Combined Refraction/Diffraction Model for Spectral Wave Conditions

REF/DIF S
Version 1.1

Documentation and User's Manual

James T. Kirby and H.Tuba Ozkan

Center for Applied Coastal Research
Department of Civil Engineering
University of Delaware, Newark, DE 19716

CACR Report No. 94-04

February, 1994

Contents

List of Figures

Introduction

The weakly nonlinear combined refraction and diffraction model described here, denoted **REF/DIF S**, simulates the behavior of a random sea over irregular bottom bathymetry, incorporating the effects of shoaling, refraction, energy dissipation and diffraction.

Although the model is developed to simulate a random sea state it can also be used to model the behavior of monochromatic waves. Additionally, a preprocessor is included which generates a two-dimensional spectrum consisting of a frequency spectrum in conjunction with a directional spreading function, discretized to yield separate wave components characterized by a frequency and a direction. These individual wave components make up the essential part of the input to the model **REF/DIF S**.

The **REF/DIF S** model has the ability to stack up parabolic models for the individual wave components, propagating them simultaneously through the domain. This approach makes it possible to calculate statistical wave parameters after each forward step in space in the parabolic scheme.

Taking advantage of this feature, **REF/DIF S** also incorporates a new statistical breaking model. The significant wave height can be computed using solutions for all wave components and can be used to predict the energy dissipation due to breaking.

The model **REF/DIF S** includes most features present in the model **REF/DIF 1** developed by Kirby and Dalrymple (1992). The reader is referred to that document for a discussion of the historical development of the model. The model **REF/DIF S** is constructed in parabolic form, and thus there is a restriction of the model to cases where the propagation direction is within $\pm 45°$ of the assumed mean wave direction.

The **REF/DIF S** model is described in detail in this manual. The first chapter includes a short summary of the theoretical background. First, the discretization of a two-dimensional spectrum into discrete wave components is described, then the developement of wave models for these individual components is described. The assumptions adopted by the model are pointed out. Then, the energy dissipation model and the breaking model are described, followed by comments on the calculation of radiation stresses and the numerical scheme.

The second chapter includes the operating manual. An overview of the operating manual can be found in Section 2.2. Chapter 3 documents the application of the model to actual examples and provides explicit descriptions of the input and output. Appendix A includes the **REF/DIF S** program listing, in Appendix B the listing of a plotting routine is given. This program can be used to construct contour plots of the wave field. Appendix C includes the *fweb* documentation of the **SPECGEN** program listing. This program discretizes a standard two-dimensional design spectrum and creates the input file for **REF/DIF S**. Appendix D includes frequently asked questions and the answers to those questions.

1 THEORETICAL BACKGROUND

1.1 Wave Climate

Although **REF/DIF S** can be used with monochromatic wave trains, it is intended to be used with two-dimensional wave spectra. The input frequency spectrum in conjunction with a directional spreading function have to be divided into discrete wave components characterized by a certain frequency and direction. Thus the water surface elevation is represented as

$$\eta(x,y,t) = \sum_{all\,f}\sum_{all\,\theta}\left\{\frac{A(x,y;f,\theta)}{2}e^{i\psi} + c.c.\right\} \tag{1}$$

where f is the frequency, θ is the direction of any individual wave component and

$$\psi = \int \mathbf{k}\cdot d\mathbf{x} - \omega t \tag{2}$$

Discrete amplitudes and directions of the input waves have to be provided by the user on the offshore grid row, corresponding to $x = 0$. The wave is prescribed by the program as

$$A(0,y)_n = A_{n_0}e^{i\ell y} \tag{3}$$

where A_{n_0} is the given wave amplitude for the wave component n and ℓ is the wavenumber in the y-direction and is related to the wavenumber k by the relationship $\ell = k_n \sin\theta_n$, where θ_n is the angle made by the wave component n to the x-axis.

The preprocessor **SPECGEN** subdivides a standard TMA spectrum and a corresponding wrapped normal directional spreading function into a desired number of discrete components and creates the input data for **REF/DIF S**. The *fweb* documentation of the preprocessor **SPECGEN** is given in Appendix C.

As mentioned earlier, computations are performed simultaneously for all wave components with each spatial step in the computational domain. Thus, after each step in the x-direction, the complex amplitudes $A(x,y)_n$ for all wave components are known. The significant wave height $H_{1/3}$ can then be computed in the following manner.

$$H_{1/3}(x,y) = \sqrt{8\sum_{n=1}^{N} |\, A(x,y)_n\,|^2} \tag{4}$$

where N is the total number of wave components and $A(x,y)_n$ is the amplitude of the wave component n.

The significant wave height $H_{1/3}$ is part of the output from the program and is also used in several places in the computational scheme, especially the wave breaking model to be discussed later in more detail.

1.2 Wave Models for Individual Spectral Components

1.2.1 Mild slope equation

The problem of water waves propagating over irregular bathymetry in arbitrary directions is a three-dimensional problem and involves complicated nonlinear boundary condition. Very few solutions to the three dimensional problem exist and those that do are only for flat bottoms. In one horizontal dimension, sophisticated models by Chu and Mei (1970) and Djordjevic and Redekopp (1978) predict the behavior of Stokes waves over slowly varying bathymetry. In order to simplify the problem in three dimensions, Berkhoff (1972), noted that the important properties of linear progressive water waves could be predicted by a weighted vertically integrated model. (The vertical integration reduces the problem to only the two horizontal dimensions, x and y.)

Berkhoff's equation is known as the mild slope equation. It is written in terms of the surface displacement, $\eta(x, y)$. The equation, in terms of horizontal gradient operator, is

$$\nabla_h \cdot (CC_g \nabla_h \eta) + k^2 CC_g \eta = 0 \tag{5}$$

Here,

$$\begin{aligned} C &= \sqrt{(g/k)\tanh kh}, \text{ the wave celerity,} \\ C_g &= C\{1 + 2kh/\sinh 2kh\}/2, \text{ the group velocity,} \end{aligned}$$

where the local water depth is $h(x, y)$ and g is the acceleration of gravity. The local wave number, $k(x, y)$, is related to the angular frequency of the waves, ω, and the water depth h by the linear dispersion relationship,

$$\omega^2 = gk \tanh kh \tag{6}$$

The model equation (5) is an approximation; however, it is quite good even for moderately large local bottom slopes (see Booij, 1983). In both deep and shallow water, it is exact. Numerous authors have applied the mild slope model to various examples, primarily using finite element techniques. See, for example, Jonsson and Skovgaard (1979), Bettess and Zienkiewicz (1977), and Houston (1981).

For the linear mild slope equation, Radder (1979) developed a parabolic model, which had several advantages over the elliptic form presented by Berkhoff. First, the boundary condition at the downwave end of the model area is no longer necessary and, secondly, very efficient solution techniques are available for the finite difference form of the model. Radder used a splitting matrix approach, which involves separating the wave field into a forward propagating wave and a backward propagating wave, and then neglecting the backward scattered wave (which is justified in most applications as only the forward propagating wave is used for design). Radder's approximation for derivatives transverse to the wave direction results in a restriction on his parabolic model: the waves must propagate within approximately 30° of the assumed wave direction. Booij (1981) also developed a splitting of the elliptic equation, but his procedure included more terms in the approximation to the lateral derivative and therefore his procedure enables the parabolic model to handle waves propagating within 45° of the assumed direction. Booij's procedure has been used in previous versions of the **REF/DIF 1** model (up through version 2.3).

More recently, Kirby (1986b) has developed an extension to the Booij approximation based on a Minimax principle, which further extends the range of validity of the model equations. The wave-current version of the resulting model is included here and in Version 2.4 of **REF/DIF 1**, however coefficients resulting in Booij's model are used.

1.2.2 Diffraction models

In contrast to the mild slope model which is valid for varying bathymetry, researchers in the area of wave diffraction were developing models for constant bottom applications. For example, Mei and Tuck (1980) developed a simple parabolic equation for wave diffraction and applied it to the diffraction of waves by a slender island. Their equation is

$$\frac{\partial A}{\partial x} = \frac{i}{2k}\frac{\partial^2 A}{\partial y^2} \tag{7}$$

where A is a *complex amplitude* related to the water surface displacement by

$$\eta = Ae^{i\,(kx-\omega t)} \tag{8}$$

Yue and Mei (1980), using a multiple scales approach, developed a nonlinear form of this equation, which accurately predicts the propagation of a third order Stokes wave. A striking result of their numerical experiments was the development of Mach stem reflection due to the reflection of obliquely incident waves from a breakwater. This phenomenon is uniquely a nonlinear effect and not predictable from a linear refraction theory.

The parabolic model described below combines the essential features of the two approaches described above. The variable depth features of the mild-slope equation (along with extensions to include effects of wave-current interaction) are retained, but the model is developed in parabolic form and in terms of a complex amplitude A.

1.2.3 Nonlinear combined refraction/diffraction models

Kirby (1983), using a Lagrangian approach, and Kirby and Dalrymple (1983a), with a multiple scales technique, developed the predecessor to the **REF/DIF S** model, which bridged the gap between the nonlinear diffraction models and the linear mild slope equation. This model can be written in several forms depending on the application. The hyperbolic form, for time dependent applications, and the elliptic form, for steady state problems, require the use of boundary conditions on all sides of the model domain. This is a difficult requirement, as the reflected wave at a boundary is not generally known *a priori*. These models, however, have the advantage that there is no restriction on the wave direction.

A detailed comparison of results of the weakly-nonlinear model of Kirby and Dalrymple (1983a) to laboratory data was shown by Kirby and Dalrymple (1984). The laboratory test, conducted at the Delft Hydraulics Laboratory by Berkhoff, Booij and Radder (1982), consisted of determining the wave amplitude over a shoal on a sloping bottom. While results predicted by ray tracing techniques were shown by Berkhoff, Booij and Radder to be very poor, the agreement between the weakly-nonlinear model and the laboratory data was excellent. Comparisons between linear and nonlinear parabolic models clearly showed the importance of the nonlinear dispersion terms in the governing equations.

1.2.4 Wave-current interaction models

Booij (1981), using a Lagrangian approach, developed a version of the mild slope equation including the influence of strong ambient current. Kirby (1984) presented the corrected form of this mild slope model, the parabolic approximation of this form was applied by Kirby and Dalrymple (1983b) and results for waves interacting with a current jet were obtained. Their equation is

$$\begin{aligned}(C_g+U)A_x+VA_y \quad &+ \quad i(\bar{k}-k)(C_g+U)A+\frac{\sigma}{2}\left\{\left(\frac{C_g+U}{\sigma}\right)_x+\left(\frac{V}{\sigma}\right)_y\right\}A \\ &- \quad \frac{i}{2\sigma}\left((p-V^2)A_y\right)_y+i\sigma\frac{k^2}{2}D|A|^2A=0 \qquad (9)\end{aligned}$$

where $\sigma=\omega-kU$ is the principle component of the intrinsic frequency, $p=CC_g$ and $\bar{k}$ = reference wave number, taken as the average wave number along the y-axis, and U is the mean current velocity in the x-coordinate direction and V is in the y-direction. The nonlinear term includes D, which is

$$D=\frac{(\cosh 4kh+8-2\tanh^2 kh)}{8\sinh^4 kh}$$

Kirby (1986a) rederived the above equation for a wide angle parabolic approximation, which allows the study of waves with larger angles of wave incidence with respect to the x-

axis. This more accurate equation was used as the basis for earlier versions of **REF/DIF 1**. The equation has been extended to include the more accurate minimax approximation (Kirby, 1986b): however the present version of **REF/DIF S** uses the previous model coefficients. The revised governing equation for the refraction, diffraction and shoaling of the discrete wave component n is given by

$$
\begin{aligned}
& (C_{gn}+U)(A_n)_x - 2\Delta_1 V(A_n)_y + i(\bar{k}_n - a_0 k_n)(C_{gn}+U)A_n \\
+ \quad & \left\{ \frac{\sigma_n}{2}\left(\frac{C_{gn}+U}{\sigma_n}\right)_x - \Delta_1\sigma_n\left(\frac{V}{\sigma_n}\right)_y \right\} A_n + i\Delta_n'\left[\left((CC_g)_n - V^2\right)\left(\frac{A_n}{\sigma_n}\right)_y\right]_y \\
- \quad & i\Delta_1\left\{\left[UV\left(\frac{A_n}{\sigma_n}\right)_y\right]_x + \left[UV\left(\frac{A_n}{\sigma_n}\right)_x\right]_y\right\} + \frac{w_n}{2}A_n + \alpha A_n \\
+ \quad & \frac{-b_1}{k_n}\left\{\left[\left((CC_g)_n - V^2\right)\left(\frac{A_n}{\sigma_n}\right)_y\right]_{yx} + 2i\left(\sigma_n V\left(\frac{A_n}{\sigma_n}\right)_y\right)_x\right\} \\
+ \quad & b_1\beta_n\left\{2i\omega_n U\left(\frac{A_n}{\sigma_n}\right)_x + 2i\sigma_n V\left(\frac{A_n}{\sigma_n}\right)_y - 2UV\left(\frac{A_n}{\sigma_n}\right)_{xy}\right. \\
+ \quad & \left.\left[\left((CC_g)_n - V^2\right)\left(\frac{A_n}{\sigma_n}\right)_y\right]_y\right\} - \frac{i}{k_n}b_1\left\{(\omega_n V)_y + 3(\omega_n U)_x\right\}\left(\frac{A_n}{\sigma_n}\right)_x \\
- \quad & \Delta_2\left\{\omega_n U\left(\frac{A_n}{\sigma_n}\right)_x + \frac{1}{2}\omega_n U_x\left(\frac{A_n}{\sigma_n}\right)\right\} + ik\omega_n U(a_0-1)\left(\frac{A_n}{\sigma_n}\right) = 0
\end{aligned}
\tag{10}
$$

where

$$
\begin{aligned}
\beta_n &= \frac{(k_n)_x}{k_n^2} + \frac{\left(k_n\left((CC_g)_n - U^2\right)\right)_x}{2k_n^2((CC_g)_n - U^2)} \\
\Delta_1 &= a_1 - b_1 \\
\Delta_2 &= 1 + 2a_1 - 2b_1 \\
\Delta_n' &= a_1 - b_1\frac{\bar{k}_n}{k_n}
\end{aligned}
\tag{11}
$$

The model coefficients a_0, a_1 and b_1 depend on the *aperture width* chosen to specify the minimax approximation; see Kirby (1986b). The combination

$$
\begin{aligned}
a_0 &= 1 \\
a_1 &= -0.5 \\
b_1 &= 0
\end{aligned}
\tag{12}
$$

recovers Radder's approximation, while the choices

$$
\begin{aligned}
a_0 &= 1 \\
a_1 &= -0.75 \\
b_1 &= -0.25
\end{aligned}
\tag{13}
$$

recover the approximation of Booij (1981) which is still used in Version 1.0 of **REF/DIF S**. The values of a_0, a_1 and b_1 used for the minimax approximation depend on the range of angles to be considered; Kirby (1986b) found that the values for a maximum angular range of 70° gave reasonable results over the range of angles typically used. The corresponding coefficient values are

$$\begin{aligned} a_0 &= 0.994733 \\ a_1 &= -0.890065 \\ b_1 &= -0.451641 \end{aligned} \tag{14}$$

These will be incorporated in future versions of **REF/DIF S** after further testing.

Equation (10) is the model equation used in **REF/DIF S** to calculate individual wave components.

In the previous two equations, the dispersion relationship relating the angular frequency of the wave, the depth and the wave number is changed to reflect the Doppler shift due to currents. The new form of Equation (6) is

$$\sigma_n^2 = (\omega_n - k_n U)^2 = g k_n \tanh k_n h \tag{15}$$

where the absolute frequency, ω_n, is related to the intrinsic frequency, σ_n, by

$$\omega_n = \sigma_n + k_n U \tag{16}$$

where the assumption that the wave is primarily travelling in the x-direction has been used.

1.3 Assumptions

The **REF/DIF S** model, in parabolic form, has a number of assumptions inherent in it and it is necessary to discuss these directly. These assumptions are:

1. Mild bottom slope. The mathematical derivation of the model equations assumes that the variations in the bottom occur over distances which are long in comparison to a wave length. For the linear model, Booij (1983) performed a comparison between an accurate numerical model and the mild slope model for waves shoaling on a beach. He found that for bottom slopes up to 1:3 the mild slope model was accurate and for steeper slopes it still predicted the trends of wave height changes and reflection coefficients correctly.

2. Weak nonlinearity. Strictly, the model is based on a Stokes perturbation expansion and is therefore restricted to applications where Stokes waves are valid. A measure of the nonlinearity is the Ursell parameter which is given as

$$U = HL^2/h^3 \tag{17}$$

When this parameter exceeds about 40, the Stokes solution is no longer valid. In order to provide a model which is valid in much shallower water, a heuristic dispersion relationship developed by Hedges (1976) was provided as an option in the previous model **REF/DIF 1**. This relationship between the frequency and the water depth is

$$\sigma^2 = gk \tanh\left(kh(1 + |A|/h)\right) \tag{18}$$

In shallow water, this equation matches that of a solitary wave, while in deep water it asymptotically approaches the linear wave result, neglecting real amplitude dispersive effects. For this reason, a model with a dispersion relationship which is a smooth patch between the Hedges form (valid in shallow water) and the Stokes relationship (valid in deep water) is used. This hybrid model is described in Kirby and Dalrymple (1986b).

A modification has been made to this description as nonlinear effects become increasingly important in situations where strong wave breaking occurs. The significant wave height at each row is therefore incorporated into the expression so that the dispersion relation for a wave component n is

$$\sigma_n^2 = gk_n \tanh\Big(k_n h(1 + H_{1/3}/2h)\Big) \tag{19}$$

A patch between this modified form (valid in shallow water) and the Stokes form (valid in deep water) is again used to obtain a smooth dispersion relationship.

There are, as a result of the different dispersion relationships possible, three options in **REF/DIF S**: (1), a linear model, (2), a Stokes-to-Hedges nonlinear model, and (3), a Stokes model,which incorporates the third order self-interaction of each component in the wave train. Of these options, the second will cover a broader range of water depths and wave heights than the others and is also modified to correct results for breaking cases.

3. The wave direction is confined to a sector $\pm 45°$ to the principal assumed wave direction, due to the use of model coefficients recovering the approximation by Booij (1981).

1.4 Energy Dissipation

1.4.1 General form

Energy dissipation in the model occurs in a number of ways depending on the situation being modelled. An energy loss term, due to Booij (1981) and expanded by Dalrymple *et al.* (1984a), permits the model to treat bottom frictional losses due to rough, porous or viscous bottoms and surface films. The linear form of the mild slope equation with dissipation is

$$\frac{\partial A_n}{\partial x} = \frac{i}{2k_n}\frac{\partial^2 A_n}{\partial y^2} - \frac{w_n}{2C_{gn}}A_n \tag{20}$$

where the dissipation factor, w_n, is given by a number of different forms depending on the nature of the energy dissipation. The factor w_n is the energy dissipation divided by the energy for a wave component n and has the units of time^{-1}.

1.4.2 Laminar surface and bottom boundary layers

At the water surface and at the bottom, boundary layers occur due to the action of viscosity. For a contaminated surface, resulting from surface films (natural or otherwise), a significant amount of damping occurs, which is dependent on the value of the fluid viscosity. From Phillips (1966), the surface film damping is

$$w_n = \frac{\sigma_n k_n \sqrt{(\nu/2\sigma_n)}(1-i)}{\tanh k_n h} \tag{21}$$

where ν is the kinematic viscosity. The term under the square root sign is related to the thickness of the boundary layer, which is generally small. At the bottom, the boundary layer damping is

$$w_n = \frac{2\sigma_n k_n \sqrt{(\nu/2\sigma)}(1-i)}{\sinh 2k_n h} \tag{22}$$

By setting the input switch, *iff(3)=1* in the **REF/DIF S** model, surface and bottom damping are computed at all locations in the model.

1.4.3 Turbulent bottom boundary layer

In the field, the likely wave conditions are such that the bottom boundary layer is turbulent. In this case, an alternative specification of the energy dissipation must be provided. Utilizing a Darcy-Weisbach friction factor, f, the dissipation term can be shown to be

$$w_n = \frac{2\sigma_n k_n f|A_n|}{3\pi \sinh 2k_n h \sinh k_n h} \tag{23}$$

See Dean and Dalrymple (1984). In order to implement this damping term, the value of $f = 0.01$ was assumed. In **REF/DIF S**, if the input data switch *iff*(1)=1, then turbulent damping is computed at all locations.

