procedure ComputeCnoidalParameters;
Procedure ComputeInitialCondition;
Procedure ComputeIrregularWaveNumber;
Procedure ComputeNewRandS;
Procedure EllipticM;
Procedure InitMatrix(J:Integer;Value:Extended;Var M:MatrixType);
Procedure InitVector(J:Integer;Value:Extended;Var V:VectorType);
Procedure Linear;
Procedure RandomWaveField;
Procedure ReadFrictionCoefficients;
Procedure Sinus;
Procedure SolitaryWaveConstants;
Procedure Step;
Procedure UserFile;
Procedure WaveParameters;

Implementation

Function CnVel(J,N:Integer):Extended;
Var
Surf:Extended;
Begin
With LeftBoundary Do
Begin
Surf:=CnSurf(J,N);
CnVel:=C*(Surf/D-Sqr(Surf/D));
End;
End;

Function CnSurf(J,N:Integer):Extended;
Var
Theta:Extended;
Begin
With LeftBoundary Do
Begin
Theta:=(N*DT/T-(J-J0)*DX/L)*2*El;
CnSurf:=EtaMin+H*Sqr(Jacobi(Theta,EM));
End;
End;
Function Fader: Extended;
Begin
If N*DT<TFade Then
  Fader:=Sin(Pi/2*N*DT/TFade)
Else
  Fader:=1;
End;

Function FourSurf(J,N: Integer): Extended;
Var
  U, AThree, ASeven: Extended;
Begin
  With LeftBoundary Do
  Begin
    AThree:=Cl*(Cl/2+1)*D*D;
    ASeven:=(Cl*Cl/2+Cl+1/3)*D*D*D;
    U:=FourVel(J,N)-Z-[2*MM+3];
    FourSurf:=(AThree*Z-[2*MM+3]*(Z'[2*MM+5]-D*U)+ASeven*
    (Z'[2*MM+6]-1/2*U*U))/(AThree*Z'[2*MM+3]+U+ASeven);
  End;
End;

Function FourVel(J,N: Integer): Extended;
Var
  Sum, Ratio, Cosine: Extended;
  M : Integer;
Begin
  With LeftBoundary Do
  Begin
    Sum:=0;
    For M:=1 To MM Do
    Begin
      Ratio:=(N+Z'[2*MM+4]*D+((1+Cl))/Cosh(N+Z'[2*MM+4]*D));
      Cosine:=(N+Z'[2*MM+4]*D)*Cos((J-J0)*DX-Z'[2*MM+3]*N*DT));
      Sum:=Sum+Ratio*Cosine;
    End;
    FourVel:=Z'[2*MM+3]+Z'[2*MM+4]*Sum;
  End;
End;

Function EllipticA(EM,El,E2: Extended): Extended;
Begin
  EllipticA:=2/EM-l-3*E2/EM/El;
End;

Function IrregularSurface(J,N: Integer): Extended;
Var
  Sum: Extended;
Begin
  Sum:=0;
  With LeftBoundary Do
  Begin
    For IFreq:=0 To NFreq Do
    Begin
      KD:=WaveNum*[IFreq]*N*DT-
      +WaveNum*[J-J0]*DX+Phase*[IFreq];
      IrregularSurface:=Sum;
    End;
  End;
End;

Function IrregularVelocity(J,N: Integer): Extended;
Var
  Arg, KD, Sum, Surf: Extended;
Begin
  Sum:=0;
  With LeftBoundary Do
  Begin
    For IFreq:=0 To NFreq Do
    Begin
      KD:=-WaveNum*[IFreq]*D;
APPENDIX B. CODE - ONE HORIZONTAL DIMENSION

\[
\text{Arg} := KD \times (1 + C1);
\]
\[
\text{Surf} := \text{Amplitude}[\text{Freq}] \times \text{Cos}(\text{Frequency}[\text{Freq}] \times DT - \text{WaveNum}[\text{Freq}] \times (J - J0) \times DX + \text{Phase}[\text{Freq}]);
\]
\[
\text{Sum} := \text{Sum} + \text{Frequency}[\text{Freq}] \times \text{Surf} \times \cosh(\text{Arg}) / \sinh(KD);
\]
\[
\text{End};
\]
\[
\text{IrregularVelocity} := \text{Sum};
\]
\[
\text{End};
\]

Function \text{Velocity}(J, N: Integer): Extended;
Begin
With LeftBoundary Do
Case UpCase(R) Of
'I': \text{Velocity} := \text{IrregularVelocity}(J, N);
'R': Case UpCase(W) Of
'C': \text{Velocity} := \text{CnVel}(J, N);
'O': \text{Velocity} := \text{SinVel}(J, N);
'F': \text{Velocity} := \text{FourVel}(J, N);
'S': \text{Velocity} := \text{SolVel}(J, N);
End;
End;
End;

Function \text{SinVel}(J, N: Integer): Extended;
Var
\text{Arg}, KD, Omega, Surf, Theta: Extended;
Begin
With LeftBoundary Do
\begin{align*}
\text{Omega} &:= 2 \times \pi / T; \\
\text{Theta} &:= \text{Omega} \times N \times DT - K \times (J - J0) \times DX; \\
\text{Arg} &:= KD \times (1 + C1) \times D; \\
\text{KD} &:= K \times D; \\
\text{Surf} &:= H / 2 \times \cos(\text{Theta}); \\
\text{SinVel} &:= \text{Omega} \times \text{Surf} \times \cosh(\text{Arg}) / \sinh(KD) + 3 / 16 \times \text{Omega} \times K \times H \times H \\
&\quad \times \cosh(2 \times \text{Arg}) / \sqrt{\sinh(KD)^2} \times \cos(2 \times \Theta); \\
\end{align*}
End;
End;

Function \text{SinSurf}(J, N: Integer): Extended;
Var
\text{Theta}, Cothkh: Extended;
Begin
With LeftBoundary Do
\begin{align*}
\text{Theta} &:= 2 \times \pi / T \times N \times DT - K \times (J - J0) \times DX; \\
\text{Cothkh} &:= \text{Coth}(K \times D); \\
\text{SinSurf} &:= H / 2 \times \cos(\text{Theta}) + K \times H \times H / 16 \times \text{Cothkh}^2 \\
&\quad \times (3 \times \text{Cothkh} \times \text{Cothkh}^2 - 1) \times \cos(2 \times \Theta); \\
\end{align*}
End;
End;

Function \text{SolVel}(J, N: Integer): Extended;
Begin
With LeftBoundary Do
\text{SolVel} := A / \sqrt{\cosh(B \times ((J - J0) \times DX - C \times (N \times DT - TDelay)))};
End;

Function \text{SolSurf}(J, N: Integer): Extended;
Begin
With LeftBoundary Do
\begin{align*}
\text{SolSurf} &:= A1 / \sqrt{\cosh(B \times ((J - J0) \times DX - C \times (N \times DT - TDelay)))} \\
&\quad + A2 / \sqrt{\cosh(B \times ((J - J0) \times DX - C \times (N \times DT - TDelay)))}; \\
\end{align*}
End;

Function \text{Surface}(J, N: Integer): Extended;
Begin
With LeftBoundary Do
Case UpCase(R) Of
'I': Surface:=IrregularSurface(J,N);
'R': Case UpCase(W) Of
'J': Surface:=CnSurf(J,N);
'O': Surface:=SinSurf(J,N);
'F': Surface:=FourSurf(J,N);
'S': Surface:=SolSurf(J,N);
End;
End;
End;

Procedure ComputeBathymetry;
Begin
If JJ>JJMax Then Error(30);
If LeftBoundary.JS>JSMax Then Error(40);
If RightBoundary.JS>JSMax Then Error(40);
Case UpCase(Bathymetry) Of
'L': Linear;
'U': UserFile;
'S': Step;
'S': Sinus;
End;
SaveBathymetry;
End;

Procedure ComputeCnoidalParameters;
Begin
EllipticM;
With LeftBoundary Do
Begin
C:=Sqrt(D*(1+EA*H/D));
L:=C*T;
EtaMin:=H*((1-E2/El)/EM-l);
End;
End;

Procedure ComputeInitialCondition;
Begin
With LeftBoundary Do
Begin
If UpCase(W)='S' Then
Begin
SampleStart:=0;
SampleStop:=NN;
End
Else
Begin
SampleStart:=NN-1-Trunc(TSample*T/DT);
SampleStop:=NN;
End;
End;
For J:=0 To JJ Do
Begin
InitVector(J,0,Delta);
InitVector(J,0,HMean);
InitVector(J,0,QMean);
InitVector(J,0,R0);
InitVector(J,0,R1);
InitVector(J,0,SMax);
InitVector(J,0,S0);
InitVector(J,0,S1);
InitVector(J,0,Tbreak);
InitVector(J,0,UMean);
InitVector(J,0,U0);
InitVector(J,0,U1);
InitMatrix(J,0,F1);
APPENDIX B. CODE - ONE HORIZONTAL DIMENSION

InitMatrix(J,0,F2);
InitMatrix(J,0,F3);
End;
JStart:=J0+3*JSearch;
JStop:=J1+3*JSearch;
End;

Procedure ComputeIrregularWaveNumber;
Begin
With LeftBoundary Do
For IFreq:=0 To NFreq Do
ComputeSineWaveNumber(WaveNum"[IFreq],D,Frequency"[IFreq]);
End;

Procedure ComputeNewRandS;
Begin
With LeftBoundary Do
Begin
SIM1:=Fader*Surface(J0-1,N);
SIM :=Fader*Surface(J0 ,N);
SIP1:=Fader*Surface(J0+1,N);
SIP2:=Fader*Surface(J0+2,N);
UIM1:=Fader*Velocity(J0-1,N);
UIM :=Fader*Velocity(J0 ,N);
UIP1:=Fader*Velocity(J0+1,N);
UIP2:=Fader*Velocity(J0+2,N);
End
End;

Procedure EllipticM;
Var
F,F1,F2,M1,M2:Extended;
I :Integer;
Begin
With LeftBoundary Do
Begin
I:=0;
F:=10;
M1:=1E-10;
E1:=Elliptic1(M1);
E2:=Elliptic2(M1);
EA:=EllipticA(M1,E1,E2);
F1:=3*H*T*T/16/D/D-M1*E1*E1/(1+EA*H/D);
M2:=1-1E-10;
E1:=Elliptic1(M2);
E2:=Elliptic2(M2);
EA:=EllipticA(M2,E1,E2);
F2:=3*H*T*T/16/D/D-M2*E1*E1/(1+EA*H/D);
While (I<MaxIter) And (Abs(F)>MaxErr) Do
Begin
I:=I+1;
EM:=(M1+M2)/2;
E1:=Elliptic1(EM);
E2:=Elliptic2(EM);
EA:=EllipticA(EM,E1,E2);
F:=3*H*T*T/16/D/D-EM*E1*E1/(1+EA*H/D);
If F>F1>0 Then
Begin
M1:=EM;
F1:=F;
End
Else
Begin
M2:=EM;
F2:=F;
End;
End;
End;
Procedure InitMatrix(J:Integer; Value:Extended; Var M:MatrixType);
Begin
  M[J][0-2]:=Value;
  M[J][1-1]:=Value;
  M[J][2-0]:=Value;
  M[J][3-1]:=Value;
End;

Procedure InitVector(J:Integer; Value:Extended; Var V:VectorType);
Begin
  V[J]:=Value;
End;

Procedure Linear;
Var
  HorzDelay:Integer;
Begin
  HorzDelay:=Trunc(JJ/4);
  For J:=0 To HorzDelay-1 Do
    D'[J]:=LeftBoundary.D;
  For J:=HorzDelay To JJ-HorzDelay Do
    D'[J]:=LeftBoundary.D+(RightBoundary.D-LeftBoundary.D)/(JJ-2*HorzDelay)*(J-HorzDelay);
  For J:=JJ-HorzDelay+1 To JJ Do
    D'[J]:=RightBoundary.D;
End;

Procedure RandomWaveField;
Const
  NOmega   =50;
  OmegaMin =0.209;
  OmegaMax =2.094;
  Cutoff   =0.995;
Var
  Alfa   :Extended;
  Beta   :Extended;
  DeltaOmega :Extended;
  Gamma   :Extended;
  OmegaCutOff:Extended;
  OmegaP   :Extended;
  Sigma   :Extended;

Procedure GenerateRandomPhase(Var Eps:Extended);
Begin
  Eps:=-Pi+2*Pi*Random;
End;

Procedure JONSWAPParameters;
Begin
  With LeftBoundary Do
    Begin
      If T<=3.6*Sqrt(H) Then
        Gamma:=5;
      If (T>3.6*Sqrt(H)) And (T<5*Sqrt(H)) Then
        Gamma:=Exp(5.75-1.15*T/Sqrt(H));
      If T>=5*Sqrt(H) Then
        Gamma:=1;
      Beta:=1.25;
      Alfa:=5.0609*Sqr(H/(T*T))*(1-0.287*Ln(Gamma));
      OmegaP:=2*Pi/T;
    End;
End;
Function JONSWAP(Omega:Extended):Extended;
Var
   W:Extended;
Begin
   W := Omega/OmegaP;
   If Omega<OmegaP Then
      Sigma := 0.07
   Else
      Sigma := 0.09;
   JONSWAP := G*G*Alfa/(Omega*Omega*Omega*Omega*Omega)*
      Exp(-Beta/(W^4)) * Exp(Ln(Gamma)*Exp(-Sqr(Omega-OmegaP)/
      (2*Sigma*Sigma*OmegaP*OmegaP)));
End;

Function PM(Omega:Extended):Extended;
Const
   AlfaPM=0.0081;
   BetaPM=0.74;
Begin
   With LeftBoundary Do
   Begin
      PM := AlfaPM/(Omega*Omega*Omega*Omega*Omega)*
      Exp(-BetaPM/Sqr(Sqr(6.5795*Omega*Sqrt(H/G))));
   End;
End;

Procedure DetermineCutOffFrequency;
Var
   MO,Omega,Sum:Extended;
Function Moment(M:Byte):Extended;
Var
   Mom:Extended;
Begin
   Mom := 0;
   With LeftBoundary Do
   Case UpCase(Sp) Of
      'J':For IFreq:=0 to NOmega Do
      Begin
         Omega := OmegaMin+IFreq*DeltaOmega;
         Mom := Mom + Exp(M*Ln(Omega))*JONSWAP(Omega);
      End;
      'P':For IFreq:=0 to NOmega Do
      Begin
         Omega := OmegaMin+IFreq*DeltaOmega;
         Mom := Mom + Exp(M*Ln(Omega))*PM(Omega);
      End;
   End;
   Moment := Mom*DeltaOmega;
End;
Begin
   Sum := 0;
   DeltaOmega := (OmegaMax-OmegaMin)/NOmega;
   MO := Moment(0)/DeltaOmega;
   Omega := OmegaMin;
   IFreq := 0;
   With LeftBoundary Do
   Case UpCase(Sp) Of
      'J':While (IFreq<NOmega) And (Sum/MO<CutOff) Do
      Begin
         Omega := OmegaMin+IFreq*DeltaOmega;
         Sum := Sum + JONSWAP(Omega);
         IFreq := IFreq+1;
      End;
      'P':While (IFreq<NOmega) And (Sum/MO<CutOff) Do
      Begin
...
Omega:=OmegaMin+IFreq*DeltaOmega;
Sum:=Sum+PM(Omega);
IFreq:=IFreq+1;
End;
If (Sum/MO<CutOff) Then
   Error(70)
Else OmegaCutOff:=OmegaMin+(IFreq-1)*DeltaOmega;
End;
Procedure GenerateRandomNumbers;
   Begin
      With LeftBoundary Do
      Begin
         DeltaOmega:=(OmegaCutOff-OmegaMin)/NFreq;
         For IFreq:=0 To NFreq Do
            Begin
               Frequency'[IFreq]' := OmegaMin+IFreq*DeltaOmega;
               GenerateRandomPhase(Phase'[IFreq]');
            End;
      Case UpCase(Sp) Of
         'J':For IFreq:=0 To NFreq Do
             Amplitude'[IFreq]' := Sqrt(2*JONSWAP(Frequency'[IFreq]')
             +DeltaOmega);
         'P':For IFreq:=0 To NFreq Do
             Amplitude'[IFreq]' := Sqrt(2*PM(Frequency'[IFreq]')
             +DeltaOmega);
      End;
      End;
   End;
Begin
   RandSeed:=0;
   If UpCase(LeftBoundary.Sp)='J' Then
      JONSWAPParemeters;
   DetermineCutOffFrequency;
   GenerateRandomNumbers;
End;
Procedure ReadFrictionCoefficients;
Begin
   Assign(DataFile,FileDir+FrictionFile);
   Reset(DataFile);
   For J:=0 To JJ Do
      ReadLn(DataFile,FricCoef'[J]');
   Close(DataFile);
End;
Procedure Sinus;
   Var
      HorzDelay:Integer;
   Begin
      HorzDelay:=Trunc(JJ/4);
      For J:=0 To HorzDelay-1 Do
         D'[J]' := LeftBoundary.D;
      For J:=HorzDelay To JJ-HorzDelay Do
         D'[J]' := (LeftBoundary.D+RightBoundary.D)/2+
                   (LeftBoundary.D-RightBoundary.D)/2*
                   Cos(Pi*(J-HorzDelay)/(JJ-2*HorzDelay));
      For J:=JJ-HorzDelay+1 To JJ Do
         D'[J]' := RightBoundary.D;
   End;
Procedure SolitaryWaveConstants;
   Var
      F,F1,F2,CC,Cell,Cel2,Height:Extended;
      Iter :Integer;
Function Dispersion(X:Extended):Extended;
Begin
Dispersion:=2*Alpha*X*X*X*X*X*X
-(3*Alpha+1/3+2*Alpha*Height)*X*X*X*X
+2*Height*(Alpha+1/3)*X*X*Alpha+1/3;
End;

Begin
With LeftBoundary Do
Begin
Height:=H;
Alpha:=(0.5*C1*C1+C1)*D*D;
End;
Cell:=0.9;
F1:=Dispersion(Cell);
Cell2:=1.5;
F2:=Dispersion(Cell2);
F:=100;
Iter:=0;
While (Abs(F)>MaxErr) And (Iter<MaxIter) Do
Begin
Iter:=Iter+1;
CC:=(Cell+Cell2)/2;
F:=Dispersion(CC);
If (F*F1)>0 Then
Begin
Cell:=CC;
F1:=F;
End
Else
Begin
Cell2:=CC;
F2:=F;
End;
End;
If Abs(F)>MaxErr Then
Error(110)
Else
With LeftBoundary Do
Begin
C:=CC;
A:=(C*C-1)/C;
A1:=(C*C-1)/(3*(Alpha+1/3-Alpha*C*C));
A2:=Sqr(C*C-1)/(2+C*C)*(Alpha+1/3+2*Alpha*C*C)
/(Alpha+1/3-Alpha*C*C);
B:=Sqrt(3/4*A1);
End;
End;

Procedure Step;
Var
JRel,JStart,JStop,JWidth:Integer;
Begin
JWidth:=Trunc(StepWidth/DX);
JStart:=Trunc((LengthX-StepWidth)/(2*DX));
JStop:=JStart+JWidth;
For J:=0 To JStart-1 Do
D^J[ ]:=LeftBoundary.D;
For J:=JStart To JStop Do
Begin
JRel:=J-JStart;
D^J[ ]:=(LeftBoundary.D+RightBoundary.D)/2 +
(LeftBoundary.D-RightBoundary.D)/2*Cos(Pi*JRel/JWidth);
End;
For J:=JStop+1 To JJ Do
D^J[ ]:=RightBoundary.D;
End;
Procedure UserFile;
Begin
  Assign(DataFile,FileDir+BathyFile);
  Reset(DataFile);
  For J:=0 To JJ Do
  Begin
    ReadLn(DataFile,D"[J]");
    D"[J]:=D"[J]/DO;
  End;
  Close(DataFile);
End;

Procedure WaveParameters;
Begin
  With LeftBoundary Do
  Case UpCase(R) Of
    'I':Begin
      If NFreq>=JJMax Then
        Error(60);
        RandomWaveField;
        ComputeIrregularWaveNumber;
        ComputeSineWaveNumber(K,D,2*Pi/T);
        C:=2*Pi/(K*T);
    End;
    'R':Case UpCase(W) Of
      'C':ComputeCnoidalParameters;
      'O':Begin
        ComputeSineWavenumber(K,D,2*Pi/T);
        C:=2*Pi/(K*T);
      End;
      'F':Begin
        FourierComputation;
        K:=Z'[2*MM+4];
        L:=2*Pi/K;
      End;
    'S':Begin
      SolitaryWaveConstants;
      T:=LengthX/(5*C);
    End;
  End;
End;
Begin
End.
B.3 Breaking.2D

Unit Breaking:

Interface

{$D+,L+,N+,Y+}$

Uses

Boundary,
MathFunc.,
Variable,
Various,
Crt;

Function FindSlope(JA,JB:Integer):Extended;
Function FindTBreak(JA,JB:Integer):Extended;
Function RollerThickness(J:Integer):Extended;
Function Slope(JA,JB:Integer):Extended;
Function SlopeOfTangent(Time:Extended):Extended;
Function Status(JSL:Integer):WaveBreaking;
Procedure ComputeEndBreakingWave(Var JSL:Integer);
Procedure ComputeNewBreakingWave(Var JSL:Integer);
Procedure ComputeOldBreakingWave(Var JSL:Integer);
Procedure ExcludeNodes;
Procedure FindNodeOfMinSlope(Var J:Integer);
Procedure IncludeNodes;
Procedure LocateOldRoller(JSL:Integer);
Procedure LocateRollerRegion(J:Integer);
Procedure PredictRollerThickness;

Implementation

Function FindSlope(JA,JB:Integer):Extended;
Var
SL,SLMin:Extended;
Begin
SLMin:=Slope(JA,JA+2);
For J:=JA+2 To JB-1 Do
Begin
SL:=Slope(J-1,J+1);
If SL<SLMin Then
SLMin:=SL;
End;
FindSlope:=SLMin;
End;

Function FindTBrek(JA,JB:Integer):Extended;
Var
TBFound:Extended;
Begin
TBFound:=0;
For J:=JA To JB Do
	If TBreak'[J]>0 Then
		TBFound:=TBreak'[J];
	End;
FindTBrek:=TBFound;

Function RollerThickness(J:Integer):Extended;
Var
Del:Extended;
Begin
Del:=(Sl'[J]-Sl'[J2]-Phi*(J-J2)*DX)*FDelta;
If Del>0 Then
	If Del<D'[J]+Sl'[J] Then
		RollerThickness:=Del
	Else
		RollerThickness:=D'[J]+Sl'[J]
Else
  RollerThickness:=0;
End;

Function Slope(JA,JB:Integer):Extended;
Begin
  Slope:=(S1'[JB]-S1'[JA])/((JB-JA)*DX);
End;

Function SlopeOfTangent(Time:Extended):Extended;
Begin
  SlopeOfTangent:=AlphaO+(AlphaB-AlphaO)*
    Exp(-Ln(2)*((Time-TB)/THalf));
End;

Function Status(JSL:Integer):WaveBreaking;
Var
  SL:Extended;
Begin
  TB:=FindTBreak(JSL-JSearch,JSL+JSearch);
  SL:=FindSlope(JSL-JSearch,JSL+JSearch);
  If TB=0 Then
    If SL<=AlphaB Then
      Status:=NewBreakingWave
    Else
      Status:=NonBreakingWave
  Else
    If SL<=AlphaO Then
      Status:=OldBreakingWave
    Else
      Status:=EndBreakingWave;
End;

Procedure ComputeEndBreakingWave(Var JSL:Integer);
Begin
  LocateOldRoller(JSL);
  ExcludeNodes;
  JSL:=J201d+JSearch;
End;

Procedure ComputeNewBreakingWave(Var JSL:Integer);
Begin
  Phi:=AlphaB;
  TB:=N*DT;
  LocateRollerRegion(JSL);
  IncludeNodes;
  SaveRollerTrace(J1,J2,'New');
  JSL:=J2+JSearch;
End;

Procedure ComputeOldBreakingWave(Var JSL:Integer);
Begin
  LocateOldRoller(JSL);
  TB:=FindTBreak(J1Old,J201d);
  Phi:=SlopeOfTangent(N*DT);
  LocateRollerRegion(J201d);
  ExcludeNodes;
  IncludeNodes;
  SaveRollerTrace(J1,J2,'Old');
  JSL:=J2+JSearch;
End;

Procedure ExcludeNodes;
Begin
  For J:=J101d To J201d Do
    Begin
      InitVector(J,0,TBreak);
      InitVector(J,0,Delta);
Procedure FindNodeOfMinSlope(Var J:Integer);
Var
JSLMin :Integer;
SL,SLMin:Extended;

Function LocalMinimum(SL1,SL2,SL3:Extended):Boolean;
Begin
LocalMinimum:=False;
If (SL2<SL1) And (SL2<SL3) Then
  If (SL1<0) And (SL2<0) And (SL3<0) Then
    LocalMinimum:=True;
End;

Function ElevationsArePositive(S1,S2,S3:Extended):Boolean;
Begin
  If (S1>0) And (S2>0) And (S3>0) Then
    ElevationsArePositive:=True
  Else
    ElevationsArePositive:=False;
End;

Begin
  Repeat
    J:=J+1
  Until ElevationsArePositive(Sl^[J-1],S1^[J],S1^[J+1])
  Or (J>=JStop);
  Repeat
    J:=J+1
  Until LocalMinimum(Slope(J-2,J),Slope(J-1,J+1),Slope(J,J+2))
  Or (TBreak^[J]>0) Or (J>=JStop);
  If TBreak^[J]>0 Then
    Begin
      JSLMin:=J;
      SLMin:=Slope(JSLMin-1,JSLMin+1);
      Repeat
        JSLMin:=JSLMin+1;
        SL:=Slope(JSLMin-1,JSLMin+1);
      Until SL<=SLMin
      Begin
        J:=JSLMin;
        SLMin:=SL;
        Until (TBreak^[JSLMin]=0) Or (JSLMin>=JStop);
      End;
    End;
  End;
End;

Procedure IncludeNodes;
Begin
  For J:=J1 To J2 Do
    Begin
      InitVector(J,TB,TBreak);
      Delta^[J]:=RollerThickness(J);
    End;
End;

Procedure Locate01dRoller(JSL:Integer);
Begin
  J101d:=JSL+2*JSearch;
  Repeat
    J101d:=J101d-1
  Until (TBreak^[J101d]>0) Or (J101d<JStart);
  If J101d=JStart Then
    Error(120);
  Repeat
B.3. Breaking 2D