1.4.4 Porous sand

Most sea bottoms are porous and the waves induce a flow into the bed. This results in wave damping due to the Darcy flow in the sand. For beds characterized by a given coefficient of permeability, C_p, the damping can be shown to be

$$w_n = \frac{g k_n C_p}{\cosh^2 k_n h} \tag{24}$$

The coefficient of permeability, C_p, has the units of (m^2) and is of order 4.5 x 10^{-11} m^2. Liu and Dalrymple (1984) show, for very permeable sands, that the damping is inversely related to C_p and a different w term must be used. However, this case is not likely to occur in nature. Porous bottom damping is computed in **REF/DIF S** when *iff*(2) = 1.

1.5 The Breaking Model

In contrast to the previous versions of **REF/DIF 1** individual parabolic models for separate wave components are stacked up in **REF/DIF S** making it possible to compute statistical properties after each spatial step in the parabolic scheme. Thus the significant and root-mean-square (rms) wave heights at each point are computed after each forward step in space using the computed complex wave amplitudes.

A breaking model by Thornton and Guza (1983) is used. They showed that the energy dissipation can be expressed as

$$\frac{\partial E C_{gn}}{\partial x} = -\epsilon_b \tag{25}$$

where the energy E and the bore dissipation ϵ_b can be expressed as

$$E = \frac{1}{8} \rho g H_{rms}^2 \tag{26}$$

$$\epsilon_b = \frac{3\sqrt{\pi}}{16} \frac{\rho g \bar{f} B^3}{\gamma^4 h^5} H_{rms}^7 \tag{27}$$

Here, $\bar{f}$ is the peak frequency of the frequency spectrum, h is the local water depth, B and γ are constants. In the testing of the model $B = 1$ and $\gamma = 0.6$ were used. The rms wave height is related to the significant wave height by

$$H_{rms} = \frac{1}{\sqrt{2}} H_{1/3} \tag{28}$$

The energy dissipation is built directly into the algorithm using an additional breaking term. No criterion for turning breaking on or off has to be stated as the breaking term is always in effect. Thus, the linear form of the mild slope equation with dissipation and breaking terms is now

$$\frac{\partial A_n}{\partial x} = \frac{i}{2k}\frac{\partial^2 A_n}{\partial y^2} - \frac{w_n}{2C_g}A_n - \alpha A_n \tag{29}$$

The coefficient α is a function of the above mentioned bore dissipation ϵ_b and is very small in magnitude when breaking does not occur but takes on significant values once breaking starts to occur, and thus energy starts to be dissipated out of the system.

$$\alpha = \frac{4\epsilon_b}{\rho g H_{rms}^2} = \frac{3\sqrt{\pi}}{4}\frac{\bar{f}B^3}{\gamma^4 h^5}H_{rms}^5 \tag{30}$$

No modifications have been made to Thornton and Guza's breaking model to account for directional effects. Only the change in the energy flux in the x-direction is considered and the energy flux in the y-direction is assumed constant. Also, no modifications have been made to account for the possibility that the breaking waves stop breaking and reform when they encounter an increase in water depth. These assumptions will require further future study.

The results of experiments performed by Vincent and Briggs (1989) show that in cases with strong wave breaking over a shoal nonlinear effects prevent the development of a focus due to the increase in the speed of the waves breaking over the crest. An attempt to account for this effect has been made by modifying the approximate nonlinear formulation by Kirby and Dalrymple (1986b) using the statistical wave height in shallow water. This formulation (Eqn. 19) has been discussed in Section 1.3.

1.6 Radiation Stresses

Radiation stresses for waves propagating at an angle θ to the x-direction are computed after each forward step in x. For a monochromatic wave the radiation stress in the y-direction due to the excess momentum flux in the x-direction is given by

$$S_{xy} = E\frac{C_g}{C}\sin\theta\cos\theta \tag{31}$$

where the wave energy $E = \frac{1}{2}\rho g |A|^2$.

For the spectral case a summation over all wave components is performed, thus S_{xy} is given by

$$S_{xy}(x,y) = \frac{1}{4}\rho g\sum_{n=1}^{N}\left(\frac{C_{gn}}{C_n}\right)(x,y)|A(x,y)_n|^2\sin 2\theta(x,y)_n \tag{32}$$

where $\theta(x,y)_n$ is the wave direction of the wave component n at the location (x,y) and is computed using the same algorithm as in **REF/DIF 1**.

Similarly the radiation stress in the x-direction due to the momentum flux in the x-direction is given by

$$S_{xx}(x,y) = \frac{1}{2}\rho g \sum_{n=1}^{N} \mid A(x,y)_n \mid^2 \left\{ \left(\frac{C_{gn}}{C_n} \right) (x,y)(1+\cos^2\theta(x,y)_n) - \frac{1}{2} \right\} \tag{33}$$

Finally the radiation stress in the y-direction due to the momentum flux in the y-direction is given by

$$S_{yy}(x,y) = \frac{1}{2}\rho g \sum_{n=1}^{N} \mid A(x,y)_n \mid^2 \left\{ \left(\frac{C_{gn}}{C_n} \right) (x,y)(1+\sin^2\theta(x,y)_n) - \frac{1}{2} \right\} \tag{34}$$

The values of these three components are written into a seperate output file *outrad.dat* after each spatial step.

1.7 Numerical Development

1.7.1 Crank-Nicolson Technique

The parabolic model is conveniently solved in finite difference form. In order to accomplish this, the study area bathymetry must be input as a grid with the (x,y) directions, divided into rectangles of Δx and Δy sizes. The complex amplitude $A(x,y)$ will then be sought at each grid location and therefore, we can keep track of A by denoting its location, not by (x,y), but by (i,j) where $x=(i-1)\Delta x$ and $y=(j-1)\Delta y$. Now we have to determine the values of $A_{n_{i,j}}$ which satisfy Equation (10) for all wave components n from 1 to N, for all i between 1 and mr and for all j between 1 and nr. The procedure involves expressing all the derivatives in the (x,y) directions in terms of the complex amplitude at various grid points. For example, the forward difference representation of

$$\frac{\partial A_n}{\partial x} = \frac{A_{n_{i+1,j}} - A_{n_{i,j}}}{\Delta x} \text{ at location } i,j, \text{ for component } n \tag{35}$$

If a forward difference is used for the x-direction and a central difference representation centered at i is used for the second derivatives in the lateral direction for all the derivatives in Equation (10), then an explicit finite difference equation results for $A_{n_{i+1,j}}$. This equation can be solved directly for all the $A_{n_{i+1,j}}, j = 1,2,3...nr$, for a given i and n, provided appropriate lateral boundary conditions are prescribed. This explicit representation is not as accurate as an implicit scheme and therefore an implicit Crank-Nicolson procedure is used for the amplitude calculations. For each wave component n and a given i row, the Crank-Nicolson scheme can be written

$$a_{n_j} A_{n_{i+1,j+1}} + b_{n_j} A_{n_{i+1,j}} + c_{n_j} A_{n_{i+1,j-1}} = B_{n_{i,j}} \tag{36}$$

The right hand side can be expressed in terms of known quantities:

$$B_{n_{i,j}} = d_{n_j} A_{n_{i,j+1}} + e_{n_j} A_{n_{i,j}} + f_{n_j} A_{n_{i,j-1}} \tag{37}$$

The amplitudes at $(i+1)$ are unknown, whereas the amplitudes at i are known from the previous calculations or boundary conditions. The coefficients a_{n_j}, b_{n_j}, c_{n_j}, d_{n_j}, e_{n_j} and f_{n_j} are complex valued, complicated functions that include nonlinear terms.

Writing the above equation for all j and for fixed i results in n tridiagonal matrix equations. These n equations can be combined resulting in a three-dimensional coefficient matrix, a two-dimensional unknown matrix and a two-dimensional known right hand side. A schematic representation of this equation is given in Figure 1. The solution for the amplitudes A_n for all j is performed at fixed i for all wave components. The tridiagonal matrix solution procedure by Carnahan *et al.* (1969), which is modified to handle complex variables, is used for the solution. Due to the nonlinearity of the finite difference equation, nonlinear terms are approximated on a first pass by using the $A_{n_{i,j}}$ values. Once the $A_{n_{i+1,j}}$ terms are computed, the equation is solved again for $A_{n_{i+1,j}}$ using now the just-calculated values in the nonlinear terms. This two-pass iterative method insures that the nonlinearities in the model are treated accurately (Kirby and Dalrymple, 1983a).

Furthermore, the model is set up to take advantage of a vector processor. The solution of the stacked tridiagonal matrix equations for all wave components is obtained using vectorization in n. This vectorization involves computing the amplitudes $A_{n_{i+1,j}}$ simultaneously for all n and at any j. This method greatly reduces the amount of computational time needed to solve the model equations.

The solution proceeds by moving one grid row in the x-direction (incrementing i by one) and, using the two-pass implicit-implicit technique, determining the complex amplitude $A_{n_{i+1,j}}$ for all wave components and for all the values of j on this row. Marching in the wave direction these calculations are repeated until all the A_n are known for all i and j. While it appears that the Crank-Nicolson procedure could be time consuming, given that there is a matrix inversion for each grid row, the coefficient matrix size is only 3 by n and the matrix inversion procedure is, in fact, very fast. The procedure is economical on storage as only the values for the rows i and $i+1$ are necessary at each calculation.

The amount of computations required is also greatly reduced by taking advantage of the physical properties of the domain. It is realized that the coefficients a_{n_j}, b_{n_j}, c_{n_j}, d_{n_j}, e_{n_j} and f_{n_j} include terms that are only related to the frequency of a wave component (e.g., the wavenumber k and wave celerity C). These terms only have to be computed for wave components with different frequencies as they will be the same for two waves with the same frequency and different directions.

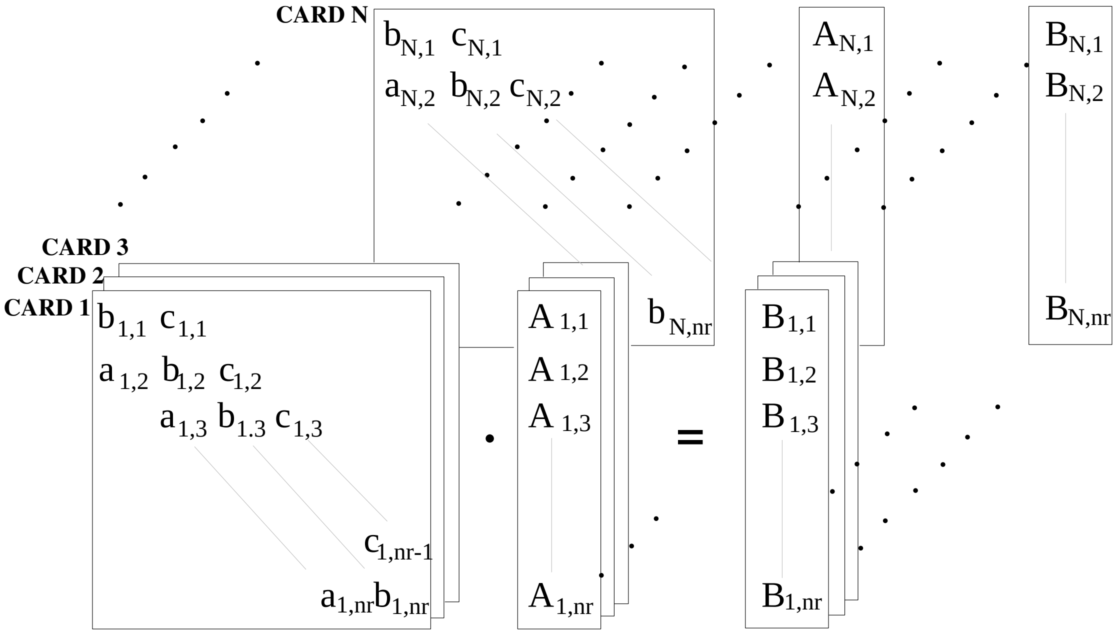

Figure 1: Vectorized Tridiagonal Matrix Equation

1.7.2 Initial and Lateral Boundary Conditions

The initial condition is vital for the parabolic model. The furthest seaward grid row, corresponding to $i = 1$, is taken as constant depth and the discretized incident wave spectrum is prescribed here. The individual wave components are then propagated over the bathymetry by the model.

As in the solution of any differential equation in a domain, the lateral boundary conditions are important. There are several ways to treat the boundaries; however, none of the presently existing boundary conditions result in the total transmission of scattered waves. Therefore, for the **REF/DIF S** model, a totally reflecting condition is generally used for each side ($j = 1$ and n). This requires that the specification of the model grid be done with care, as the reflection of the incident wave from the lateral boundaries can propagate into the region of interest rapidly and result in erroneous results.

In general, the width of the model should be such that no reflection occurs until far downwave of the region of interest. As a precaution, a graphical representation of the computed wave field should be examined to determine where the reflection from the boundaries is important. By using the switch, *ibc*, partially transmitting boundaries can be used (Kirby, 1986c). In general, this boundary condition will result in less reflection in the model domain; however, since some reflection will occur, it is recommended that runs be carried out with the reflecting boundary conditions in order to assess the regions potentially affected by reflection from the model boundaries.

1.7.3 Subgrids

In order to reduce the amount of data input and yet provide the user the ability to prescribe the fine scale bathymetry in areas of interest, **REF/DIF S** utilizes a coarse scale user-specified reference grid and a fine scale subgrid, which can have many times the resolution of the reference grid. The principal purpose of the subgrid is to provide enough computational points to the numerical model to preserve accuracy. The user specifies the number of subgrid divisions in the y-direction with the parameter *nd*. If *nd*=1, then the subgrid spacing in the y-direction is the same as the reference grid. If *nd*=2, then the model uses twice as many computational points in the y-direction as there are in the reference grid. In the propagation direction, x, the model will automatically determine the subgrid spacing if *ispace* has been set to unity. Otherwise, the user provides the subgrid spacing using the input, *mr*, which permits variable spacing in the x-direction. For subgrids, the input flag, *isp*, must be set to one and an array, *isd*, must be specified.

1.7.4 Damping

When the large angle parabolic approximation is used as a basis for the computation of wave fields around islands, the presence of wave breaking, and resulting sharp lateral variations in wave height, leads to the generation of high-wavenumber spectral components in the computed complex amplitude A in the lateral (y) direction. Kirby (1986a) has shown that these components have propagation velocities which can become large in an unbounded fashion; as a result, they can propagate across the grid, filling the computational domain with high-wavenumber noise.

The high frequency noise generated by the breaking process is controlled by an algorithm which has recently been developed by Kirby (1993). Reference should be made to that document for a description of the algorithm. The damping is built into the computational algorithm and is turned on at all times. The damping coefficient *cdamp* is specified in the code for **REF/DIF S** at the beginning of the subroutine *fdcalc* (after the statement functions). The effects of several values of damping coefficients on the wave fields are documented in Kirby (1993).

2 USER'S MANUAL

This chapter provides the operating manual for the program **REF/DIF S**. Chapter 3 provides sample documentation and calculations for example problems.

2.1 REF/DIF S Program History

REF/DIF S is essentially an enhancement to **REF/DIF 1** . **REF/DIF S** allows for the simultaneous computation of wave components with different frequencies and directions in order to simulate a random sea. Such a simulation would have to be made using sequential computations in **REF/DIF 1** . As wave amplitudes for all components are computed simultaneously at any grid location, statistical analysis can be performed. **REF/DIF S** also incorporates a statistical wave breaking model.

REF/DIF S should be regarded as a statistical model. It should be noted that some options valid for **REF/DIF 1** are not included in the structure of **REF/DIF S**.

1. The desired two-dimensional spectrum is input as discrete wave components having frequencies and directions associated with them. Because of the large amount of wave components needed to describe a spectrum properly, it is fairly unlikely that a user would desire to input the complex amplitude of each wave on the offshore boundary. Therefore, this option is turned off. This corresponds to setting the switch *iinput* in the data file to unity.

2. Another option existing in **REF/DIF 1** gives the user the opportunity to save the last row of computed amplitudes. Because of the large number of wave components present this option would result in a file that would allocate a large amount of disk space. Anticipating that this option would not be desired, it is turned off. This is achieved by setting *ioutput* to unity.

3. **REF/DIF S** expects an initial wave field in form of discrete wave components (*iwave*=1). These wave components may represent any desired two-dimensional spectrum. Therefore, there is no need for an option, *iwave*=2, which in **REF/DIF 1** corresponds to a directional spreading of the input wave condition. So *iwave* has to be set to unity.

   Even if any mistakes are made during input of these three switches, the program will automatically correct those and set the switches to the mentioned values.

4. The value of the tidal offset has to be a constant with respect to all frequencies. This was not a requirement in **REF/DIF 1** , where separate components are treated as completely independent simulations.

5. The number of waves per frequency *nwavs* is a constant, too. Thus there can only be a uniform number of directional wave components for each frequency component.

REF/DIF S needs the following additional information as input:

1. As a discretized spectrum is input into the model, the smooth spectrum cannot be reconstructed. Therefore, the peak frequency of the original spectrum that was discretized by a preprocessor has to be input by the user.

2. The output from **REF/DIF S** is given in terms of the statistical wave height. An option is built into the model in case the user might want to analyze any specific wave component. If turned on this option enables amplitude information about that specific component to be written into a separate output file *specamp.dat*.

Because of the possibility that the user wants to use an input file constructed for **REF/DIF 1**, these additional lines are not necessary in cases where only one wave component is specified. However, if more than one wave component is specified in the input file, **REF/DIF S** will require these additional lines to be present in the input file.

2.2 Overview of Operating Manual

This section provides a description of the program structure (section 2.3), followed by some notes on problems which are likely to be encountered during the installation and use of the program on different computer systems (section 2.4). Section 2.5 presents the two levels of grid information used by the program. Section 2.6 gives information about preprocessing spectral input. The input data file structure is then discussed. The program reads data in two essentially separate groups. The first group of data establishes the size of the model grid and gives the wave conditions to be studied; this group is discussed in section 2.7. The second group of data gives the reference grid data values and defines any user-specified subgrids; this is discussed in section 2.8. The structure of the program output is discussed in section 2.9. A listing of the program is included in Appendix A. A listing of a plotting program used to interpret the model output is given in Appendix B. An *fweb* documentation of a preprocessor used to generate input data files is given in Appendix C. A list of frequently asked questions is given in Appendix D.

2.3 Program Outline and Flow Chart

The model **REF/DIF S** is organized in one main program *refdifs* and eleven subroutines. The program does not contain calls to any external package, and should be free standing on any system. The program is written in a subset of FORTRAN 77 and should be a legal code for any compiler which accepts the full FORTRAN 77 standard language. Exceptions might be **open** statements, which might have to be altered due to macine and site specific conditions.

REF/DIF S is structured in two levels; a main level, which reads in and checks input data and then starts the operation of the wave model level, and the model level itself, which performs the actual finite difference calculations. The flow charts for the two levels are given in Figures 2 and 3. A short description of each routine in the model follows.

1. *refdifs*: Main program controls the calls to *inref* and *inwave* to read in data, and to *model*, which performs the actual calculations. No calculations are performed by this routine.

2. *inref* : Called by *refdifs*. *inref* reads in data controlling reference grid dimensions and the grid interpolation scheme from logical device number *iun*(5), and reads in the reference grid values of depth *dr*, x-direction velocity *ur* and y-direction velocity *vr* from logical device number *iun*(1). Some data checking is performed. If data is read in in English units, *inref* converts it to MKS units using the *dconv* factor. Output files are initialized. A description of the data files may be found in sections 2.5-2.8 of this manual. At the end of the subroutine, control is returned to *refdifs*.

3. *inwave* : Called by *refdifs*. *inwave* reads in data specifying the initial wave field along the first row of reference grid points. Data is read from logical device number *iun*(5). Conversion to MKS units is performed for data read in in English units. Control is returned to *refdifs*.

4. *model* : Called from *refdifs*. *model* controls execution of the computational part of the program. *model* performs the following series of operations:

   (a) Initializes program by calculating the incident wave field on the first grid row for all wave components.

   (b) Then, for each grid block in the reference grid:

      - Calls *grid* to perform the grid interpolation specified in the input data.
      - Calls *con* to calculate constants on the first row for all frequency components in the interpolated grid.
      - For each row in the interpolated grid:
         - Calls *con* to calculate constants on row for all frequency components.
         - Calls *fdcalc* to perform the numerical integration of the parabolic equation on row for all wave components.