J101d:=J101d-1
Until (TBreak'[J101d]=0) Or (J101d<=JStart);
J101d:=J101d+1;
J201d:=J101d;
Repeat
J201d:=J201d+1
Until (TBreak'[J201d]=0) Or (J201d>=JStop);
J201d:=J201d-1;
End;

Procedure LocateRollerRegion(J:Integer);
Begin
J2:=J;
Repeat
J2:=J2+1
Until (Slope(J2-1,J2)>Phi) Or (J2>=JStop);
J2:=J2-1;
J1:=J2;
Repeat
J1:=J1-1
Until (RollerThickness(J1)=0) Or (J1<=JStart);
J1:=J1+1;
End;

Procedure PredictRollerThickness;
Var
JSL,JSL0ld:Integer;
Begin
JSL0ld:=0;
JSL:=JStart;
While JSL<=JStop Do
Begin
FindNodeOfMinSlope(JSL);
Case Status(JSL) Of
NewBreakingWave:ComputeNewBreakingWave(JSL);
OldBreakingWave:ComputeOldBreakingWave(JSL);
EndBreakingWave:ComputeEndBreakingWave(JSL);
End;
If JSL0ld=JSL Then
Error(130)
Else
JSL0ld:=JSL;
JSL:=JSL+1;
End;
End;

Begin
End.
B.4 Fourier.2D

Unit Fourier;

Interface

{$D+, L+, N+, Y+}$

Uses

Variable,

MathFunc,

Various;

Function DContDEta(J:Integer):Extended;
Function DContDK(J:Integer):Extended;
Function DContDQ:Extended;
Function DContDU(I,J:Integer):Extended;
Function DContDUO(J:Integer):Extended;
Function DMomDC(J:Integer):Extended;
Function DMomDEta:Extended;
Function DMomDK(J:Integer):Extended;
Function DMomDR:Extended;
Function DMomDU(I,J:Integer):Extended;
Function Dynamic(J:Integer):Extended;
Function F2MM3:Extended;
Function F2MM4:Extended;
Function F2MM5:Extended;
Function F2MM6:Extended;
Function Kinematic(J:Integer):Extended;
Function Sigma1(J:Integer):Extended;
Function Sigma2(J:Integer):Extended;
Function Sigma3(J:Integer):Extended;
Function Sigma4(J:Integer):Extended;
Procedure CoefficientMatrix;
Procedure ComputeSineWaveNumber(Var WaveNum:Extended;Depth, Omega:Extended);
Procedure Gauss(A:QuadMatrix;X,B:VectorType;N:Integer);
Procedure GaussNewton;
Procedure FourierComputation;
Procedure Increment;
Procedure InitialGuess;
Procedure Initialize;
Procedure RightHandSide;
Procedure Terminate;

Implementation

Function DContDEta(J:Integer):Extended;
Begin
With LeftBoundary Do
  DContDEta:=Z-[MM+2]+Z^[2+MM+4]*Sigma(J);
End;
Function DContDK(J:Integer):Extended;
Begin
With LeftBoundary Do
  DContDK:=(D+Z^[J+1])*Sigma1(J)+Z^[2*MM+4]*Sigma3(J)
-((1/2*C1+C1+1/3)*D+D*Sqr(Z^[2*MM+4]))*
  (3*Sigma2(J)+Z^[2*MM+4]*Sigma4(J));
End;
Function DContDQ:Extended;
Begin
  DContDQ:=-1;
End;
Function DContDU(I,J:Integer):Extended;
Begin

With LeftBoundary Do
  DContDU:=(D+Sqrt(I*Z-[(2*MM+4)*D])*(I*Z-[(2*MM+4)*D]*Cos(I*J*Pi/MM));
End;

Function DContDUO(J:Integer):Extended;
Begin
  With LeftBoundary Do
  DContDUO:=D+Z[J+1];
End;

Function DMomDC(J:Integer):Extended;
Begin
  With LeftBoundary Do
  DMomDC:=Cl*(1/2*Cl+l)*D*Sqrt(Z-[(2*MM+4)])*Z-[(2*MM+4)]*Sigma2(J);
End;

Function DMomDEta:Extended;
Begin
  DMomDEta:=1;
End;

Function DMoinDK(J:Integer) :Extended;
Var
  Sigma:Extended;
Begin
  With LeftBoundary Do
  Begin
    Sigma:=Sigmal(J);
    DMomDK:=(Z'[MM+2]+Z-[(2*MM+4)]*Sigma)+(Sigma+Z-[(2*MM+4)]*Sigma3(J))*
      (Cl*(1/2*Cl+l)*D*Sqrt(Z-[(2*MM+4)])*Z-[(2*MM+4)]*I)*
      (3*Sigma2(J)+Z-[(2*MM+4)]*Sigma4(J));
  End;
End;

Function DMomDR:Extended;
Begin
  DMomDR:=-l;
End;

Function DMomDU(I,J:Integer):Extended;
Begin
  With LeftBoundary Do
  DMomDU:=(Z'[MM+2]+Z-[(2*MM+4)]*Sigma)+Cl*(1/2*Cl+l)*D*Sqrt(Z-[(2*MM+4)])*Z-[(2*MM+4)]*I*
    Cosh(I*Z-[(2*MM+4)]*D*(1+Cl))/Cosh(I*Z-[(2*MM+4)]*D)*Cos(I*J*Pi/MM);
End;

Function DMomDUO(J:Integer):Extended;
Begin
  With LeftBoundary Do
  DMomDUO:=Z'[MM+2]+Z-[(2*MM+4)]*Sigma(J);
End;

Function Dynamic(J:Integer):Extended;
Begin
  With LeftBoundary Do
  Dynamic:=l/2*Sqrt(Z'[MM+2]+Z-[(2*MM+4)]*Sigma(J))+(Sigma)+Z'[J+1]+*
    C1*(1/2*Cl+l)*D*Sqrt(Z-[(2*MM+4)])*Z-[(2*MM+4)]*Sigma2(J)*Z-[(2*MM+4)];
End;

Function F2MM3:Extended;
Var
  Sum:Extended;
Begin
  With LeftBoundary Do
  Begin
    ...
Begin
Sum:=Z'[1]+Z'[MM+1];
For J:=2 To MM Do
    F2MM3:=Sum+2*Z'[J];
End;
Sum:=Sum/(2*MM);
End;

Function F2MM3:Extended;
Begin
With LeftBoundary Do
    F2MM3:=Z'[1]-Z'[MM+1]-H;
End;

Function F2MM4:Extended;
Begin
With LeftBoundary Do
    F2MM4:=Z'[1]-Z'[MM+1]-H;
End;

Function F2MM5:Extended;
Begin
With LeftBoundary Do
    F2MM5:=Z'[2*MM+3]*Z'[2*MM+4]-2*Pi;
End;

Function F2MM6:Extended;
Begin
With LeftBoundary Do
    If UpCase(Current)='E' Then
        F2MM6:=Z'[2*MM+3]-Curr+Z'[2*MM+4]-2*Pi
    Else
        F2MM6:=Z'[2*MM+3]-Curr+Z'[2*MM+4]-2*Pi;
End;

Function Kinematic(J:Integer):Extended;
Begin
With LeftBoundary Do
    Kinematic:=(D+Z'[J+1])*(Z'[MM+2]+Z'[2*MM+4]*Sigma(J))-
        (1/2*(1+Cl)*D+(1+Cl)/2*(1+Cl)*D+D+Sqr(Z'[2*MM+4])*Z'[2*MM+4]*
            Sigma2(J)-Z'[2*MM+5];
End;

Function Sigma(J:Integer):Extended;
Var
    Sum,Ratio,Cosine:Extended;
    M :Integer;
Begin
With LeftBoundary Do
Begin
    Sum:=0;
    For M:=1 To MM Do
Begin
        Ratio:=Cosh(M*Z'[2*MM+4]*D*(1+Cl))/Cosh(M*Z'[2*MM+4]*D);
        Cosine:=M*M*M*Z'[2*MM+4]*Cos(J*M*Pi/MM);
        Sum:=Sum+Ratio*Cosine;
End;
    Sigma:=Sum;
End;
End;

Function Sigma2(J:Integer):Extended;
Var
    Sum,Ratio,Cosine:Extended;
    M :Integer;
Begin
With LeftBoundary Do
Begin
    Sum:=0;
    For M:=1 To MM Do
Begin
        Ratio:=Cosh(M*Z'[2*MM+4]*D*(1+Cl))/Cosh(M*Z'[2*MM+4]*D);
        Cosine:=M*M*M*Z'[2*MM+4]*Cos(J*M*Pi/MM);
        Sum:=Sum+Ratio*Cosine;
    End;
End;
End;
Function Sigma3(J:Integer):Extended;
Var
  Sum, Alf, Bet : Extended;
  Arg1, Arg2, Cosine, Ratio : Extended;
  X1, X2, X3, X4 : Extended;
  M : Integer;
Begin
  With LeftBoundary Do
  Begin
    Sum := 0;
    For M := 1 To MM Do
    Begin
      Cosine := M*M*Z'[MM+2*M]*Cos(J*M*Pi/MM);
      Alf := M*D*(1+C1);
      Arg1 := Alf*Z'[2*MM+4];
      Bet := M*D;
      Arg2 := Bet*Z'[2*MM+4];
      X1 := Alf*Sinh(Arg1)*Cosh(Arg2);
      X2 := Bet*Cosh(Arg1)*Sinh(Arg2);
      X3 := X2 - X1;
      X4 := Sqr(Cosh(Arg2));
      Ratio := X3/X4;
      Sum := Sum + Ratio*Cosine;
    End;
    Sigma3 := Sum;
  End;
End;

Function Sigma4(J:Integer):Extended;
Var
  Sum, Alf, Bet : Extended;
  Arg1, Arg2, Cosine, Ratio : Extended;
  X1, X2, X3, X4 : Extended;
  M : Integer;
Begin
  With LeftBoundary Do
  Begin
    Sum := 0;
    For M := 1 To MM Do
    Begin
      Cosine := M*M*M*Z'[MM+2*M]*Cos(J*M*Pi/MM);
      Alf := M*D*(1+C1);
      Arg1 := Alf*Z'[2*MM+4];
      Bet := M*D;
      Arg2 := Bet*Z'[2*MM+4];
      X1 := Alf*Sinh(Arg1)*Cosh(Arg2);
      X2 := Bet*Cosh(Arg1)*Sinh(Arg2);
      X3 := X2 - X1;
      X4 := Sqr(Cosh(Arg2));
      Ratio := X3/X4;
      Sum := Sum + Ratio*Cosine;
    End;
    Sigma4 := Sum;
  End;
End;

Procedure ComputeSineWaveNumber(Var WaveNum:Extended; Depth, Omega:Extended);
Var
  F, F1, F2, W1, W2 : Extended;
  Iter : Integer;
Begin
  W1 := 1E-6;
  F1 := Omega*Omega - W1*Depth*Tanh(W1*Depth);
  End;
W2:=1E3;
F2:=Omega*Omega-W2*Depth*Tanh(W2*Depth);
F:=100;
Iter:=0;
While (Abs(F)>MaxErr) And (Iter<MaxIter) Do
Begin
Iter:=Iter+1;
WaveNum:=(W1+W2)/2;
F:=Omega*Omega-WaveNum*Depth*Tanh(WaveNum*Depth);
If (F+F1)>0 Then
Begin
W1:=WaveNum;
F1:=F;
End
Else
Begin
W2:=WaveNum;
F2:=F;
End;
End;
If Abs(F)>MaxErr Then
Error(100)
Else
WaveNum:=W1;
End;

Procedure Gauss(A:QuadMatrix;X,B:VectorType;N:Integer);
Var
Change,Sum :Extended;
I1,I2,I3,IMax:Integer;
Begin
For I1:=1 To N-1 Do
Begin
IMax:=I1;
For I2:=I1+1 To N Do
If Abs(A[I1]-[I1])>Abs(A[IMax]-[I1]) Then IMax:=I2;
For I3:=1 To N Do
Begin
Change:=A[I1]-[13];
A[I1]-[13]:=A[IMax]-[13];
A[IMax]-[13]:=Change;
End;
Change:=B[I1];
B[I1]:=B[IMax];
B[IMax]:=Change;
If Abs(A[I1]-[I1])<1E-10 Then
Error(80);
For I2:=I1+1 To N Do
Begin
A[I2]-[I1]:=A[I2]-[I1]/A[I1]-[I1];
For I3:=I1+1 To N Do
A[I2]-[I3]:=A[I2]-[I3]-A[I2]-[I1]*A[I1]-[I3];
B[I2]:=B[I2]-A[I2]-[I1]*B[I1];
End;
End;
For I2:=N DownTo 1 Do
Begin
Sum:=0;
For I3:=I2+1 To N Do
Sum:=Sum+A[I2]-[I3]*X[I3];
X[I2]:=(B[I2]-Sum)/A[I2]-[I2];
End;
End;

Procedure CoefficientMatrix;
Begin
With LeftBoundary Do 
Begin
For J:=1 To 2*MM+6 Do
  For I:=1 To 2*MM+6 Do
    Coef["J"]*[I]:=0;
  For J:=1 To MM+1 Do
    Begin
      Coef["J"]*[J]:=DContDEta(J-1);
      Coef["J"]*[MM+2]:=DContDDO(J-1);
      For I:=MM+3 To 2*MM+2 Do
        Coef["J"]*[I]:=DContDU(I-MM-2,J-1);
      Coef["J"]*[2*MM+4]:=DContDX(J-1);
      Coef["J"]*[2*MM+5]:=DContDQ;
    End;
  For J:=MM+2 To 2*MM+2 Do
    Begin
      Coef["J"]*[J-MM-l]:=DMomDEta;
      Coef["J"]*[MM+2]:=DMomDUO(J-MM-2);
      For I:=MM+3 To 2*MM+2 Do
        Coef["J"]*[I]:=DMomDU(I-MM-2,J-MM-2);
      Coef["J"]*[2*MM+3]:=DMomDC(J-MM-2);
      Coef["J"]*[2*MM+4]:=DMomDX(J-MM-2);
      Coef["J"]*[2*MM+6]:=DMomDR;
    End;
  Coef*[2*MM+3] *[MM+1]:=1/(2*MM);
  For I:=2 To MM Do
    Coef*[2*MM+3] *[I]:=1/MM;
  Coef*[2*MM+4] *[MM+1]:=1/(2*MM);
  Coef*[2*MM+4] *[1]:=1;
  Coef*[2*MM+4] *[MM+1]:=1;
  Coef*[2*MM+5] *[2*MM+3]:=Z*2*[2*MM+4]*T;
  Coef*[2*MM+5] *[2*MM+4]:=Z*2*[2*MM+3]*T;
  If UpCase(Current)='E' Then
    Begin
      Coef*[2*MM+6] *[MM+2]:=1;
      Coef*[2*MM+6] *[2*MM+3]:=1;
    End
  Else
    Begin
      Coef*[2*MM+6] *[2*MM+3]:=1;
      Coef*[2*MM+6] *[2*MM+5]:=1;
    End;
  End;
End;

Procedure RightHandSide;
Begin
With LeftBoundary Do
Begin
End;
End;

Procedure GaussNewton;
Var
  NewErr:Extended;
  I,J :Integer;
Begin
With LeftBoundary Do
Begin
End;
End;

Procedure GaussNewton;
Var
  NewErr:Extended;
  I,J :Integer;
Begin
With LeftBoundary Do
Begin
End;
End;
While ((KNRI) And (NewErr>MaxErr)) Do
Begin
NewErr:=0;
I:=I+1;
CoefficientMatrix;
RightHandSide;
Gauss(Coeff,DF,RHS,2*MM+6);
For J:=1 To 2*MM+6 Do
Begin
If Abs(DF'[J])>NewErr Then
NewErr:=Abs(DF'[J]/Z'[J]);
Z'[J]:=Z'[J]+DF'[J];
End;
End;
End;
If NewErr>MaxErr Then
Error(90);
End;

Procedure Increment;
Begin
With LeftBoundary Do
Begin
H:=Alpha*H;
For J:=1 To MM+1 Do
Z'[J]:=Alpha*Z'[J];
For J:=MM+3 To 2*MM+2 Do
Z'[J]:=Alpha*Z'[J];
End;
End;

Procedure InitialGuess;
Var
 Omega:Extended;
Begin
With LeftBoundary Do
Begin
H:=H/NH;
Alpha:=Exp(Ln(NH)/(NH-1));
Omega:=2*Pi/T;
ComputeSineWaveNumber(Z'[2*MM+4],D,Omega);
Z'[2*MM+3]:=Omega/Z'[2*MM+4];
For J:=1 To MM+1 Do
Z'[J]:=H/2*cos((J-1)*Pi/MM);
Z'[2+MM+3]=-H/4/Z'[2+MM+3]*Z'[2+MM+4];
For J:=MM+4 To 2*MM+2 Do
Z'[J]:=0;
Z'[2*MM+5]:=-D*Z'[2*MM+3];
Z'[2*MM+6]:=Sqr(Z'[2*MM+3])/2;
End;
End;

Procedure Initialize;
Begin
AllocateVector(LeftBoundary,Z);
AllocateVector(DF);
AllocateVector(RHS);
AllocateQuadratic(Coeff);
End;

Procedure Terminate;
Begin
DeAllocateVector(DF);
DeAllocateVector(RHS);
DeAllocateQuadratic(Coeff);
End;
B.4. Fourier.2D

Procedure FourierComputation;
Var
  Count:Integer;
Begin
  Initialize;
  InitialGuess;
  GaussNewton;
  For Count:=2 To NH Do Begin
    Increment;
    GaussNewton;
    End;
  Terminate;
End;

Begin
End.
APPENDIX B.  CODE - ONE HORIZONTAL DIMENSION

B.5 Mathfunc.2D

Unit MathFunc;

Interface
{$D+,L+,N+,Y+}

Uses
    Variable,
    Various;

Function ArcSin(Z:Extended):Extended;
    Begin
        Arcsin:=Arctan(Z/Sqrt(-Z*Z+1));
    End;

Function Cosh(Z:Extended):Extended;
    Begin
        Cosh:=(Exp(Z)+Exp(-Z))/2;
    End;

Function Coth(Z:Extended):Extended;
    Begin
        Coth:=1/Tanh(Z);
    End;

Function Ellipticl(X:Extended):Extended;
    Begin
        Ellipticl:=RF(0,1-X,1);
    End;

Function Ellipticll(Phi,M:Extended):Extended;
    Var
        Sine:Extended;
    Begin
        Sine:=Sin(Phi);
        Ellipticll:=Sine*RF(Sqr(Cos(Phi)),1-Sine*Sine*M,1);
    End;

Function Elliptic2(X:Extended):Extended;
    Begin
        Elliptic2:=RF(0,1-X,1)-X/3*RD(0,1-X,1);
    End;

Function Jacobi(U,M:Extended):Extended;
    Const
        ArrSize=20;
    Var
        A,B,C:Array[0..ArrSize] Of Extended;
        I,W :Integer;
Phi2 : Extended;
Begin
A[0] := 1;
B[0] := Sqrt(N-M);
C[0] := Sqrt(N);
N := 0;
While (N <= ArrSize-1) And (Abs(CCN3) >= MaxErr) Do
Begin
N := N + 1;
C[N] := 0.5 * (A[N-1] - B[N-1]);
End;
Phi2 := Exp(N * Ln(2)) * A[N] * U;
For I := N Downto 1 Do
Phi2 := 0.5 * (Phi2 + Arcsin(C[I] / A[I] * Sin(Phi2)));
Jacobi := Cos(Phi2);
End;

Function Maximum(X, Y, Z: Extended): Extended;
Begin
If (X >= Y) And (X >= Z) Then Maximum := X;
If (Y >= X) And (Y >= Z) Then Maximum := Y;
If (Z >= X) And (Z >= Y) Then Maximum := Z;
End;

Function RD(X, Y, Z: Extended): Extended;
Const
ErrTol = 0.05;
C1 = 0.214285714286;
C2 = 0.166666666667;
C3 = 0.409090909090;
C4 = 0.115384615385;
C5 = 0.102272727273;
C6 = 0.173076923078;
Var
Alamb, Ave, DelX, DelY, DelZ, EA, EB, EC, EE, Fac: Extended;
SqrtX, SqrtY, SqrtZ, Sum, Xt, Yt, Zt: Extended;
Begin
Xt := X;
Yt := Y;
Zt := Z;
Sum := 0;
Fac := 1;
DelX := 1;
DelY := 2;
DelZ := 3;
While Maximum(Abs(DelX), Abs(DelY), Abs(DelZ)) > ErrTol Do
Begin
SqrtX := Sqrt(Xt);
SqrtY := Sqrt(Yt);
SqrtZ := Sqrt(Zt);
ALamb := SqrtX * (SqrtY + SqrtZ) + SqrtY * SqrtZ;
Sum := Sum + Fac / (SqrtZ * (Zt + ALamb));
Fac := 0.25 * Fac;
Xt := 0.25 * (Xt + ALamb);
Yt := 0.25 * (Yt + ALamb);
Zt := 0.25 * (Zt + ALamb);
Ave := 0.2 * (Xt + Yt + 3 * Zt);
DelX := (Ave - Xt) / Ave;
DelY := (Ave - Yt) / Ave;
DelZ := (Ave - Zt) / Ave;
End;
EA := DelX + DelY;
EB := DelZ + DelZ;
EC := EA - EB;
ED := EA - 6 * EB;
EE := ED + 2 * EC;
APPENDIX B. CODE - ONE HORIZONTAL DIMENSION

RD:=3*Sum+Fac*(C1+ED-C6+DelZ*EE)+(C2+Ed+DelZ*(-C3+EC+DelZ*C4*EA))/(Ave*Sqrt(Ave));
End;

Function RF(X,Y,Z:Extended):Extended;
Const
ErrTo1=0.08;
Third=0.3333333333333333;
C1=0.04166666666666666;
C2=0.1;
C3=0.06818181818181818;
C4=0.7142857142857142;
Var
Alamb,Ave,Delx,Delx,Delz,E2,E3,SqrtX,SqrtY,SqrtZ,Xt,Yt,Zt:Extended;
Begin
Xt:=X;
Yt:=Y;
Zt:=Z;
DelX:=l;
DelY:=2;
DelZ:=3;
While Maximum(Abs(DelX),Abs(DelY),Abs(DelZ))>ErrTo1 Do
Begin
SqrtX:=Sqrt(Xt);
SqrtY:=Sqrt(Yt);
SqrtZ:=Sqrt(Zt);
Alamb:=SqrtX*(SqrtY+SqrtZ)+SqrtY*SqrtZ;
Xt:=0.25*(Xt+Alamb);
Yt:=0.25*(Yt+Alamb);
Zt:=0.25*(Zt+Alamb);
Ave:=Third*(Xt+Yt+Zt);
DelX:=(Ave-Xt)/Ave;
DelY:=(Ave-Yt)/Ave;
DelZ:=(Ave-Zt)/Ave;
End;
E2:=Delx+Delz*Delz;
E3:=Delx*Delz;
RF:=(1+(C1+E2-C2-C3*E3+E3+C4*E3)/Sqrt(Ave));
End;

Function Sinh(Z:Extended):Extended;
Begin
Sinh:=(Exp(Z)-Exp(-Z))/2;
End;

Function Tan(Z:Extended):Extended;
Begin
Tan:=Sin(Z)/Cos(Z);
End;

Function Tanh(Z:Extended):Extended;
Begin
Tanh:=(Exp(Z)-Exp(-Z))/(Exp(Z)+Exp(-Z));
End;

Procedure DoubleSweep;

Procedure ForwardSweep;
Begin
E'[0]:=-A1'[0]/D2'[0];
F'[0]:=-D3'[0]/D2'[0];
For J:=1 To J0-1 Do
Begin
F'[J]:=-D3'[J]/(D2'[J]+D1'[J]*F'[J-1]);
End;
E'[J0]:=(A1'[J0]+D3'[J0]+E'[J0-1])
      /((D2'[J0]+D1'[J0]+F'[J0-1]));
F'[J0] := -D3'[J0]/(D2'[J0]+D1'[J0]*F'[J0-1]);
E'[J0+1]:=(R1'[J0+1]-D1'[J0+1]*U10-D1'[J0+1]*E'[J0])
/(D2'[J0+1]+D1'[J0+1]*F'[J0]);
F'[J0+1]:= -D3'[J0+1]/(D2'[J0+1]+D1'[J0+1]*F'[J0]);
For J:=J0+2 To JJ Do
Begin
E'[J] := (R1'[J]-D1'[J]*E'[J-1])/(D2'[J]+D1'[J]*F'[J-1]);
F'[J] := -D3'[J]/(D2'[J]+D1'[J]*F'[J-1]);
End;
End;

Procedure BackwardSweep;
Begin
Vel:=E'[JJ];
ErrorU:=ErrorU+Abs(Vel-U1'[JJ]);
U1'[JJ]:=Vel;
For J:=JJ-1 DownTo 0 Do
Begin
Vel:=E'[J]+F'[J]*U1'[J+1];
ErrorU:=ErrorU+Abs(Vel-U1'[J]);
U1'[J]:=Vel;
End;
End;
Begin
ForwardSweep;
BackwardSweep;
End;

Begin
End.
APPENDIX B. CODE - ONE HORIZONTAL DIMENSION

B.6 Solution.2D

Unit Solution;

Interface

{D+L+N+Y+}

Uses

Boundary,

Breaking,

Fourier,

MathFunc,

Variable,

Various,

Crt;

Function ErrorExceedsLimit:Boolean;

Function Flow(J:Integer):Extended;

Procedure ApplyDissipativeInterface;

Procedure CoefficientMatrix;

Procedure ComputeCoefficients;

Procedure ComputeF1;

Procedure ComputeF2;

Procedure ComputeF3;

Procedure ComputeMeanSetupAndFlow;

Procedure ComputeRelativeErrors;

Procedure ComputeSpongeCoefficients;

Procedure Corrector;

Procedure CorrectorRelativeErrors;

Procedure CorrectorRightHandSideForU;

Procedure InitComputation;

Procedure InitCounters;

Procedure InitPredictor;

Procedure Predictor;

Procedure PredictorRightHandSideForU;

Procedure UpdateComputation;

Procedure UpdateSetupAndFlow;

Procedure UpdateVariables;

Implementation

Function ErrorExceedsLimit:Boolean;

Begin
If (RelErrorS<MaxIterError) And (RelErrorU<MaxIterError) Then
  ErrorExceedsLimit:=False
Else
  ErrorExceedsLimit:=True;
End;

Function Flow(J:Integer):Extended;

Var

UX,UXX:Extended;

Begin
If (J>=2) And (J<=JJ-2) Then
  Begin
    UX:=(UO-[J-2]-8*UO-[J-1]+8*UO-[J+1]-UO-[J+2])/(12*DX);
    UXX:=(UO-[J-1]-2*UO-[J]+UO-[J+1])/(DX*DX);
  End
Else
  If (J=0) Or (J=1) Then
    Begin
      UX:=(UO-[0]-UO-[2])/(2*DX);
      UXX:=(UO-[0]-2*UO-[1]+UO-[2])/(DX*DX);
    End
End;
Else
  If (J=JJ-1) Or (J=JJ) Then
  Begin
    UX:=(-U0-[JJ-2]+U0-[JJ-1]+U0-[JJ])/(2*DX);
    UXX:=(U0-[JJ-2]-2*U0-[JJ-1]+U0-[JJ])/(DX*DX);
  End;
Flow:=(D'[J]+S0'[J]-Delta'[J]+D'[J]+D'[J]*DX2'[J]+(C1+0.5))*U0'[J]
+2*D'[J]+D'[J]+DX1'[J]*((C1+0.5)*UX+D'[J]+D'[J]*D'[J]
+(0.5*C1+C1+C1/3)/UX
+CorrectC*Sqrt(D'[J]+Delta'[J]);
End;

Procedure ApplyDissipativeInterface;
Var
  Q1:Extended;
Begin
  Q1:=I/50;
  For J:=J0+JSearch To JJ-JSearch Do
  IF ((S1'[J-1]>S1'[J]) And (S1'[J]<S1'[J+1])) Or
      ((S1'[J-1]<S1'[J]) And (S1'[J]>S1'[J+1])) Then
    S1'[J]:=Q1*S1'[J-1]+(1-2*Q1)*S1'[J]+Q1*S1'[J+1];
End;

Procedure CoefficientMatrix;
Begin
  D1'[0]:=0;
  D2'[0]:=1;
  D3'[0]:=0;
  For J:=1 To JJ-1 Do
  Begin
    D1'[J]:=A1'[J]/(2*DX)+A2'[J]/(DX*DX);
    D2'[J]:=A1'[J]/(2*DX)+A2'[J]/(DX*DX);
    D3'[J]:=A2'[J]/(2*DX)+A3'[J]/(DX*DX);
  End;
End;

Procedure ComputeCoefficients;
Var
  DX3:Extended;
Begin
  DX1'[0]:=(-D'[0]+D'[2])/(2*DX);
  DX2'[0]:=D'[0]-2*D'[1]+D'[2])/(DX*DX);
  DX3:=(3*D'[0]+10*D'[1]-12*D'[2]+6*D'[3]-D'[4])/(2*DX*DX*DX);
  A1'[1]:=1+C1*D'[1]*DX2'[1];
  A2'[1]:=2*C1*D'[1]*DX1'[1];
  A3'[1]:=C1*(C1/2+C1)*D'[1];
  A4'[1]:=(C1+0.5)*(2*DX1'[1]*DX2'[1]+D'[1]*DX3)*D'[1];
  A5'[1]:=(C1+0.5)*3*D'[1]*DX2'[1]+4*DX1'[1]*DX1'[1])*D'[1];
  A6'[1]:=(3/2*C1+C1+C1)*D'[1]*DX1'[1];
  A7'[1]:=(0.5*C1*C1+C1+1/3)*D'[1]*D'[1]*DX1'[1];
  For J:=2 To JJ-2 Do
  Begin
    DX1'[J]:=(D'[J-2]-8*D'[J-1]+8*D'[J+1]-D'[J+2])/(12*DX);
    DX2'[J]:=(D'[J-1]-2*D'[J]+D'[J+1])/(DX*DX);
    DX3:=(3*D'[J-2]+2*D'[J-1]-2*D'[J]+D'[J+2])/(2*DX*DX*DX);
    A1'[J]:=1+C1*D'[J]*DX2'[J];
    A2'[J]:=2*C1*D'[J]*DX1'[J];
    A3'[J]:=-C1*(C1/2+C1)*D'[J];
    A4'[J]:=(C1+0.5)*(2*DX1'[J]*DX2'[J]+D'[J]*DX3)*D'[J];
APPENDIX B. CODE - ONE HORIZONTAL DIMENSION

\[ A5^*[[J]] := (C1 + 0.5) \times (3 \times D - [J] \times DX2^*[[J]] + 4 \times DX1^*[[J]] \times DX1^*[[J]]) \\
A6^*[[J]] := (3/2 \times C1 \times C1 + 6 \times C1 + 2) \times D - [J] \times DX1^*[[J]] \\
A7^*[[J]] := (0.5 \times C1 \times C1 + C1 + 1/3) \times D - [J] \times DX1^*[[J]] + DX1^*[[J]] \\
\]

\[ DX1^*[[JJ-1]] := (-D - [JJ-2] + D - [JJ-1]) / (2 \times DX) \\
DX2^*[[JJ-1]] := (D - [JJ-2] - 2 \times D - [JJ-1] + D - [JJ-J]) / (DX \times DX) \\
DX3^* := (D - [JJ-4] - 4 \times D - [JJ-3] - 12 \times D - [JJ-2] - 10 \times D - [JJ-1] + 3 \times D - [JJ-J]) / (2 \times DX \times DX) \\
A1^*[[JJ]] := 1 + C1 \times D - [JJ] \times DX2^*[[JJ-1]] \\
A2^*[[JJ]] := 2 \times C1 \times D - [JJ] \times DX1^*[[JJ-1]] \\
A3^*[[JJ]] := C1 \times (C1/2 + 1) \times D - [JJ-J] \times DX1^*[[JJ-1]] \\
A4^*[[JJ]] := (C1 + 0.5) \times (2 \times DX1^*[[JJ-1]] + DX2^*[[JJ-1]] + D - [JJ-1]) \times DX1^*[[JJ-1]] \\
A5^*[[JJ]] := (0.5 \times C1 \times C1 + C1 + 1/3) \times D - [JJ-J] \times DX1^*[[JJ-1]] \\
\]

\[ DX1^*[[JJ]] := (-D - [JJ-2] + D - [JJ]) / (2 \times DX) \\
DX2^*[[JJ]] := (D - [JJ-2] - 2 \times D - [JJ-1] + D - [JJ-J]) / (DX \times DX) \\
\]

Procedure ComputeF1;
Begin
For J:=2 To JJ-2 Do
\[ F1^*[[J]] := - (D - [J-2] + S1 - [J-2]) \times U1^*[[J-2]] \\
- 8 \times (D - [J-1] + S1 - [J-1]) \times U1^*[[J-1]] \\
+ 8 \times (D - [J] + S1 - [J]) \times U1^*[[J]] \\
- (D - [J+2] + S1 + [J+2]) \times U1^*[[J+2]]) / (12 \times DX) \\
- (Sqrt(D - [J-2]) - U1^*[[J-2]]) \times \Delta t^*[[J-2]] \\
- 8 \times (Sqrt(D - [J-1]) - U1^*[[J-1]]) \times \Delta t^*[[J-1]] \\
+ 8 \times (Sqrt(D - [J+1]) - U1^*[[J+1]]) \times \Delta t^*[[J+1]] \\
- (Sqrt(D - [J+2]) - U1^*[[J+2]]) \times \Delta t^*[[J+2]]) / (12 \times DX) \times CorrectC \\
- A4^*[[J]] \times U1^*[[J]] \\
- A5^*[[J]] \times (U1^*[[J-2]] - 8 \times U1^*[[J-1]] + U1^*[[J+2]]) / (12 \times DX) \\
- A6^*[[J]] \times (U1^*[[J-1]] - 2 \times U1^*[[J]] + U1^*[[J+1]]) / (DX \times DX) \\
- A7^*[[J]] \times (-U1^*[[J-2]] + 2 \times U1^*[[J-1]] - 2 \times U1^*[[J+1]] + U1^*[[J+2]]) / (2 \times DX \times DX \times DX) \\
\]

\[ F1^*[[J0-1]] := F1^*[[J0]] - (D - [J0+1] + S1^1) \times U1^*[[J0+1]] \\
+ A5^*[[J0]] \times (8 \times U1^*[[J0+1]] - U1^*[[J0]]) / (12 \times DX) \\
+ A7^*[[J0]] \times (2 \times U1^*[[J0+1]] - U1^*[[J0]]) / (2 \times DX \times DX) \\
\]

\[ F1^*[[J0]] := F1^*[[J0-1]] + (8 \times (D - [J0+1] + S1^1) \times U1^*[[J0+1]] - A5^*[[J0]] \times U1^*[[J0]]) / (12 \times DX) \\
+ A6^*[[J0]] \times (8 \times U1^*[[J0+1]] + U1^*[[J0]]) / (DX \times DX) \\
+ A7^*[[J0]] \times (2 \times U1^*[[J0+1]] + U1^*[[J0]]) / (2 \times DX \times DX) \\
\]

Function ConvectiveTerm(J:Integer):Extended;
Begin
If J<J0+1 Then
ConvectiveTerm := (U1^*[[J-2]] - 8 \times U1^*[[J-1]] + 8 \times U1^*[[J+1]] - U1^*[[J+2]]) / (12 \times DX) \times U1^*[[J]]
Else
If (Abs(Slope(J-1,J))>=0.3) Or (Abs(Slope(J,J))>=0.3) Then
If ((S1^J-J]<S1^J) And (S1^J<J1^J+1)) Or ((S1^J-J]<S1^J) And (S1^J>S1^J+1)) Then
Function Excess(J:Integer):Extended;
Begin
    Excess:=CorrectC*CorrectC*
             (Delta**[J-2]**+(Delta**[J-2]**-8*Delta**[J-1]**+8*Delta**[J-1]**))
             /12*(1/(D**[J]+S1**[J]));
End;

Procedure ComputeF2;
Begin
    F2**[J]**:= -(Ul**[J]+Ul**[J])/(2*DX)*Ul**[J];
    For J:=2 To JJ-2 Do
    Begin
        F2**[J]**:= -(ConvectiveTerm(J) -Excess(J) -(SpongeL**[J]+SpongeR**[J])
                   *A1**[J]*U1**[J] -FricCoef**[J]*U1**[J]*Abs(Ul**[J])/(D**[J]+S1**[J])/2;
    End;
    F2**[JJ-1]**:= -(Ul**[JJ-2]+Ul**[JJ])/(2*DX)*Ul**[JJ-1];
    F2**[J0]**:= F2**[J0]** + (8*SIP1-U1**[J0]+U1**[J0]+U1**[J0]+U1**[J0])/(12*DX);
    F2**[J0+1]**:= F2**[J0+1]** + (-SIM1+8*SO1-U1**[J0]+U1**[J0]+U1**[J0])/(12*DX);
    F2**[J0+2]**:= F2**[J0+2]** + (-SO1-U1**[J0]+U1**[J0]+U1**[J0])/12*DX;
End;

Procedure ComputeMeanSetupAndFlow;
Begin
    Assign(DataFile,FileDir+FileName+'M.Dat');
    Rewrite(DataFile);
    For J:=0 To JJ Do
    Begin
        HMean**[J]**:=HMean**[J]**/(1Sample-1);
    End;
End;
\begin{verbatim}
SMean'[J] := SMean'[J] / IMean;
QMean'[J] := QMean'[J] / IMean;
UMean'[J] := UMean'[J] / IMean;
End;
For J:=0 To JJ Do
  WriteLn(DataFile, LeftBoundary.X+J*DX:.12:.4, QMean'[J]:.12:.4,
           SMean'[J]:.12:.4, UMean'[J]:.12:.4, SMin'[J]:.12:.4, SMax'[J]:.12:.4,
           HMean'[J]/LeftBoundary.H:.12:.4, D'[J]/D[0]:.12:.4);
Close(DataFile);
End;
Procedure ComputeRelativeErrors;
Begin
  SumS:=0;
  SumU:=0;
  For J:=0 To JJ Do
    Begin
      SumS:=SumS+Abs(SI'[J]);
      SumU:=SumU+Abs(UI'[J]);
    End;
    If SumS>0 Then
      RelErrorS:=ErrorS/SumS
    Else
      RelErrorS:=0;
    If SumU>0 Then
      RelErrorU:=ErrorU/SumU
    Else
      RelErrorU:=0;
  End;
End;
Procedure ComputeSpongeCoefficients;
Const
  SpongeMax=0.75;
Begin
  For J:=0 To JJ Do
    Begin
      SpongeL'[J] := 0;
      SpongeR'[J] := 0;
    End;
  With LeftBoundary Do
    If JS>0 Then
      For J:=0 To JS Do
        SpongeL'[J] := SpongeMax*Sqr((JS-J)/JS);
  With RightBoundary Do
    If JS>0 Then
      For J:=JJ-JS To JJ Do
        SpongeR'[J] := SpongeMax*Sqr((J-(JJ-JS))/JS);
End;
Procedure Corrector;
Begin
  While ErrorExceedsLimit And (I<1Max) Do
    Begin
      CorrectElevation;
      CorrectorRightHandSideForU;
      DoubleSweep;
      ComputeRelativeErrors;
      ComputeF1;
      ComputeF2;
      ComputeF3;
      I:=I+1;
    End;
End;
Procedure CorrectElevation;
Begin
  For J:=2 To JJ-2 Do
    Begin

\end{verbatim}
ErrorS:=ErrorS+Abs(Surf-S1°[J]);
S1°[J]:=Surf;
End;

Surf:=S1°[0]/5+S1°[2]*2-S1°[3]*2 +S1°[4] -S1°[5]/5;
ErrorS:=ErrorS+Abs(Surf-S1°[1]);
S1°[1]:=Surf;

Surf:=S1°[1];
ErrorS:=ErrorS+Abs(Surf-S1°[0]);
S1°[0]:=Surf;

Surf:=-S1°[JJ-6]/5+S1°[JJ-4] -S1°[JJ-3]*2+S1°[JJ-2]*2 +S1°[JJ]/5;
ErrorS:=ErrorS+Abs(Surf-S1°[JJ-1]);
S1°[JJ-1]:=Surf;

Surf:=S1°[JJ-1];
ErrorS:=ErrorS+Abs(Surf-S1°[JJ]);
S1°[JJ]:=Surf;
End;

Procedure CorrectorRightHandSideForU;
Begin
For J:=1 To JJ-1 Do
R1°[J]:=RO°[J] +DT/24*(9*F2°[J]-[ 1] +19*F2°[J]-[ 0] - 5*F2°[J]-[-1] + F2°[J]-[-2]) + (F3°[J]-[1]-F3°[J]-[0]) / (D°[J]+S1°[J]);
End;

Procedure InitComputation;
Begin
ReadDataFromFile;
AllocateMemory;
NonOimensionalizeData;
ComputeBathymetry;
ReadFrictionCoefficients;
ComputeSpongeCoefficients;
ComputeTimeStep;
ComputeInitialCondition;
WaveParameters;
ComputeCoefficients;
CoefficientMatrix;
InitCounters;
ComputeMassAndEnergy;
InitialMass:=Mass;
ShowTime;
Assign(ControlFile,FileDir+FileName+'C.Dat'); Rewrite(ControlFile);
Assign(PrintFile,FileDir+FileName+'P.Dat'); Rewrite(PrintFile);
Assign(RollerFile,FileDir+FileName+'R.Dat'); Rewrite(RollerFile);
SaveUAndS;
End;

Procedure InitCounters;
APPENDIX B. CODE - ONE HORIZONTAL DIMENSION

Begin
ISample:=1;
IWave:=1;
RelErrorS:=1;
RelErrorU:=1;
IMean:=0;
IOut:=0;
N:=0;
End;

Procedure InitPredictor;
Begin
RelErrorS:=1;
RelErrorU:=1;
I:=0;
End;

Procedure PredictElevation;
Begin
For J:=2 To JJ-2 Do
  S1'(J):=S0'(J)
    +DT/12*(23*F1'(J)'[ 0]
      -16*F1'(J)'[ -1]
      + 5*F1'(J)'[-2]);
S1'(0):=S1'(1);
S1'(JJ-1):=S1'(JJ-6)/5+S1'(JJ-4)
    -S1'(JJ-3)*2+S1'(JJ-2)*2
    +S1'(JJ )/5;
S1'(JJ):=S1'(JJ-1);
End;

Procedure Predictor;
Begin
  InitPredictor;
  PredictElevation;
  PredictorRightHandSideForU;
  DoubleSweep;
  ComputeF1;
  ComputeF2;
  ComputeF3;
End;

Procedure PredictorRightHandSideForU;
Begin
For J:=1 To JJ-1 Do
  R1'(J):=R0'(J)
    +DT/12*(23*F2'(J)'[ 0]
      -16*F2'(J)'[ -1]
      + 5*F2'(J)'[-2]);
End;

Procedure UpdateComputation;
Begin
UpdateVariables;
If N Trunc(NN/NOut)=0 Then
  SaveUAndS;
  CheckMassEnergyAndIter;
  UpdateSetupAndFlow;
  ShowTime;
End;
Procedure UpdateSetupAndFlow;
Var
Q:Extended;
Begin
If (N>=SampleStart) And (N<=SampleStop) Then
Begin
IMean:=IMean+1;
For J:=0 To JJ Do
Begin
If SO−[J]>SMax−[J] Then SMax−[J]:=SO−[J];
If SO−[J]<SMin−[J] Then SMin−[J]:=SO−[J];
SMean−[J]:=SMean−[J]+SO−[J];
Case J-J0 Of
-2,-1,0:Q:=Flow(J0-3);
1,2,3 :Q:=Flow(J0+3);
Else
Q:=Flow(J);
End;
UMean−[J]:=UMean−[J]+(Q-CorrectC*Sqrt(D−[J])*Delta−[J])/(D−[J]+SO−[J]-Delta−[J]);
QMean−[J]:=QMean−[J]+Q;
End;
End;
With LeftBoundary Do
Begin
If (N*DT<IWave*T) And ((N+l)*DT>=IWave*T) Then
Begin
IWave:=IWave+l;
For J:=0 To JJ Do
SO−[J]:=SO−[J]-(Mass-InitialMass)/((JJ-1)*DX);
End;
If ( N *DT< SampleStart*DT+ISample*T) And
((N+l)*DT>=SampleStart*DT+ISample*T) Then
Begin
For J:=0 To JJ Do
Begin
HMean−[J]:=HMean−[J]+SMax−[J]-SMin−[J];
InitVector(J,0,SMin);
InitVector(J,0,SMax);
End;
ISample:=ISample+l;
End;
End;
For J:=0 To JJ Do Delta−[J]:=0;
End;
End;
End;
Procedure UpdateVariables;
Begin
For J:=0 To JJ Do
Begin
F1−[J]−[−2]:=F1−[J]−[−1];
F1−[J]−[−1]:=F1−[J]−[ 0];
F1−[J]−[ 0]:=F1−[J]−[ 1];
F2−[J]−[−2]:=F2−[J]−[−1];
F2−[J]−[−1]:=F2−[J]−[ 0];
F2−[J]−[ 0]:=F2−[J]−[ 1];
F3−[J]−[−1]:=F3−[J]−[ 0];
F3−[J]−[ 0]:=F3−[J]−[ 1];
R0−[J]:=R1−[J];
SO−[J]:=S1−[J];
U0−[J]:=U1−[J];
End;
N:=N+l;
ComputeNewRandS;
End;
Begin
End.
B.7 Variable.2D

Unit Variable;

Interface

{$D+, L+, N+, Y+}$

Const

FourRes = 46;
JMax = 5;
JJMax = 2000;
JSearch = 5;
JSMax = 200;
MaxIter = 100;
NNMax = 3E5;
TDelay = 30;
TFade = 10;
TSample = 20;
Cl = -0.531;
G = 9.796;
MaxMassError = 10.0;
MaxErr = 1E-8;
MaxIterError = 1E-3;

Type

TimeMemory = -2..1;
SpaceMemory = 0..JMax;
RowOne = Array[TimeMemory] Of Extended;
RowPtrOne = 'RowOne;
Matrix = Array[SpaceMemory] Of RowPtrOne;
MatrixType = 'Matrix;
RowTwo = Array[SpaceMemory] Of Extended;
VectorType = 'RowTwo;
RowThree = Array[1..FourRes] Of Extended;
RowPtrThree = 'RowThree;
Quadratic = Array[1..FourRes] Of RowPtrThree;
QuadMatrix = 'Quadratic;
WaveBreaking = (NonBreakingWave, NewBreakingWave, OldBreakingWave, EndBreakingWave);

LeftBoundaryType = Record

Current : Char;
R : Char;
Sp : Char;
W : Char;
A : Extended;
A1 : Extended;
A2 : Extended;
B : Extended;
C : Extended;
Curr : Extended;
D : Extended;
EA : Extended;
EM : Extended;
E1 : Extended;
E2 : Extended;
H : Extended;
X : Extended;
L : Extended;
EtaMin : Extended;
T : Extended;
JSrLongInt;
MM : Longint;
NFreq : LongInt;
Amplitude : VectorType;
Frequency : VectorType;
Phase : VectorType;
WaveNum : VectorType;
Z : VectorType;
End;
RightBoundaryType = Record
  D : Extended;
  X : Extended;
  JS : LongInt;
End;

Var
  LeftBoundary : LeftBoundaryType;
  RightBoundary : RightBoundaryType;
  Bathymetry : Char;
  Alpha : Extended;
  Alpha0 : Extended;
  AlphaB : Extended;
  CorrectC : Extended;
  Cr : Extended;
  DT : Extended;
  DX : Extended;
  DO : Extended;
  Energy : Extended;
  FDelta : Extended;
  HalfTime : Extended;
  InitialMass : Extended;
  LengthX : Extended;
  Mass : Extended;
  RelErrorS : Extended;
  RelErrorU : Extended;
  ErrorS : Extended;
  ErrorU : Extended;
  Phi : Extended;
  SIM1 : Extended;
  SI0 : Extended;
  SIP1 : Extended;
  SIP2 : Extended;
  StepWidth : Extended;
  SumS : Extended;
  Surf : Extended;
  SumU : Extended;
  TB : Extended;
  THalf : Extended;
  TMax : Extended;
  UIM1 : Extended;
  U10 : Extended;
  UIP1 : Extended;
  UIP2 : Extended;
  Vel : Extended;
  I : LongInt;
  IFreq : LongInt;
  IMean : LongInt;
  IOut : LongInt;
  ISample : LongInt;
  IWave : LongInt;
  J : LongInt;
  J0 : LongInt;
  J1 : LongInt;
  J2 : LongInt;
  J3 : LongInt;
  J1Old : LongInt;
  J2Old : LongInt;
  JJ : LongInt;
  JStart : LongInt;
  JStop : LongInt;
  N : LongInt;
  NH : LongInt;
  NN : LongInt;
  NRI : LongInt;
APPENDIX B. CODE - ONE HORIZONTAL DIMENSION

NOut : Longint;
SampleStart : Longint;
SampleStop : Longint;
F1 : MatrixType;
F2 : MatrixType;
F3 : MatrixType;
Coef : QuadMatrix;
BathyFile : String;
FileDir : String;
FileName : String;
FrictionFile : String;
ControlFile : Text;
DataFile : Text;
PrintFile : Text;
RollerFile : Text;
A1 : VectorType;
A2 : VectorType;
A3 : VectorType;
A4 : VectorType;
A5 : VectorType;
A6 : VectorType;
A7 : VectorType;
D : VectorType;
Delta : VectorType;
DF : VectorType;
DX1 : VectorType;
DX2 : VectorType;
D1 : VectorType;
D2 : VectorType;
D3 : VectorType;
E : VectorType;
F : VectorType;
FricCoef : VectorType;
HMean : VectorType;
QMean : VectorType;
RHS : VectorType;
RO : VectorType;
R1 : VectorType;
SMax : VectorType;
SMean : VectorType;
SMin : VectorType;
SpongeL : VectorType;
SpongeR : VectorType;
SO : VectorType;
SI : VectorType;
TBreak : VectorType;
UMean : VectorType;
UO : VectorType;
Ul : VectorType;

Implementation

Begin
End.
B.8 Various.2D

Unit Various;

Interface

\{D+,L+,N+,Y+\}

Uses

\begin{align*}
& \text{Variable, Crt;}
& \text{Procedure AllocateMatrix(Var M:MatrixType);}
& \text{Procedure AllocateMemory;}
& \text{Procedure AllocateQuadratic(Var M:QuadMatrix);}
& \text{Procedure CheckMassEnergyAndIter;}
& \text{Procedure ComputeMassAndEnergy;}
& \text{Procedure ComputeTimeStep;}
& \text{Procedure DeAllocateQuadratic(Var M:QuadMatrix);}
& \text{Procedure DeAllocateVector(Var V:VectorType);}
& \text{Procedure DeAllocateMatrix(Var M:MatrixType);}
& \text{Procedure DeAllocateMemory;}
& \text{Procedure DeAllocateVector(Var V:VectorType);}
& \text{Procedure EndComputation;}
& \text{Procedure Error(ErrCode:Integer);}
& \text{Procedure NonDimensionalizeData;}
& \text{Procedure ReadDataFromFile;}
& \text{Procedure SaveBathymetry;}
& \text{Procedure SaveRollerTrace(JA,JB:Integer;ControlString:String);}
& \text{Procedure SaveUAndS;}
& \text{Procedure ShowTime;}
\end{align*}

Implementation

\begin{align*}
\text{Procedure AllocateMatrix(Var M:MatrixType);}
& \text{Begin}
& \quad \text{If MaxAvail} \lt \text{SizeOf(M) Then}
& \quad \quad \text{Error(20)}
& \quad \text{Else}
& \quad \quad \text{Begin}
& \quad \quad \quad \text{GetMem(M,(JJMax+1)*SizeOf(RowPtrOne));}
& \quad \quad \quad \text{For J:=0 To JJMax Do}
& \quad \quad \quad \quad \text{GetMem(M^J,SizeOf(RowOne));}
& \quad \quad \quad \text{End;}
& \quad \quad \text{End;}
& \text{End;}
\end{align*}

\begin{align*}
\text{Procedure AllocateMemory;}
& \text{Begin}
& \quad \text{AllocateVector(A1);}
& \quad \text{AllocateVector(A2);}
& \quad \text{AllocateVector(A3);}
& \quad \text{AllocateVector(A4);}
& \quad \text{AllocateVector(A5);}
& \quad \text{AllocateVector(A6);}
& \quad \text{AllocateVector(A7);}
& \quad \text{AllocateVector(D);}
& \quad \text{AllocateVector(DX1);}
& \quad \text{AllocateVector(DX2);}
& \quad \text{AllocateVector(Dl);}
& \quad \text{AllocateVector(D2);}
& \quad \text{AllocateVector(D3);}
& \quad \text{AllocateVector(E);}
& \quad \text{AllocateVector(F);}
& \quad \text{AllocateVector(FricCoef);}
& \quad \text{AllocateVector(HMean);}
& \quad \text{AllocateVector(QMean);}
& \quad \text{AllocateVector(RO);}
& \quad \text{AllocateVector(R1);}
\end{align*}
AllocateVector(SMax);
AllocateVector(SMean);
AllocateVector(SMin);
AllocateVector(SD);
AllocateVector(SpongeL);
AllocateVector(SpongeR);
AllocateVector(SI);
AllocateVector(TBreak);
AllocateVector(Delta);
AllocateVector(UMean);
AllocateVector(UO);
AllocateVector(U1);