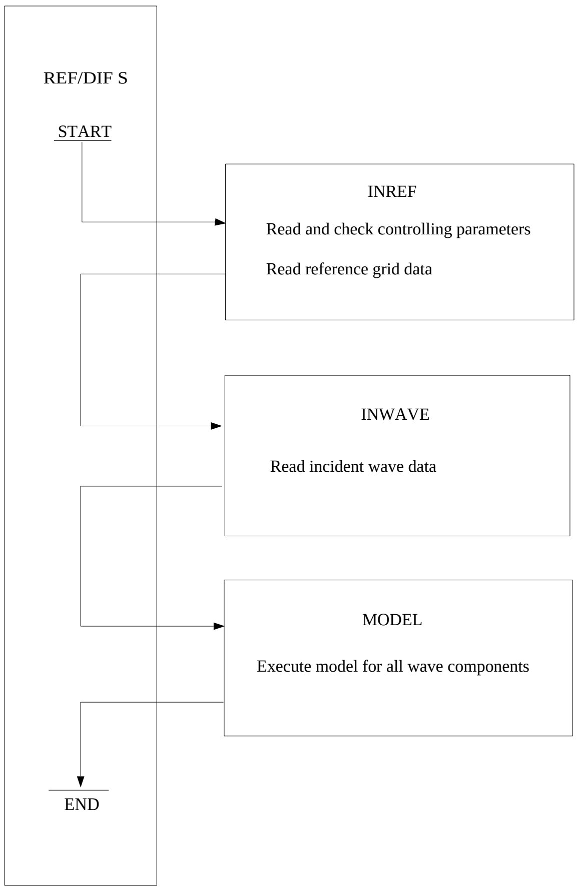

Figure 2: REF/DIF S: *refdifs* program level

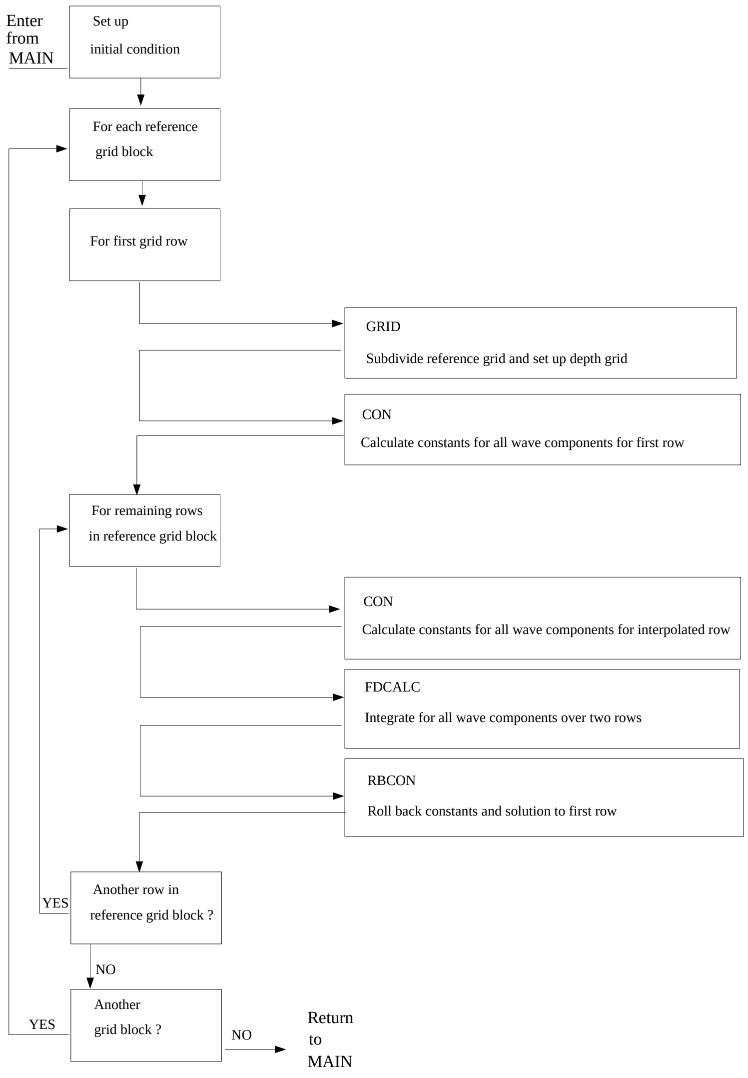

Figure 3: REF/DIF S: *model* subroutine level

– Calls *rbcon* to roll back solution to previous row for all wave components.

The model execution is then complete. Control is returned to *refdifs*.

5. *grid* : Called by *model*. *grid* performs the required interpolation over a single grid block of the reference grid as specified in the input data.The interpolation is performed as described in Section 2.5. *grid* checks to see whether a user-specified subgrid feature should be read in, and reads it in from logical unit number $iun(2)$ if needed. The interpolated depth grid is then corrected for tidal offset, and checked for surface-piercing features. These features are modified using the "thin-film" approach; see Kirby and Dalrymple (1986a). Control is returned to *model*.

6. *con* : Called by *model*. *con* calculates various constants for all frequency components for a row in the reference grid created by *grid*. Control is returned to *model*.

7. *fdcalc* : Called from *model*. *fdcalc* performs the integration of the governing parabolic equation over the grid defined in *grid*. The coefficients of the finite difference form of the parabolic equation are developed according to the Crank-Nicolson method. A complete description of the equations and the treatment of nonlinearities may be found in Kirby (1986a) and Kirby and Dalrymple (1986b).

   The parabolic equation is integrated for one row in the subdivided *ir* grid block and for all wave components. Control is then returned to *model*.

8. *rbcon*: Called by *model*. *rbcon* overwrites previous values of constants and solution with computed values and re-initializes arrays containing these computed values. So most recently calculated values are rolled back to the previous row for all wave components.

9. *wnum*: Called by *model*. *wnum* performs a Newton-Raphson solution of the linear wave-current dispersion relation to obtain values of the wavenumber k at one point for all frequency components.

10. *vwnum*: Called by *grid* and *con*. *vwnum* performs a Newton-Raphson solution of the linear wave-current dispersion relation to obtain values of the wavenumber k for a grid row for all frequency components.

11. *vtrida*: Vectorizable utility routine which is called by *fdcalc* to perform the double sweep elimination to solve the implicit set of equations for all wave components.

12. *diss*: Called by *con*. *diss* calculates frictional dissipation coefficients based on values of the switches read in by *inref* for all wave components.

2.4 Special Installation Instructions

Several features of the program **REF/DIF S** may require some modification or customizing during program installation on various systems. **REF/DIF S** is written using the features of FORTRAN 77. The code is arranged to take advantage of automatic vectorization in the CRAY CF77 compiler; however, no explicit vector code is used.

If the program is to be used on machines with no upward limit placed on the size of the compiled, executable code, only two variables have to be checked during the initial installation of the program. These are the logical device numbers for the controlling input date file and the device number for line printer output. These numbers may vary from system to system. They appear in the program as:

iun(4): logical device number for line printer output. (initialized in the main program *refdifs*).

iun(5): logical device number for controlling input data file. (initialized near the top of subroutine *inref*).

The supplied version of **REF/DIF S** has those values set to *iun*(4)="*" and *iun*(5)=5. If a device number is to be given for *iun*(4), the choice of 6 for output should be fairly standard (one exception is the DEC-10, which uses the device number 5 for both input from file or screen and output to the line printer). The version of the program currently used on SUN workstations has the default device indicator "*" substituted for *iun*(4) everywhere. Output can be directed to a data file using the directed output switch in either UNIX or DOS.

Many systems require that access to disk files be initialized and terminated by the use of **open** and **close** statements in the program code. Since the parameter list of the **open** statement varies from system to system, the user should take care that the **open** statements are compatible with the system software being used. The **open** statements appear near the top of *refdifs*, *inref* and *model*. The corresponding **close** statements appear near the end of *refdifs*. The present model is internally configured to run on a SUN system.

When writing **open** statements, a choice of a fixed record length of 132 (line printer width) will safely bracket all the **format** specifications in the program.

REF/DIF S has at this time been run on SUN Sparcstation 2's, IBM RS 6000 and CRAY YMP machines, all using versions of UNIX operating systems.

2.5 Computational Grids and Grid Interpolation

The reference grid terminology is defined in Figure 4. The grid consists of a mesh of points with dimensions $mr \times nr$ in x and y. The values of mr and nr must be less than or equal to the parameters *ixr* and *iyr* whose values are set in the **parameter** statements. Reference grid data of depth *dr* and ambient current components *ur* and *vr* are defined at the grid points. The program assumes that the x, y coordinate system is established with the origin at grid point (*ir*,*jr*)=(1,1). In this manual, we make the distinction between the terms "grid row " *ir*, which is the row of points *jr = 1, nr* at location *ir*, and "grid block" *ir*, which is the physical space between grid rows *ir* and *ir*+1.

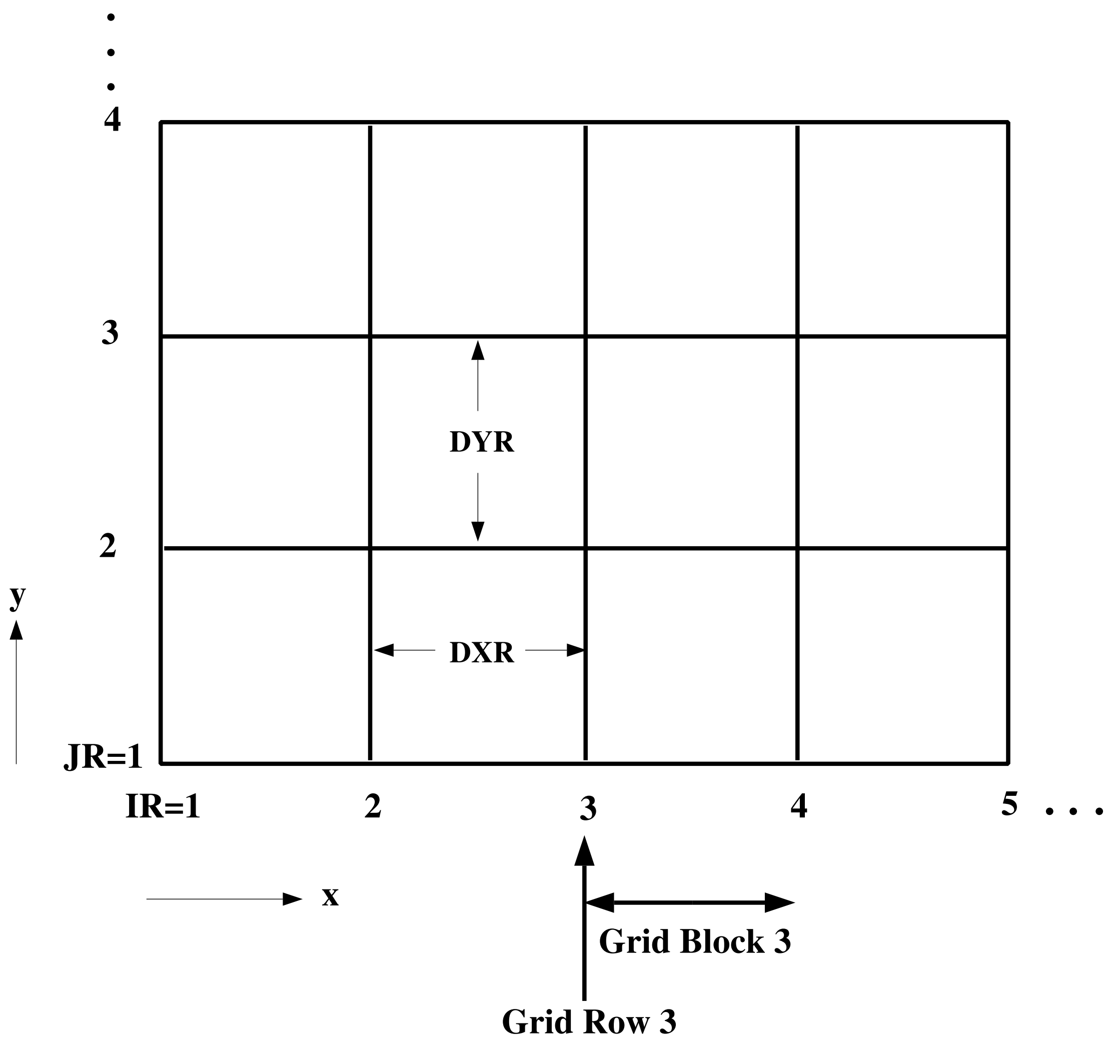

Figure 4: Reference grid notation

The reference points are separated by spacings dxr and dyr which are uniform in the x- and y-directions. The spacings dxr and dyr may have arbitrary, independent values.

Values of dr, ur and vr constitute the "reference grid data". Section 2.8 describes the required input data file for these quantities.

The computational grid for any particular application should be chosen with care. Since **REF/DIF S** tries to use at least 5 points per wavelength of the shortest wave being modelled, the length of the computational domain in the propagation direction is restricted (with the given **parameter** statements). This range can be extended by increasing ixr and/or ix in the **parameter** statements. The width of the model domain should be chosen such that interference from the boundaries does not affect the study area. If the reflective boundary conditions are used, the extent of boundary influence is usually obvious, particularly when viewing two-dimensional plots of the significant wave height $H_{1/3}$.

2.5.1 Grid Subdivision

The only major feature of **REF/DIF S** which is not described in Chapter 1 is the ability to subdivide the given reference grid into a more finely subdivided computational grid. This would usually be done in cases where the reference grid spacing is too large to be used directly for calculations. In this case, the user may specify how the reference grid is to be subdivided, or the user may tell the program to attempt its own subdivisions.

The maximum reference grid dimensions have been set fairly arbitrarily and may be changed by modifying the **parameter** statements at the beginning of each routine in the program. The grid is large enough to comply with typical grids for various tidal computational models, which may be used to specify the ambient currents, and get small enough so that data values do not occupy too much internal storage. It is anticipated that the spacings *dxr* and *dyr*, if based on such a model, may be too large for an accurate integration of the parabolic model to be performed. Consequently, the model has been arranged so that the reference grid block between any two adjacent reference grid rows may be subdivided down to a finer mesh in order to provide sufficient resolution in the computational scheme. This subdivision process is performed internally in the program (with an exceptional feature to be described below), and may be controlled by the user or the program.

Remember that the computational scheme proceeds by marching in the x-direction, and, therefore, the only reference grid information required at any particular step is the data on row *ir*, where computation starts, and on row *ir*+1, where computation ends. The fine, subdivided mesh is set up on the intervening grid block. An example of a particular subdivision of a grid block is shown in Figure 5. Here, the choices of x-direction subdivision *md*(*ir*)=5 and *nd*=2 are illustrated, with *md* and *nd* being the number of spaces each reference grid cell is divided into, rather than the number of extra points being inserted.

Several restrictions are placed on the choice of *nd* and *md*'s. First, the maximum dimensions of the subdivided grid cell is given by the parameters *ix* and *iy*. This implies that any *md* can be at most (*ix*-1) and *nd* can be at most *(iy-1)/(nr-1)*. The maximum number of added spaces may be increased by increasing *ix* and *iy* in the **parameter** statements. Further, the y-subdivision specified by *nd* is applied uniformly along each grid row for the full extent to the reference grid. No provision is made for variable grid spacing in the y-direction. Grid spacing in the x-direction is arbitrary, so *md*'s may differ arbitrarily for each grid block.

The user must specify the single value of *nd* in the input data. Two choices may be made regarding *md*'s, however.

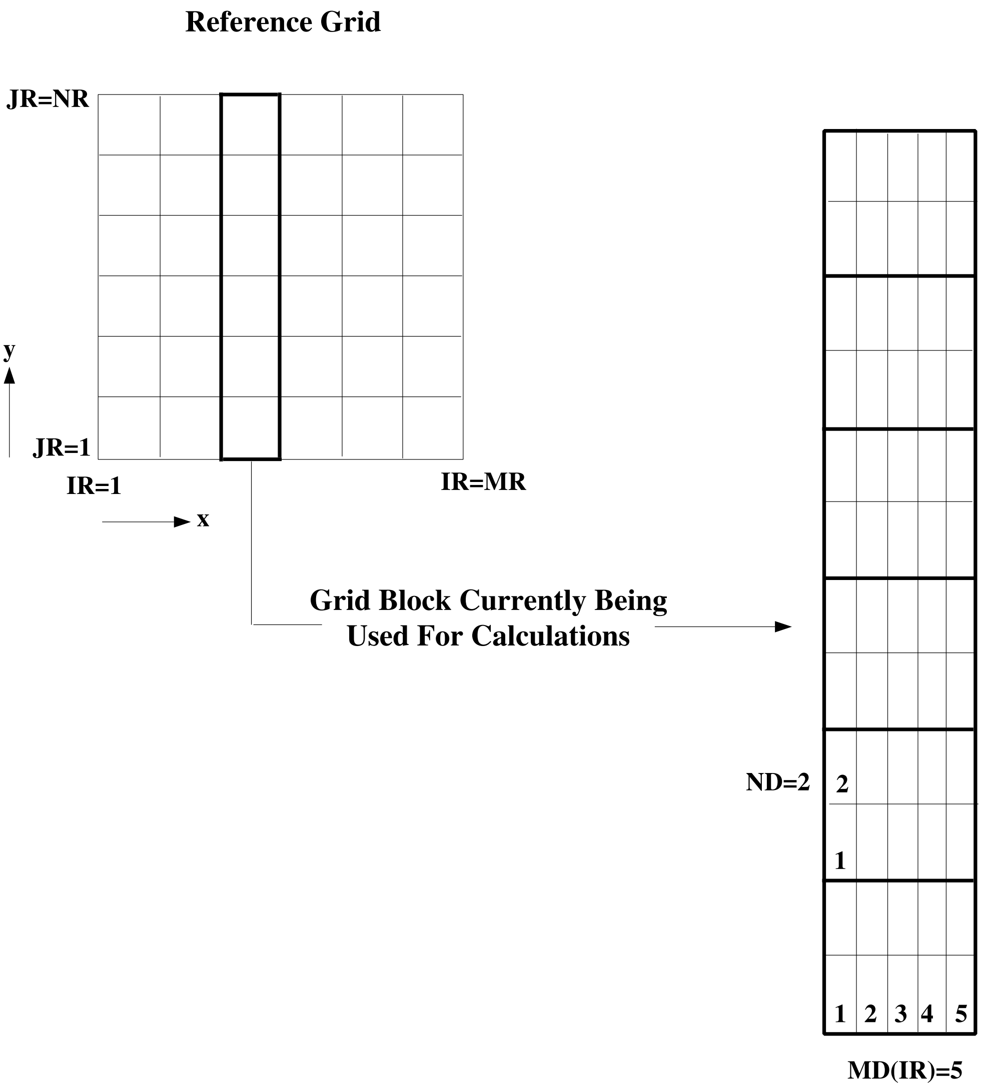

Figure 5: Sample grid subdivision

1. The user may let the program calculate *md*. The program proceeds by calculating an average wavenumber along the reference grid row, and uses this to estimate the wavelength in the grid block. The program then chooses a subdivision so that at least 5 grid points per wavelength of the shortest wave being modeled will occur in the grid block. The program checks to see that this desired number of subdivisions does not exceed the maximum. If it does, the program reduces the number to the maximum and prints a warning message indicating that the grid block can not be subdivided finely enough. Computed results in this case must be regarded as being suspect. The number of subdivisions used is indicated on the output. The user chooses this option by setting the switch *ispace*=0 on input.

2. The user may specify each value of *md*(*ir*) from *ir*=1 to (*mr*-1). This is done by setting the switch *ispace*=1 on input, and the values of *md* are then read in from the input data file. Note that this choice is necessary if the user-defined subgrids discussed below are to be used, since the user will be inputting subgrids with pre-specified spacings. As it is presently written, the program will only print a warning if it encounters subgrids with *ispace*=0 chosen; extensive garbling of input data may result.

After the subdivided grid block is established, the program uses this grid as the actual computation grid. New values of depth d and ambient current u and v are calculated at the extra grid points by fitting a twisted surface to the reference grid using linear interpolation. An example of the resulting bottom topography for a single grid cell is shown in Figure 6.

2.5.2 User-specified Subgrids

In some applications, an important topographic feature may be present at a subgrid scale within the reference grid. Examples include artificial islands, shoals, borrow pits, etc., which are superimposed on an otherwise slowly-varying topography which is represented by the sort of grid resolution appropriate to tidal models. An illustration of such a feature is shown in Figure 7, where a poorly resolved feature occupies portions of four reference grid cells. For cases such as this, the program includes the option for the user to input user-defined, subdivided grid cells in order to specify these features at the level of the computational grid.