With LeftBoundary Do
  If UpCase(R)='I' Then
    Begin
      AllocateVector(Phase);
      AllocateVector(Amplitude);
      AllocateVector(Frequency);
      AllocateVector(WaveNum);
      End;
    AllocateMatrix(Fl);
    AllocateMatrix(F2);
    AllocateMatrix(F3);
  End;

Procedure AllocateQuadratic(Var M:QuadMatrix);
Begin
  If MaxAvail<SizeOf(M) Then
    Halt(1)
  Else
    Begin
      GetMem(M,FourRes*SizeOf(RowPtrThree));
      For J:=1 To FourRes Do
        GetMem(M'
          [J],SizeOf(RowThree));
      End;
    End;

Procedure AllocateVector(Var V:VectorType);
Begin
  If MaxAvail<SizeOf(V) Then
    Error(10)
  Else
    GetMem(V,SizeOf(RowTwo));
  End;

Procedure DeAllocateMatrix(Var M:MatrixType);
Begin
  For J:=0 To JJMax Do
    FreeMem(M'
      [J],SizeOf(RowOne));
  FreeMem(M,(JJMax+1)*SizeOf(RowPtrOne));
End;

Procedure DeAllocateMemory;
Begin
  DeAllocateVector(A1);
  DeAllocateVector(A2);
  DeAllocateVector(A3);
  DeAllocateVector(A4);
  DeAllocateVector(A5);
  DeAllocateVector(A6);
  DeAllocateVector(A7);
  DeAllocateVector(D);
  DeAllocateVector(D1);
  DeAllocateVector(D2);
  DeAllocateVector(D1);
  DeAllocateVector(D2);
  DeAllocateVector(D3);
  DeAllocateVector(E);
DeAllocateVector(F);
DeAllocateVector(FricCoef);
DeAllocateVector(HMean);
DeAllocateVector(RO);
DeAllocateVector(R1);
DeAllocateVector(SMax);
DeAllocateVector(SMean);
DeAllocateVector(SMin);
DeAllocateVector(SO);
DeAllocateVector(SpongeL);
DeAllocateVector(SpongeR);
DeAllocateVector(Sl);
DeAllocateVector(TBreak);
DeAllocateVector(Delta);
DeAllocateVector(UMean);
DeAllocateVector(UI);
DeAllocateMatrix(Fl);
DeAllocateMatrix(F2);
DeAllocateMatrix(F3);
End;

With LeftBoundary Do
Begin
  If UpCase(W)='F' Then
  DeAllocateVector(Z);
  If UpCase(R)='I' Then
  Begin
    DeAllocateVector(Phase);
    DeAllocateVector(Amplitude);
    DeAllocateVector(Frequency);
    DeAllocateVector(WaveNum);
  End;
End;

DeAllocateMatrix(Fl);
DeAllocateMatrix(F2);
DeAllocateMatrix(F3);
End;

Procedure CheckMassEnergyAndIter;
Var
  MassError:Extended;
Begin
  ComputeMassAndEnergy;
  MassError:=(Mass/InitialMass-1)*100;
  If Abs(MassError)>MaxMassError Then
  Begin
    Close(ControlFile);
    Close(PrintFile);
    Close(RollerFile);
    Error(140)
  End
  Else
  Begin
    WriteLn(ControlFile,N*DT:12:4,I:6,MassError:12:4,Energy:12:4,
    Sl-[0]:12:4,S1-[JJ]:12:4);
    GotoXY(40,6);
    Write('Percentage mass error: ',MassError:0:3);
    GotoXY(40,7);
    Write('Number of corrections: ',I:3);
  End;
End;

Procedure ComputeMassAndEnergy;
Begin
  Mass:=0.5*(D-[0]+30-[0]+0-[JJ]+30-[JJ] +U0-[0]*U0-[0]*D-[0]+30-[0]*30-[0]
             +U0-[JJ]*U0-[JJ]*D-[JJ]+30-[JJ]*30-[JJ]);
  For J:=1 To JJ-1 Do
    Mass:=Mass+D*[J]+30*[J];
  Mass:=Mass*DX;
  Energy:=0.5*(U0-[0]*U0-[0]*D-[0]+30-[0]*30-[0] +U0-[JJ]*U0-[JJ]*D-[JJ]+30-[JJ]*30-[JJ]);
  For J:=1 To JJ-1 Do
Energy := Energy + U0\[J\]*D\[J\]*S0\[J\];
Energy := 0.5*Energy*DX;
End;

Procedure ComputeTimeStep;
Var
MaxDepth: Extended;
Begin
MaxDepth := D\[0\];
For J:=1 To JJ Do
If D\[J\] > MaxDepth Then
MaxDepth := D\[J\];
DT := Cr*DX/Sqrt(MaxDepth);
NN := 1 + Trunc(TMax/DT);
If NN > NNMax Then Error(50);
End;

Procedure DeAllocateQuadratic(Var M: QuadMatrix);
Begin
For J:=1 To FourRes Do
FreeMem(M\[J\], SizeOf(RowThree));
FreeMem(M, FourRes*SizeOf(RowPtrThree));
End;

Procedure DeAllocateVector(Var V: VectorType);
Begin
FreeMem(V, SizeOf(RowTwo));
End;

Procedure EndComputation;
Begin
DeAllocateMemory;
Close(ControlFile);
Close(PrintFile);
Close(RollerFile);
Error(0);
End;

Procedure Error(ErrCode: Integer);
Begin
GoToXY(1,25);
Case ErrCode Of
0: Write('Computation complete...');
10: Write('>>Error 10<< Not enough memory to allocate a vector...');
20: Write('>>Error 20<< Not enough memory to allocate a matrix...');
30: Write('>>Error 30<< The number of discretization points exceeds ',
JMax,'...');
40: Write('>>Error 40<< The number of sponge layer nodes exceeds ',
JSMax,'...');
50: Write('>>Error 50<< The number of time steps exceeds ',NNMax,'...');
60: Write('>>Error 60<< The number of frequency components exceeds ',
JMax,'...');
70: Write('>>Error 70<< Energy of specified wave spectrum not within ',
'allowed frequency range...');
80: Write('>>Error 80<< Singular system of equations in Fourier ',
'algorithm...');
90: Write('>>Error 90<< No convergence in Fourier algorithm...');
100: Write('>>Error 100<< No convergence in sinusoidal wave dispersion ',
'relation...');
110: Write('>>Error 110<< No convergence in solitary wave dispersion ',
'relation...');
120: Write('>>Error 120<< The program failed to locate a roller...');
130: Write('>>Error 130<< Circular roller determination...');
140: Write('>>Error 140<< Mass conservation failure (more than ',
'MaxMassError:4:2,' '%)...');
End;
Halt(1);
End;
Procedure NonDimensionalizeData;
Begin
With LeftBoundary Do
Begin
DO:=D;
X:=X/DO;
D:=D/DO;
H:=H/DO;
T:=T*Sqrt(G/DO);
THalf:=T*HalfTime;
Curr:=Curr/Sqrt(G*DO);
End;
With RightBoundary Do
Begin
X:=X/DO;
D:=D/DO;
End;
If UpCase(Bathymetry)='P' Then
StepWidth:=StepWidth/DO;
TMax:=TMax*Sqrt(G/DO);
LengthX:=(RightBoundary.X-LeftBoundary.X);
DX:=LengthX/JJ;
End;
Procedure ReadDataFromFile;
Procedure Initialize;
Begin
ClrScr;
WriteLnC'Program ABM executing...');
WriteLn;
WriteCDirectory for input and output
Directory for input and output
GetDir(0,FileDir);
FileDir:=FileDir+'\';
WriteLn(FileDir);
WriteCRead input from file (excl. extension): ');
ReadLn(FileName);
Assign(DataFile,FileDir+FileName+'.Dat');
Reset(DataFile);
GotoXY(1,10);
WriteLnC'Initializing...');
End;
Procedure BathymetryData;
Begin
ReadLnCDataFile,Bathymetry);
If UpCase(Bathymetry)='P' Then
ReadLnCDataFile,StepWidth);
If UpCase(Bathymetry)='U' Then
ReadLnCDataFile,BathyFile);
ReadLnCDataFile,FrictionFile);
ReadLnCDataFile,JJ);
End;
Procedure LeftBoundaryData;
Begin
With LeftBoundary Do
Begin
ReadLn(DataFile,X);
ReadLn(DataFile,D);
ReadLn(DataFile,JS);
RJ:=JS+3;
Case UpCase(R) Of
'I':Begin
ReadLn(DataFile,Sp);
ReadLn(DataFile,NFreq);
APPENDIX B. CODE - ONE HORIZONTAL DIMENSION

```
ReadLn(DataFile,H);
ReadLn(DataFile,T);
End;
'R':Begin
ReadLn(DataFile,W);
ReadLn(DataFile,H);
If UpCase(W) In ['C', 'O', 'F'] Then
  ReadLn(DataFile,T);
If UpCase(W)='F' Then
  Begin
    ReadLn(DataFile,Current);
    ReadLn(DataFile,Curr);
    ReadLn(DataFile,MM);
    ReadLn(DataFile,NH);
    ReadLn(DataFile,NRI);
  End;
End;
End;
End;
Procedure RightBoundaryData;
Begin
  With RightBoundary Do
    Begin
      ReadLn(DataFile.X);
      ReadLn(DataFile.D);
    End;
  End;
End;
Procedure GeneralData;
Begin
  ReadLn(DataFile,NOut);
  ReadLn(DataFile,TMax);
  ReadLn(DataFile,Cr);
  ReadLn(DataFile,AlphaA);
  ReadLn(DataFile,AlphaB);
  ReadLn(DataFile,FDelta);
  ReadLn(DataFile,HalfTime);
  ReadLn(DataFile,CorrectC);
End;
Begin
  Initialize;
  BathymetryData;
  LeftBoundaryData;
  RightBoundaryData;
  GeneralData;
  Close(DataFile);
End;
Procedure SaveBathymetry;
Var
  St: String;
Begin
  St:=PileDir+FileName+'b.dat';
  Assign(DataFile,St);
  Rewrite(DataFile);
  For J:=0 To JJ Do
    WriteLn(DataFile,LeftBoundary.X+J*DX:12:4,0:12,-D'[J]:12:4);
  Close(DataFile);
End;
Procedure SaveRollerTrace(JA,JB:Integer; ControlString:String);
Begin
  For J:=JA To JB Do
    WriteLn(RollerFile,LeftBoundary.X+J*DX:12:4,N*DT:12:4,
```
B.8. Various 2D

TBreak\^[J\]:12:4, Delta\^[J\]:12:4, Phi:12:4, ControlString:6);
End;

Procedure SaveUAndS;
Var
  Number, St: String;
Begin
  GotoXY(1, 10);
  ClrEol;
  Str(IOOut:0, Number);
  St:=FileName+Number+' .dat';
  Writeln('Saving computed data in file '+'St'+ at time t=','N*DT:6:2,'...');
  WriteLn(PrintFile, IOOut:6, N*DT:12:4);
  Assign(DataFile, St);
  Rewrite(DataFile);
  For J:=0 To JJ Do
    Writeln(DataFile, LeftBoundary.X+J*DX:12:4, U0\^[J\]:12:4, 
             SO\^[J\]:12:4);
  Close(DataFile);
  IOOut:=IOOut+1;
End;

Procedure ShowTime;
Var
  St: String;
Begin
  Str(N*DT:0:2, St);
  GotoXY(1, 6);
  Write('Elapsed time .........: '+'St');
  GotoXY(1, 7);
  Str(TMax:0:2, St);
  Write('Total computation time: '+'St');
End;
Begin
End.
Appendix C

Code - Two Horizontal Dimensions

C.1 Abm.3D

Program ABM;
{$D+,L+,N+,Y+}$

Uses
  Boundary,
  Fourier,
  Grid,
  Solution,
  Variable,
  Various,
  Crt;

Begin
  InitComputation;
  While N<=NN Do
    Begin
      Predictor;
      Corrector;
      UpdateComputation;
    End;
  EndComputation;
End.
APPENDIX C. CODE - TWO HORIZONTAL DIMENSIONS

C.2 Boundary.3D

Unit Boundary;
Interface
{$D+,L+,N+,Y+}$
Uses
Variable,
Various;

Procedure ComputeInitialCondition;
Procedure PredictNewBoundary;
Procedure CorrectNewBoundary;
Procedure PAbsorb11(I,J:Integer);
Procedure PAbsorb12(I,J:Integer);
Procedure PAbsorb13(I,J:Integer);
Procedure PAbsorb14(I,J:Integer);
Procedure PAbsorb15(I,J:Integer);
Procedure PAbsorb21(I,J:Integer);
Procedure PAbsorb25(I,J:Integer);
Procedure PAbsorb31(I,J:Integer);
Procedure PAbsorb35(I,J:Integer);
Procedure PAbsorb41(I,J:Integer);
Procedure PAbsorb45(I,J:Integer);
Procedure PAbsorb51(I,J:Integer);
Procedure PAbsorb52(I,J:Integer);
Procedure PAbsorb53(I,J:Integer);
Procedure PAbsorb54(I,J:Integer);
Procedure PAbsorb55(I,J:Integer);
Procedure CAbsorb11(I,J:Integer);
Procedure CAbsorb12(I,J:Integer);
Procedure CAbsorb13(I,J:Integer);
Procedure CAbsorb14(I,J:Integer);
Procedure CAbsorb21(I,J:Integer);
Procedure CAbsorb25(I,J:Integer);
Procedure CAbsorb31(I,J:Integer);
Procedure CAbsorb35(I,J:Integer);
Procedure CAbsorb41(I,J:Integer);
Procedure CAbsorb45(I,J:Integer);
Procedure CAbsorb51(I,J:Integer);
Procedure CAbsorb52(I,J:Integer);
Procedure CAbsorb53(I,J:Integer);
Procedure CAbsorb54(I,J:Integer);
Procedure CAbsorb55(I,J:Integer);
Procedure CReflect11(I,J:Integer);
Procedure CReflect12(I,J:Integer);
Procedure CReflect13(I,J:Integer);
Procedure CReflect14(I,J:Integer);
Procedure CReflect15(I,J:Integer);
Procedure CReflect21(I,J:Integer);
Procedure CReflect25(I,J:Integer);
Procedure CReflect31(I,J:Integer);
Procedure CReflect35(I,J:Integer);
Procedure CReflect41(I,J:Integer);
Procedure CReflect45(I,J:Integer);
Procedure CReflect51(I,J:Integer);
Procedure CReflect52(I,J:Integer);
Procedure CReflect53(I,J:Integer);
Procedure CReflect54(I,J:Integer);
Procedure CReflect55(I,J:Integer);
Procedure CReflect11(I,J:Integer);
Procedure CReflect12(I,J:Integer);
Procedure CReflect13(I,J:Integer);
Procedure CReflect14(I,J:Integer);
Procedure CReflect15(I,J:Integer);
C.2. Boundary: 3D

Procedure CReflect21(I,J: Integer);
Procedure CReflect25(I,J: Integer);
Procedure CReflect31(I,J: Integer);
Procedure CReflect35(I,J: Integer);
Procedure CReflect41(I,J: Integer);
Procedure CReflect45(I,J: Integer);
Procedure CReflect51(I,J: Integer);
Procedure CReflect52(I,J: Integer);
Procedure CReflect53(I,J: Integer);
Procedure CReflect54(I,J: Integer);
Procedure CReflect55(I,J: Integer);

Implementation

Procedure ComputeInitialCondition;
Begin
Write(PrintFile,'Procedure ComputeInitialCondition.......
InitMatrix(0,HMax);
InitMatrix(0,HMin);
InitMatrix(0,ROX);
InitMatrix(0,R1X);
InitMatrix(0,R0Y);
InitMatrix(0,R1Y);
InitMatrix(1, Mu);
InitMatrix(0, So);
InitMatrix(0, S1);
InitMatrix(0, S1Tot);
InitMatrix(0, S1Ref);
InitMatrix(0, u0);
InitMatrix(0, u1);
InitMatrix(0, u1Tot);
InitMatrix(0, u1Ref);
InitMatrix(0, v0);
InitMatrix(0, v1);
InitMatrix(0, v1Tot);
InitMatrix(0, v1Ref);
InitMatrix2(0, F1);
InitMatrix2(0, F2X);
InitMatrix2(0, F2Y);
InitMatrix2(0, F2XT);
InitMatrix2(0, F2YT);
For I:=0 To II Do
For J:=0 To JJ Do
If NodeType[I][J]=0 Then
Begin
S0[I][J]:=H;
S1[I][J]:=H;
End;
WriteLn(PrintFile,' - Ok');
End;

Procedure PredictNewBoundary;
Begin
For I:=0 To II Do
For J:=0 To JJ Do
Case NodeType[I][J] Of
-11: PReflect11(I,J);
-12: PReflect12(I,J);
-13: PReflect13(I,J);
-14: PReflect14(I,J);
-15: PReflect15(I,J);
-21: PReflect21(I,J);
-25: PReflect25(I,J);
-31: PReflect31(I,J);
-35: PReflect35(I,J);
-41: PReflect41(I,J);
-45: PReflect45(I,J);
-51: PReflect51(I,J);
APPENDIX C. CODE - TWO HORIZONTAL DIMENSIONS

-52: PReflect52(I,J);
-53: PReflect53(I,J);
-54: PReflect54(I,J);
-55: PReflect55(I,J);
11: PAbsorb11(I,J);
12: PAbsorb12(I,J);
13: PAbsorb13(I,J);
14: PAbsorb14(I,J);
15: PAbsorb15(I,J);
21: PAbsorb21(I,J);
25: PAbsorb25(I,J);
31: PAbsorb31(I,J);
35: PAbsorb35(I,J);
41: PAbsorb41(I,J);
45: PAbsorb45(I,J);
51: PAbsorb51(I,J);
52: PAbsorb52(I,J);
53: PAbsorb53(I,J);
54: PAbsorb54(I,J);
55: PAbsorb55(I,J);

End;

Procedure CorrectNewBoundary;
Begin
For I:=0 To II Do
For J:=0 To JJ Do
Case NodeType[I][J] Of
-11: CReflect11(I,J);
-12: CReflect12(I,J);
-13: CReflect13(I,J);
-14: CReflect14(I,J);
-15: CReflect15(I,J);
-21: CReflect21(I,J);
-25: CReflect25(I,J);
-31: CReflect31(I,J);
-35: CReflect35(I,J);
-41: CReflect41(I,J);
-45: CReflect45(I,J);
-51: CReflect51(I,J);
-52: CReflect52(I,J);
-53: CReflect53(I,J);
-54: CReflect54(I,J);
-55: CReflect55(I,J);
11: CAbsorb11(I,J);
12: CAbsorb12(I,J);
13: CAbsorb13(I,J);
14: CAbsorb14(I,J);
15: CAbsorb15(I,J);
21: CAbsorb21(I,J);
25: CAbsorb25(I,J);
31: CAbsorb31(I,J);
35: CAbsorb35(I,J);
41: CAbsorb41(I,J);
45: CAbsorb45(I,J);
51: CAbsorb51(I,J);
52: CAbsorb52(I,J);
53: CAbsorb53(I,J);
54: CAbsorb54(I,J);
55: CAbsorb55(I,J);
End;

Procedure PAbsorb1(I,J: Integer);
Begin
R1X[I][J]:=0;
R1Y[I][J]:=0;
S1[I][J]:=0;
End;
End;

Procedure PAbsorb12(I,J:Integer);
Begin
  R1X'[I]'[J]:=0;
  R1Y'[I]'[J]:=0;
  S1'[I]'[J]:=0;
End;

Procedure PAbsorb13(I,J:Integer);
Begin
  R1X'[I]'[J]:=0;
  R1Y'[I]'[J]:=0;
  S1'[I]'[J]:=0;
End;

Procedure PAbsorb14(I,J:Integer);
Begin
  R1X'[I]'[J]:=0;
  R1Y'[I]'[J]:=0;
  S1'[I]'[J]:=0;
End;

Procedure PAbsorb15(I,J:Integer);
Begin
  R1X'[I]'[J]:=0;
  R1Y'[I]'[J]:=0;
  S1'[I]'[J]:=0;
End;

Procedure PAbsorb21(I,J:Integer);
Begin
  R1X'[I]'[J]:=0;
  R1Y'[I]'[J]:=0;
  S1'[I]'[J]:=0;
End;

Procedure PAbsorb25(I,J:Integer);
Begin
  R1X'[I]'[J]:=0;
  R1Y'[I]'[J]:=0;
  S1'[I]'[J]:=0;
End;

Procedure PAbsorb31(I,J:Integer);
Begin
  R1X'[I]'[J]:=0;
  R1Y'[I]'[J]:=0;
  S1'[I]'[J]:=0;
End;

Procedure PAbsorb35(I,J:Integer);
Begin
  R1X'[I]'[J]:=0;
  R1Y'[I]'[J]:=0;
  S1'[I]'[J]:=0;
End;

Procedure PAbsorb41(I,J:Integer);
Begin
  R1X'[I]'[J]:=0;
  R1Y'[I]'[J]:=0;
  S1'[I]'[J]:=0;
End;

Procedure PAbsorb45(I,J:Integer);
Begin
  R1X'[I]'[J]:=0;

```plaintext
R1Y[I][J] := 0;
S1[I][J] := 0;
End;

Procedure PAbsorb51(I,J:Integer);
Begin
R1X[I][J] := 0;
R1Y[I][J] := 0;
S1[I][J] := 0;
End;

Procedure PAbsorb52(I,J:Integer);
Begin
R1X[I][J] := 0;
R1Y[I][J] := 0;
S1[I][J] := 0;
End;

Procedure PAbsorb53(I,J:Integer);
Begin
R1X[I][J] := 0;
R1Y[I][J] := 0;
S1[I][J] := 0;
End;

Procedure PAbsorb54(I,J:Integer);
Begin
R1X[I][J] := 0;
R1Y[I][J] := 0;
S1[I][J] := 0;
End;

Procedure PAbsorb55(I,J:Integer);
Begin
R1X[I][J] := 0;
R1Y[I][J] := 0;
S1[I][J] := 0;
End;

Procedure CAbsorb11(I,J:Integer);
Begin
R1X[I][J] := 0;
R1Y[I][J] := 0;
S1[I][J] := 0;
End;

Procedure CAbsorb12(I,J:Integer);
Begin
R1X[I][J] := 0;
R1Y[I][J] := 0;
S1[I][J] := 0;
End;

Procedure CAbsorb13(I,J:Integer);
Begin
R1X[I][J] := 0;
R1Y[I][J] := 0;
S1[I][J] := 0;
End;

Procedure CAbsorb14(I,J:Integer);
Begin
R1X[I][J] := 0;
R1Y[I][J] := 0;
S1[I][J] := 0;
End;

Procedure CAbsorb15(I,J:Integer);
```
Begin
R1X*[I]*[J] := 0;
R1Y*[I]*[J] := 0;
S1*[I]*[J] := 0;
End;

Procedure CAbsorb21(I,J:Integer);
Begin
R1X*[I]*[J] := 0;
R1Y*[I]*[J] := 0;
S1*[I]*[J] := 0;
End;

Procedure CAbsorb25(I,J:Integer);
Begin
R1X*[I]*[J] := 0;
R1Y*[I]*[J] := 0;
S1*[I]*[J] := 0;
End;

Procedure CAbsorb31(I,J:Integer);
Begin
R1X*[I]*[J] := 0;
R1Y*[I]*[J] := 0;
S1*[I]*[J] := 0;
End;

Procedure CAbsorb35(I,J:Integer);
Begin
R1X*[I]*[J] := 0;
R1Y*[I]*[J] := 0;
S1*[I]*[J] := 0;
End;

Procedure CAbsorb41(I,J:Integer);
Begin
R1X*[I]*[J] := 0;
R1Y*[I]*[J] := 0;
S1*[I]*[J] := 0;
End;

Procedure CAbsorb45(I,J:Integer);
Begin
R1X*[I]*[J] := 0;
R1Y*[I]*[J] := 0;
S1*[I]*[J] := 0;
End;

Procedure CAbsorb51(I,J:Integer);
Begin
R1X*[I]*[J] := 0;
R1Y*[I]*[J] := 0;
S1*[I]*[J] := 0;
End;

Procedure CAbsorb52(I,J:Integer);
Begin
R1X*[I]*[J] := 0;
R1Y*[I]*[J] := 0;
S1*[I]*[J] := 0;
End;

Procedure CAbsorb53(I,J:Integer);
Begin
R1X*[I]*[J] := 0;
R1Y*[I]*[J] := 0;
S1*[I]*[J] := 0;
End;
Procedure CAbsorb54(I,J:Integer);
Begin
  R1X'[I]'[J]:=0;
  R1Y'[I]'[J]:=0;
  S1'[I]'[J]:=0;
End;

Procedure CAbsorb55(I,J:Integer);
Begin
  R1X'[I]'[J]:=0;
  R1Y'[I]'[J]:=0;
  S1'[I]'[J]:=0;
End;

Procedure PReflect11(I,J:Integer);
Begin
  R1X'[I]'[J]:=0;
  R1Y'[I]'[J]:=0;
  S1'[I]'[J]:=S1'[I+1]'[J+1];
End;

Procedure PReflect12(I,J:Integer);
Begin
  R1X'[I]'[J]:=0;
  R1Y'[I]'[J]:=R1Y'[I+1]'[J];
  S1'[I]'[J]:=S1'[I+1]'[J];
End;

Procedure PReflect13(I,J:Integer);
Begin
  R1X'[I]'[J]:=0;
  R1Y'[I]'[J]:=R1Y'[I+1]'[J];
  S1'[I]'[J]:=S1'[I+1]'[J];
End;

Procedure PReflect14(I,J:Integer);
Begin
  R1X'[I]'[J]:=0;
  R1Y'[I]'[J]:=R1Y'[I+1]'[J];
  S1'[I]'[J]:=S1'[I+1]'[J];
End;

Procedure PReflect15(I,J:Integer);
Begin
  R1X'[I]'[J]:=0;
  R1Y'[I]'[J]:=0;
  S1'[I]'[J]:=S1'[I+1]'[J-1];
End;