The use of user-defined subgrids implies that the user will be choosing the grid spacing of the subgrids. Since grid spacings must be uniform across a grid block, the user should choose the option, *ispace*=1 and specify the values of *md* in the input file, *indat.dat*. Then, the program will look for a data array of the correct, pre-specified dimension when it reads

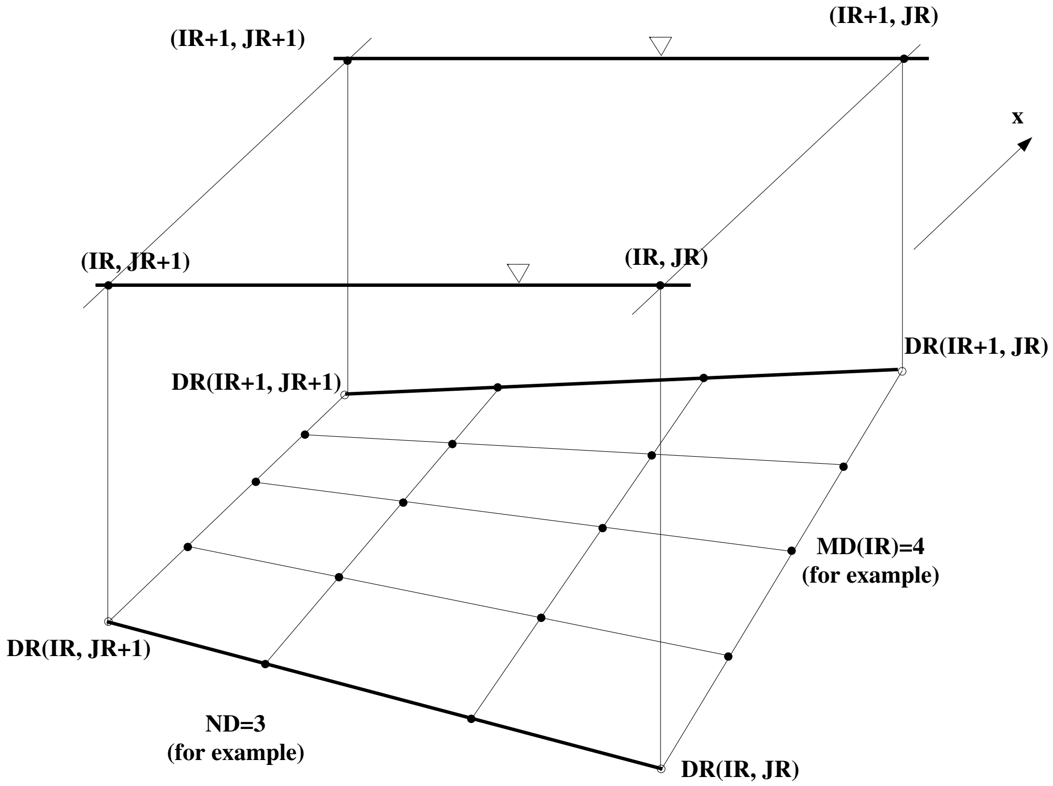

Figure 6: Interpolation of depth data

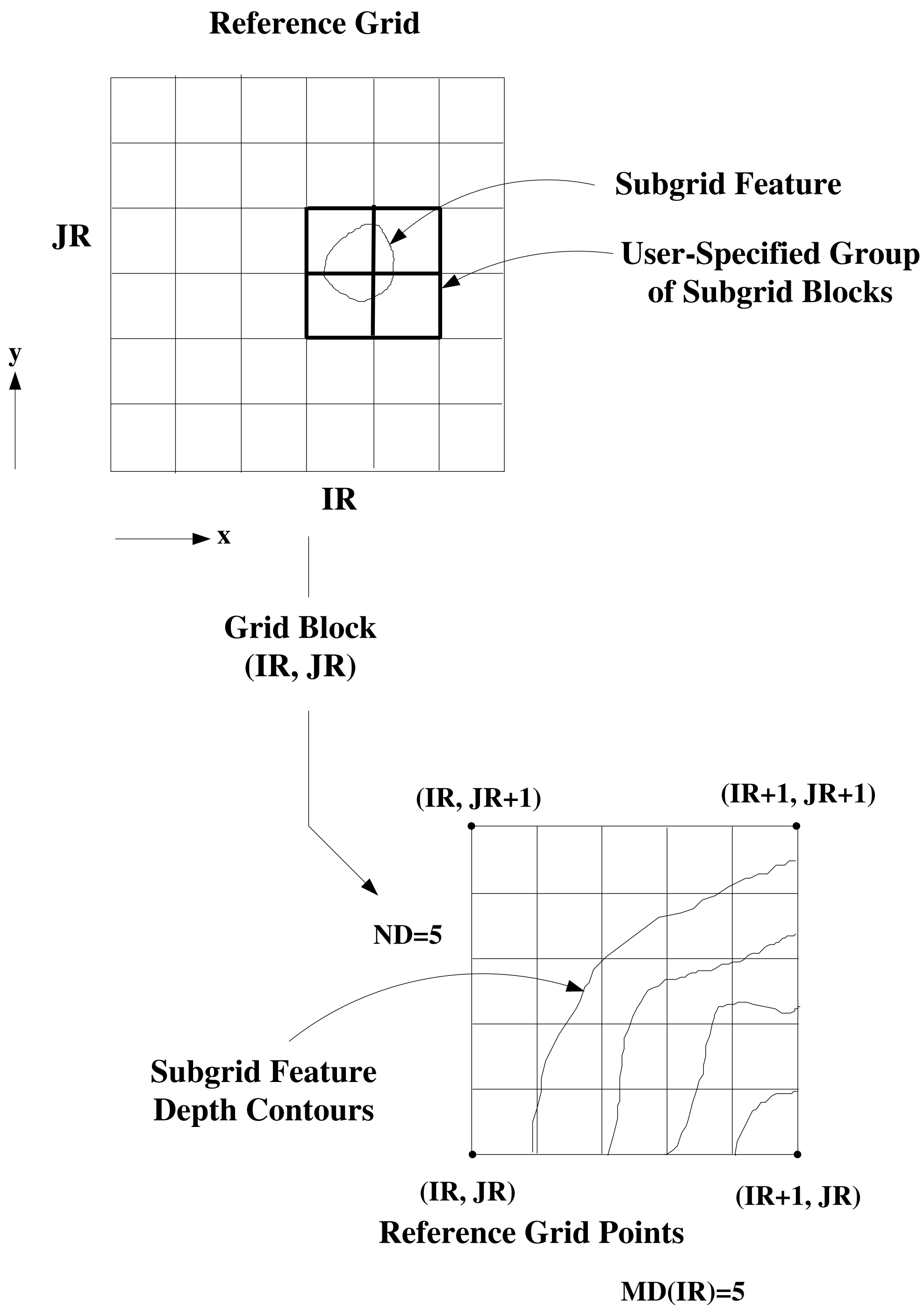

Figure 7: User-defined subgrids

the subgrid data from the file, *subdat.dat* (unit number $iun(2)$).

Several aspects of the subgrid data should be noted. Data for depth d and current velocities u and v need to be specified at each of the subgrid points. These data are input in the same units, and with the same datum for depth as the data for the reference grid. Also, note that the subgrid includes the points on its boundaries; for example, the single subgrid shown in Figure 7 has a dimension of 6 by 6.

The data values on the outer borders of the subgrids should be setup to match with the linearly interpolated values in the region external to the user-defined subgrid. If the data do not vary linearly along the subgrid boundaries, there may be some mismatch between the subgrid boundaries and the external region. An exception to this rule occurs when two subgrids adjoin each other. Then, care should be taken that the data along the common boundary match. These common boundary point data are duplicated in the input data, since each subgrid data set includes the boundaries.

Instructions and formats for creating the subgrid data file *subdat.dat* are included in section 2.8.

2.6 Program Input: Preprocessing a Spectral Input

In many cases, the spectral density at the offshore boundary might be available as data and the user might want to use this information as input to the model. As can be seen in the next section, the program **REF/DIF S** requires the input of discrete wave components that describe the desired two-dimensional spectrum. In order to use spectral data at the off-shore boundary, the user has to discretize the spectrum and input the individual wave components in the manner described in the next section.

Several approaches to the task of discretizing a two-dimensional spectrum can be used. In order to clarify the procedure to be followed, the preprocessor **SPECGEN** is included in Appendix C. This program is specificly designed to create input files for three of the example cases in Chapter 3, which simulate experiments by Vincent and Briggs (1989). **SPECGEN** discretizes a TMA frequency spectrum (Hughes, 1984) used in conjunction with a wrapped normal directional distribution (Borgman, 1985). A similar procedure has to be followed for the discretization of any other design or measured spectrum.

SPECGEN divides the two-dimensional spectrum into bins of constant volume, therefore the spectrum is well resolved in the region of the peak frequency and the mean angle of incidence. As the volume of each bin is constant, the energy, and hence the amplitude, corresponding to each wave component is constant. The midpoints of each bin in frequency and in direction are chosen as the representative values for frequency and direction for that bin.

The discretization of the two-dimensional spectrum performed by **SPECGEN** results in wave components of amplitude A with an associated frequency f and an angle of incidence θ with respect to the assumed propagation direction. **SPECGEN** outputs this information in the format of a *indat.dat* file. The format of this file is described in the next section.

In order to use any design or measured spectrum as input, the user has to discretize the spectrum and create discrete wave components of amplitude A with an associated frequency f and an angle of incidence θ. The method of discretization used is not relevant to **REF/DIF S**

Figure 8 shows plots of a divided TMA frequency spectrum and a divided wrapped normal directional spreading function. Discretization has been performed by **SPECGEN**.

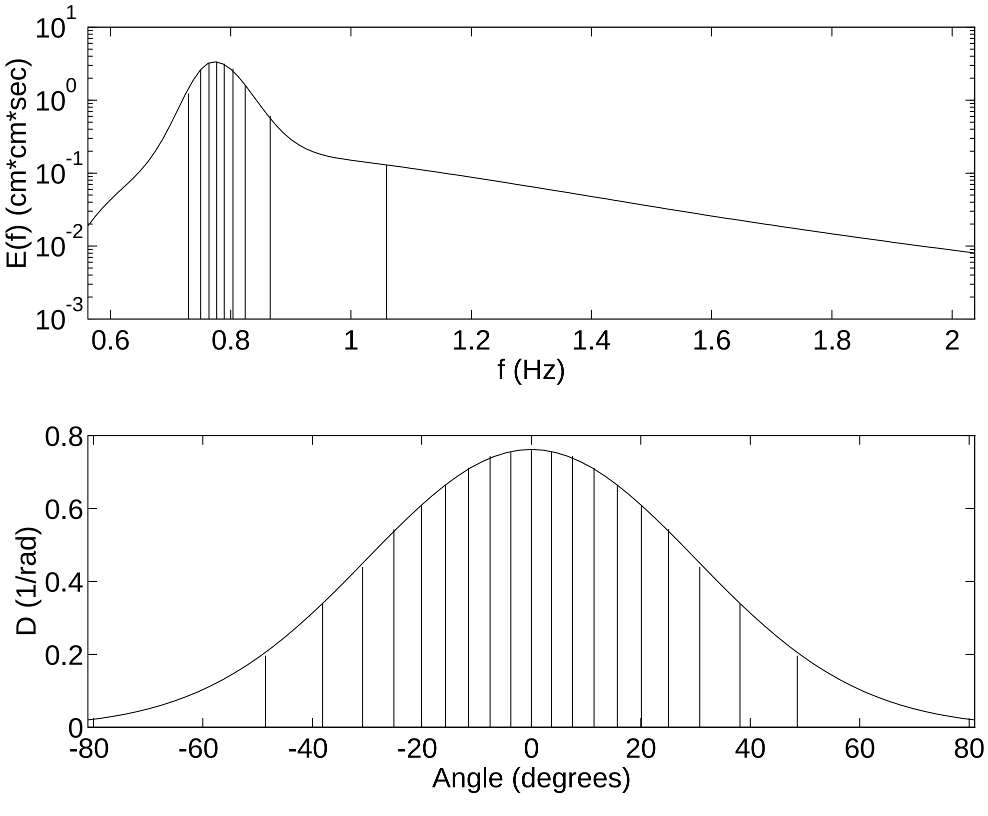

Figure 8: Divided Frequency Spectrum and Directional Spreading Function

2.7 Program Input: Model Control and Wave Data

This section discusses the structure of the input data file *indat.dat* read in through logical device number $iun(5)$. This file contains all the information needed to control the operations of the program, and all of the input wave data. The file *indat.dat* is kept fairly standard so that already existing data files for **REF/DIF 1** can be used with no changes if a monochromatic wave is described and with only minor changes if more than one wave component is described. Data files for **REF/DIF S** can be used with **REF/DIF 1** without any alterations. As the program structures are slightly different some switches for options existing in **REF/DIF 1** are set to default values. The preprocessor **SPECGEN** will set the proper values of those switches. In addition, the default values are stated here and reference is made to Section 2.1 for details.

Reference grid values of depth *dr* and currents *ur* and *vr* are treated in the following section.

Data is read from *iun*(5) in both the *inref* and *inwave* subroutines. Each line of input data is in free format. The input file is set up as follows:

1. *iun(1), iun(2), iun(3)*:

   Logical device numbers for reference grid data, user-specified subdivision data, and reference grid output. Note that the values 5, 22 or 36 should not be chosen for *iun(1), iun(2), iun(3)* as these are default values chosen by the program for output files.

2. *mr*, *nr* :

   Reference grid dimensions. Maximum values are *ixr*, *iyr* respectively.

3. *iu, ntype, icur, ibc*:

   iu is switch for physical units; *iu*=1, MKS; *iu*=2, English. (program defaults to *iu*=1 if an error is made on input).

   ntype is switch for nonlinearity; *ntype*=0, linear model; *ntype*=1, composite model; *ntype*=2, Stokes wave model.

   icur is switch for input current data. *icur*=0, no currents input; *icur*=1, currents input.

   ibc is the boundary condition switch. *ibc*=0, closed boundaries; *ibc*=1, open boundaries.

   Both *icur* and *ibc* default to zero on input error.

4. *dxr, dyr, dt* :

   Reference grid x-spacing and y-spacing (uniform over entire reference grid) and depth tolerance value.

5. *ispace, nd*:

   ispace is switch controlling subdivisions; *ispace*=0, program attempts its own x-

subdivisions. *ispace*=1, user specifies x-subdivisions.
nd is the number of y-direction subdivisions (needed in either case).

If *ispace*=1, the x-direction subdivisions are inserted here (one for each reference grid block):

$(md(ir),\ ir=1,\ mr\text{-}1)$

6. *iff(1), iff(2), iff(3)*:
   Dissipation switches; *iff(1)*=1;turn on turbulent boundary layer; *iff(2)*=1; turn on porous bottom damping; *iff(3)*=1; turn on laminar boundary layers. No damping if all values are zero.

7. *isp*:
   Switch for user-specified sub-grid specifications. *isp*=0, no subgrids to be read; *isp*=1, subgrids will be read.

8. *iinput, ioutput*:
   The switches *iinput* and *ioutput* control options that exist in **REF/DIF 1**, but that are not included in the structure of **REF/DIF S**. Nevertheless, this line of input is required for the user to be able to use this input file with **REF/DIF 1**, if desired. For more information, the reader is referred back to Section 2.1. For a proper run with **REF/DIF S** this line of the input file should be

   ```
   1 1
   ```

   Both *iinput* and *ioutput* default to unity on input error.

   As *iinput*=1, the rest of *indat.dat* is as follows:

9. *iwave, nfreqs*:
   iwave is the switch for wave field type. If *iwave*=1 the program expects discrete wave components to be input. Any other options are not incorporated into **REF/DIF S**. More information can be found in Section 2.1. Thus, set *iwave*=1. *iwave* defaults to unity on input error.
   nfreqs is the number of frequency components to be run.

   As *iwave*=1, the remainder of the file is as follows:

10. *freqs(ifreq), tide*:
    Wave period for each frequency component and tidal offset, which is a constant for all frequencies.

11. *nwavs*:

    Number of directional components per frequency, again a constant for all frequencies.

12. *amp(ifreq, iwavs)*, *dir(ifreq, iwavs)*:

    Amplitude (not height) and direction (in degrees) for each component wave.

    repeat items (11) and (12) *nwavs* times

    repeat entire block from item (10) to item (12) *nfreqs* times

    The following lines are needed in addition for the execution of **REF/DIF S** in cases where more than one wave component is desired.

13. *fpeak*:

    Peak frequency of input frequency spectrum in Hz.

14. *iopt*:

    iopt specifies whether computed amplitudes for any specific wave component are to be stored in a separate file *specamp.dat*. *iopt*=1 turns the option on, *iopt*=0 turns it off.

    If *iopt*=0, the data file terminates here. No additional information is necessary.
    If *iopt*=1 the following information is needed.

15. *ifcomp,idcomp*:

    This line specifies the wave component for which output will be produced. *ifcomp* specifies which frequency component is desired and *idcomp* specifies which directional component is desired.

Examples of data files are given in Chapter 3.

2.8 Program Input: Reference Grid and Subgrid Data

This section describes two files which provide arrays of data on the various grids.

Reference Grid Data File

This file (named *refdat.dat*) is accessed as logical device number *iun*(1). Its contents consist of the arrays of depth *dr*, x-velocity *ur*, and y-velocity *vr* at the reference grid points. This file is accessed only once per model run, and its entire contents are read in by subroutine *inref*. If *icur*=0, only the depth data *dr* need to be specified.

Data for this file should be written in the following format:

```
do 1 ir=1, mr
```

```
        write(unit,#) (dr(ir, jr), jr=1, nr)
1       continue

        (then, if icur=1)

        do 2 ir=1, mr
        write(unit,#) (ur(ir, jr), jr=1, nr)
2       continue

        do 3 ir=1, mr
        write(unit,#) (vr(ir, jr), jr=1, nr)
3       continue
#       format(16f8.4)
```

The data may be in either MKS or English units; set the units switch *iu* in the *iun*(5) data file *indat.dat* accordingly.

Subgrid Data File

This file (named *subdat.dat*) is accessed as logical device number *iun*(2). If no user-defined subgrids are to be read in, this file may be omitted. The file consists of two parts; an integer array of 1's and 0's indicating which reference grid cells are to be defined by the user, and then a sequence of groups of arrays of d, u and v, one group for each subgrid.

The integer array *isd* is dimensioned (*mr*-1) by (*nr*-1), with one point for each spacing in the reference grid. The array contains a 0 if that cell is not to be user-defined, and a one if it is. For example, the (*mr*,*nr*)=(7,6) reference grid shown in Figure 7 has four cells which are to be read in as user-defined subgrids.

The array *isd* should be written in the data file first, using the format:

```
        do 1 ir=1,mr-1
        write(unit,#)(isd(ir,jr),jr=1,nr-1)
1       continue
#       format(15I4)
```

Now, a group of arrays of d, u and v must be entered into the data file, with one group corresponding to each value of 1 in *isd*. The program accesses the data file by x-row (*ir*) and then by y-column (*jr*). For the above example, the subgrid array groups should thus be stored in the order of the coordinate pairs (4,3), (4,4), (5,3), (5,4). The dimensions of each subgrid are given by m=md(*ir*)+1 and ns=nd+1. The borders of adjacent subgrids share the same common boundary points.

Each group of subgrid data should be written using a format similar to:

```
        write(unit,#)((d(i,j),j=1,ns),i=1,m)

        (then, if icur=1)

        write(unit,#)((u(i,j),j=1,ns),i=1,m)
        write(unit,#)((v(i,j),j=1,ns),i=1,m)

#       format(16f8.4)
```

using the appropriate value of m for the grid block in question. Data may be in MKS or English units, depending on the value of iu in the input data file. The integer array *isd* is read by *inref*, and the individual subgrids are read by *grid*.

2.9 Program Output: Screen Output

This section discusses the output sent to the screen (or the chosen output device with logical unit number $iun(4)$). Screen output consists of three types: title and header information which is printed in order to identify the run and the operating parameters, run-time error messages (either warnings or terminal error messages), and the results of calculations.

Header Information

At the start of each run, a set of two title pages are printed. Page 1 identifies the program and then prints out messages identifying the parameters which set up the model run, as read in by subroutine *inref*. A sample title page 1 is shown in Figure 9.

Title page 2 gives the input wave conditions which were read in by *inwave*. Part of a sample title page 2 is given in Figure 10.

Dimensional quantities are printed on the title page in the units used for input. Title page 1 gives an indication whether quantities are in MKS or English units.

Run-time Error Messages

REF/DIF S performs some data checking and checking of calculations during a run. This checking may result in warnings or terminal errors which are beyond calculation errors which would lead to standard FORTRAN error messages. A list of possible errors and the resulting messages follow.

1. Reference grid dimensions were specified as being too large on input. $mr > ixr$ and/or $nr > iyr$.

```
                    Refraction-Diffraction Model for
                    Spectral Wave Conditions


                    REF/DIF~S  Version 1.1


                    Center for Applied Coastal Research
                    Department of Civil Engineering
                    University of Delaware
                    Newark, Delaware 19716


     James T. Kirby and H.Tuba Ozkan, July 1992, February 1994
0


                    input section, reference grid values



 reference grid dimensions  mr=251
                            nr=275


 reference grid spacings   dxr=  0.1000
                           dyr=  0.1000

 physical unit switch iu=1,  input in mks units

 icur=0, no current values read from input files

 ibc=1, open lateral boundaries

 ispace =0 chosen, program will attempt its own reference grid subdivisions

 y-direction subdivision according to nd=  1

 ntype = 1, stokes model matched to hedges model


   switches for dissipation terms

 0   turbulent boundary layer
 0   porous bottom
 0   laminar boundary layer

 isp=0, no user defined subgrids
iinput = 1, program specifies initial row of a
```

Figure 9: Sample title page 1

```
                    input section, wave data values



iwave=1, discrete wave amps and directions


the model is to be run for 10 separate frequency components
```

Figure 10: Sample title page 2

Message: dimensions for reference grid too large; stopping.

Result: Program stops.

Error occurs in: *inref*

Action: Change the values of ixr and iyr in the **parameter** statements, such that $ixr \geq mr$ and $iyr \geq nr$.

2. User specifies a y-direction subdivision nd which will cause the number of y grid points n to exceed the maximum iy.

   Message: y-direction subdivision nd="*nd*" too fine. maximum number of y grid points will be exceeded. execution terminating.

   Result: Program stops.

   Error occurs in: *inref*

   Action: Change the values of iy in the **parameter** statements, such that $iy \geq (nr - 1)nd + 1$. Also, see Section 2.5.1.

3. User specifies an x-direction subdivision on one of the grid blocks ir which exceeds the maximum amount (ix-1). As a result, the dimension of the subdivided grid will be too large.

   Message: x-direction subdivision md="*md*" too fine on grid block "*ir*", execution terminating.

   Result: program stops

   Error occurs in: *inref*

Action: Change the values of *ix* in the **parameter** statements, such that $ix \geq$ (*md(ir)*+1) at the grid block *ir* indicated in the error message. At this point the user should check if $ix \geq (md(ir)_{max}+1)$, where $md(ir)_{max}$ is the maximum number of x-direction subdivision on the grid blocks, to prevent reoccurance of this error. See Section 2.5.1.

4. A depth value occurs in the reference grid which differs from the average of its neighbors by more than the tolerance value *dt* specified on input. This is basically a data checking feature. Printed values are in meters.

   Message: Depth "*dr*" (m) at reference grid location "*ir,jr*" differs from the average of its neighbors by more than "*dt*" (m). Execution continuing.

   Result: None by program.

   Error occurs in: *inref*

   Action: Data in file *refdat.dat* should be checked for mistakes and corrected if wrong.

5. An ambient current value occurs which implies that the flow would be supercritical at the given location. This serves as both a check for anomalously large current values, and an indicator of possible subsequent computational problems.

   Message: ambient current at reference grid location "*ir,jr*" is supercritical with *froude number* = "*froude number*", execution continuing

   Result: None by program.

   Error occurs in: *inref*

   Action: Data in file *refdat.dat* should be checked for mistakes and corrected if wrong.

6. If the user specifies that predetermined subgrids are to be read in, while at the same time telling the program to performs its own subdivisions, the computed dimensions of the subgrid may be different than those of the subgrid included in the input. Runs requiring user-specified subgrids should choose the *ispace*=1 option. If an incompatible set of dimensions occurs, the program will either garble the input array or run out of data.

   Message: Warning: input specifies that user will be supplying specified subgrids (isp=1), while program has been told to generate its own subgrid spacings (ispace=0). possible incompatibility in any or all subgrid blocks.

   Result: None by program.

   Error occurs in: *inref*

   Action: Should restart unit with correct *ispace*, *isp* values. See Section 2.5.1.

7. User specifies number of frequency components which exceeds the maximum amount *icomp*.

   Message: Too many frequency components; stopping.

   Result: Program stops.

   Error occurs in: *inwave*

   Action: Change the value of *icomp* in the **parameter** statements, such that $icomp \geq nfreqs$. The value of *nnii* in the **parameter** statements has to be changed, as well, such that $nnii \geq (icomp)(nnd)$.

8. User specifies number of directional components which exceeds the maximum amount *nnd*.

   Message: Too many directional components; stopping.

   Result: Program stops.

   Error occurs in: *inwave*

   Action: Change the value of *nnd* in the **parameter** statements, such that $nnd \geq nwavs$. The value of *nnii* in the **parameter** statements has to be changed, as well, such that $nnii \geq (icomp)(nnd)$.

9. While calculating its own subdivision spacings, the model may try to put more division in a reference grid block than is allowed by dimension *ix*. If this occurs, the program uses the maximum number of subdivisions allowed (*ix*-1), but prints a message indicating that the reference grid spacing is too large with respect to the waves being calculated. This problem may be circumvented by increasing the size of *ix* in **parameter** statements.

   Message: model tried to put more spaces (md="*md*") than allowed in grid block "*ir*"

   Result: Program performs fixup and continues. Model resolution and accuracy may be poor.

   Error occurs in: *grid*

   Action: A finer reference grid should be used or the value of *ix* in the **parameter** statements should be increased, such that $ix \geq (md(ir)+1)$. Also, see Section 2.5.1.

10. While using the Stokes wave form of the model, *ntype*=2 the model may encounter large values of the Ursell number, indicating that the water is too shallow for that model to be appropriate. The cutoff point recognized by the program is $(A/h)/(kh)^2 = 0.5$.

Message: Warning: Ursell number = "*U*" encountered at grid location "i,j" during calculation of wave component "*ii*". Should be using Stokes-Hedges model (*ntype* = 1) due to shallow water.

Result: None by program.

Error occurs in: *fdcalc*

Action: The program should be terminated and re-run with the composite nonlinear model.

11. The Newton-Raphson iteration for wavenumber k may not converge on the point (1,1) of the computatinal grid in the specified number of steps. This may occur for waves on strong opposing currents.

    Message: Wavenumber failed to converge for frequency component "*ifreq*" on point (1,1).

    K = last iterated value of wavenumber

    D = depth

    T = period calculated from last iterated value of k

    U = x-direction velocity

    F = value of objective function (should be =0 for convergence)

    Result: Program continues with last iterated value of k. Computed results are of questionable accuracy.

    Error occurs in: *wnum*

    Action: Check if the test case suits the assumptions of the program. Also, an error in the input files might be possible cause.

12. A similar error as in (9) can occur at any other grid location.

    Message: Wavenumber failed to converge for frequency component "*ifreq*" on column "*j*".

    Result: Program continues with last iterated value of k. Computed results are of questionable accuracy.

    Error occurs in: *vwnum*

    Action: Check if the test case suits the assumptions of the program. Also, an error in the input files might be possible cause.

Results of Calculations

As part of the screen output, the model first prints the x-position for each reference grid row. Also, for each grid row, the values of significant wave height are printed for each reference grid location in 12(f10.4) format using the physical units (MKS or English) used for input.

2.10 Program Output: Stored Output

This section discusses the output stored in two disk files *outdat.dat* and *outrad.dat* (logical unit numbers *iun*(3) and 22, respectively) with a resolution of the reference grid. Also the optional output file *specamp.dat* (logical unit number 36) will be discussed. All data is stored using free format. The data file *outdat.dat* has been used to generate the plots given in Chapter 3.

Output stored in *outdat.dat*

Information about the depth grid and the significant wave heights computed at each reference grid point are stored in the file *outdat.dat.* The program first stores the values of *nr* and *mr*, and the values of the reference grid y-locations.

1. *nr*, *mr*
2. *yr(jr), jr=1,* nr

   Then, for each x-position *ir* in the reference grid, the program stores the following information.

3. *xr*(*ir*)
4. *d*(*ir,jr*), *jr*=1, *nr*
5. *h13*(*ir,jr*), *jr*=1, *nr*

The set of output (3–5) is repeated for each x-position *ir.* The data for depth *d* include the effect of the tidal offset (input as *tide*) and also will show the thin film where exposed, dry land has been replaced by a small water depth. *h13* is the significant wave height at any reference grid location. The data are stored using the units (MKS or English) used for input.

Output stored in *outrad.dat*

Information about radiation stresses is stored in the file *outrad.dat*. The program first stores the dimensions of the reference grid *nr* and *mr*.

1. *nr*, *mr*

   Then for each x-position in the reference grid, the following information is stored.

2. S_{xx}(*ir,jr*), *jr*=1, *nr*

3. S_{xy}(*ir,jr*), *jr*=1, *nr*

4. S_{yy}(*ir,jr*), *jr*=1, *nr*

(2–4) is repeated for each row *ir*. Here S_{xx} is the radiation stress in the x-direction due to the momentum flux in the x-direction, S_{xy} is the radiation stress in the y-direction due to the momentum flux in the x-direction and S_{yy} is the radiation stress in the y-direction due to the momentum flux in the y-direction.

Output stored in *specamp.dat*

Information about the reference grid and the complex amplitudes of any specified wave component can be stored in the file *specamp.dat* when the switch *iopt* is set to unity and the desired component is specified in the input. This option provides the user with the opportunity to examine the behavior of any wave component in more detail. The file *specamp.dat* is set up similar to *outdat.dat* with the only difference that complex amplitudes are stored instead of the significant wave height at all reference grid points. The structure of *specamp.dat* is as follows.

1. *nr*, *mr*

2. *yr(jr), jr=1,* nr

   Then, for each x-position *ir* in the reference grid, the program stores the following information.

3. *xr*(*ir*), *psibar(ir)*

4. *d*(*ir,jr*), *jr*=1, *nr*

5. *a*(*ir,jr*), *jr*=1, *nr*

The set of output (3–5) is repeated for each x-position *ir*. Here, *psibar(ir)* is the value of reference phase function with no random phase included and *a* is the complex amplitude of the chosen wave component.

A listing of the program code is given in Appendix A. Also, an example of a plotting program using data stored in *outdat.dat* is included in Appendix B to provide guidance on reading the stored output file. An *fweb* documentation of the preprocessor **SPECGEN** is given in Appendix C.

3 EXAMPLE CALCULATIONS

This chapter presents calculations performed using the spectral combined refraction-diffraction model **REF/DIF S**.

Each section of this chapter describes in full the model's application to a specific problem. Following a description of the problem, defining parameters needed to create the input files are stated. Only part of the input file *indat.dat* will be presented for each example case. The rather long part where the user inputs periods and tides for each frequency as well as amplitudes and directions of all wave components for that frequency is not presented. Thus the wave parameters are omitted but the first and last parts of the data file are presented. The omitted lines correspond to items (10–12) described in Section 2.7 of this manual. Reference will be made to preprocessing programs that were used to create the input data.

The data files are then used to run the program **REF/DIF S** with no job-specific modifications to the program involved. Program output is then presented in such a way as to adequately indicate the results and compared to data at hand.

The output for the various examples has been presented using some plotting programs which are external to the main body of the supplied program **REF/DIF S**. A specialized program has been included in Appendix B in order to provide some guidance in reading the data files generated by the main program. However, plotting routines are likely to vary from one computer system to another. The extra programs are therefore likely to be extensively machine-specific to the systems on which the computations were performed.

Four example cases will be presented. Results of all four examples will be compared to available laboratory data. The results of the first example case will be compared to experiments described by Mase and Kirby (1992). The rest of the examples will be numerical reproductions of experimental test cases chosen out of a series of tests performed by Vincent and Briggs (1989).

Only two separate depth grids have to be constructed for the simulations of these four example cases. The first domain is a channel with a minimal width, a short region of constant depth and a beach with a $1:20$ slope, corresponding to the setup used in the experiments described by Mase and Kirby (1992). The second is a two-dimensional domain with an elliptical shoal over an otherwise constant depth bottom used by Vincent and Briggs during their experiments. Both depth grids may be generated using the preprocessor **DATGEN**. Depth grids are stored in the file *refdat.dat*.

Section 3.1 provides a simulation of an experiment performed by Mase and Kirby (1992) with waves shoaling and breaking on a plane beach. This experiment only involves a unidirectional frequency spectrum. Section 3.2 presents simulations of multidirectional waves over a submerged shoal. The example cases in this section involve breaking and non-breaking waves, as well as broad and narrow directional distributions. Simulations of examples involving wide directional distributions make extensive use of the wide angle capabilities of **REF/DIF S**.

3.1 Unidirectional Waves Shoaling on a Plane Beach

3.1.1 Experimental Domain

Mase and Kirby (1992) conducted wave flume experiments using a Pierson-Moskowitz spectrum without directional spreading. The waves were generated in constant depth of 47 cm and shoaled and dissipated on a 1 : 20 beach. Wave gages were placed at several locations on the beach. Figure 11 shows the setup of the experiment. The depth grid for this case can be created using **DATGEN** using the information about the domain in this section. In the simulations of this experiment a constant depth region of 2 m length was used.

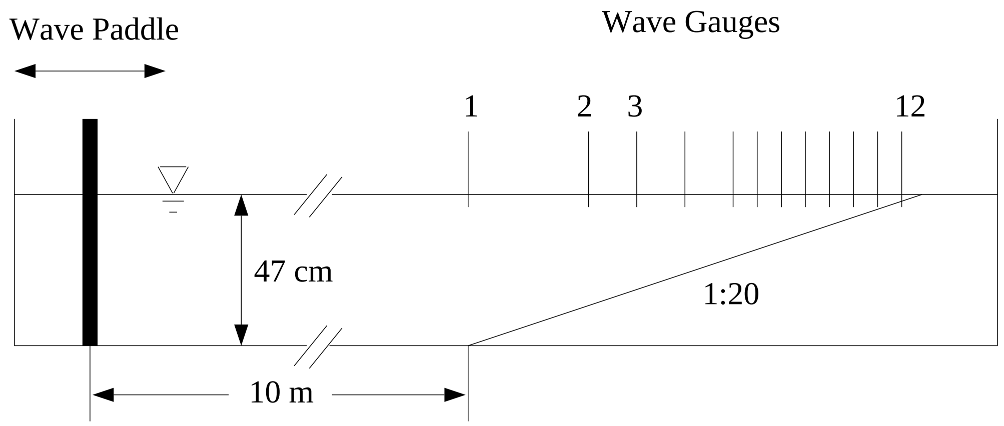

Figure 11: Experimental Setup (from Mase and Kirby (1992))

The comparisons between this data set and the model will be made using Case 1 in Mase and Kirby (1992). The available data are in terms of time series at the wave gage locations. The time series are used to compute the significant wave height for comparison with model results.

3.1.2 Model Comparison

Measured data at the offshore gage is used to create a smoothed incident frequency spectrum for the model input. The incident spectrum is divided into 50 discrete wave components

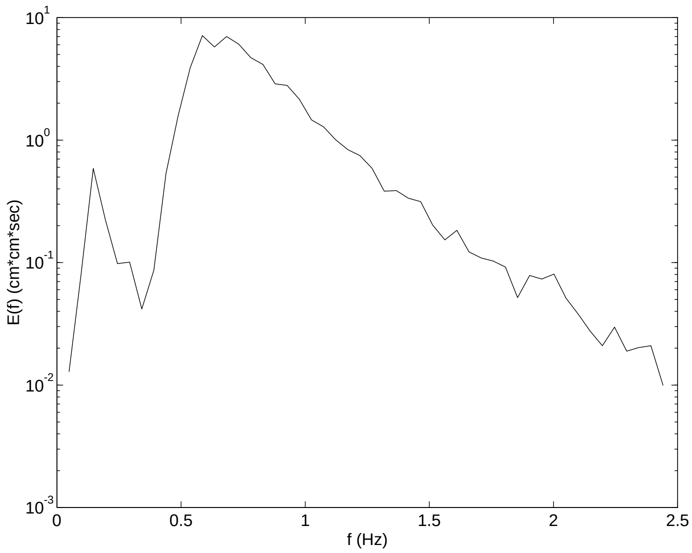

Figure 12: Incident Frequency Spectrum for Unidirectional Waves on a Beach

with constant bin width Δf. Figure 12 shows the incident frequency spectrum. Thus, the measured spectrum is used in the input rather than the target spectrum. The resulting *indat.dat* file is provided to the user with this manual.

The reference grid consists of 228 $\times$ 5 grid points with dx = 0.05 m and dy = 0.1 m. A depth tolerance value of *dt*=0.05 m was used. Also, MKS units were used (*iu*=1) and the linear model was applied (*ntype*=1). No currents (*icur*=0) or tides (*tide*=0) were considered. The program was run using closed lateral boundaries (*ibc*=0) and no subdivisions were specified by the user (*ispace*=0). Also no damping (*iff(1)=iff(2)=iff(3)*=0) and no user-specified subgrids (*isp*=0) were considered. The option to examine one of the wave components more closely was turned off (*iopt*=0).

The input data file *indat.dat* is partially given below. Note that the first seven lines are identical with the first seven lines of the input file *specgen.dat* for the preprocessor **SPECGEN**, which is documented in Appendix C. Also, see Section 2.6.

```
1 2 3
228 5
1 0 0 0
0.05 0.1 0.05
0 1
0 0 0
0
1 1
  1  50
```

The omitted part mentioned above is inserted here. The last two lines of the file *indat.dat* are as follows.

```
0.768317
0
```

Results for this case are presented in Figure 13. It can be seen that the model shoals the waves up to the breaking point. The location of the breaking point and the wave height at that location agree well with the data. The wave height decay after breaking starts is also very well predicted. The data points close to the shore show the presence of setup, which the model cannot predict.

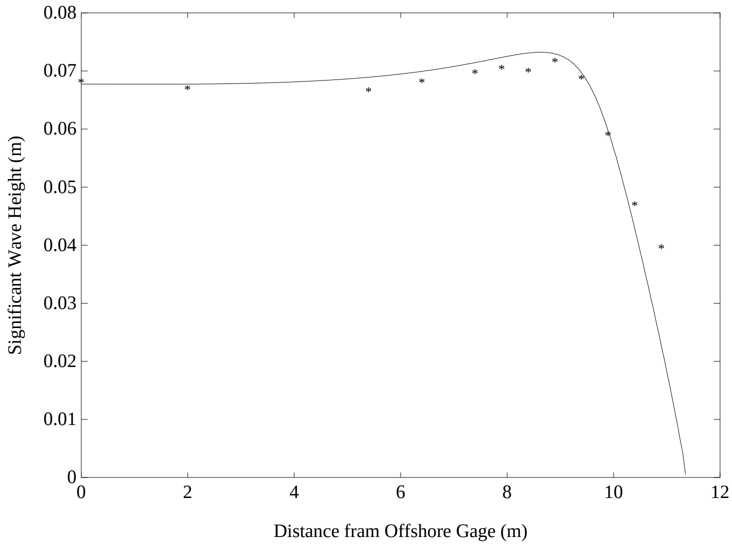

Figure 13: Waves Shoaling on a Beach: Significant Wave Height for Data (*) and Model (—–)

3.2 Multidirectional Waves over a Submerged Shoal

3.2.1 Experimental Domain

Vincent and Briggs (1989) conducted their experiments in a wave tank that was 35 m (114 ft) wide and 29 m (96 ft) long. The waves were generated by the directional spectral wave generator, which was 27.43 m (90 ft) long. The center of an elliptical shoal was located at $x = 6.10$ m and $y = 13.72$ m. The elliptical shoal had a major radius of 3.96 m, a minor radius of 3.05 m and a maximum height of 30.48 cm at the center. Maximum water depth was 47 cm. Expressions for the shoal perimeter and the elevation of each point on the shoal are given by Vincent and Briggs (1989). Reference should be made to that document for further information about the domain.