Procedure PReflect21(I,J:Integer);
Begin
  R1X'[I]'[J]:=R1X'[I]'[J+1];
  R1Y'[I]'[J]:=0;
  S1'[I]'[J]:=S1'[I]'[J+1];
End;

Procedure PReflect25(I,J:Integer);
Begin
  R1X'[I]'[J]:=R1X'[I]'[J-1];
  R1Y'[I]'[J]:=0;
  S1'[I]'[J]:=S1'[I]'[J-1];
End;

Procedure PReflect31(I,J:Integer);
Begin
  R1X'[I]'[J]:=R1X'[I]'[J+1];
  R1Y'[I]'[J]:=0;
C.2. Boundary: 3D

\[ S_{1} \cdot [i] \cdot [j] := S_{1} \cdot [i] \cdot [j+1]; \]
End;

Procedure PReflect35(I, J: Integer);
Begin
\[ R_{1X}[i][j] := R_{1X}[i][j-1]; \]
\[ R_{1Y}[i][j] := 0; \]
\[ S_{1}[i][j] := S_{1}[i][j-1]; \]
End;

Procedure PReflect41(I, J: Integer);
Begin
\[ R_{1X}[i][j] := R_{1X}[i][j+1]; \]
\[ R_{1Y}[i][j] := 0; \]
\[ S_{1}[i][j] := S_{1}[i][j+1]; \]
End;

Procedure PReflect45(I, J: Integer);
Begin
\[ R_{1X}[i][j] := R_{1X}[i][j+1]; \]
\[ R_{1Y}[i][j] := 0; \]
\[ S_{1}[i][j] := S_{1}[i][j+1]; \]
End;

Procedure PReflect51(I, J: Integer);
Begin
\[ R_{1X}[i][j] := 0; \]
\[ R_{1Y}[i][j] := 0; \]
\[ S_{1}[i][j] := S_{1}[i-1][j+1]; \]
End;

Procedure PReflect52(I, J: Integer);
Begin
\[ R_{1X}[i][j] := 0; \]
\[ R_{1Y}[i][j] := R_{1Y}[i-1][j]; \]
\[ S_{1}[i][j] := S_{1}[i-1][j]; \]
End;

Procedure PReflect53(I, J: Integer);
Begin
\[ R_{1X}[i][j] := 0; \]
\[ R_{1Y}[i][j] := R_{1Y}[i-1][j]; \]
\[ S_{1}[i][j] := S_{1}[i-1][j]; \]
End;

Procedure PReflect54(I, J: Integer);
Begin
\[ R_{1X}[i][j] := 0; \]
\[ R_{1Y}[i][j] := R_{1Y}[i-1][j]; \]
\[ S_{1}[i][j] := S_{1}[i-1][j]; \]
End;

Procedure PReflect55(I, J: Integer);
Begin
\[ R_{1X}[i][j] := 0; \]
\[ R_{1Y}[i][j] := 0; \]
\[ S_{1}[i][j] := S_{1}[i-1][j]; \]
End;

Procedure CReflect11(I, J: Integer);
Begin
\[ R_{1X}[i][j] := 0; \]
\[ R_{1Y}[i][j] := 0; \]
\[ Surf := S_{1}[i+1][j+1]; \]
\[ Errors := ErrorS + Abs(Surf - S_{1}[i][j]); \]
\[ S_{1}[i][j] := Surf; \]
End;
Procedure CReflect12(I,J:Integer);
Begin
R1X'[I]'[J]:=0;
R1Y'[I]'[J]:=R1Y'[I+1]'[J];
Surf:=S1'[I+1]'[J];
Errors:=Errors+Abs(Surf-S1'[I]'[J]);
S1'[I]'[J]:=Surf;
End;

Procedure CReflect13(I,J:Integer);
Begin
R1X'[I]'[J]:=0;
R1Y'[I]'[J]:=R1Y'[I+1]'[J];
Surf:=S1'[I+1]'[J];
Errors:=Errors+Abs(Surf-S1'[I]'[J]);
S1'[I]'[J]:=Surf;
End;

Procedure CReflect14(I,J:Integer);
Begin
R1X'[I]'[J]:=0;
R1Y'[I]'[J]:=R1Y'[I+1]'[J];
Surf:=S1'[I+1]'[J];
Errors:=Errors+Abs(Surf-S1'[I]'[J]);
S1'[I]'[J]:=Surf;
End;

Procedure CReflect15(I,J:Integer);
Begin
R1X'[I]'[J]:=0;
R1Y'[I]'[J]:=0;
Surf:=S1'[I+1]'[J-1];
Errors:=Errors+Abs(Surf-S1'[I]'[J]);
S1'[I]'[J]:=Surf;
End;

Procedure CReflect21(I,J:Integer);
Begin
R1X'[I]'[J]:=R1X'[I]'[J+1];
R1Y'[I]'[J]:=0;
Surf:=S1'[I]'[J+1];
Errors:=Errors+Abs(Surf-S1'[I]'[J]);
S1'[I]'[J]:=Surf;
End;

Procedure CReflect25(I,J:Integer);
Begin
R1X'[I]'[J]:=R1X'[I]'[J-1];
R1Y'[I]'[J]:=0;
Surf:=S1'[I]'[J-1];
Errors:=Errors+Abs(Surf-S1'[I]'[J]);
S1'[I]'[J]:=Surf;
End;

Procedure CReflect31(I,J:Integer);
Begin
R1X'[I]'[J]:=R1X'[I]'[J+1];
R1Y'[I]'[J]:=0;
Surf:=S1'[I]'[J+1];
Errors:=Errors+Abs(Surf-S1'[I]'[J]);
S1'[I]'[J]:=Surf;
End;

Procedure CReflect36(I,J:Integer);
Begin
R1X'[I]'[J]:=R1X'[I]'[J-1];
R1Y'[I]'[J]:=0;
Surf:=S1'[I]'[J-1];
C.2. Boundary 3D

Errors := ErrorS + Abs(Surf - S1[I][J]);
S1[I][J] := Surf;
End;

Procedure CReflect41(I,J:Integer);
Begin
    R1X[I][J] := R1X[I][J+1];
    R1Y[I][J] := 0;
    Surf := S1[I][J+1];
    Errors := ErrorS + Abs(Surf - S1[I][J]);
    S1[I][J] := Surf;
End;

Procedure CReflect45(I,J:Integer);
Begin
    R1X[I][J] := R1X[I][J-1];
    R1Y[I][J] := 0;
    Surf := S1[I-1][J];
    Errors := ErrorS + Abs(Surf - S1[I][J]);
    S1[I][J] := Surf;
End;

Procedure CReflect51(I,J:Integer);
Begin
    R1X[I][J] := 0;
    R1Y[I][J] := 0;
    Surf := S1[I-1][J+1];
    Errors := ErrorS + Abs(Surf - S1[I][J]);
    S1[I][J] := Surf;
End;

Procedure CReflect52(I,J:Integer);
Begin
    R1X[I][J] := 0;
    R1Y[I][J] := R1Y[I-1][J];
    Surf := S1[I-1][J];
    Errors := ErrorS + Abs(Surf - S1[I][J]);
    S1[I][J] := Surf;
End;

Procedure CReflect53(I,J:Integer);
Begin
    R1X[I][J] := 0;
    R1Y[I][J] := R1Y[I-1][J];
    Surf := S1[I-1][J];
    Errors := ErrorS + Abs(Surf - S1[I][J]);
    S1[I][J] := Surf;
End;

Procedure CReflect54(I,J:Integer);
Begin
    R1X[I][J] := 0;
    R1Y[I][J] := R1Y[I-1][J];
    Surf := S1[I-1][J];
    Errors := ErrorS + Abs(Surf - S1[I][J]);
    S1[I][J] := Surf;
End;

Procedure CReflect55(I,J:Integer);
Begin
    R1X[I][J] := 0;
    R1Y[I][J] := 0;
    Surf := S1[I-1][J-1];
    Errors := ErrorS + Abs(Surf - S1[I][J]);
    S1[I][J] := Surf;
End;
End.
C.3 Fourier.3D

Unit Fourier;

Interface

{D*, L*, N*, Y*}

Uses
Variable,
Various;

Function DContDEta(J:Integer):Extended;
Function DContDK(J:Integer):Extended;
Function DContDQ:Extended;
Function DContDU(I,J:Integer):Extended;
Function DContDUO(J:Integer):Extended;
Function DMomDC(J:Integer):Extended;
Function DMomDEta:Extended;
Function DMomDK(J:Integer):Extended;
Function DMomDR:Extended;
Function DMomDU(I,J:Integer):Extended;
Function Dynamic(J:Integer):Extended;
Function F2MM3:Extended;
Function F2MM4:Extended;
Function F2MM5:Extended;
Function F2MM6:Extended;
Function Kinematic(J:Integer):Extended;
Function Sigma1(J:Integer):Extended;
Function Sigma2(J:Integer):Extended;
Function Sigma3(J:Integer):Extended;
Function Sigma4(J:Integer):Extended;
Function SineWaveNumber:Extended;
Procedure CoefficientMatrix;
Procedure Gauss;
Procedure GaussNewton;
Procedure FourierComputation;
Procedure Increment;
Procedure InitialGuess;
Procedure Initialize;
Procedure RightHandSide;
Procedure Terminate;

Implementation

Function DContDEta(J:Integer):Extended;
Begin
DContDEta:=Z^[MM+2]+Z'[2*MM+4]*Sigmal(J);
End;

Function DContDK(J:Integer):Extended;
Begin
DContDK:=(l+Z-CJ+l]*[Sigmal(J)+Z-C2*MM+4]*Sigma3(J))
-(l/2*Cl*Cl+Cl+l/3)*Sqr(Z-[2*MM+4])
*(3*Sigma2(J)+Z-[2*MM+4]*Sigma4(J));
End;

Function DContDQ:Extended;
Begin
DContDQ:=-l;
End;

Function DContDU(I,J:Integer):Extended;
Begin
DContDU:=(l+Z-[J+l]-(l/2*Cl*Cl+Cl+l/3)*Sqr(I*Z-[2*MM+4]))*
Cosh(I*Z-[2*MM+4]*Cl)/Cosh(I*Z-[2*MM+4])*
I*Z-[2*MM+4]*Cos(I*J*Pi/MM);
Function DContDUO(J:Integer):Extended;
Begin
  DContDUO:=1+Z'[J+1];
End;

Function DMomDC(J:Integer):Extended;
Begin
  DMomDC:=C1*(1/2*C1+1)*Sqr(Z'[2+MM+4])+Z'[2+MM+4]*Sigma2(J);
End;

Function DMomDEta:Extended;
Begin
  DMomDEta:=1;
End;

Function DMomDK(J:Integer):Extended;
Var
  Sigma:Extended;
Begin
  Sigma:=Sigmal(J);
  DMomDK:=(Z'[MM+2]+Z'[2+MM+4]*Sigma)+(Sigma+Z'[2+MM+4]*Sigma3(J)) +
            Cl*(1/2*C1+1)*Sqr(Z'[2+MM+4])*(Sigma+Sqrt2(J)+
            (3+Sigmal2(J)+Z'[2+MM+4]*Sigma6(J)));
End;

Function DMomDR:Extended;
Begin
  DMomDR:=-1;
End;

Function DMomDU(I,J:Integer):Extended;
Begin
  DMomDU:=(Z'[MM+2]+Z'[2+MM+4]*Sigmal(J)+Cl*(1/2*C1+1) +
            Z'[2+MM+3]*Sqr(Z'[2+MM+4])+1)*Z'[2+MM+4]*Cosh(Z'[2+MM+4]*I)*
            Cosh(I+Z'[2+MM+4])/(2*MM+1);
End;

Function Dynamic(J:Integer):Extended;
Begin
  Dynamic:=(1/2*Sqr(Z'[MM+2]+Z'[2+MM+4]*Sigmal(J))+Z'[J+1]+ 
            Cl*(1/2*C1+1)*Z'[2+MM+3]*Sqr(Z'[2+MM+4]*Z'[2+MM+4])*
            Sigma2(J)+Z'[2+MM+6];
End;

Function F2MM3:Extended;
Var
  Sum:Extended;
Begin
  Sum:=Z'[1]+Z'[MM+1];
  For J:=2 To MM Do
    Sum:=Sum+2*Z'[J];
  F2MM3:=Sum/(2*MM);
End;

Function F2MM4:Extended;
Begin
  F2MM4:=Z'[1]-Z'[MM+1]-H;
End;

Function F2MM5:Extended;
Begin
Function F2MM5:Extended;
Begin
End;

Function F2MM6:Extended;
Begin
If UpCase(Current)='E' Then
  F2MM6:=2\^[MM+2]-Curr+Z\^[2*MM+3]
Else
  F2MM6:=Z\^[2*MM+5]-Curr+Z\^[2*MM+3];
End;

Function Kinematic(J:Integer):Extended;
Begin
Kinematic: = (l+Z\^[J+l])*(Z\^[MM+2]+2\^[2*MM+4]*Sigmal(J))-
            (1/2*Cl*Cl+Cl+l/3)*Sqr(Z\^[2*MM+4])+Z\^[2*MM+4]*
            Sigma2(J)-Z\^[2*MM+6];
End;

Function SigmaKJ:Integer):Extended;
Var
  Sum,Ratio,Cosine:Extended;
  M:Integer;
Begin
  Sum:=0;
  For M:=1 To MM Do
  Begin
    Ratio:=Cosh(M*Z\^[MM+2+M])/Cosh(M*Z\^[2*MM+4]);
    Cosine:=M*Z\^[MM+2+M]*Cos(J*M*Pi/MM);
    Sum:=Sum+Ratio*Cosine;
  End;
  Sigmal:=Sum;
End;

Function Sigma2(J:Integer):Extended;
Var
  Sum,Ratio,Cosine:Extended;
  M:Integer;
Begin
  Sum:=0;
  For M:=1 To MM Do
  Begin
    Ratio:=Cosh(M*Z\^[MM+2+M])/(1+Cl))/Cosh(M*Z\^[2*MM+4]);
    Cosine:=M*Z\^[MM+2+M]*Cos(J*M*Pi/MM);
    Sum:=Sum+Ratio*Cosine;
  End;
  Sigma2:=Sum;
End;

Function Sigma3(J:Integer):Extended;
Var
  Sum,Alf,Bet:Extended;
  Arg1,Arg2,Cosine,Ratio:Extended;
  X1,X2,X3,X4:Extended;
  M:Integer;
Begin
  Sum:=0;
  For M:=1 To MM Do
  Begin
    Cosine:=M*Z\^[MM+2+M]*Cos(J*M*Pi/MM);
    Alf:=w*H(1+Cl);
    Arg1:=Alf*Z\^[2*MM+4];
    Bet:=M;
    Arg2:=Bet*Z\^[2*MM+4];
    X1:=Alf*Sinh(Arg1)+Cosh(Arg2);
    X2:=Bet*Cosh(Arg1)*Sinh(Arg2);
    X3:=X2-X1;
    X4:=Sqr(Cosh(Arg2));
    Ratio:=X3/X4;
  End;
End;
APPENDIX C. CODE - TWO HORIZONTAL DIMENSIONS

Sum:=Sum+Ratio*Cosine;
End;
Sigma3:=Sum;
End;

Function Sigma4(J: Integer): Extended;
Var
Sum, Alf, Bet: Extended;
Arg1, Arg2, Cosine, Ratio: Extended;
X1, X2, X3, X4: Extended;
M: Integer;
Begin
Sum:=0;
For M:=1 To MM Do
Begin
Cosine:=M*M*M*Z-[MM+2+M]*Cos(J*M*Pi/MM);
Alf:=M*(1+C1);
Bet:=M;
Arg2:=Bet*Z-[2*MM+4];
X1:=Alf*Sinh(Arg1)*Cosh(Arg2);
X2:=Bet*Cosh(Arg1)*Sinh(Arg2);
X3:=X2-X1;
X4:=Sqr(Cosh(Arg2));
Ratio:=X3/X4;
Sum:=Sum+Ratio*Cosine;
End;
Sigma4:=Sum;
End;

Function SineWaveNumber: Extended;
Var
F, F1, F2, W1, W2, Omega: Extended;
Iter: Integer;
Begin
Omega:=2*Pi/T;
W1:=1E-6;
F1:=Omega*Omega-W1*Tanh(W1);
W2:=1E3;
F2:=Omega*Omega-W2*Tanh(W2);
F:=100;
Iter:=0;
While (Abs(F)>MaxErr) And (Iter<MaxIter) Do
Begin
Iter:=Iter+1;
K:=(W1+W2)/2;
F:=Omega*Omega-K*Tanh(K);
If (F*F1)>0 Then
Begin
W1:=K;
F1:=F;
End
Else
Begin
W2:=K;
F2:=F;
End;
End;
If Abs(F)>MaxErr Then
Error(50)
Else
SineWaveNumber:=K;
End;

Procedure Gauss;
Var
Change, Sum: Extended;
I1, I2, I3, IMax: Integer;
C.3. Fourier.3D

Begin
For I1:=1 To N-1 Do
Begin
IMax:=I1;
For I2:=I1+1 To N Do
  If Abs(A[I1]-A[I2])>Abs(A[IMax]-A[I1]) Then IMax:=I2;
For I3:=1 To N Do
Begin
  Change:=A[I1]-A[I3];
  A[I1]:=A[IMax];
  A[IMax]:=Change;
End;
Change:=B[I1];
B[I1]:=B[IMax];
B[IMax]:=Change;
If Abs(A[I1]-A[I1])<1E-10 Then Error(30);
End;
End;
For I2:=1 To N Do
Begin
  Sum:=0;
  For I3:=1 To N Do
    Sum:=Sum+A[I2]*DZ[I3];
  DZ[I2]:=(B[I2]-Sum)/A[I2];
End;
End;

Procedure CoefficientMatrix;
Begin
For J:=1 To N Do
  For I:=1 To N Do
    A[J][I]:=0;
For J:=1 To MM+1 Do
  Begin
    A[J][J]:=DContDEta(J-1);
    A[J][MM+2]:=DContDU0(J-1);
    For I:=MM+3 To 2*MM+2 Do
      A[J][I]:=DContDU(I-MM-2,J-1);
    A[J][2*MM+4]:=DContDK(J-1);
    A[J][2*MM+5]:=DContDQ;
  End;
For J:=MM+2 To 2*MM+2 Do
  Begin
    A[J][J-MM-1]:=DMomDEta;
    A[J][MM+2]:=DMomDU0(J-MM-2);
    For I:=MM+3 To 2*MM+2 Do
      A[J][I]:=DMomDU(I-MM-2,J-MM-2);
    A[J][2*MM+3]:=DMomDC(J-MM-2);
    A[J][2*MM+4]:=DMomDQ;
  End;
A[2*MM+3][1]:=1/(2*MM);
For I:=2 To MM Do
  Begin
    A[2*MM+3][I]:=1/MM;
    A[2*MM+4][I]:=1/(2*MM);
    A[2*MM+4][1]:=1;
    A[2*MM+5][MM+1]=-1;
  End;
If UpCase(Current)='E' Then
Begin
    A'[2+MM+6]'[MM+2]:=1;
    A'[2+MM+6]'[2+MM+3]:=1;
End
Else
Begin
    A'[2+MM+6]'[2+MM+3]:=1;
    A'[2+MM+6]'[2+MM+6]:=1;
End;
End;

Procedure RightHandSide;
Begin
For J:=l To MM+1 Do
    B'[J]:=Kinematic(J-1);
For J:=MM+2 To 2*MM+2 Do
    B'[J]:=Dynamic(J-MM-2);
B'[2+MM+3]:=F2MM3;
B'[2+MM+4]:=F2MM4;
B'[2+MM+5]:=F2MM5;
B'[2+MM+6]:=F2MM6;
End;

Procedure GaussNewton;
Var
    NewErr:Extended;
    I,J :Integer;
Begin
    NewErr:=1;
    I:=0;
    While ((KNRI) And (NewErr>MaxErr)) Do
    Begin
        NewErr:=0;
        I:=I+1;
        CoefficientMatrix;
        RightHandSide;
        Gauss;
        For J:=l To N Do
        Begin
            If Abs(DZ'[J])>NewErr Then
                NewErr:=Abs(DZ'[J]/Z'[J]);
            Z'[J]:=Z'[J]+DZ'[J];
        End;
    End;
End;

Procedure Increment;
Begin
    H:=Alpha*H;
    For J:=l To MM+1 Do
        Z'[J]:=Alpha*Z'[J];
    For J:=MM+3 To 2*MM+2 Do
        Z'[J]:=Alpha*Z'[J];
End;

Procedure InitialGuess;
Begin
    H:=H/NH;
    Alpha:=Exp(Ln(NH)/(NH-1));
    Z'[2+MM+4]:=SineWaveNumber;
    Z'[2+MM+3]:=2*Pi/(Z'[2+MM+4]*T);
    For J:=l To MM+1 Do
        Z'[J]:=H/2*Cos((J-1)*Pi/MM);
    Z'[MM+2]:=-Z'[2+MM+3];
    Z'[MM+3]:=-H/(4*Z'[2+MM+3]*Z'[2+MM+4]);
    For J:=MM+4 To 2*MM+2 Do
\[ \begin{align*}
Z[J] &= 0; \\
\end{align*} \]

End;

Procedure Initialize;
Begin
AllocateVector1(B);
AllocateVector1(DZ);
AllocateVector1(Z);
AllocateMatrix3(A);
N := 2*MM+6;
End;

Procedure Terminate;
Begin
DeAllocateVector1(B);
DeAllocateVector1(DZ);
DeAllocateMatrix3(A);
C := Z[2*MM+3];
K := Z[2*MM+4];
CosTheta := Cos(Theta);
SinTheta := Sin(Theta);
KX := K*CosTheta;
KY := K*SinTheta;
L := 2*Pi/K;
End;

Procedure FourierComputation;
Var
Count: Integer;
Begin
Write(PrintFile, 'Procedure FourierComputation...........
');
Initialize;
InitialGuess;
GaussNewton;
For Count := 2 To NH Do
Begin
  Increment;
  GaussNewton;
End;
Terminate;
WriteLn(PrintFile, '  -Ok');
End;

Begin
End.
C.4 Grid.3D

Unit Grid;

Interface
{$D+,L+,N+,Y+}$

Uses
  Variable,
  Various;

Function Min(A,B:Integer):Integer;
Function Max(A,B:Integer):Integer;
Procedure RoughX1Boundary(IMin,IMax,J:Integer);
Procedure RoughX3Boundary(IMin,IMax,J:Integer);
Procedure RoughY2Boundary(JMin,JMax,I:Integer);
Procedure RoughY4Boundary(JMin,JMax,I:Integer);
Procedure RoughBoundary;
Procedure FineBoundary;
Procedure ComputeNodeTypes;
Procedure ComputeSpongeNodes;
Procedure ExternalNodes;
Procedure SaveWaveAndNodeGrid;

Implementation

Function Min(A,B:Integer):Integer;
Begin
  If A<=B Then
   Min:=A
  Else
   Min:=B;
End;

Function Max(A,B:Integer):Integer;
Begin
  If A>=B Then
   Max:=A
  Else
   Max:=B;
End;

Procedure RoughX1Boundary(IMin,IMax,J:Integer);
Begin
  If (J>=0) And (J<=JJ) Then
     For I:=IMin+1 To IMax-1 Do
      NodeType[I]
      [J]:=31;
   If (J>=0) And (J+1<=JJ) Then
     For I:=IMin+1 To IMax-1 Do
      NodeType[I]
      [J+1]:=32;
End;

Procedure RoughX3Boundary(IMin,IMax,J:Integer);
Begin
  If (J-1>=0) And (J<=JJ) Then
     For I:=IMin+1 To IMax-1 Do
      NodeType[I]
      [J-1]:=34;
  If (J>=0) And (J<=JJ) Then
     For I:=IMin+1 To IMax-1 Do
      NodeType[I]
      [J]:=35;
End;

Procedure RoughY2Boundary(JMin,JMax,I:Integer);
Begin
  If (I-1>=0) And (I<=II) Then
     For J:=JMin+1 To JMax-1 Do
      NodeType[I-1]
      [J]:=43;
If \((I\geq 0)\) And \((I\leq II)\) Then 
For \(J:=JMin+1\) To \(JMax-1\) Do 
NodeType\([I\ ][J]\] :=33;
End;

Procedure RoughX4Boundary(JMin,JMax,I:Integer);
Begin
If \((I\geq 0)\) And \((I+1\leq II)\) Then 
For \(J:=JMin+1\) To \(JMax-1\) Do 
NodeType\([I+1][J]\] :=23;
End;

Procedure FineBoundary;
Var
IA,JA:Integer;
Begin
For \(N:=1\) To \(NM+NI\) Do 
Begin
IA:=I1[N];
JA:=J1[N];
End;

Case CT[N] Of
11,111:Begin
If \(IMin=IMax\) Then
RoughY4Boundary(JMin,JMax,IMin);
End;

51,151:Begin
If \(IMin=IMax\) Then
RoughY2Boundary(JMin,JMax,IMin);
End;

55,155:Begin
If \(IMin=IMax\) Then
RoughX3Boundary(IMin,IMax,JA);
End;

15,115:Begin
If \(IMin=IMax\) Then
RoughX3Boundary(IMin,IMax,IA);
End;
End;
Case CT[N] Of
  11: Begin
    NodeType[IA][JA] := 11;
    NodeType[IA+1][JA] := 21;
    NodeType[IA][JA+1] := 12;
    NodeType[IA+1][JA+1] := 22;
    End;
  15: Begin
    NodeType[IA][JA-1] := 14;
    NodeType[IA+1][JA-1] := 24;
    NodeType[IA][JA] := 15;
    NodeType[IA+1][JA] := 25;
    End;
  51: Begin
    NodeType[IA-1][JA] := 41;
    NodeType[IA][JA] := 51;
    NodeType[IA-1][JA+1] := 42;
    NodeType[IA][JA+1] := 52;
    End;
  55: Begin
    NodeType[IA-1][JA-1] := 44;
    NodeType[IA][JA-1] := 54;
    NodeType[IA-1][JA+1] := 45;
    NodeType[IA][JA] := 55;
    End;
  111: Begin
    If (IA<II) And (JA<JJ) Then
      Begin
        NodeType[IA][JA] := 111;
        NodeType[IA+1][JA] := 33;
        NodeType[IA][JA+1] := 33;
      End;
    If (IA<II) And (JA=JJ) Then
      Begin
        NodeType[IA][JA-1] := 14;
        NodeType[IA+1][JA-1] := 24;
        NodeType[IA][JA] := 15;
        NodeType[IA+1][JA] := 25;
      End;
    If (IA=II) And (JA<JJ) Then
      Begin
        NodeType[IA-1][JA] := 41;
        NodeType[IA][JA] := 51;
        NodeType[IA-1][JA+1] := 42;
        NodeType[IA][JA+1] := 52;
      End;
    If (IA=II) And (JA=JJ) Then
      NodeType[IA][JA] := 0;
    End;
  115: Begin
    If (IA<II) And (JA=0) Then
      Begin
        NodeType[IA][JA] := 11;
        NodeType[IA+1][JA] := 21;
        NodeType[IA][JA+1] := 12;
        NodeType[IA+1][JA+1] := 22;
      End;
    If (IA<II) And (JA>0) Then
      Begin
        NodeType[IA][JA] := 33;
        NodeType[IA+1][JA] := 15;
        NodeType[IA][JA+1] := 33;
      End;
    If (IA=II) And (JA=0) Then
      NodeType[IA][JA] := 0;
    If (IA=II) And (JA>0) Then
      Begin
        NodeType[IA-1][JA-1] := 44;
      End;