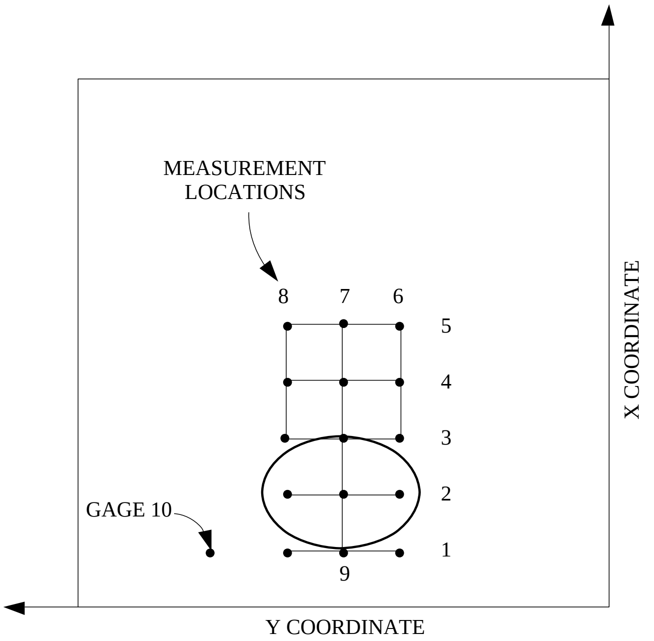

Figure 14: Experimental Setup (from Vincent and Briggs (1989))

Figure 14 shows the experimental setup in the basin and the measurement transects 1–9. Data collection for the cases of interest to this study were performed only along Transect 4.

In all the cases a TMA spectrum (Hughes, 1984) in conjunction with a wrapped normal directional spreading function (Borgman, 1985) was used to establish the target spectrum. The TMA spectrum is given by the energy density $E(f)$ for frequency f

$$E(f) = \frac{\alpha g^2}{(2\pi)^4 f^5} \exp\left\{-1.25\left(\frac{f_m}{f}\right)^4 + (\ln\gamma)\exp\left[\frac{-(f-f_m)^2}{2\sigma^2 f_m^2}\right]\right\}\phi(f,h) \tag{38}$$

where α is the Philips' constant, f_m is the peak frequency and γ is the peak enhancement factor. Furthermore, σ is the shape parameter defined by

$$\sigma = \begin{cases} \sigma_a = 0.07 & \text{if } f < f_m \\ \sigma_b = 0.09 & \text{if } f \geq f_m \end{cases} \tag{39}$$

The factor $\phi(f,h)$ incorporates the effect of the depth h and is computed following Hughes (1984) by

$$\phi = \begin{cases} 0.5\,(\omega_h)^2 & \text{if } \omega_h < 1 \\ 1 - 0.5\,(2-\omega_h)^2 & \text{if } 1 \leq \omega_h \leq 2 \\ 1 & \text{if } \omega_h > 2 \end{cases} \tag{40}$$

where

$$\omega_h = 2\pi f\sqrt{\frac{h}{g}} \tag{41}$$

Narrow or broad frequency spectra used in the experiment can be obtained by assigning the values 20 and 2 to the parameter γ, respectively. Representative plots of the narrow and broad frequency spectra are given in Figure 15.

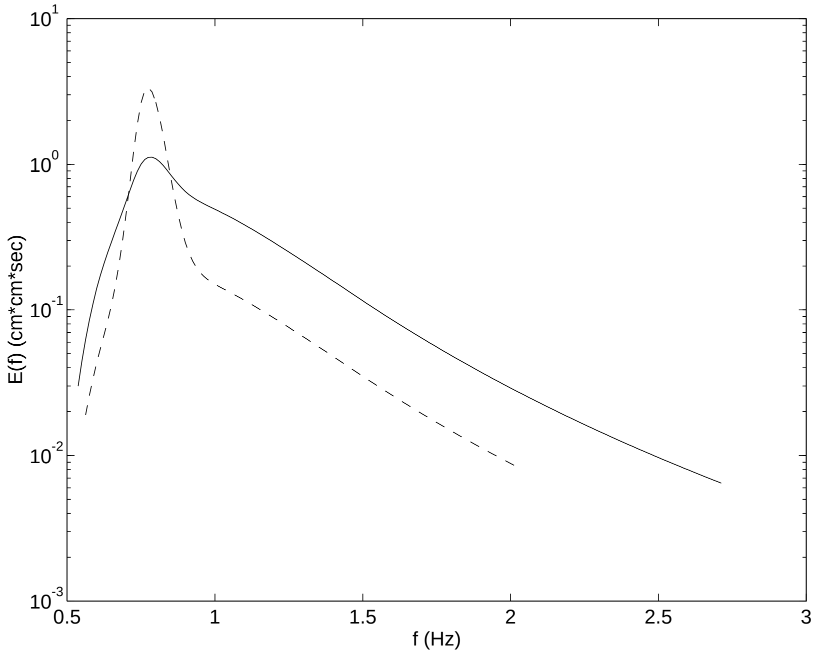

Figure 15: Narrow (– – –) and Broad (——) Frequency Spectra with $H_{1/3} = 0.0254$

The directional spreading function (Borgman, 1985) is given by

$$D(\theta) = \frac{1}{2\pi} + \frac{1}{\pi}\sum_{j=1}^{J} \exp\left[-\frac{(j\sigma_m)^2}{2}\right] \cos j\,(\theta - \theta_m) \tag{42}$$

where

θ_m = mean wave direction = 0°.

J = number of terms in the series (chosen to be 20 in the numerical calculations).

The parameter σ_m is chosen to be 10° or 30° giving a narrow or broad directional spreading, respectively. These spreading characteristics were used in the experiment. Representative plots of the narrow and broad directional spreading functions are given in Figure 16.

The two-dimensional spectrum is then given by the product

$$S(f, \theta) = E(f)\, D(\theta) \tag{43}$$

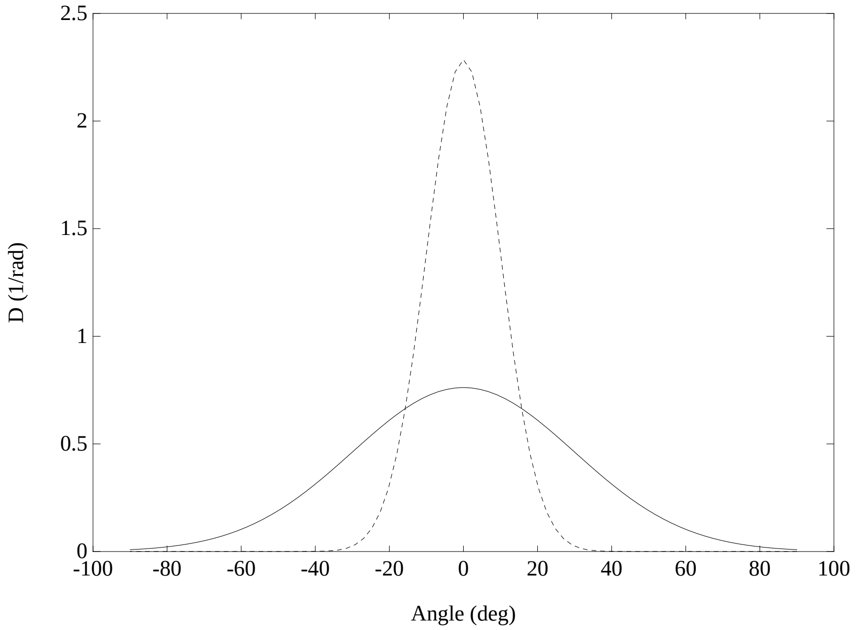

Figure 16: Narrow (– – –) and Broad (——) Directional Spreading Functions

The two-dimensional spectrum created by these two functions has to be divided up into discrete components to create input for **REF/DIF S**, which can be performed using **SPECGEN**. Section 2.6 and Appendix C give more information about this preprocessor.

Also, the reference grid was established and stored in *refdat.dat* using the preprocessor **DATGEN**. No additional information is necessary for this process.

3.2.2 Model Comparison

Three cases out of this data set will be used for comparisons with the model. The first two cases do not involve breaking waves, whereas the last case involves extensive breaking of the waves over the submerged shoal. Table 1 summarizes the wave parameters for these cases.

In order to recreate the results for these three example cases, the user has to create a *specgen.dat* input file for the preprocessor **SPECGEN**. The structure of this file is explained in Appendix C. The necessary information about the test cases can be found in this section as well as the previous section. The preprocessor **SPECGEN** will create an *indat.dat* file, that can be used to run **REF/DIF S**. The *indat.dat* files for the examples are included below, reference can be made to those listings.

The bottom bathymetry input file *refdat.dat* can be created using **DATGEN**. Note that

Table 1: Test Conditions for Waves over a Submerged Shoal (Vincent and Briggs, 1989)

Test Type	*Case ID*	T_p (sec)	$H_{1/3}$ (cm)	α	γ	σ_m $(\,^\circ)$
Non-breaking	*N*4	1.3	2.54	0.00047	20	10
	*B*4	1.3	2.54	0.00047	20	30
Breaking	*B*5	1.3	19.00	0.08650	2	30

DATGEN will give the user the option to create an *indat.dat* manually. The user must skip that option as the necessary *indat.dat* is created by **SPECGEN** and would otherwise be overwritten.

Non-Breaking Waves

These cases donot involve breaking waves and are simulated to show that the breaking term that is always in effect does not induce an unrealistic wave height reduction.

Case N4

The first case is denoted as Case N4 by Vincent and Briggs (1989) and involves a narrow frequency spectrum used in conjunction with a narrow directional spreading. Distinct values are given to the free parameters to define the spectrum.

$$
\begin{aligned}
H_{1/3} &= 0.0254m \\
f_m &= \frac{1}{1.3sec} = 0.769231Hz \\
f_{max} &= 5Hz \\
\gamma &= 20 \\
\theta_m &= 0^\circ \\
\sigma_m &= 10^\circ
\end{aligned}
$$

SPECGEN was used to generate the input data file *indat.dat*. The spectrum was divided into 10 frequency components and 20 directional components, thus a total of 200 waves were present.

The model was run using grid dimensions 251 $\times$ 275 with a resolution of $dx = dy = 0.1$ m. A depth tolerance value of dt=0.05 m was used. MKS units were used (iu=1) and the Stokes-to-Hedges nonlinear model was applied to test the behaviour of the modified nonlinear terms in the absense of breaking waves (*ntype*=1). No currents (*icur*=0) or tides (*tide*=0) were considered. The program was run using partially transmitting boundaries (*ibc*=1) and no subdivisions were specified by the user (*ispace*=0). Also no damping (*iff(1)*=*iff(2)*=*iff(3)*=0) and no user-specified subgrids (*isp*=0) were considered. The option to examine one of the wave components more closely was turned off (*iopt*=0).

Part of the input file *indat.dat* is given below. Note that the first seven lines are identical with the first seven lines of the input file *specgen.dat* for the preprocessor **SPECGEN**, which is documented in Appendix C. Also, see Section 2.6.

```
1  2  3
251  275
1  1  0  1
  1.00000E-01    1.00000E-01    5.00000E-02
0  1
0  0  0
0
1  1
1  10
```

The omitted part consisting of the amplitudes and directions of the individual components is inserted here. The last two lines of the file *indat.dat* are as follows.

```
 0.769231
0
```

The output is presented in two ways. Figure 17 shows a contour plot of the normalized significant wave height. Figure 18 shows a transect at x = 12.2 m and the comparison to available data at that transect. It can be seen that the general shape of the curve in Figure 18 is well modeled. The fact that the peak of the wave height for this non-breaking wave case is well represented shows that the coefficient of the breaking term stays small and does not dissipate significant energy from the system.

Case B4

In case B4 a spectrum with narrow frequency and broad directional spreading is generated. The free parameters have the following values.

$$
\begin{aligned}
H_{1/3} &= 0.0254m \\
f_m &= \frac{1}{1.3sec} = 0.769231Hz \\
f_{max} &= 5Hz \\
\gamma &= 20
\end{aligned}
$$

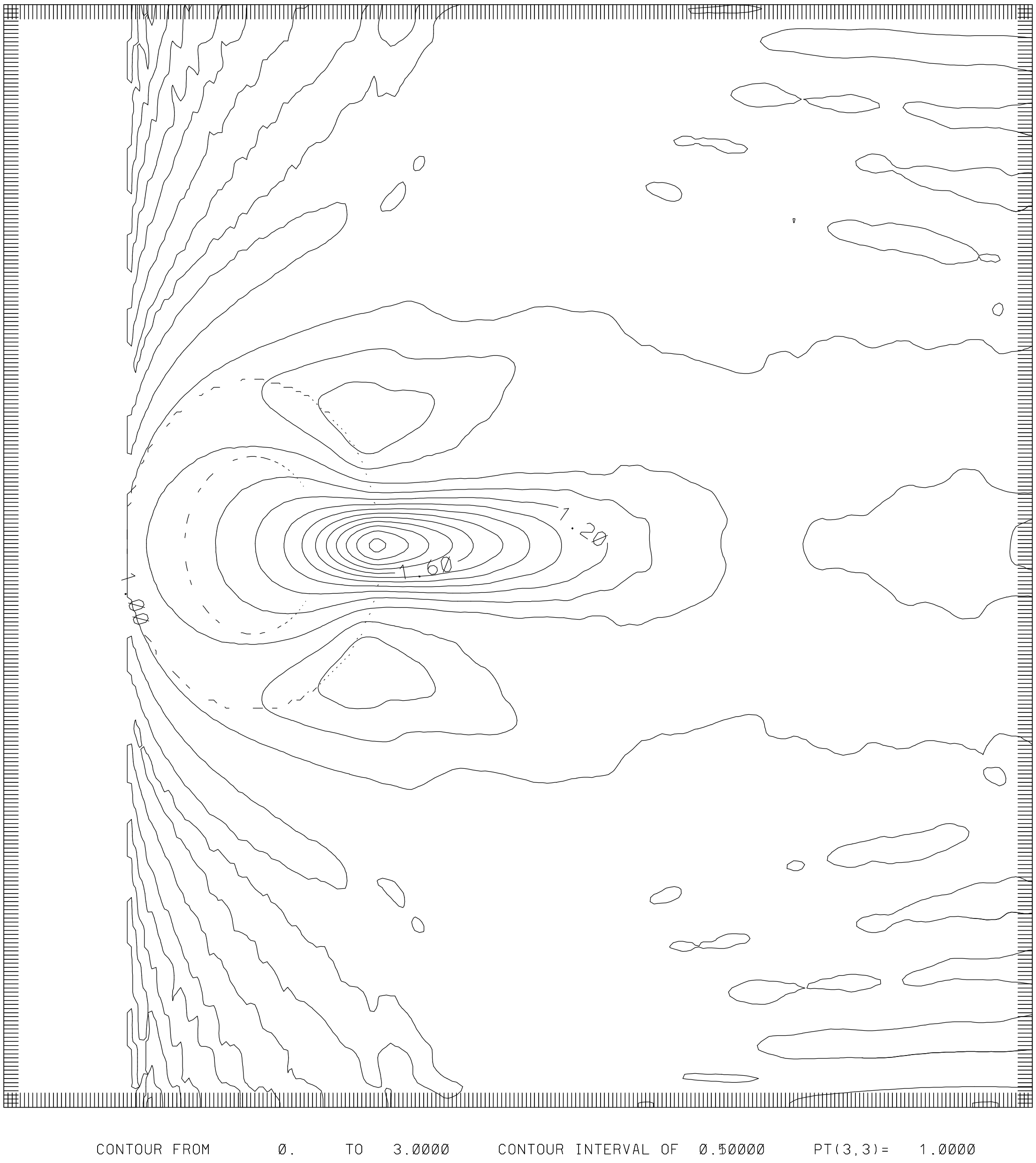

Figure 17: Case N4: Contour Plot of Normalized Significant Wave Height – Model Results

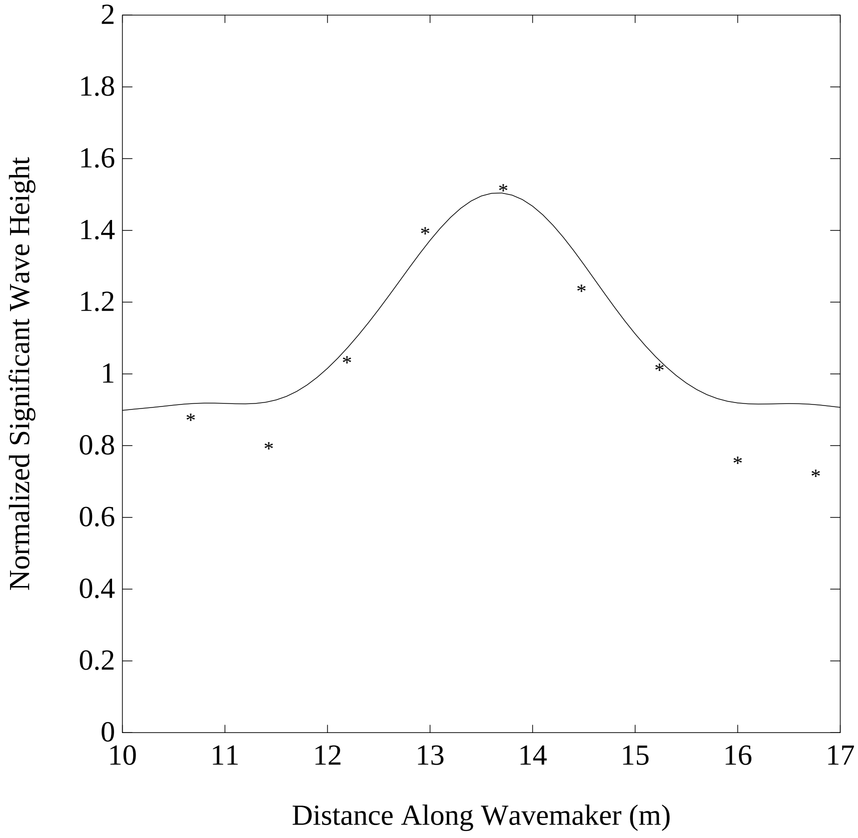

Figure 18: Case N4: Normalized Significant Wave Height for Data (*) and Model Results (—–) at Transect at $x = 12.2$ m

$$\theta_m = 0°$$
$$\sigma_m = 30°$$

SPECGEN was used to generate the input data file *indat.dat*. Again the spectrum was divided into 10 frequency components and 20 directional components, thus a total of 200 waves were present.

The parameters defining the domain and controling the program options are identical to those of Case N4 with the exception of *ntype*. In this case the linear model is used as the satisfactory behavior of the Stokes-to-Hedges nonlinear model was shown for Case N4.

The input data file *indat.dat* is given partially below. Again note that the first seven lines of this file correspond to the first seven lines of the input file *specgen.dat* for **SPECGEN**, which is documented in Appendix C. Also, see Section 2.6.

```
1 2 3
```

```
251  275
1  0  0  1
  1.00000E-01    1.00000E-01    5.00000E-02
0  1
0  0  0
0
1  1
1  10
```

The omitted part mentioned above is inserted here. The last two lines of the file *indat.dat* are as follows.

```
 0.769231
0
```

Figure 19 shows a contour plot of the normalized significant wave height. Figure 20 shows a transect at $x = 12.2m$ and the comparison to available data at that transect. These results also confirm that the breaking term in the model remains sufficiently small for non-dissipative cases.

Breaking Waves

Case B5

In this example Case B5 of the Vincent and Briggs experiments is modeled. The parameters used as input to the preprocessor **SPECGEN** to create the input file for **REF/DIF S** are as follows.

$$
\begin{aligned}
H_{1/3} &= 0.19m \\
f_m &= \frac{1}{1.3sec} = 0.769231Hz \\
f_{max} &= 5Hz \\
\gamma &= 2 \\
\theta_m &= 0^\circ \\
\sigma_m &= 30^\circ
\end{aligned}
$$

Thus, a random sea with a broad frequency and a broad directional spectrum is generated. The spectrum is again divided into 10 frequency and 20 directional components, yielding 200 wave components. The Stokes-to-Hedges nonlinear model is used as strong nonlinear effects occur in this case. Again no bottom friction, no tides or currents were considered. The reference grid is constructed with the same resolution as in the first two cases simulated.