NodeType['IA']['JA-1']=54;
NodeType['IA-1']['JA ']=45;
NodeType['IA ']['JA ']=55;
End;
End;
151:Begin
If (IA=0) And (JA<JJ) Then
Begin
NodeType['IA ']['JA']=11;
NodeType['IA+1']['JA']=21;
NodeType['IA ']['JA+1']=12;
NodeType['IA+1 ']['JA+1']=22;
End;
If (IA=0) And (JA=JJ) Then
NodeType['IA ']['JA']=0;
If (IA>0) And (JA<JJ) Then
Begin
NodeType['IA-1']['JA ']=33;
NodeType['IA ']['JA ']=151;
NodeType['IA '] ['JA+1']=33;
End;
If (IA>0) And (JA=JJ) Then
Begin
NodeType['IA-1 '] ['JA-1']=44;
NodeType['IA ']['JA-1']=54;
NodeType['IA-1 '] ['JA]=45;
NodeType['IA '] ['JA']=55;
End;
End;
End;
155:Begin
If (IA=0) And (JA=0) Then
NodeType['IA '] ['JA ']=0;
If (IA=0) And (JA>0) Then
Begin
NodeType['IA '] ['JA-l']=14;
NodeType['IA+1 '] ['JA-1']=24;
NodeType['IA '] ['JA']=15;
NodeType['IA+1 '] ['JA']=25;
End;
If (IA>0) And (JA=0) Then
Begin
NodeType['IA-1 '] ['JA']=41;
NodeType['IA '] ['JA']=51;
NodeType['IA-1 '] ['JA+1']=42;
NodeType['IA '] ['JA+1']=52;
End;
If (IA>0) And (JA>0) Then
Begin
NodeType['IA '] ['JA-1']=33;
NodeType['IA-1 '] ['JA']=33;
NodeType['IA '] ['JA']=155;
End;
End;
End;
End;

Procedure ComputeSpongeNodes;
Const
GammaMax=0.75;
Var
IMin,JMin,IMax,JMax,Scale:Integer;
Value :Extended;

Function SpongeValue(J,JS:Integer):Extended;
Begin
SpongeValue:=1+GammaMax*Sqr(J/JS);
End;
BEGIN
FOR N:=1 TO NM+NI DO
  IF SP[N]>0 THEN
    BEGIN
      IMin:=Min(I1[N],I2[N]);
      IMax:=Max(I1[N],I2[N]);
      JMin:=Min(J1[N],J2[N]);
      JMax:=Max(J1[N],J2[N]);
      IF IMin=IMax THEN
        CASE CT[N] OF
          11,15,111,115:
            FOR I:=IMin To IMin+SP[N] DO
              BEGIN
                Scale:=IMin+SP[N]-I;
                Value:=SpongeValue(Scale,SP[N]);
                FOR J:=JMin To JMax DO
                  IF Value>Mu[I][J] THEN
                    IF NodeType[I][J]>0 THEN
                      Mu[I][J]:=Value;
              END;
            END;
          51,55,151,155:
            FOR I:=IMin-SP[N] TO IMin DO
              BEGIN
                Scale:=I-(IMin-SP[N]);
                Value:=SpongeValue(Scale,SP[N]);
                FOR J:=JMin To JMax DO
                  IF Value>Mu[I][J] THEN
                    IF NodeType[I][J]>0 THEN
                      Mu[I][J]:=Value;
              END;
            END;
        END;
      END;
      IF JMin=JMax THEN
        CASE CT[N] OF
          11,51,111,151:
            FOR J:=JMin To JMin+SP[N] DO
              BEGIN
                Scale:=JMin+SP[N]-J;
                Value:=SpongeValue(Scale,SP[N]);
                FOR I:=IMin To IMax DO
                  IF Value>Mu[I][J] THEN
                    IF NodeType[I][J]>0 THEN
                      Mu[I][J]:=Value;
              END;
            END;
          15,55,115,155:
            FOR J:=JMin-SP[N] TO JMin DO
              BEGIN
                Scale:=J-(JMin-SP[N]);
                Value:=SpongeValue(Scale,SP[N]);
                FOR I:=IMin To IMax DO
                  IF Value>Mu[I][J] THEN
                    IF NodeType[I][J]>0 THEN
                      Mu[I][J]:=Value;
              END;
            END;
        END;
    END;
  END;
END;

FUNCTION FN(N:integer):integer;
BEGIN
  FN:=NM+1+(N-NM-1) mod NI;
END;

PROCEDURE ExternalNodes;
VAR
  AP,NP,CMin,CMax,IMin,IMax,JMin,JMax,NodeType,Sign:integer;
  ExternalArea: boolean;
BEGIN
  "Code here...
END;
For \( N := NM + 1 \) To \( NM + NI \) Do

Begin
  IMin := Min(I1[N], I2[N]);
  IMax := Max(I1[N], I2[N]);
  If IMin = IMax Then
    Begin
      JMin := Min(J1[N], J2[N]);
      JMax := Max(J1[N], J2[N]);
      AP := FN(N);
      NP := FN(N+1);
      If (JMin>0) And (JMin<JJ) Then
        Begin
          If J1[AP]=JMin Then
            CMin:=CT[AP]
          Else
            CMin:=CT[NP];
          Case CMin Of
            11:
              Sign:=-1;
            51:
              Sign:=1;
            115:
              Begin
                JMin:=JMin+1;
                Sign:=-1;
              End;
            155:
              Begin
                JMin:=JMin+1;
                Sign:=1;
              End;
          End;
        End;
        If (JMax>0) And (JMax<JJ) Then
          Begin
            If J1[AP]=JMax Then
              CMax:=CT[AP]
            Else
              CMax:=CT[NP];
            Case CMax Of
              15:
                Sign:=-1;
              55:
                Sign:=1;
              111:
                Begin
                  JMax:=JMax-1;
                  Sign:=-1;
                End;
              151:
                Begin
                  JMax:=JMax-1;
                  Sign:=1;
                End;
          End;
        End;
      End;
    End;
    For J:=JMin To JMax Do
      WaveNode'[IMin]'[J]:=Sign;
  End;
End;
For J:=0 To JJ Do

Begin
  I:=0;
  ExternalArea:=False;
  If WaveNode'[0]'[J]=1 Then
    Begin
      ExternalArea:=True;
      NodeType'[0]'[J]:=0;
    End;
End;
APPENDIX C. CODE - TWO HORIZONTAL DIMENSIONS

End;
While I<II Do
Begin
While (ExternalArea) And (I<II) Do
Begin
J:=I+1;
If WaveNode^[I]-[J]=-1 Then
ExternalArea:=False
Else
NodeType^[I]-[J]:=0;
End;
While (Not ExternalArea) And (I<II) Do
Begin
J:=I+1;
If WaveNode^[I]-[J]=1 Then
ExternalArea:=True;
End;
End;
End;

Procedure AbsorptionOrReflection;
Var
IMin,IMax,JMin,JMax: Integer;
Begin
For N:=1 To NM+NI Do
Begin
IMin:=Min(I1[N],I2[N]);
IMax:=Max(I1[N],I2[N]);
JMin:=Min(J1[N],J2[N]);
JMax:=Max(J1[N],J2[N]);
If BT[N]="R" Then
Begin
If IMin=IMax Then
For J:=JMin To JMax Do
If NodeType^[IMin]-[J]>0 Then
NodeType^[IMin]-[J]:=-NodeType^[IMin]-[J];
If JMin=JMax Then
For I:=IMin To IMax Do
If NodeType^[I]-[JMin]>0 Then
NodeType^[I]-[JMin]:=-NodeType^[I]-[JMin];
End;
End;
End;
End;

Procedure WaveBoundary;
Var
IMin,IMax,JMin,JMax,Sign: Integer;
Begin
For I:=0 To II Do
Begin
For J:=0 To JJ Do
WaveNode^[I]-[J]:=0;
End;
For N:=NM+NI+1 To NM+NI+NW Do
Begin
IMin:=Min(I1[N],I2[N]);
IMax:=Max(I1[N],I2[N]);
JMin:=Min(J1[N],J2[N]);
JMax:=Max(J1[N],J2[N]);
Case UpCase(BT[N]) Of
'P':Sign:=-1;
'T':Sign:=1;
End;
If IMin=IMax Then
For J:=JMin To JMax Do
WaveNode^[IMin]-[J]:=Sign;
If JMin=JMax Then
For I:=IMin To IMax Do
WaveNode^[I]-[JMin]:=Sign;
Procedure SaveWaveAndNodeGrid;
Begin
  Assign(DataFile, FileDir + FileName + '.Grd');
  Rewrite(DataFile);
  For I := 0 To II Do
    For J := 0 To JJ Do
      WriteLn(DataFile, I:6, J:6, NodeType'[I][J]:8,
              WaveNode'[I][J]:8, Mu'[I][J]:12:4);
  Close(DataFile);
End;

Procedure ComputeNodeTypes;
Begin
  Write(PrintFile, 'Procedure ComputeNodeTypes, RoughBoundary;
          FineBoundary;
          ExternalNodes;
          AbsorptionOrReflection;
          WaveBoundary;
          WriteLn(PrintFile, ' - Ok');
End;

Begin
End.
C.5 Solution 3D

Unit Solution;

Interface

{$D+,L+,N+,Y+}$

Uses
  Boundary,
  Fourier,
  Grid,
  Variable,
  Various,
  Crt;

Function ErrorExceedsLimit:Boolean;
Function Surface(I,J,N: Integer):Extended;
Function Velocity(I,J,N:Integer):Extended;
Procedure AddF1Incident;
Procedure AddF2XIncident;
Procedure AddF2YIncident;
Procedvure AddF2YTIncident;
Procedure CoefficientMatrixX;
Procedure CoefficientMatrixY;
Procedure ComputeDepthDerivatives;
Procedure ComputeRelativeErrors;
Procedure ComputeF1;
Procedure ComputeF2X;
Procedure ComputeF2Y;
Procedure ComputeF2XT;
Procedure ComputeF2YT;
Procedure ComputeNewIncidentWave;
Procedure CornerPoints;
Procedure Corrector;
Procedure CorrectRHSX;
Procedure CorrectRHSY;
Procedure InitComputation;
Procedure InitControlVariables;
Procedure InitPredictor;
Procedure PredictElevation;
Procedure Predictor;
Procedure PredictRHSX;
Procedure PredictRHSY;
Procedure SpongeComputation;
Procedure UpdateComputation;
Procedure UpdateVariables;
Procedure WaveEnvelope;

Implementation

Function ErrorExceedsLimit:Boolean;
Begin
  If (RelErrorS<MaxIterError) And
      (RelErrorU<MaxIterError) And
      (RelErrorV<MaxIterError) Then
    ErrorExceedsLimit:=False
  Else
    ErrorExceedsLimit:=True;
  Errors:=0;
  ErrorU:=0;
  ErrorV:=0;
End;

Function Surface(I,J,N:Integer):Extended;
Var
C.5. Solution.3D

\[ U, A_{\text{Three}}, A_{\text{Seven}} : \text{Extended}; \]

\[ \text{Begin} \]

\[ A_{\text{Three}} := C_1 \cdot (C_1 / 2 + 1); \]

\[ A_{\text{Seven}} := (C_1 \cdot C_1 / 2 + C_1 / 3); \]

\[ U := \text{Velocity}(I, J, N) - Z^{[2 \cdot MM + 3]}; \]

\[ \text{Surface} := (A_{\text{Three}} \cdot Z^{[2 \cdot MM + 3]} \cdot (Z^{[2 \cdot MM + 5]} - U) + A_{\text{Seven}} = (Z^{[2 \cdot MM + 6]} - 1 / 2 \cdot U) / (A_{\text{Three}} \cdot Z^{[2 \cdot MM + 3]} + U); \]

\[ \text{End}; \]

\[ \text{Function Velocity}(I, J, N : \text{Integer}) : \text{Extended}; \]

\[ \text{Var} \]

\[ \text{Sum, Ratio, Cosine} : \text{Extended}; \]

\[ M : \text{Integer}; \]

\[ \text{Begin} \]

\[ \text{Sum} := 0; \]

\[ \text{For } M := 1 \text{ To } MM \text{ Do} \]

\[ \text{Ratio} := \cosh(\pi \cdot (M \cdot I)) / \cosh(\pi \cdot K); \]

\[ \text{Cosine} := \pi \cdot Z^{[MM + 6]} \cdot \cos(M \cdot (K \cdot I \cdot \text{DelI} \cdot X + K \cdot J \cdot \text{DelI} \cdot Y - 2 \cdot \pi \cdot I \cdot N \cdot \text{DT} / T)); \]

\[ \text{Sum} := \text{Sum} + \text{Ratio} \cdot \text{Cosine}; \]

\[ \text{End}; \]

\[ \text{Velocity} := Z^{[2 \cdot MM + 3]} \cdot Z^{[MM + 6]} \cdot K \cdot \text{Sum}; \]

\[ \text{End}; \]

\[ \text{Procedure AddF1Incident}; \]

\[ \text{Begin} \]

\[ \text{Write(PrintFile, 'Procedure AddF1Incident..............')}; \]

\[ \text{For } I := 0 \text{ To } II \text{ Do} \]

\[ \text{For } J := 0 \text{ To } JJ \text{ Do} \]

\[ \text{Case WaveNode}^{[I]} \cdot [J] \text{ Of} \]

\[ -1:F1^{[I]} [J] [I] := \]

\[ \text{F1}^{[I]} [J] [I] + \text{FX} (U_{\text{Tot}}, I, J) \]

\[ + \text{U}_{\text{Ref}} [I] [J] \cdot \text{HX} (S_{\text{Tot}}, I, J, U_{\text{Ref}}) \]

\[ + \text{S}_{\text{Ref}} [I] [J] \cdot \text{HX} (U_{\text{Tot}}, I, J, S_{\text{Ref}}) \]

\[ + D^{[I]} [J] \cdot \text{FY} (V_{\text{Tot}}, I, J) \]

\[ + \text{V}_{\text{Ref}} [I] [J] \cdot \text{HY} (S_{\text{Tot}}, I, J, V_{\text{Ref}}) \]

\[ + \text{S}_{\text{Ref}} [I] [J] \cdot \text{HY} (V_{\text{Tot}}, I, J, S_{\text{Ref}}) \]

\[ + A_1 \cdot 3 \cdot D^{[I]} [J] \cdot D^{[I]} [J] \cdot \text{DX}^{[I]} [J] \]

\[ \cdot (\text{FXX} (U_{\text{Tot}}, I, J) + \text{FYY} (V_{\text{Tot}}, I, J)) \]

\[ + A_1 \cdot 3 \cdot D^{[I]} [J] \cdot D^{[I]} [J] \cdot D^{[I]} [J] \cdot \text{DX}^{[I]} [J] \]

\[ \cdot (\text{FXXX} (U_{\text{Tot}}, I, J) + \text{FXY} (V_{\text{Tot}}, I, J)) \]

\[ + A_2 \cdot 2 \cdot D^{[I]} [J] \cdot \text{DX}^{[I]} [J] \]

\[ \cdot (\text{DGXX} (U_{\text{Tot}}, I, J) + \text{DGXY} (V_{\text{Tot}}, I, J)) \]

\[ + A_1 \cdot 3 \cdot D^{[I]} [J] \cdot D^{[I]} [J] \cdot \text{DY}^{[I]} [J] \]

\[ \cdot (\text{FX} (U_{\text{Tot}}, I, J) + \text{FY} (V_{\text{Tot}}, I, J)) \]

\[ + A_1 \cdot 3 \cdot D^{[I]} [J] \cdot D^{[I]} [J] \cdot D^{[I]} [J] \cdot \text{DY}^{[I]} [J] \]

\[ \cdot (\text{FYY} (V_{\text{Tot}}, I, J) + \text{FF} (V_{\text{Tot}}, I, J)) \]

\[ + A_2 \cdot 2 \cdot D^{[I]} [J] \cdot \text{DY}^{[I]} [J] \]

\[ \cdot (\text{DGXY} (U_{\text{Tot}}, I, J) + \text{DGYY} (V_{\text{Tot}}, I, J)) \]

\[ + A_1 \cdot 3 \cdot D^{[I]} [J] \cdot D^{[I]} [J] \cdot \text{DX}^{[I]} [J] \]

\[ \cdot (\text{DGXX} (U_{\text{Tot}}, I, J) + \text{DGYY} (V_{\text{Tot}}, I, J)); \]

\[ 1:F1^{[I]} [J] [I] := \]

\[ \text{F1}^{[I]} [J] [I] + \text{FX} (U_{\text{Ref}}, I, J) \]

\[ + \text{U}_{\text{Tot}} [I] [J] \cdot \text{HX} (S_{\text{Ref}}, I, J, U_{\text{Tot}}) \]

\[ + \text{S}_{\text{Tot}} [I] [J] \cdot \text{HX} (U_{\text{Ref}}, I, J, S_{\text{Tot}}) \]

\[ + D^{[I]} [J] \cdot \text{FY} (V_{\text{Ref}}, I, J) \]

\[ + \text{V}_{\text{Tot}} [I] [J] \cdot \text{HY} (S_{\text{Ref}}, I, J, V_{\text{Tot}}) \]

\[ + \text{S}_{\text{Tot}} [I] [J] \cdot \text{HY} (V_{\text{Ref}}, I, J, S_{\text{Tot}}) \]

\[ + A_1 \cdot 3 \cdot D^{[I]} [J] \cdot D^{[I]} [J] \cdot \text{DX}^{[I]} [J] \]

\[ \cdot (\text{FXX} (U_{\text{Ref}}, I, J) + \text{FYY} (V_{\text{Ref}}, I, J)) \]

\[ + A_1 \cdot 3 \cdot D^{[I]} [J] \cdot D^{[I]} [J] \cdot \text{DY}^{[I]} [J] \]

\[ \cdot (\text{FX} (U_{\text{Ref}}, I, J) + \text{FY} (V_{\text{Ref}}, I, J)) \]

\[ + A_2 \cdot 2 \cdot D^{[I]} [J] \cdot \text{DY}^{[I]} [J] \]

\[ \cdot (\text{DGXX} (U_{\text{Ref}}, I, J) + \text{DGYY} (V_{\text{Ref}}, I, J)) ; \]
procedure AddF2XIncident;
begin
  write(printfile, 'procedure AddF2XIncident...');
  for I:=0 to II do
    for J:=0 to JJ do
      case waveNode[I,J] of
        -1: f2X[I,J][1] := f2X[I,J][1] +
            (fx(s1Tot,I,J) +
             hx(UlTot,I,J,UlRef)*UlRef[I,J] +
             hy(UlTot,I,J,VlRef)*VlRef[I,J]);
        1: f2X[I,J][1] := f2X[I,J][1] -
            (fxCS1Ref,I,J) +
            hx(UlRef,I,J,UlTot)*UlTot[I,J] +
            hy(UlRef,I,J,VlTot)*VlTot[I,J]);
      end;
   end;
   write(printfile, ' - ok');
end;

procedure AddF2XTIncident;
begin
  write(printfile, 'procedure AddF2XTIncident...');
  for I:=0 to II do
    for J:=0 to JJ do
      case waveNode[I,J] of
        -1: f2XT[I,J][1] := f2XT[I,J][1] +
            B1*D[I,J]*fxy(VlTot,I,J) +
            B2*DGXY(VlTot,I,J); 
        1: f2XT[I,J][1] := f2XT[I,J][1] -
            (B1*D[I,J]*fxy(VlRef,I,J) +
             B2*DGXY(VlRef,I,J));
      end;
   end;
   write(printfile, ' - ok');
end;

procedure AddF2YIncident;
begin
  write(printfile, 'procedure AddF2YIncident...');
  for I:=0 to II do
    for J:=0 to JJ do
      case waveNode[I,J] of 
        -1: f2Y[I,J][1] := f2Y[I,J][1] +
            fy(s1Tot,I,J) +
            hx(VlTot,I,J,UlRef)*UlRef[I,J] +
            hy(VlTot,I,J,VlRef)*VlRef[I,J]);
        1: f2Y[I,J][1] := f2Y[I,J][1] -
            (fyCS1Ref,I,J) +
            hx(VlRef,I,J,UlTot)*UlTot[I,J] +
            hy(VlRef,I,J,VlTot)*VlTot[I,J]);
      end;
   end;
   write(printfile, ' - ok');
end;
C.5. Solution.3D

End;
WriteLn(PrintFile,' - Ok');
End;

Procedure AddF2YTIincident;
Begin
Write(PrintFile,'Procedure AddF2YTIincident ..................');
For I:=0 To II Do
For J:=0 To JJ Do
Case WaveNode[I][J] Of
-1:F2YT[I][J][II]+D[I][J]*(
B1*D[I][J]+FXY(UITot,I,J)
+ B2*DGXY(UITot,I,J));
1:F2YT[I][J][I]D[I][J]*(
B1*D[I][J]+FXY(UIRef,I,J)
+ B2*DGXY(UIRef,I,J));
End;
WriteLn(PrintFile,' - Ok');
End;

Procedure CoefficientMatrixX;
Begin
For J:=0 To JJ Do
Begin
D1X[0][J]:=0;
D2X[0][J]:=1;
D3X[0][J]:=0;
For I:=1 To II-1 Do
Case Abs(NodeType[I][J]) Of
21,22,23,24,25,31,32,33,34,35,41,42,43,44,45,111,115,151,155:
Begin
D1X[I][J]:=D[I][J]*D[I][J]*(B1+B2)/(DelX*DelX)
- B2*D[I][J]*DX[I][J]/DelX;
D2X[I][J]:=1+B2*D[I][J]*DXX[I][J]
- 2*D[I][J]*D[I][J]*(B1+B2)/(DelX*DelX);
D3X[I][J]:=D[I][J]*D[I][J]*(B1+B2)/(DelX*DelX)
+ B2*D[I][J]*DX[I][J]/DelX;
End;
Else
Begin
D1X[I][J]:=0;
D2X[I][J]:=1;
D3X[I][J]:=0;
End;
End;
D1X[II][J]:=0;
D2X[II][J]:=1;
D3X[II][J]:=0;
End;
End;

Procedure CoefficientMatrixY;
Begin
For I:=0 To II Do
Begin
D1Y[I][0]:=0;
D2Y[I][0]:=1;
D3Y[I][0]:=0;
For J:=1 To JJ-1 Do
Case Abs(NodeType[I][J]) Of
12,13,14,15,22,23,24,32,33,34,35,42,43,44,45,52,53,54,111,115,151,155:
Begin
D1Y[I][J]:=D[I][J]*D[I][J]*(B1+B2)/(DelY*DelY)
- B2*D[I][J]*DYY[I][J]/DelY;
D2Y[I][J]:=1+B2*D[I][J]*DYY[I][J]
- 2*D[I][J]*D[I][J]*(B1+B2)/(DelY*DelY);
APPENDIX C. CODE - TWO HORIZONTAL DIMENSIONS

End;

Else
Begin
D1Y'[J] := 0;
D2Y'[J] := 1;
D3Y'[J] := 0;
End;

D1Y'-[I]-[J] := 0;
D2Y'-[I]-[J] := 1;
D3Y'-[I]-[J] := 0;
End;

End;

Procedure ComputeDepthDerivatives;
Begin
Write(PrintFile, 'Procedure ComputeDepthDerivatives...');
For I := 0 To II Do
For J := 0 To JJ Do
Begin
DX'[I]-[J] := FX(D, I, J);
DY'[I]-[J] := FY(D, I, J);
DXX'[I]-[J] := FXX(D, I, J);
DXY'[I]-[J] := FXY(D, I, J);
DYY'[I]-[J] := FYY(D, I, J);
DXXX'[I]-[J] := FXXX(D, I, J);
DXXY'[I]-[J] := DXXY(D, I, J);
DXY'[I]-[J] := FXY(D, I, J);
DYYY'[I]-[J] := FYYY(D, I, J);
End;
WriteLn(PrintFile, ' - Ok');
End;

Procedure ComputeRelativeErrors;
Begin
SumS := 0;
SumU := 0;
SumV := 0;
For I := 0 To II Do
For J := 0 To JJ Do
If NodeType'[I]-[J] <> 0 Then
Begin
SumS := SumS + Abs(S1'[I]-[J]);
SumU := SumU + Abs(U1'[I]-[J]);
SumV := SumV + Abs(V1'[I]-[J]);
End;

If SumS > 0 Then
RelErrorS := ErrorS / SumS
Else
RelErrorS := 0;
If SumU > 0 Then
RelErrorU := ErrorU / SumU
Else
RelErrorU := 0;
If SumV > 0 Then
RelErrorV := ErrorV / SumV
Else
RelErrorV := 0;
End;

Procedure ComputeF1;
Begin
Write(PrintFile, 'Procedure ComputeF1.................');
For I := 0 To II Do
For J := 0 To JJ Do
F1'[I]-[J] := -(
\[ DX^3[I]^3[1] \cdot U^3[I]^3[1] \cdot J \]
\[ + D^3[I]^3[1] \cdot [FX(U1, I, J)] \]
\[ + U^3[I]^3[1] \cdot HX(S1, I, J, U1) \]
\[ + S1^3[I]^3[1] \cdot HX(U1, I, J, S1) \]
\[ + D^3[I]^3[1] \cdot FY(V1, I, J) \]
\[ + V1^3[I]^3[1] \cdot HY(S1, I, J, V1) \]
\[ + S1^3[I]^3[1] \cdot HY(V1, I, J, S1) \]
\[ + A1^3[D^3[I]^3[1]] \cdot [D^3[I]^3[1]] \cdot [DX^3[I]^3[1]] \]
\[ + A1^3[D^3[I]^3[1]] \cdot [D^3[I]^3[1]] \cdot [DFXX(U1, I, J) + DFXY(V1, I, J)] \]
\[ + A2^3[D^3[I]^3[1]] \cdot [D^3[I]^3[1]] \cdot [DFXXX(U1, I, J) + DFXXY(V1, I, J)] \]
\[ + A3^3[D^3[I]^3[1]] \cdot [D^3[I]^3[1]] \cdot [DFXY(U1, I, J) + DFYY(V1, I, J)] \]
\[ + A2^3[D^3[I]^3[1]] \cdot [D^3[I]^3[1]] \cdot [DFXYY(U1, I, J) + DFYYY(V1, I, J)]; \]