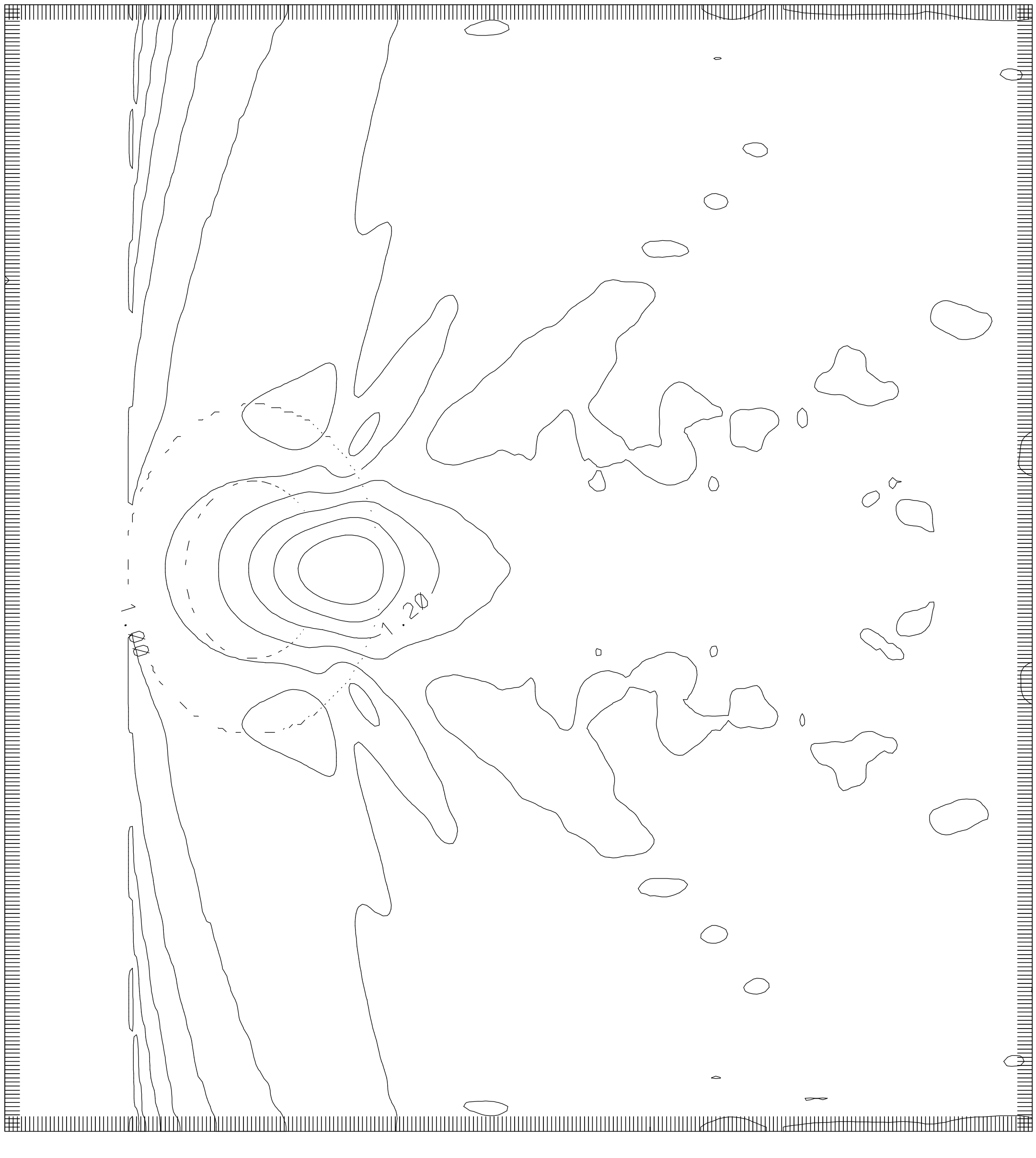

Figure 19: Case B4: Contour Plot of Normalized Significant Wave Height – Model Results

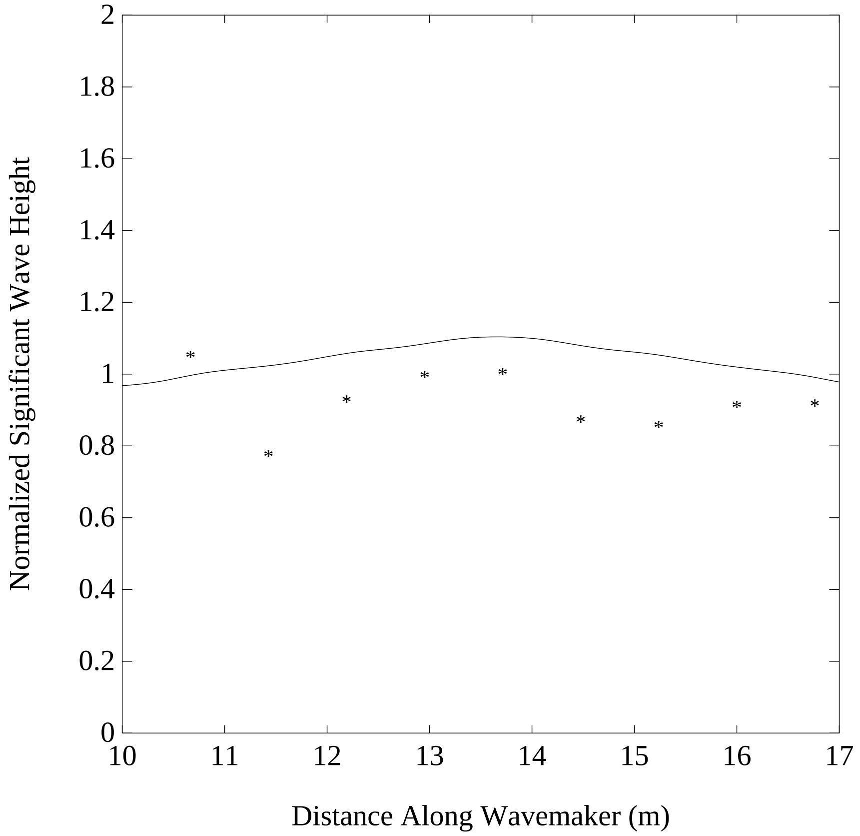

Figure 20: Case B4: Normalized Significant Wave Height for Data (*) and Model Results (—–) at Transect at $x = 12.2$ m

and all other model controlling parameters are the same as well. The reader is referred to the discussion of Cases N4 and B4 for that information.

Part of the input file *indat.dat* is listed below. Note that the first seven lines are identical to those of the input file *specgen.dat* for the program **SPECGEN**, which is documented in Appendix C. Also, see Section 2.6.

```
1  2  3
251  275
1  1  0  1
  1.00000E-01    1.00000E-01    5.00000E-02
0  1
0  0  0
0
1  1
1  10
```

The last two lines of the input file are as follows.

```
 0.769231
0
```

Results are shown in Figure 21 in terms of a contour plot of the normalized significant wave height and in Figure 22 which shows a comparison between the data and the results of the model for the normalized significant wave height at the transect at $x = 12.2$ m.

The results show that the model is partially succesful in predicting the defocusing effect behind the shoal that arises because of nonlinear effects. This effect can not be predicted using the linear model. It is seen though that the data shows a tendency to recover the initial significant wave height at $y = 10$ m and $y = 17$ m, the model does not predict this behavior. The discrepency between the data and the model results is believed to arise because the breaking model currently used in **REF/DIF S** does not take directional effects into account.

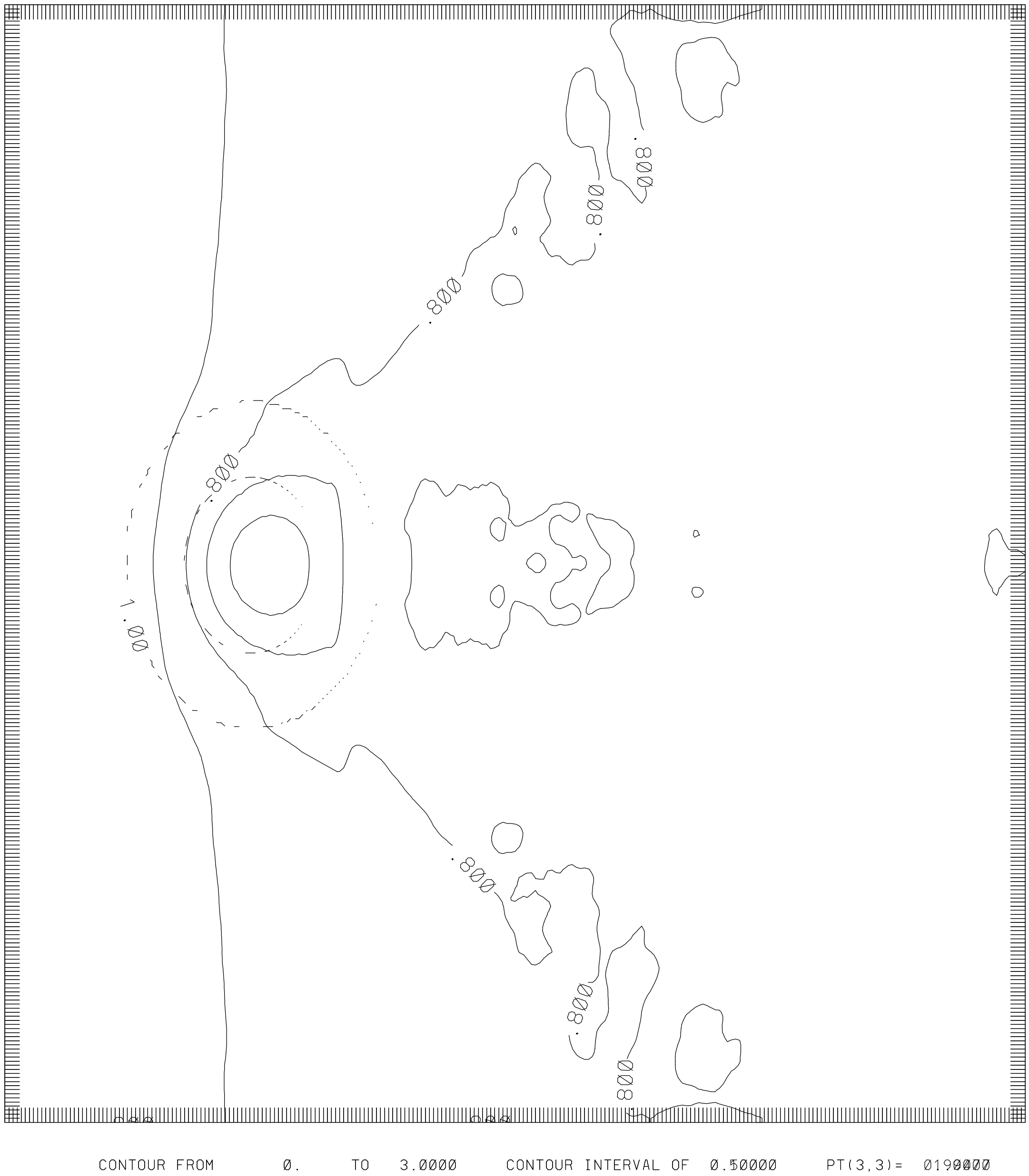

Figure 21: Case B5: Contour Plot of Normalized Significant Wave Height – Model Results

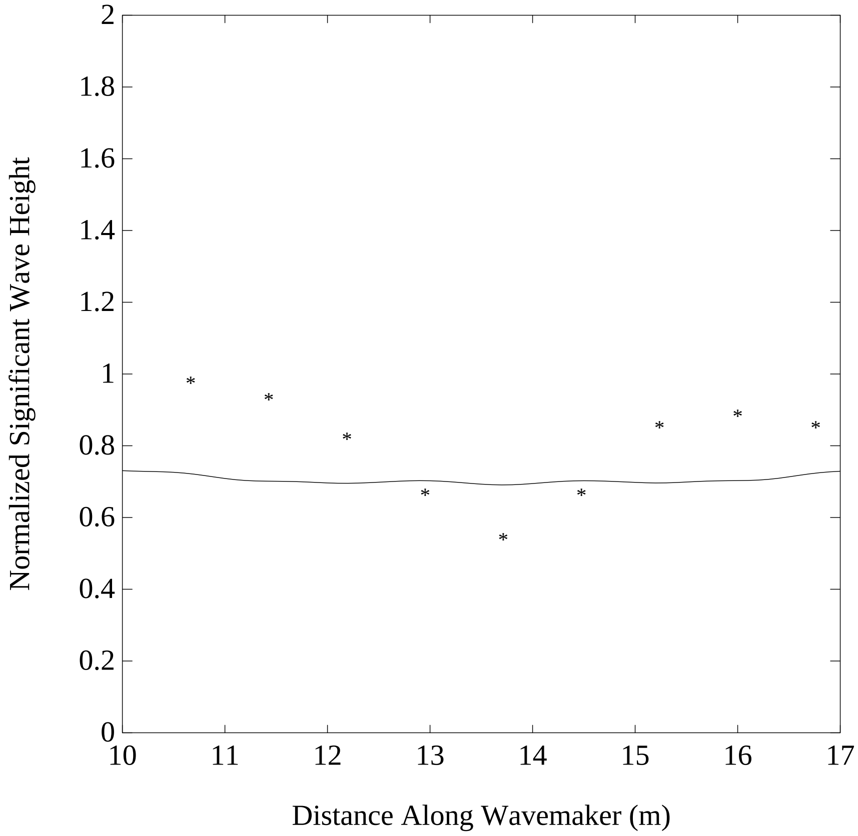

Figure 22: Case B5: Normalized Significant Wave Height for Data (*) and Model Results (—–) at Transect at $x = 12.2$ m

4 REFERENCES

Berkhoff, J.C.W., 1972, "Computation of combined refraction-diffraction," *Proc. 13th Int. Conf. Coastal Engrg.*, ASCE, Vancouver.

Berkhoff, J.C.W., N. Booij and A.C. Radder, 1982, "Verification of numerical wave propagation models for simple harmonic linear waves," *Coastal Engineering*, 6, 255-279.

Bettess, P. and O.C. Zienkiewicz, 1977, "Diffraction and refraction of surface waves using finite and infinite elements," *Int. J. for Numerical Methods in Engrg.*, 1, 1271-1290.

Booij, N., 1981, *Gravity Waves on Water with Non-uniform Depth and Current*, Doctoral dissertation, Technical University of Delft, The Netherlands, 131 pp.

Booij, N., 1983, "A note on the accuracy of the mild-slope equation," *Coastal Engineering*, 7, 191-203.

Borgman, L.E. (1985), "Directional spectrum estimation for the S_{xy} gauges." *Technical Report*, Coast. Engrg. Res. Center, Waterways Experiment Station, Vicksburg, MS, 1-104.

Carnahan, B., H.A. Luther and J.O. Wilkes, 1969, *Applied Numerical Methods*, Wiley.

Chu, V.C. and C.C. Mei, 1970, "On slowly varying Stokes waves," *J. Fluid Mech.*, 41, 873-887.

Dalrymple, R.A., J.T. Kirby and P.A. Hwang, 1984a, "Wave diffraction due to areas of energy dissipation," *J. Waterway, Port, Coastal and Ocean Div.*, ASCE, 110, 67-79.

Dean, R.G. and R.A. Dalrymple, 1984, *Water Wave Mechanics for Engineers and Scientists*, Englewood Cliffs: Prentice-Hall.

Djordjevic, V.D. and L.G. Redekopp, 1978, "On the development of packets of surface gravity waves moving over and uneven bottom," *Z. Angew. Math. and Phys.*, 29, 950-962.

Hedges, T.S., 1976, "An empirical modification to linear wave theory," *Proc. Inst. Civ. Eng.*, 61, 575-579.

Houston, J.R., 1981, "Combined refraction-diffraction of short waves using the finite element method", *Applied Ocean Res.*, 3, 163-170.

Hughes, S.A., 1984, "The TMA shallow-water spectrum description and applications", Tech. Rept. CERC-84-7, Coast. Engrg. Res Center, Waterways Experiment Station, Vicksburg, Miss.

Jonsson, I.G. and O. Skovgaard, 1979, "A mild-slope wave equation and its application to tsunami calculations," *Mar. Geodesy*, 2, 41-58.

Kirby, J .T., 1983, "Propagation of weakly-nonlinear surface water waves in regions with varying depth and current", ONR Tech. Rept. 14, Res. Rept. CE-83-37, Department of Civil Engineering, University of Delaware, Newark.

Kirby, J.T. and R.A. Dalrymple, 1983a, "A parabolic equation for the combined refraction-diffraction of Stokes waves by mildly varying topography," *J. Fluid Mech.*, 136, 543-566.

Kirby, J.T. and R.A. Dalrymple, 1983b, "The propagation of weakly nonlinear waves in the presence of varying depth and currents," *Proc. XX^{th} Congress I.A.H.R.*, Moscow.

Kirby, J.T., 1984, "A note on linear surface wave-current interaction," *J. Geophys. Res.*, 89, 745-747.

Kirby, J.T. and R.A. Dalrymple, 1984, "Verification of a parabolic equation for propagation of weakly non-linear waves," *Coastal Engineering*, 219-232.

Kirby, J. T., 1986a, "Higher-order approximations in the parabolic equation method for water waves", *J. Geophys. Res.*, **91**, 933-952.

Kirby, J. T., 1986b, "Rational approximations in the parabolic equation method for water waves", *Coastal Engineering*,

Kirby, J. T., 1986c, "Open boundary condition in parabolic equation method", *J. Waterway, Port, Coast. and Ocean Eng.*, **112**, 460-465.

Kirby, J. T. and Dalrymple, R. A., 1986a, "Modeling waves in surfzones and around islands", *J. Waterway, Port, Coast. and Ocean Eng.*, **112**, 78-93.

Kirby, J. T. and Dalrymple, R. A., 1986b, "An approximate model for nonlinear dispersion in monochromatic wave propagation models", *Coast. Eng.*, **9**, 545-561.

Kirby, J. T. and Dalrymple, R. A., 1992, " **REF/DIF 1** Version 2.4, Documentation and User's Manual", CACR 92-04, Coast. Engrg. Res Center, Waterways Experiment Station, Vicksburg, Miss

Kirby, J. T., 1993, "Damping of high frequency noise in the large angle parabolic equation method", manuscript.

Krommes, J. A., 1992, "The *web* system of structured software design and documentation for C, C++, Fortran, Ratfor, and TeX", draft report.

Liu, P.L.-F. and R.A. Dalrymple, 1984, "The damping of gravity water waves due to percolation," *Coastal Engineering.*

Mase, H. and Kirby, J.T., 1992, "Modified frequency-domain KdV equation for random wave shoaling," *Proc. 23rd Intl. Conf. Coast. Engrg.*, Venice, 474-487.

Mei, C.C. and E.O. Tuck, 1980, "Forward scattering by thin bodies," *SIAM J. Appl. Math.*, 39, 178-191.

Phillips, O.M., 1966, *The Dynamics of the Upper Ocean*, Cambridge University Press.

Radder, A.C., 1979, "On the parabolic equation method for water-wave propagation," *J. Fluid Mech.*, 95, 159-176.

Thornton, E.B. and Guza, R.T., 1983, "Transformation of wave height distribution," *J. Geophys. Res.*, 88, 5925-5938.

Vincent, C.L. and Briggs, M.J., 1989, "Refraction-diffraction of irregular waves over a mound," *J. Waterway, Port, Coast. and Ocean Eng.*, 115(2), 269-284.

Yue, D.K.P. and C.C. Mei, 1980, "Forward diffraction of Stokes waves by a thin wedge," *J. Fluid Mech.*, 99, 33-52.

5 Appendix A: REF/DIF S Program Listing

The following section contains the listing for **REF/DIF S**. An attempt has been made to make the code readible by using comments at appropriate locations in the program. The anotations may be improved significantly using the *fweb* documentation program (Krommes, 1992).