WriteLn(PrintFile, ' - Ok');
End;

Procedure ComputeF2X;
Begin
Write(PrintFile, 'Procedure ComputeF2X.
For I:=0 To II Do
For J:=0 To JJ Do
F2X^3[I]^3[1][J][I][J] := -(FX(U1, I, J)]
\[ + HX(U1, I, J, U1) \cdot U^3[I]^3[1][1][J] \]
\[ + HY(U1, I, J, V1) \cdot V^3[I]^3[1][1][J] \]
\[ + Friction^3[I]^3[1][1][J] \cdot U^3[I]^3[1][1][J]); \]
WriteLn(PrintFile, ' - Ok');
End;

Procedure ComputeF2Y;
Begin
Write(PrintFile, 'Procedure ComputeF2Y.
For I:=0 To II Do
For J:=0 To JJ Do
F2Y^3[I]^3[1][J][I][J] := -(FY(V1, I, J)]
\[ + HX(V1, I, J, U1) \cdot U^3[I]^3[1][1][J] \]
\[ + HY(V1, I, J, V1) \cdot V^3[I]^3[1][1][J] \]
\[ + Friction^3[I]^3[1][1][J] \cdot V^3[I]^3[1][1][J]); \]
WriteLn(PrintFile, ' - Ok');
End;

Procedure ComputeF2XT;
Begin
Write(PrintFile, 'Procedure ComputeF2XT.
For I:=0 To II Do
For J:=0 To JJ Do
F2XT^3[I]^3[1][J][I][J] := -(B1*D^3[I]^3[1][1][J]*[FY(V1, I, J)]
\[ + B2*DFXY(V1, I, J); \]
WriteLn(PrintFile, ' - Ok');
End;

Procedure ComputeF2YT;
Begin
Write(PrintFile, 'Procedure ComputeF2YT.
For I:=0 To II Do
For J:=0 To JJ Do
F2YT[I][J-1]=-D[I][J]*
B[I][J]*FXY(01,1,I,J)
+ B2*DFXY(01,1,I,J));
WriteLn(PrintFile,' - Ok');
End;

Procedure ComputeNewIncidentWave;
Begin
For I:=0 To II Do
For J:=0 To JJ Do
  Case WaveNode[I][J] Of
    -1:Begin
      SlRef[I][J]:=Fader*Surface(I,J,N);
      UlRef[I][J]:=Fader*CosTheta*Velocity(I,J,N);
      VlRef[I][J]:=Fader*SinTheta*Velocity(I,J,N);
    End;
    1:Begin
      SlTot[I][J]:=Fader*Surface(I,J,N);
      UlTot[I][J]:=Fader*CosTheta*Velocity(I,J,N);
      VlTot[I][J]:=Fader*SinTheta*Velocity(I,J,N);
    End;
  End;
End;

Procedure CorrectElevation;
Begin
Write(PrintFile,'Procedure CorrectElevation...');
For I:=1 To II-1 Do
For J:=1 To JJ-1 Do
  If NodeType[I][J]<>0 Then
    Begin
      Surf:=S0[I][J]
      +DT/24*(9*Fl[I][J]+19*Fl[I][J-1]
      - 5*Fl[I][J-2])
      + Fl[I][J-1]*; Errors:=ErrorS+Abs(Surf-S1[I][J]);
      S1[I][J]:=Surf;
    End;
  WriteLn(PrintFile,' - Ok');
End;

Procedure CornerPoints;
Begin
For I:=1 To II-1 Do
For J:=1 To JJ-1 Do
  Case Abs(NodeType[I][J]) Of
    111:Begin
      R1X[I][J]:=(R1X[I][J]+R1X[I+1][J]+R1X[I][J+1])/-3;
      R1Y[I][J]:=(R1Y[I][J]+R1Y[I+1][J]+R1Y[I][J+1])/-3;
      S1[I][J]:=(S1[I][J]+S1[I+1][J]+S1[I][J+1])/-3;
    End;
    115:Begin
      R1X[I][J]:=(R1X[I][J]+R1X[I+1][J]+R1X[I][J-1])/-3;
      R1Y[I][J]:=(R1Y[I][J]+R1Y[I+1][J]+R1Y[I][J-1])/-3;
      S1[I][J]:=(S1[I][J]+S1[I+1][J]+S1[I][J-1])/-3;
    End;
    151:Begin
      R1X[I][J]:=(R1X[I][J]+R1X[I-1][J]+R1X[I][J+1])/-3;
      R1Y[I][J]:=(R1Y[I][J]+R1Y[I-1][J]+R1Y[I][J+1])/-3;
      S1[I][J]:=(S1[I][J]+S1[I-1][J]+S1[I][J+1])/-3;
    End;
    155:Begin
      R1X[I][J]:=(R1X[I][J]+R1X[I-1][J]+R1X[I][J+1])/-3;
      R1Y[I][J]:=(R1Y[I][J]+R1Y[I-1][J]+R1Y[I][J+1])/-3;
      S1[I][J]:=(S1[I][J]+S1[I-1][J]+S1[I][J+1])/-3;
    End;
  End;
End;
R1X[I,J] := (R1X[I-1,J] + R1X[I,J+1] + R1X[I,J-1]) / 3;
R1Y[I,J] := (R1Y[I-1,J] + R1Y[I,J+1] + R1Y[I,J-1]) / 3;
S1[I,J] := (S1[I,J+1] + S1[I-1,J] + S1[I,J-1]) / 3;

End;
End;
End;

Procedure Corrector;
Begin
While ErrorExceedsLimit And (ICorrect<IMax) Do
Begin
CorrectElevation;
CorrectRHSX;
CorrectRHSY;
CorrectNewBoundary;
For I:=0 To II Do
DoubleSweepY(I);
For J:=0 To JJ Do
DoubleSweepX(J);
ComputeRelativeErrors;
ComputeF1;
AddF1Incident;
ComputeF2X;
AddF2XIncident;
ComputeF2Y;
AddF2YIncident;
ComputeF2XT;
AddF2XTIncident;
ComputeF2YT;
AddF2YTIncident;
ICorrect:=ICorrect+1;
End;
SpongeComputation;
End;

Procedure CorrectRHSX;
Begin
Write(PrintFile,'Procedure CorrectRHSX.....................');
For I:=1 To II-1 Do
For J:=1 To JJ-1 Do
If NodeType[I,J]<>0 Then
R1X[I,J] := R0X[I,J] + DT/24*(9 + F2X[I,J]+1 + 19*F2X[I,J]-0 - 5*F2X[I,J]-1 + F2X[I,J]-2) + F2XT[I,J]+1 - F2XT[I,J]-0;
WriteLn(PrintFile,'Ok');
End;

Procedure CorrectRHSY;
Begin
Write(PrintFile,'Procedure CorrectRHSY.....................');
For I:=1 To II-1 Do
For J:=1 To JJ-1 Do
If NodeType[I,J]<>0 Then
WriteLn(PrintFile,'Ok');
End;

Procedure InitComputation;
APPENDIX C. CODE - TWO HORIZONTAL DIMENSIONS

Begin
ReadDataFromFile;
Assign(PrintFile,FileDir+FileName+'.Log');
Rewrite(PrintFile);
StoreTimeAndDate;
AllocateMemory;
ComputeNodeTypes;
NonDimensionalize;
ReadBathymetry;
CoefficientMatrixX;
CoefficientMatrixY;
ComputeTimeStep;
ComputeInitialCondition;
ComputeSpongeNodes;
SaveWaveAndNodeGrid;
FourierComputation;
ClrScr;
InitControlVariables;
SaveUAndS;
End;

Procedure InitControlVariables;
Begin
RelErrorS:=l;
RelErrorU:=l;
RelErrorV:=l;
IOut:=0;
N:=0;
End;

Procedure InitPredictor;
Begin
ICorrect:=0;
RelErrorS:=l;
RelErrorU:=l;
RelErrorV:=l;
End;

Procedure PredictElevation;
Begin
Write(PrintFile,'Procedure PredictElevation..............');
For I:=l To II-1 Do
  For J:=1 To JJ-1 Do
    If NodeType[I][J]>0 Then
      S1[I][J]:=S0[I][J]
      +0T/12*(23*Fl[I][J]-[0]
        -16*Fl[I][J][-1]
        +5*Fl[I][J][-2]);
      WriteLnCPrintFile,'  - Ok');
End;

Procedure Predictor;
Begin
  InitPredictor;
  PredictElevation;
  PredictRHSX;
  PredictRHSY;
  PredictNewBoundary;
  For I:=0 To II Do
    DoubleSweepY(I);
  For J:=0 To JJ Do
    DoubleSweepX(J);
  ComputeFl;
  AddFlIncident;
  ComputeF2X;
  AddF2XIncident;
  ComputeF2Y;

AddF2YIncident;
ComputeF2XT;
AddF2XTIncident;
ComputeF2YT;
AddF2 YTIncident;
End;

Procedure PredictRHSX;
Begin
Write(PrintFile,'Procedure PredictRHSX.
For I:=1 To II-1 Do
For J:=1 To JJ-1 Do
If NodeType[I][J]<>0 Then
R1X[I][J]:=ROX[I][J]
+DT/12*(23*F2X[I][J][0]
-16*F2X[I][J][1]
+5*F2X[I][J][2]
+2*F2XT[I][J][0]
-3*F2XT[I][J][1]
+F2XT[I][J][2];
WriteLn(PrintFile,'  - Ok');
End;

Procedure PredictRHSY;
Begin
Write(PrintFile,'Procedure PredictRHSY.
For I:=l To II-1 Do
For J:=l To JJ-1 Do
If NodeType[I][J]<>0 Then
R1Y[I][J]:=ROY[I][J]
+DT/12*(23*F2Y[I][J][0]
-16*F2Y[I][J][1]
+5*F2Y[I][J][2]
+2*F2YT[I][J][0]
-3*F2YT[I][J][1]
+F2YT[I][J][2];
WriteLn(PrintFile,'  - Ok');
End;

Procedure SpongeComputation;
Begin
For I:=0 To  II Do
For J:=0 To  JJ Do
Begin
S1[I][J]:=S1[I][J]/Mu[I][J];
RIX[I][J]:=R1X[I][J]/Mu[I][J];
RIY[I][J]:=R1Y[I][J]/Mu[I][J];
End;
End;

Procedure UpdateComputation;
Begin
Write(PrintFile,'Procedure UpdateComputation.
CornerPoints;
UpdateVariables;
WaveEnvelope;
WriteLn(PrintFile,'     - Ok');
If N Mod Trunc(NN/Out)=0 Then
SaveUNandS;
GotoXYCl.l);
WriteLn(Percentage completed: ',N/NN*100:8:2);
WriteLn('Iterations required : ,ICorrect:8);
WriteLn('Maximum Error on S : ,RelErrorS:8:6);
WriteLn('Maximum Error on U : ,RelErrorU:8:6);
WriteLn('Maximum Error on V : ,RelErrorV:8:6);
End;

Procedure UpdateVariables;
Begin
For I:=0 To II Do
For J:=0 To JJ Do
Begin
  F1[I][J][2]:=F1[I][J][1];
  F1[I][J][1]:=F1[I][J][0];
  F1[I][J][0]:=F1[I][J][-1];
  F2X[I][J][2]:=F2X[I][J][1];
  F2X[I][J][1]:=F2X[I][J][0];
  F2X[I][J][0]:=F2X[I][J][-1];
  F2Y[I][J][1]:=F2Y[I][J][0];
  F2Y[I][J][0]:=F2Y[I][J][-1];
  F2X[I][J][2]:=F2X[I][J][1];
  F2X[I][J][1]:=F2X[I][J][0];
  F2X[I][J][0]:=F2X[I][J][-1];
  F2Y[I][J][1]:=F2Y[I][J][0];
  F2Y[I][J][0]:=F2Y[I][J][-1];
End;
N:=N+1;
ComputeNewIncidentWave;
End;

Procedure WaveEnvelope;
Begin
If (N>=NStart) And (N<NStop) Then
  For I:=0 To II Do
  For J:=0 To JJ Do
  Begin
    If S0[I][J]>HMax[I][J] Then HMax[I][J]:=S0[I][J];
    If S0[I][J]<HMin[I][J] Then HMin[I][J]:=S0[I][J];
  End;
End;

Begin
End.
C.6 Variable.3D

Unit Variable;

Interface

{$D+,L+,N+,Y+}$

Const

\begin{align*}
\text{FourRes} & = 26; \\
\text{IMax} & = 6; \\
\text{IJMax} & = 200; \\
\text{MaxIter} & = 100; \\
\text{MM} & = 10; \\
\text{NH} & = 6; \\
\text{NRI} & = 20; \\
\text{NNMax} & = 1E5; \\
\text{SamplePeriod} & = 1; \\
\text{TFade} & = 2; \\
\text{C1} & = -0.531; \\
\text{G} & = 9.796; \\
\text{MaxErr} & = 1E-8; \\
\text{MaxIterError} & = 1E-3;
\end{align*}

Type

\begin{align*}
\text{TimeMemory} & = -2..1; \\
\text{SpaceMemory} & = 0..\text{IJMax}; \\
\text{Vector1} & = \text{Array[SpaceMemory]} \text{ Of Extended}; \\
\text{VectorPtr1} & = \text{Vector1}; \\
\text{VectorTypel} & = \text{VectorPtr1}; \\
\text{Matrix1} & = \text{Array[SpaceMemory]} \text{ Of VectorPtr1}; \\
\text{MatrixTypel} & = \text{Matrix1}; \\
\text{TimeVec} & = \text{Array[TimeMemory]} \text{ Of Extended}; \\
\text{TimeVecPtr} & = \text{TimeVec}; \\
\text{Vector2} & = \text{Array[SpaceMemory]} \text{ Of TimeVecPtr}; \\
\text{VectorPtr2} & = \text{Vector2}; \\
\text{Matrix2} & = \text{Array[SpaceMemory]} \text{ Of VectorPtr2}; \\
\text{MatrixTypel} & = \text{Matrix2}; \\
\text{Vector3} & = \text{Array[1..FourRes]} \text{ Of Extended}; \\
\text{VectorPtr3} & = \text{Vector3}; \\
\text{Matrix3} & = \text{Array[1..FourRes]} \text{ Of VectorPtr3}; \\
\text{MatrixTypel} & = \text{Matrix3}; \\
\text{Vector4} & = \text{Array[SpaceMemory]} \text{ Of Integer}; \\
\text{VectorPtr4} & = \text{Vector4}; \\
\text{Matrix4} & = \text{Array[SpaceMemory]} \text{ Of VectorPtr4}; \\
\text{MatrixTypel} & = \text{Matrix4};
\end{align*}

Var

\begin{align*}
\text{Current} & : \text{Char}; \\
\text{A1} & : \text{Extended}; \\
\text{A2} & : \text{Extended}; \\
\text{Alpha} & : \text{Extended}; \\
\text{B1} & : \text{Extended}; \\
\text{B2} & : \text{Extended}; \\
\text{C} & : \text{Extended}; \\
\text{Curr} & : \text{Extended}; \\
\text{CosTheta} & : \text{Extended}; \\
\text{Cr} & : \text{Extended}; \\
\text{DT} & : \text{Extended}; \\
\text{DelX} & : \text{Extended};
\end{align*}
DeLY : Extended;
ErrorS : Extended;
ErrorU : Extended;
ErrorV : Extended;
H : Extended;
K : Extended;
KX : Extended;
KY : Extended;
L : Extended;
LengthX : Extended;
LengthY : Extended;
RelErrorS : Extended;
RelErrorU : Extended;
RelErrorV : Extended;
SinTheta : Extended;
SumS : Extended;
SumU : Extended;
SumV : Extended;
Surf : Extended;
T : Extended;
Theta : Extended;
TMax : Extended;
Vel : Extended;
WD : Extended;

I : LongInt;
ICorrect : LongInt;
II : LongInt;
IOut : LongInt;
J : LongInt;
JJ : LongInt;
N : LongInt;
NFreq : LongInt;
NI : LongInt;
NM : LongInt;
NN : LongInt;
NOut : LongInt;
NStart : LongInt;
NStop : LongInt;
NW : LongInt;

D : MatrixType1;
DX : MatrixType1;
DY : MatrixType1;
DXX : MatrixType1;
DXY : MatrixType1;
DXXX : MatrixType1;
DXXY : MatrixType1;
DXXYY : MatrixType1;
DXYY : MatrixType1;
DYYY : MatrixType1;
D1X : MatrixType1;
D2X : MatrixType1;
D3X : MatrixType1;
D1Y : MatrixType1;
D2Y : MatrixType1;
D3Y : MatrixType1;
Friction : MatrixType1;
HMin : MatrixType1;
HMax : MatrixType1;
Mu : MatrixType1;
ROX : MatrixType1;
R1X : MatrixType1;
ROY : MatrixType1;
R1Y : MatrixType1;
SO : MatrixType1;
S1 : MatrixType1;
S1Ref : MatrixType1;
S1Tot : MatrixType1;
U0 : MatrixType1;
U1 : MatrixType1;
U1Ref : MatrixType1;
U1Tot : MatrixType1;
V0 : MatrixType1;
V1 : MatrixType1;
V1Ref : MatrixType1;
V1Tot : MatrixType1;
F1 : MatrixType2;
F2X : MatrixType2;
F2Y : MatrixType2;
F2XT : MatrixType2;
F2YT : MatrixType2;
A : MatrixType3;
NodeType : MatrixType4;
WaveNode : MatrixType4;
B : VectorType1;
DZ : VectorType1;
E : VectorType1;
F : VectorType1;
Z : VectorType1;
BT : Array[1..100] of Char;
CT : Array[1..100] of Integer;
I1 : Array[1..100] of Integer;
J2 : Array[1..100] of Integer;
J1 : Array[1..100] of Integer;
J2 : Array[1..100] of Integer;
SP : Array[1..100] of Integer;
BathyFile : String;
FileDir : String;
FileName : String;
DataFile : Text;
PrintFile : Text;

Implementation

Begin
End.
C.7 Various.3D

Unit Various;

Interface
{$D+,L+,N+,Y+}$

Uses
Variable,
Crt,
Dos;

Function Cosh(Z:Extended):Extended;
Function Fader:Extended;
Function FX(F:MatrixType1;I,J:Integer):Extended;
Function FY(F:MatrixType1;I,J:Integer):Extended;
Function FXX(F:MatrixType1;I,J:Integer):Extended;
Function FYY(F:MatrixType1;I,J:Integer):Extended;
Function FXXX(F:MatrixType1;I,J:Integer):Extended;
Function FXXY(F:MatrixType1;I,J:Integer):Extended;
Function FXY(F:MatrixType1;I,J:Integer):Extended;
Function FYY(F:MatrixType1;I,J:Integer):Extended;
Function DFXX(F:MatrixType1;I,J:Integer):Extended;
Function DFXY(F:MatrixType1;I,J:Integer):Extended;
Function DFYY(F:MatrixType1;I,J:Integer):Extended;
Function DFXXX(F:MatrixType1;I,J:Integer):Extended;
Function DFXXY(F:MatrixType1;I,J:Integer):Extended;
Function DFXY(F:MatrixType1;I,J:Integer):Extended;
Function DFYY(F:MatrixType1;I,J:Integer):Extended;
Function DGXX(G:MatrixType1;I,J:Integer):Extended;
Function DGXY(G:MatrixType1;I,J:Integer):Extended;
Function DGYY(G:MatrixType1;I,J:Integer):Extended;
Function DGXXX(G:MatrixType1;I,J:Integer):Extended;
Function DGXXY(G:MatrixType1;I,J:Integer):Extended;
Function DGXY(G:MatrixType1;I,J:Integer):Extended;
Function DGYY(G:MatrixType1;I,J:Integer):Extended;
Function HX(F:MatrixType1;I,J:Integer;G:MatrixType1):Extended;
Function HY(F:MatrixType1;I,J:Integer;G:MatrixType1):Extended;
Function Sinh(Z:Extended):Extended;
Function Tanh(Z:Extended):Extended;

Procedure AllocateMemory;
Procedure AllocateMatrix1(Var M:MatrixType1);
Procedure AllocateMatrix2(Var M:MatrixType2);
Procedure AllocateMatrix3(Var M:MatrixType3);
Procedure AllocateMatrix4(Var M:MatrixType4);
Procedure AllocateVector1(Var V:VectorType1);
Procedure ComputeTimeStep;
Procedure DeAllocateMemory;
Procedure DeAllocateMatrix1(Var M:MatrixType1);
Procedure DeAllocateMatrix2(Var M:MatrixType2);
Procedure DeAllocateMatrix3(Var M:MatrixType3);
Procedure DeAllocateMatrix4(Var M:MatrixType4);
Procedure DeAllocateVector1(Var V:VectorType1);
Procedure DoubleSweepX(J:Integer);
Procedure DoubleSweepY(I:Integer);
Procedure EndComputation;
Procedure Error(ErrCode:Integer);
Procedure InitMatrix1(Value:Extended;Var M:MatrixType1);
Procedure InitMatrix2(Value:Extended;Var M:MatrixType2);
Procedure InitVector1(Value:Extended;Var V:VectorType1);
Procedure NonDimensionalize;
Procedure ReadBathymetry;
Procedure ReadDataFromFile;
Procedure SaveUAndS;
Procedure StoreTimeAndDate;
Implementation

Function Cosh(Z:Extended):Extended;
Begin
  Cosh:=(Exp(Z)+Exp(-Z))/2;
End;

Function Fader:Extended;
Begin
  If N*DT<TFade Then
    Fader:=Sin(Pi/2*N*DT/TFade)
  Else
    Fader:=1;
End;

Function FX(F:MatrixType1;I,J:Integer):Extended;
Begin
  Case Abs(NodeType[I,J]) Of
    11,12,13,14,15:
      FX:=(F[I+1,J]-F[I,J])/DelX;
    21,22,23,24,25,41,42,43,44,45:
      FX:=(F[I-1,J]-F[I,J]+F[I+1,J])/2*DelX;
    31,32,33,34,35:
      FX:=(-F[I-2,J]+8*F[I-1,J]-8*F[I+1,J]+F[I+2,J])/(12*DelX);
    41,42,43,44,51,52,53,54,55:
      FX:=(F[I-1,J]-F[I,J])/DelX;
    111,115,151,155:
      FX:=(F[I-1,J]-F[I+1,J])/(2*DelX);
  Else
    FX:=0;
  End;
End;

Function FY(F:MatrixType1;I,J:Integer):Extended;
Begin
  Case Abs(NodeType[I,J]) Of
    11,21,31,41,51:
      FY:=(F[I,J]-F[I,J+1])/DelY;
    12,22,32,42,52,14,24,34,44,54:
      FY:=(F[I,J-1]-F[I,J]+F[I,J+1])/2*DelY;
    13,23,33,43,53:
      FY:=(-F[I-2,J]+8*F[I-1,J]-8*F[I+1,J]-F[I+2,J])/(12*DelY);
    15,25,35,45,55:
      FY:=(F[I,J-1]-F[I,J])/DelY;
    111,115,151,155:
      FY:=(F[I,J]-F[I,J-1])/DelY;
  End;
End;
APPENDIX C. CODE - TWO HORIZONTAL DIMENSIONS

\[
\begin{align*}
\text{End;}& \\
\text{Else}& \\
\text{FT} &:= 0; \\
\text{End;}& \\
\text{End;}& \\
\text{Function FXX(F:MatrixType;I,J:Integer):Extended;}& \\
\text{Begin}& \\
\text{Case Abs(NodeType[I]-[J]) Of}& \\
21,22,23,24,25,31,32,33,34,35,41,42,43,44,45,111,115,151,155:& \\
\text{FXX} &:= (F[i-1] - [J] - 2*F[i] - [J] + F[i+1] - [J])/(DelX*DelX);& \\
\text{Else}& \\
\text{FXX} &:= 0; \\
\text{End;}& \\
\text{End;}& \\
\text{Function FXY(F:MatrixType;I,J:Integer):Extended;}& \\
\text{Begin}& \\
\text{Case Abs(NodeType[I]-[J]) Of}& \\
22,23,24,32,33,34,42,43,44,11,111:& \\
15,115:& \\
51,151:& \\
55,155:& \\
\text{Else}& \\
\text{FXY} &:= 0; \\
\text{End;}& \\
\text{End;}& \\
\text{Function FYY(F:MatrixType;I,J:Integer):Extended;}& \\
\text{Begin}& \\
\text{Case Abs(NodeType[I]-[J]) Of}& \\
12,13,14,22,23,24,32,33,34,42,43,44,52,53,54,111,115,151,155:& \\
\text{FYY} &:= (F[i] - [J-1] - 2*F[i] + F[i+1] - [J+1])/DelY; \\
\text{Else}& \\
\text{FYY} &:= 0; \\
\text{End;}& \\
\end{align*}
\]
Function FXXX(F:MatrixTypel;I,J:Integer):Extended;
Begin
Case Abs((NodeType[I]-[J]) Of
31,32,33,34,35,111,115,151,155:
FXXX:=((-F-[I-2]-[J]
+2*F-[I-1]-[J]
-2*F-[I+1]-[J]
+ F-[I+2]-[J]
)/(2*DelX*DelX*DelX));
Else
FXXX:=0;
End;
End;

Function FXXY(F:MatrixTypel;I,J:Integer):Extended;
Begin
Case Abs((NodeType[I]-[J]) Of
22,23,24,32,33,34,42,43,44:
FXXY:=((-F-[I-1]-[J-1]
+2*F-[I ]-[J-1]
- F-[I+1]-[J-1]
+ F-[I-1]-[J+1]
-2*F-[I ]-[J+1]
+ F-[I+1]-[J+1]
)/(2*DelX*DelX*DelY));
111,151:
FXXY:=(( F-[I-1]-[J+1]
-2*F-[I ]-[J+1]
+ F-[I+1]-[J+1]
- F-[I-1]-[J ]
+2*F-[I ]-[J ]
- F-[I+1]-[J ]
)/(DelX*DelX*DelY));
115,155:
FXXY:=( F-[I+1]-[J-1]
-2*F-[I ]-[J-1]
+ F-[I+1]-[J+1]
- F-[I-1]-[J-1]
+2*F-[I ]-[J-1]
- F-[I+1]-[J-1]
)/(DelX*DelX*DelY);
Else
FXXY:=0;
End;
End;

Function FXYY(F:MatrixTypel;I,J:Integer):Extended;
Begin
Case Abs((NodeType[I]-[J]) Of
22,23,24,32,33,34,42,43,44:
FXYY:=((-F-[I-1]-[J]
+2*F-[I-1]-[J]
- F-[I+1]-[J]
+2*F-[I+1]-[J]
+ F-[I+1]-[J+1]
)/(2*DelX*DelY*DelY));
111,115:
FXYY:=(( F-[I+1]-[J]
-2*F-[I ]-[J]
+ F-[I+1]-[J]
- F-[I-1]-[J-1]
+2*F-[I ]-[J-1]
- F-[I+1]-[J-1]
)/(DelX*DelY*DelY));
111,155:
FXYY:=( F-[I+1]-[J-1]
-2*F-[I ]-[J-1]
+ F-[I+1]-[J-1]
- F-[I-1]-[J ]
+2*F-[I ]-[J ]
- F-[I+1]-[J ]
)/(DelX*DelY*DelY);
APPENDIX C. CODE - TWO HORIZONTAL DIMENSIONS

151,155:
  FXYY:=\((F[1][j]-[j-1])\)
  \(-2*F[1][j][j]\)
  \(+ F[1][j][j+1]\)
  \(- F[1-1][j-1]\)
  \(+2*F[1-1][j]\)
  \(- F[1-1][j+1]\)
  \)/(DeltaX*DeltaY*DeltaY);
  
  Else
  FXYY:=0;
  End;
End;

Function FYYY(F: MatrixType1; i, j: Integer): Extended;
Begin
  Case Abs(NodeType[i][j]) of
    13,23,33,43,53,111,115,151,155:
      FYYY:=\((-F[i][j-2]\)
       \(+2*F[i][j-1]\)
       \(-2*F[i][j+1]\)
       \(+ F[i][j+2]\)
       \)/(2*DeltaY*DeltaY*DeltaY);
  Else
    FYYY:=0;
  End;
End;
Function DFXX(F: MatrixType1; i, j: Integer): Extended;
Begin
  DFXX:=DXX[i][j]*FX[i][j]
       \(+2*DX[i][j]*FX[i][j]\)
       \(+ D[i][j]*FX[i][j]\)
  End;
Function DFXY(F: MatrixType1; i, j: Integer): Extended;
Begin
  DFXY:=DXY[i][j]*FY[i][j]
       \(+ DX[i][j]*FY[i][j]\)
       \(+ D[i][j]*FY[i][j]\)
  End;
Function DFYY(F: MatrixType1; i, j: Integer): Extended;
Begin
  DFYY:=DYY[i][j]*FY[i][j]
       \(+2*DY[i][j]*FY[i][j]\)
       \(+ D[i][j]*FY[i][j]\)
  End;
Function DFXXX(F: MatrixType1; i, j: Integer): Extended;
Begin
  DFXXX:=DXXX[i][j]*FX[i][j]
       \(+3*DX[i][j]*FX[i][j]\)
       \(+ D[i][j]*FX[i][j]\)
  End;
Function DFXXY(F: MatrixType1; i, j: Integer): Extended;
Begin
  DFXXY:=DXXY[i][j]*FX[i][j]
       \(+2*DX[i][j]*FX[i][j]\)
       \(+ D[i][j]*FX[i][j]\)
  End;
Function DFXYY(F: MatrixType1; i, j: Integer): Extended;
Begin
  DFXYY:=DFXYY[i][j]
       \(+2*DX[i][j]*FX[i][j]\)
       \(+ D[i][j]*FX[i][j]\)
  End;
Begin
DFXY := DXYY[I][J]*F[I][J]
    + DYY[I][J]*FX(F,I,J)
    + 2*DY[I][J]*FY(F,I,J)
    + 2*DY[I][J]*FXY(F,I,J)
    + D[I][J]*FYY(F,I,J);
End;

Function DFYYY(F:MatrixType1;I,J:Integer):Extended;
Begin
DFYYY := DYYY[I][J]*F[I][J]
    + 3*DYY[I][J]*FY(F,I,J)
    + 3*DY[I][J]*FYY(F,I,J)
    + D[I][J]*FYYY(F,I,J);
End;

Function DGXX(G:MatrixType1;I,J:Integer):Extended;
Begin
DGXX := 2*DX[I][J]*FX(G,I,J)
    + D[I][J]*FXX(G,I,J);
End;

Function DGXY(G:MatrixType1;I,J:Integer):Extended;
Begin
DGXY := DY[I][J]*FX(G,I,J)
    + DX[I][J]*FY(G,I,J)
    + D[I][J]*FXY(G,I,J);
End;

Function DGYY(G:MatrixType1;I,J:Integer):Extended;
Begin
DGYY := 2*DY[I][J]*FY(G,I,J)
    + D[I][J]*FYY(G,I,J);
End;

Function DGXXX(G:MatrixType1;I,J:Integer):Extended;
Begin
DGXXX := 3*DXX[I][J]*FX(G,I,J)
    + 3*DX[I][J]*FXX(G,I,J)
    + D[I][J]*FXXX(G,I,J);
End;

Function DGXXY(G:MatrixType1;I,J:Integer):Extended;
Begin
DGXXY := 2*DXY[I][J]*FX(G,I,J)
    + DX[I][J]*FY(G,I,J)
    + 2*DY[I][J]*FXY(G,I,J)
    + D[I][J]*FXXY(G,I,J);
End;

Function DGXYY(G:MatrixType1;I,J:Integer):Extended;
Begin
DGXYY := DYY[I][J]*FX(G,I,J)
    + 2*DY[I][J]*FY(G,I,J)
    + 2*DY[I][J]*FYY(G,I,J)
    + D[I][J]*FXYY(G,I,J);
End;

Function DGYYY(G:MatrixType1;I,J:Integer):Extended;
Begin
DGYYY := 3*DYY[I][J]*FY(G,I,J)
    + 3*DY[I][J]*FYY(G,I,J)
    + D[I][J]*FYYY(G,I,J);
End;
Function HX(F:MatrixType;I,J:Integer;G:MatrixType):Extended;
Begin
  Case Abs(NodeType'[I]'[J]) Of
    11,12,13,14,15:
      HX:=(-F'[I]'-[J]+F'[I+1]'-[J]) / DelX;
    21,22,23,24,25,41,42,43,44,45:
      HX:=(-F'[I-1]-[J]+F'[I+1]-[J]) / (2*DelX);
    31,32,33,34,35:
      HX:=(F'[I-2]-[J]-8*F'[I-1]-[J]+8*F'[I+1]-[J]-F'[I+2]-[J]) / (12*DelX);
    51,52,53,54,55:
      HX:=(-F'[I-1]-[J]+F'[I]-[J]) / DelX;
    111,115,151,155:
      If G'[I]-[J]>=0 Then
        HX:=(-F'[I-1]-[J]+F'[I]-[J])/DelX
      Else
        HX:=(-F'[I]-[J]+F'[I+1]-[J])/DelX;
      Else
        HX:=0;
      End;
  End;
End;

Function HY(F:MatrixType;I,J:Integer;G:MatrixType):Extended;
Begin
  Case Abs(NodeType'[I]'[J]) Of
    11,21,31,41,51:
      HY:=(-F'[I]-[J]+F'[I]-[J+1])/DelY;
    12,22,32,42,52,14,24,34,44,54:
      HY:=(-F'[I]-[J-1]+F'[I]-[J+1]) / (2*DelY);
    13,23,33,43,53:
      HY:=(F'[I]-[J-2]-8*F'[I]-[J-1]+8*F'[I]-[J+1]-F'[I]-[J+2]) / (12*DelY);
    15,25,35,45,55:
      HY:=(-F'[I]-[J-1]+F'[I]-[J+1]) / DelY;
    111,115,151,155:
      If G'[I]-[J]>=0 Then
        HY:=(-F'[I]-[J]+F'[I]-[J])/DelY
      Else
        HY:=(-F'[I]-[J]+F'[I+1]-[J])/DelY;
      Else
        HY:=0;
      End;
  End;
End;
C.7. Various 3D

Function Sinh(Z:Extended):Extended;
Begin
Sinh:=(Exp(Z)-Exp(-Z))/2;
End;

Function Tanh(Z:Extended):Extended;
Begin
Tanh:=(Exp(Z)-Exp(-Z))/(Exp(Z)+Exp(-Z));
End;

Procedure AllocateMemory;
Begin
Write(PrintFile,'Procedure AllocateMemory...');
AllocateMatrix1(0);
AllocateMatrix1(DX);
AllocateMatrix1(DY);
AllocateMatrix1(DXX);
AllocateMatrix1(DXY);
AllocateMatrix1(DYY);
AllocateMatrix1(DXXY);
AllocateMatrix1(DYYY);
AllocateMatrix1(D1X);
AllocateMatrix1(D2X);
AllocateMatrix1(D3X);
AllocateMatrix1(D1Y);
AllocateMatrix1(D2Y);
AllocateMatrix1(D3Y);
AllocateMatrix1(Friction);
AllocateMatrix1(HMax);
AllocateMatrix1(HMin);
AllocateMatrix1(ROX);
AllocateMatrix1(R1X);
AllocateMatrix1(R0Y);
AllocateMatrix1(R1Y);
AllocateMatrix1(SD);
AllocateMatrix1(S1);
AllocateMatrix1(S1Ref);
AllocateMatrix1(S1Tot);
AllocateMatrix1(Mu);
AllocateMatrix1(UO);
AllocateMatrix1(UI);
AllocateMatrix1(U1Ref);
AllocateMatrix1(U1Tot);
AllocateMatrix1(V0);
AllocateMatrix1(V1);
AllocateMatrix1(V1Ref);
AllocateMatrix1(V1Tot);
AllocateMatrix2(F1);
AllocateMatrix2(F2X);
AllocateMatrix2(F2Y);
AllocateMatrix2(F2XT);
AllocateMatrix2(F2YT);
AllocateMatrix4(NodeType);
AllocateMatrix4(WaveNode);
AllocateVector1(E);
AllocateVector1(F);
WriteLn(PrintFile,' - Ok');
End;

Procedure AllocateMatrix1(Var M:MatrixType1);
Begin
GetMem(M,(II+1)*SizeOf(Pointer));
For I:=0 To II Do GetMem(M[I],(JJ+1)*SizeOf(Extended));
APPENDIX C. CODE - TWO HORIZONTAL DIMENSIONS

End;

Procedure AllocateMatrix2(Var M:MatrixType2);
Begin
  GetMem(M,(II+l)*SizeOf(Pointer));
  For I:=0 To II Do
    Begin
      GetMem(M[I],(JJ+l)*SizeOf(Pointer));
      For J:=0 To JJ Do
        GetMem(M[I][J],4*SizeOf(Extended));
    End;
  End;
End;

Procedure AllocateMatrix3(Var M:MatrixType3);
Begin
  GetMem(M,FourRes*SizeOf(Pointer));
  For J:=1 To FourRes Do
    GetMem(M[J],FourRes*SizeOf(Extended));
End;

Procedure AllocateMatrix4(Var M:MatrixType4);
Begin
  GetMem(M,(II+l)*SizeOf(Pointer));
  For I:=0 To II Do
    GetMem(M[I],(JJ+l)*SizeOf(Integer));
End;

Procedure AllocateVector1(Var V:VectorType1);
Begin
  GetMem(V,SizeOf(Vector1));
End;

Procedure ComputeTimeStep;
Const
  DLimit=0.1;
Var
  SumDT,Del1:Extended;
  Number :Integer;
Begin
  Write(PrintFile, 'Procedure ComputeTimeStep

  Del:=(DelX+DelY)/2;
  SumDT:=0;
  Number:=0;
  For I:=0 To II Do
    For J:=0 To JJ Do
      If D[I][J]>=DLimit Then
        Begin
          Number:=Number+1;
          SumDT:=SumDT+1/Sqrt(D[I][J]);
        End;
  SumDT:=SumDT*Cr*Del;
  DT:=SumDT/Number;
  NN:=1+Trunc(TMax/DT);
  NStart:=NN-1-Trunc(SamplePeriod*T/DT);
  NStop:=NN;
  If NN>NHMax Then Error(20);
  WriteLn(PrintFile,' -Ok');
End;

Procedure DeAllocateMemory;
Begin
  DeAllocateMatrix1(D);
  DeAllocateMatrix1(DX);
  DeAllocateMatrix1(DY);
  DeAllocateMatrix1(DXX);
  DeAllocateMatrix1(DXY);
  DeAllocateMatrix1(DYY);
  DeAllocateMatrix1(DXXX);

DeAllocateMatrix1(DXXY);
DeAllocateMatrix1(DXY);
DeAllocateMatrix1(DTTY);
DeAllocateMatrix1(D1X);
DeAllocateMatrix1(D2X);
DeAllocateMatrix1(D3X);
DeAllocateMatrix1(D1Y);
DeAllocateMatrix1(D2Y);
DeAllocateMatrix1(D3Y);
DeAllocateMatrix1(Friction);
DeAllocateMatrix1(HMax);
DeAllocateMatrix1(HMin);
DeAllocateMatrix1(ROX);
DeAllocateMatrix1(R1X);
DeAllocateMatrix1(ROY);
DeAllocateMatrix1(R1Y);
DeAllocateMatrix1(SO);
DeAllocateMatrix1(S1);
DeAllocateMatrix1(S1Ref);
DeAllocateMatrix1(S1Tot);
DeAllocateMatrix1(SI);
DeAllocateMatrix1(VO);
DeAllocateMatrix1(V1);
DeAllocateMatrix1(V1Ref);
DeAllocateMatrix1(V1Tot);
DeAllocateMatrix2(F1);
DeAllocateMatrix2(F2X);
DeAllocateMatrix2(F2Y);
DeAllocateMatrix2(F2XT);
DeAllocateMatrix2(F2YT);
DeAllocateMatrix4(Node); 
DeAllocateMatrix4(WaveNode);
DeAllocateVector1(E);
DeAllocateVector1(F);
DeAllocateVector1(I);

Procedure DeAllocateMatrix1(Var M:MatrixTypel);
Begin
For I:=0 To II Do
  FreeMem(M[I],(JJ+1)*SizeOf(Extended));
  FreeMem(M,(II+1)*SizeOf(Pointer));
End;

Procedure DeAllocateMatrix2(Var M:MatrixType2);
Begin
For I:=0 To II Do
  Begin
    For J:=0 To JJ Do
      FreeMem(M[I][J],4*SizeOf(Extended));
      FreeMem(M,(II+1)*SizeOf(Pointer));
  End;
  FreeMem(M,(II+1)*SizeOf(Pointer));
End;

Procedure DeAllocateMatrix3(Var M:MatrixType3);
Begin
For J:=1 To FourRes Do
  FreeMem(M[J],FourRes*SizeOf(Extended));
  FreeMem(M,FourRes*SizeOf(Pointer));
End;

Procedure DeAllocateMatrix4(Var M:MatrixType4);
Begin
APPENDIX C. CODE - TWO HORIZONTAL DIMENSIONS

For I:=0 To II Do
   FreeMem(M[I],(JJ+l)*SizeOf(Integer));
   FreeMem(M,II*l*SizeOf(Pointer));
End;

Procedure DeAllocateVectorl(Var V:VectorTypel);
Begin
   FreeMem(V.SizeOf(Vectorl));
End;

Procedure DoubleSweepX(J:Integer);

Procedure ForwardSweep;
Var
   RIXModified:Extended;
Begin
   E[0]:=R1X[0][J]/D2X[0][J];
   F[0]:=-D3X[0][J]/D2X[0][J]*E[0];
   For I:=1 To II-l Do
      Begin
          RIXModified:=R1X[I][J];
          Case WaveNode[I][J] Of
          -1:Begin
              If WaveNode[I-1][J]=1 Then
                  RIXModified:=RIXModified+D1X[I][J]*UlTot[I-1][J];
              If WaveNode[I+1][J]=1 Then
                  RIXModified:=RIXModified+D3X[I][J]*UlTot[I+1][J];
          End;
          1:Begin
              If WaveNode[I-1][J]=-1 Then
                  RIXModified:=RIXModified-D1X[I][J]*UlRef[I-1][J];
              If WaveNode[I+1][J]=-1 Then
                  RIXModified:=RIXModified-D3X[I][J]*UlRef[I+1][J];
          End;
      End;
      E[I]:=(RIXModified-D1X[I][J]*E[I-1])/(D2X[I][J]+D1X[I][J]*F[I-1]);
      F[I]:=-D3X[I][J]/(D2X[I][J]+D1X[I][J]*F[I-1]);
      End;
   E[II]:=(R1X[II][J]-D1X[II][J]*E[II-l])/(D2X[II][J]+D1X[II][J]*F[II-l]);
   F[II]:=-D3X[II][J]/(D2X[II][J]+D1X[II][J]*F[II-l]);
End;

Procedure BackwardSweep;
Begin
   Vel:=E[II];
   ErrorU:=ErrorU+Abs(Vel-Ul[II][J]);
   Ul[II][J]:=Vel;
   For I:=II-l DownTo 0 Do
      Begin
          Vel:=E[I]+F[I]*Ul[I+1][J];
          ErrorU:=ErrorU+Abs(Vel-Ul[I][J]);
          Ul[I][J]:=Vel;
      End;
   End;

Begin
   ForwardSweep;
   BackwardSweep;
End;

Procedure DoubleSweepY(I:Integer);

Procedure ForwardSweep;
Var
   RIXModified:Extended;
Begin
E'[0] := R1Y'[I]'[0]/D2Y'[I]'[0];
F'[0] := D3Y'[I]'[0]/D2Y'[I]'[0];
For J:=1 To JJ-1 Do
Begin
  R1YModified := R1Y'[I]'[J];
  Case WaveNode'[I]'[J] Of
    -1: Begin
      If WaveNode'[I]'[J-1] = -1 Then
        R1YModified := R1YModified + D1Y'[I]'[J] * VlTot'[I]'[J-1];
      If WaveNode'[I]'[J+1] = -1 Then
        R1YModified := R1YModified + D3Y'[I]'[J] * VlTot'[I]'[J+1];
      End;
    1: Begin
      If WaveNode'[I]'[J-1] = 1 Then
        R1YModified := R1YModified - D1Y'[I]'[J] * VlRef'[I]'[J-1];
      If WaveNode'[I]'[J+1] = 1 Then
        R1YModified := R1YModified - D3Y'[I]'[J] * VlRef'[I]'[J+1];
      End;
    End;
  End;
  E'[J] := (R1YModified - D1Y'[I]'[J] * E'[J-1]) /
         (D2Y'[I]'[J] + D1Y'[I]'[J] * F'[J-1]);
         (D2Y'[I]'[J] + D1Y'[I]'[J] * F'[J-1]);
End;
E'[JJ] := (R1Y'[I]'[JJ] - D1Y'[I]'[JJ] * E'[JJ-1]) /
         (D2Y'[I]'[JJ] + D1Y'[I]'[JJ] * F'[JJ-1]);
         (D2Y'[I]'[JJ] + D1Y'[I]'[JJ] * F'[JJ-1]);
End;

Procedure BackwardSweep;
Begin
  Vel := E'[JJ];
  ErrorV := ErrorV + Abs(Vel - Vl'[I]'[JJ]);
  Vl'[I]'[JJ] := Vel;
  For J:=JJ-1 DownTo 0 Do
  Begin
    Vel := E'[J] + F'[J] * Vl'[I]'[J+1];
    ErrorV := ErrorV + Abs(Vel - Vl'[I]'[J]);
    Vl'[I]'[J] := Vel;
  End;
End;

Procedure EndComputation;
Begin
  Assign(DataFile, FileDir + FileName + 'H.Dat');
  Rewrite(DataFile);
  For I:=0 To II Do
  For J:=0 To JJ Do
  Begin
  End;
  Close(DataFile);
  DeAllocateMemory;
  WriteLn(PrintFile, 'Computation complete...');
  Close(PrintFile);
End;

Procedure Error(ErrCode: Integer);
Begin
  Case ErrCode Of
    10: Write(PrintFile, 'Error 10',
            ' The number of discretization points exceeds ', IJMax,'...');
    20: Write(PrintFile, 'Error 20',
            ' The number of time steps exceeds ', NNMax,'...');
    30: Write(PrintFile, 'Error 30',
            ' The number of time steps exceeds ', NNMax,'...');
  End;
PROCEDURE InitMatrix1(Value:Extended; Var M:MatrixTypel);
Begin
  For I:=0 To II Do
    For J:=0 To JJ Do
      M[I,J]:=Value;
  End;
End;

PROCEDURE InitMatrix2(Value:Extended; Var M:MatrixTypel);
Begin
  For I:=0 To II Do
    For J:=0 To JJ Do
      Begin
        M[I,J][2]:=Value;
        M[I,J][1]:=Value;
        M[I,J][0]:=Value;
        M[I,J][1]:=Value;
      End;
  End;
End;

PROCEDURE InitVector1(Value:Extended; Var V:VectorTypel);
Begin
  For J:=0 To IJMax Do
    V[J]:=Value;
End;

PROCEDURE NonDimensionalize;
Begin
  Write(PrintFile,'Procedure NonDimensionalize............');
  LengthX:=LengthX/WD;
  LengthY:=LengthY/WD;
  DelX:=LengthX/II;
  DelY:=LengthY/JJ;
  TMax:=TMax*Sqrt(G/WD);
  H:=H/WD;
  T:=T*Sqrt(G/WD);
  Curr:=Curr*Sqrt(G*WD);
  WriteLn(PrintFile,'  - Ok');
End;

PROCEDURE ReadBathymetry;
Begin
  Write(PrintFile,'Procedure ReadBathymetry............');
  Assign(DataFile,FileDir+BathyFile);
  Reset(DataFile);
  For I:=0 To II Do
    For J:=0 To JJ Do
      Begin
        ReadLn(DataFile,D[I,J].Friction[I,J]);
        D[I,J]:=D[I,J]/WD;
      End;
  Close(DataFile);
  Bl:=Cl*Cl/2;
  B2:=C1;
A1 := B1 - 1/6;
A2 := C1 + 1/2;
WriteLn(PrintFile, ' - Ok');
End;

Procedure ReadDataFromFile;
Var
 Control: String[21];
 S : String;

Function Upper(S: String): String;
Begin
 For I := 1 To Length(S) Do
  S[I] := Uppercase(S[I]);
 Upper := S;
End;

Function GetChar(S: String): Char;
Begin
 Delete(S, 1, Length(S) - 1);
 GetChar := S[1];
End;

Begin
ClrScr;
WriteLn('Program ABM3D executing...');
WriteLn;
Write('Directory for input and output:
');
GetDir(0, FileDir);
FileDir := FileDir + '\';
WriteLn(FileDir);
Write('Read input from file (excl. extension): '); 
ReadLn(FileName);
Window(1, 6, 80, 24);
Assign(DataFile, FileDir + FileName + '.Dat');
Reset(DataFile);
Read(DataFile, Control);
Read(DataFile, BathyFile);
For I := 1 To Length(BathyFile) Do
 If BathyFile[I] = ' ' Then
  Delete(BathyFile, I, 1);
Read(DataFile, Control);
Read(DataFile, LengthX);
Read(DataFile, Control);
Read(DataFile, LengthY);
Read(DataFile, Control);
NM := 4;
If Upper(Control) = 'BEGINMAINBOUNDARY...' Then
 Halt(1)
Else
 For N := 1 To NM Do
 Begin
  Read(DataFile, CT[N], I1[N], J1[N], I2[N], J2[N], SP[N]);
  Read(DataFile, S);
  BT[N] := GetChar(S);
 End;
II := I2[1];
JJ := J2[2];
If (II > IMax) Or (JJ > JMax) Then Error(10);
NI := 0;
Read(DataFile, Control);
If Upper(Control) = 'ENDMAINBOUNDARY...' Then
 Error(80);
Read(DataFile, Control);
If Upper(Control) = 'BEGININTERNALBOUNDARY' Then
 Begin
  Read(DataFile, NI);
  For I := NM + 1 To NM + NI Do

Begin
  Read(DataFile,CT[I],I1[I],J1[I],I2[I],J2[I],SP[I]);
  ReadLn(DataFile,S);
  BT[I]:=GetChar(S);
End;
ReadLnCDataFile,S);
BT[I]:=GetChar(S);
End;
If Upper(Control)<> 'ENDINTERNALBOUNDARY..' Then
  Error(70);
ReadCDataFile,Control);
End;
NW:=0;
If Upper(Control)='BEGINWAVEBOUNDARY....' Then
  Begin
    ReadLn(DataFile,NW);
    For I:=NM+NI+1 To NM+NI+NW Do
      Begin
        Read(DataFile,Il[I],Jl[I],I2[I],J2[I]);
        ReadLn(DataFile,S);
        BT[I]:=GetChar(S);
      End;
    ReadLn(DataFile,Control);
    If Upper(Control)<>'ENDWAVEBOUNDARY......' Then
      Error(60);
    ReadCDataFile,Control);
  End;
ReadLn(DataFile,Cr);
ReadCDataFile,Control);
ReadLn(DataFile,TMax);
ReadCDataFile,Control);
ReadLn(DataFile,NOut);
ReadCDataFile,Control);
ReadLn(DataFile,Theta);
ReadCDataFile,Control);
ReadLn(DataFile,WD);
ReadCDataFile,Control);
ReadLn(DataFile,Control);
ReadLn(DataFile,Control);
ReadLn(DataFile,Control);
ReadLn(DataFile,T);
ReadCDataFile,Control);
ReadLn(DataFile,Current);
ReadCDataFile,Control);
ReadLn(DataFile,Current);
Close(DataFile);
 Theta:=Theta*Pi/180;
End;
Procedure SaveUAndS;
Var
  Number,St:String;
Begin
  StrClOutrO,Number);
  St:=FileName+Number+'.Dat';
  WriteLn(PrintFile,'Saving computed data in file '+St+' at time t=','
    N*DT:6:2,...');
  Assign(DataFile,St);
  Rewrite(DataFile);
  For I:=0 To II Do
    For J:=0 To JJ Do
      WriteLnCDataFile,I*DelX:12:4,J*DelY:12:4,SO'-[I]-[J]:12:4,
        UO'-[I]-[J]:12:4,V0'-[I]-[J]:12:4);
  Close(DataFile);
  I0ut:=I0ut+1;
End;
Procedure StoreTimeAndDate;
Var
  H,M,S,HUND:Word;
Begin
GetDate(H,M,S,HUND);
Write(PrintFile,'Run time was '/',M,'-',H,' ');
GetTime(H,M,S,HUND);
WriteLn(PrintFile,H,':',M,':',S);
End;

Begin
End.