The main program **REF/DIF S** and the eleven subroutines are listed here in the following order:

1. *refdifs*
2. *inref*
3. *inwave*
4. *model*
5. *grid*
6. *con*
7. *fdcalc*
8. *rbcon*
9. *wnum*
10. *vwnum*
11. *vtrida*
12. *diss*

```
c*-------------------------------------------------------------------
c*
c*    program:  ref/dif s
c*    version:  1.1
c*
c*    program to calculate the forward scattered wave field in
c*    regions with slowly varying depth and current, including the
c*    effects of refraction and diffraction.  program is based on
c*    parabolic equation method.
c*--------------------------------------------------------------------
      parameter(ixr=251,iyr=275,ix=11,iy=275,ncomp=10,
     1nnd=20,nnii=200)
      common/ref1/mr,nr,ispace,nd,md(ixr),iu,dconv(2),if(3),icur,ibc,
     1dconv2(2)
      common/ref2/dr(ixr,iyr),ur(ixr,iyr),vr(ixr,iyr),iun(8),iinput,
     1ioutput
      common/ref3/dxr,dyr,xr(ixr),yr(iyr),x(ix),y(iy)
      common/ref4/isd(ixr,iyr)
      common/block1/d(ix,iy),u(ix,iy),v(ix,iy),m,n,dx,dy
      common/con1/q(2,iy,ncomp),p(2,iy,ncomp),sig(2,iy,ncomp)
      common/con2/k(2,iy,ncomp),kb(2,ncomp),w(2,iy,nnii),dd(2,iy,ncomp)
      common/nlin/an,anl,ntype
      common/wav1/iwave,nfreqs,freqs(ncomp),nwavs,istore(nnii)
     1,nii,fpeak
      common/wav2/amp(ncomp,nnd),dir(ncomp,nnd),tide
      common/comp/a(2,iy,nnii),psibar(ncomp),az(iy,nnii),h13(iy)
      real k,kb
      complex a,w,az
      dconv(1)=1.
      dconv(2)=0.30488
      dconv2(1)=1.
      dconv2(2)=0.453
c*--------------------------------------------------------------------
c*    specify unit number for line printer output.  standard choice
c*    would be *; some systems may vary.
c*    this line must be set during installation.
c*--------------------------------------------------------------------
c*    print headers on output
c*--------------------------------------------------------------------
      write(*,100)
c*--------------------------------------------------------------------
c*    read control parameters and reference grid data
c*--------------------------------------------------------------------
      call inref
c*--------------------------------------------------------------------
c*    read control parameters and initializing wave data
c*--------------------------------------------------------------------
      call inwave
c*--------------------------------------------------------------------
c*    pass program control to subroutine model
c*    for each frequency component specified in inwave, model
```

```
c*    executes the model throughout the entire grid and then
c*    reinitializes the model for the next frequency
c*----------------------------------------------------------------------
      call model
c*---------------------------------------------------------------------
c*    all done
c*    close output data files if open and close statements are
c*    being used.
c*----------------------------------------------------------------------
      do 1 i=1,3
      close(iun(i))
 1    continue
      close(22)
      if(iopt.eq.1) close(36)
      stop
 100  format(//////20x,'Refraction-Diffraction Model for'/
     120x,'Spectral Wave Conditions'///
     120x,'REF/DIF S  Version 1.1'///
     120x,'Center for Applied Coastal Research'/
     120x,'Department of Civil Engineering'/
     120x,'University of Delaware'/
     120x,'Newark, Delaware 19716'///
     110x,'James T. Kirby and H.Tuba Ozkan, July 1992, February 1994')
      end
```

```
c*-----------------------------------------------------------------
c*
      subroutine inref
c*
c*    subroutine reads in and checks dimensions and values for
c*    large scale reference grid.  wave parameters for the
c*    particular run are read in later by subroutine inwave.
c*    the following unit (device) numbers are assumed:
c*
c*          unit      contents
c*
c*           iun(1)   input reference grid values of d, u, and v
c*           iun(2)   input user specified subgrid divisions
c*           iun(3)   output results at reference grid locations to
c*                    disk file
c*           *        assumed unit for line printer output
c*                    (must be specified in program, machine
c*                     specific. * is sun workstation version)
c*           iun(5)   dimension information and run instructions
c*                    (must be specified in program, machine
c*                     specific)
c*
c*    variable definitions
c*
c*        mr,nr  -  reference grid dimensions (maximum 100,100)
c*        dxr,dyr-  grid spacing for reference grid
c*        iu     -  physical unit descriptor ( 1=mks, 2=english)
c*                  default value is 1, mks units
c*        dt     -  depth tolerence value (to check for anomalous
c*                  depth values)
c*        ispace -  switch to control grid subdivision.
c*                  =0, program attempts its own subdivisions
c*                  =1, user specifies subdivisions
c*        nd     -  y direction subdivision (ispace=0 or 1)
c*                  (must be .lt. iy/nr-1)
c*        md(mr-1)- x direction subdivisions (if ispace=1)
c*                  (must be .le. ix-1)
c*        ntype  -  nonlinearity control parameter
c*                  =0, linear model
c*                  =1, stokes matched to hedges in shallow water
c*                  =2, stokes throughout
c*        icur   -  switch to tell program if current data is to
c*                  be used and read on input
c*                  =0, no input current data
c*                  =1, input current data to be read
c*                  program defaults to icur=0
c*        ibc    -  boundary condition switch
c*                  =0, use closed lateral boundaries
c*                  =1, use open lateral conditions
c*                  program defaults to ibc=0
c*        dr     -  depths at grid points
```

```
c*                  >0, submerged areas
c*                  <0, elevation above surface datum
c*        ur     -  x velocities at grid points
c*                  (only entered if icur=1)
c*        vr     -  y velocities at grid points
c*                  (only entered if icur=1)
c*
c*    subroutine is called from main and returns control to
c*    calling location
c*-------------------------------------------------------------------
      parameter(ixr=251,iyr=275,ix=11,iy=275,ncomp=10,
     1nnd=20,nnii=200)
      common/ref1/mr,nr,ispace,nd,md(ixr),iu,dconv(2),if(3),icur,ibc,
     1dconv2(2)
      common/ref2/dr(ixr,iyr), ur(ixr,iyr), vr(ixr,iyr),iun(8),iinput,
     1ioutput
      common/ref3/dxr,dyr,xr(ixr),yr(iyr),x(ix),y(iy)
      common/ref4/isd(ixr,iyr)
      common/block1/d(ix,iy),u(ix,iy),v(ix,iy),m,n,dx,dy
      common/nlin/an,anl,ntype
c*-----------------------------------------------------------------------
c*    constants
c*-----------------------------------------------------------------------
      g=9.80621
c*----------------------------------------------------------------------
c*    define device number iun(5) for reference grid input
c*    this line must be set during program installation, prior to
c*    compilation.  also construct an open statement if needed.
c*----------------------------------------------------------------------
      iun(5)=5
      open(unit=iun(5),file='indat.dat',status='old')
c*----------------------------------------------------------------------
c*    remaining unit numbers in i4 format
c*----------------------------------------------------------------------
      read(iun(5),*)(iun(i),i=1,3)
      open(unit=iun(1),file='refdat.dat',status='old')
      open(unit=iun(2),file='subdat.dat')
      open(unit=iun(3),file='outdat.dat')
      open(unit=22,file='outrad.dat')
c*----------------------------------------------------------------------
c*    headers on output file *
c*-----------------------------------------------------------------------
      write(*,106)
c*-----------------------------------------------------------------------
c*    read control data from unit iun(5)
c*-----------------------------------------------------------------------
      read(iun(5),*) mr,nr
      read(iun(5),*) iu, ntype, icur, ibc
      read(iun(5),*) dxr, dyr, dt
      read(iun(5),*) ispace, nd
      if(ispace.eq.1) then
      read(iun(5),*) (md(i),i=1,mr-1)
      else
      endif
```

```
      write(*,107) mr,nr,dxr,dyr
      if(iu.eq.1) write(*,114) iu
      if(iu.eq.2) write(*,115) iu
      if(icur.eq.0) write(*,200)
      if(icur.eq.1) write(*,201)
      if(ibc.eq.0) write(*,202)
      if(ibc.eq.1) write(*,203)
      if(ispace.eq.0)write(*,108)
      if(ispace.eq.1)write(*,109)
      write(*,119) nd
      if(ntype.eq.0) write(*,110)
      if(ntype.eq.1) write(*,111)
      if(ntype.eq.2) write(*,112)
c*----------------------------------------------------------------------
c*    check input from unit iun(5)
c*----------------------------------------------------------------------
      if((mr.gt.ixr).or.(nr.gt.iyr)) then
        stop 'dimensions for reference grid too large, stopping'
      else
      end if
      if((iu.ne.1).and.(iu.ne.2)) iu=1
      dt=dt*dconv(iu)
      dxr=dxr*dconv(iu)
      dyr=dyr*dconv(iu)
      if(dt.eq.0.) dt=2.
      if(nd.gt.(ifix(float(iy-1)/float(nr-1)))) then
        write(*,102) nd
        stop
      else
      endif
      if(ispace.eq.1) then
        test=0.
        do 1 i=1,mr-1
        if(md(i).gt.(ix-1)) then
          write(*,103) md(i),i
          test=1.
        else
        endif
 1      continue
      if(test.eq.1.) stop
      else
      end if
c*--------------------------------------------------------------------
c*    read depth grid and velocities from unit iun(1)
c*--------------------------------------------------------------------
      do 2 i=1,mr
      read(iun(1),101)(dr(i,j),j=1,nr)
 2    continue
      if(icur.eq.1)then
      do 3 i=1,mr
      read(iun(1),101)(ur(i,j),j=1,nr)
 3    continue
      do 4 i=1,mr
      read(iun(1),101)(vr(i,j),j=1,nr)
```

```
 4    continue
      endif
c*-----------------------------------------------------------------------
c*    convert depth and currents
c*-----------------------------------------------------------------------
      do 5 i=1,mr
      do 5 j=1,nr
      dr(i,j)=dr(i,j)*dconv(iu)
 5    continue
      if(icur.eq.1)then
      do 55 i=1,mr
      do 55 j=1,nr
      ur(i,j)=ur(i,j)*dconv(iu)
      vr(i,j)=vr(i,j)*dconv(iu)
 55   continue
      endif
c*-----------------------------------------------------------------------
c*    check for large depth changes and large currents
c*    in reference grid data
c*------------------------------------------------------------------------
      do 6 i=2,mr-1
      do 6 j=2,nr-1
      dcheck=(dr(i+1,j)+dr(i-1,j)+dr(i,j-1)+dr(i,j+1))/4.
      if(abs(dcheck-dr(i,j)).gt.dt)
     1  write(*,104) dr(i,j), i,j,dt
 6    continue
      if(icur.eq.1) then
      do 7 i=1,mr
      do 7 j=1,nr
      if(dr(i,j).le.0.0) go to 7
      fr=(ur(i,j)*ur(i,j)+vr(i,j)*vr(i,j))/(g*dr(i,j))
      if(fr.gt.1.) write(*,105) i,j,fr
 7    continue
      endif
c*---------------------------------------------------------------------
c*   establish coordinates for reference grid
c*---------------------------------------------------------------------
      do 8 ir=1,mr
      xr(ir)=float(ir-1)*dxr
 8    continue
      do 9 jr=1,nr
      yr(jr)=float(jr-1)*dyr
 9    continue
c*----------------------------------------------------------------------
c*    establish y coordinates for interpolated grid
c*----------------------------------------------------------------------
      n=nd*(nr-1)+1
      dy=dyr/float(nd)
      do 10 j=1,n
      y(j)=float(j-1)*dy
 10   continue
c*-----------------------------------------------------------------------
c*    write grid information on output unit iun(3)
c*-----------------------------------------------------------------------
```

```
      write(iun(3),*) nr,mr
      write(iun(3),*) (yr(jr)/dconv(iu),jr=1,nr)
c*-----------------------------------------------------------------
c*    read in changes for friction values
c*
c*        if(1)=1, turbulent boundary layer damping everywhere
c*        if(2)=1, porous bottom damping everywhere
c*        if(3)=1, laminar boundary layer damping everywhere
c*-----------------------------------------------------------------
      read(iun(5),*) (if(i),i=1,3)
      do 11 i=1,3
      if((if(i).ne.0).and.(if(i).ne.1)) if(i)=0
 11   continue
      write(*,116)(if(i),i=1,3)
c*---------------------------------------------------------------
c*    specify whether or not user specified subgrids are to be
c*    read in during model operation.
c*    isp=0, no subgrids specified
c*    isp=1, subgrids to be read in later from unit iun(2)
c*------------------------------------------------------------------
      read(iun(5),*) isp
      if(isp.eq.0) write(*,117)
      if(isp.eq.1) write(*,118)
      if((isp.eq.1).and.(ispace.eq.0))write(*,113)
      if(isp.eq.0)then
      do 14 ir=1,mr
      do 14 jr=1,nr
      isd(ir,jr)=0
 14   continue
      else
      do 15 ir=1,mr-1
      read(iun(2),100)(isd(ir,jr),jr=1,nr-1)
 15   continue
      end if
c*--------------------------------------------------------------------
c*    input done, return to main program
c*---------------------------------------------------------------------
      return
 100  format(15i4)
 101  format(16f8.4)
 102  format(' y-direction subdivision nd=',i4,'too fine.'/
     1' maximum number of y grid points will be exceeded.'/
     1' execution terminating.')
 103  format(' x-direction subdivision md=',i4,
     1'too fine on grid block',2x,i3/' execution terminating')
 104  format(' depth',2x,f7.2,'(m) at reference grid location',
     12(2x,i3)/' differs from the average of its neighbors by',
     1' more than',2x,f7.2,'(m).'/' execution continuing')
 105  format(' ambient current at reference grid location'
     1,2(2x,i3),' is supercritical with froude number =',f7.4/
     1' execution continuing')
 106  format('0'///20x,'input section, reference grid values'///)
 107  format(' reference grid dimensions  mr=',i3/
     1      '                             nr=',i3///
```

```
     1       ' reference grid spacings   dxr=',f8.4/
     1       '                           dyr=',f8.4)
 108  format(' '/' ispace =0 chosen, program will attempt its own ',
     1'reference grid subdivisions')
 109  format(' '/' ispace =1 chosen, subdivision spacings will be',
     1' input as data')
 110  format(' '/' ntype = 0, linear model')
 111  format(' '/' ntype = 1, stokes model matched to hedges model')
 112  format(' '/' ntype = 2, stokes model')
 113  format(' warning: input specifies that user will be supplying',
     1' specified subgrids (isp=1),'/
     1' while program has been told to generate its own subgrid',
     1' spacings (ispace=0).'/
     1' possible incompatibility in any or all subgrid blocks')
 114  format(' '/' physical unit switch iu=',i1,
     1',  input in mks units')
 115  format(' '/' physical unit switch iu=',i1,
     1',  input in english units')
 116  format(' '//'   switches for dissipation terms'//
     1' ',i1,'   turbulent boundary layer'/
     1' ',i1,'   porous bottom'/
     1' ',i1,'   laminar boundary layer')
 117  format(' '/' isp=0, no user defined subgrids')
 118  format(' '/' isp=1, user defined subgrids to be read')
 119  format(' '/' y-direction subdivision according to nd=',i3)
 200  format(' '/' icur=0, no current values read from input files')
 201  format(' '/' icur=1, current values read from data files')
 202  format(' '/' ibc=0, closed (reflective) lateral boundaries')
 203  format(' '/' ibc=1, open lateral boundaries')
      end
```

```
c*-------------------------------------------------------------------
c*
      subroutine inwave
c*
c*    read in wave parameters.
c*    variable definitions
c*
c*        iinput  -  determine method of specifying the first
c*                   row of computational values
c*                    =1, input values from indat.dat according to
c*                   value of iwave
c*       =2, input complex a values from file wave.dat
c*
c*        ioutput -  determine whether last row of complex amplitudes
c*                   are stored in separate file owave.dat
c*                    =1, extra data not stored
c*                    =2, extra data stored in file owave.dat
c*
c*    if iinput=1:
c*
c*        iwave   -  input wave type
c*                    =1, input discrete wave amplitudes and
c*                        directions
c*                    =2, read in dominant direction, total average
c*                        energy density, and spreading factor
c*        nfreqs  -  number of frequency components to be used
c*                    (separate model run for each component)
c*        freqs   -  wave frequency for each of nfreqs runs
c*        tide    -  tidal offset for each of nfreqs runs
c*
c*    if iwave = 1
c*
c*        nwavs   -  number of discrete wave components at
c*                   each of nfreqs runs
c*        amp     -  initial amplitude of each component
c*        dir     -  direction of each discrete component in
c*                   + or - degrees from the x-direction
c*
c*    if iinput=2:
c*
c*       freqs    -  wave frequency for one run
c*       tide     -  tidal offset for one run
c*-------------------------------------------------------------------
      parameter(ixr=251,iyr=275,ix=11,iy=275,ncomp=10,
     1nnd=20,nnii=200)
      common/ref1/mr,nr,ispace,nd,md(ixr),iu,dconv(2),if(3),icur,ibc,
     1dconv2(2)
      common/ref2/dr(ixr,iyr),ur(ixr,iyr),vr(ixr,iyr),iun(8),iinput,
     1ioutput
      common/wav1/iwave,nfreqs,freqs(ncomp),nwavs,istore(nnii)
     1,nii,fpeak
```

```
      common/wav2/amp(ncomp,nnd),dir(ncomp,nnd),tide
      pi=3.1415927
c*---------------------------------------------------------------------
c*    read value of iinput, ioutput
c*---------------------------------------------------------------------
      read(iun(5),*) iinput,ioutput
      if(iinput.ne.1) iinput=1
      if(ioutput.ne.1) ioutput=1
      if(iinput.eq.1)then
      write(*,*)'iinput = 1, program specifies initial row of a'
c*---------------------------------------------------------------------
c*    read iwave, nfreqs
c*---------------------------------------------------------------------
      write(*,102)
      read(iun(5),*) iwave, nfreqs
      if(iwave.ne.1) iwave=1
      write(*,103)
      if(nfreqs.gt.ncomp) then
      write(*,104)
      stop
      endif
      write(*,105) nfreqs
c*---------------------------------------------------------------------
c*    for each frequency, enter the wave period and tidal offset
c*---------------------------------------------------------------------
      do 3 ifreq=1,nfreqs
      read(iun(5),*) freqs(ifreq), tide
      write(*,107) ifreq,freqs(ifreq),tide
      freqs(ifreq)=2.*pi/freqs(ifreq)
      tide=tide*dconv(iu)
c*---------------------------------------------------------------------
c*    if iwave = 1, read the number of discrete components
c*---------------------------------------------------------------------
      if(iwave.eq.1) then
      read(iun(5),*) nwavs
      if(nwavs.gt.nnd) then
      write(*,109)
      stop
      endif
      do 1 iwavs=1,nwavs
      read(iun(5),*) amp(ifreq,iwavs), dir(ifreq,iwavs)
      write(*,106) iwavs,amp(ifreq,iwavs),dir(ifreq,iwavs)
      dir(ifreq,iwavs)=dir(ifreq,iwavs)*pi/180.
      amp(ifreq,iwavs)=amp(ifreq,iwavs)*dconv(iu)
 1    continue
      else
      end if
 3    continue
      endif
      return
 100  format(15i4)
 101  format(16f8.4)
 102  format('1'///20x,' input section, wave data values'///)
 103  format(' '///' iwave=1, discrete wave amps and directions')
```

```
104  format('Too many frequency components; stopping')
105  format(' '///' the model is to be run for',i3,' separate',
    1' frequency components')
106  format(' '/' wave component ',i2,', amplitude =',f8.4,
    1', direction=',f8.4)
107  format(' '//' frequency component ',i2//
    1' wave period=',f8.4,'sec., tidal offset=',f8.4)
108  format(' '/' total variance density =',f8.4,', spreading factor
    1n=',i2,' seed number =',i5)
109  format('Too many directional components; stopping')
     end
```

```
c*-----------------------------------------------------------------
c*
      subroutine model
c*
c*    this subroutine is the control level for the actual wave
c*    model.  data read in during inref and inwave is conditioned
c*    and passed to the wave model.  this routine is executed
c*    once for each frequency component specified in inwave.
c*
c*    the wave model is split in three parts which are run
c*    sequentially for each reference grid row.
c*
c*       grid   -  subroutine performs the interpolation of
c*                 depth and current values.
c*       con    -  calculate the constants needed by the finite
c*                 difference scheme.
c*       fdcalc -  perform the finite difference calculations.
c*-----------------------------------------------------------------
      parameter(ixr=251,iyr=275,ix=11,iy=275,ncomp=10,
     1nnd=20,nnii=200)
      common/ref1/mr,nr,ispace,nd,md(ixr),iu,dconv(2),if(3),icur,ibc,
     1dconv2(2)
      common/ref2/dr(ixr,iyr),ur(ixr,iyr),vr(ixr,iyr),iun(8),iinput,
     1ioutput
      common/ref3/dxr,dyr,xr(ixr),yr(iyr),x(ix),y(iy)
      common/ref4/isd(ixr,iyr)
      common/block1/d(ix,iy),u(ix,iy),v(ix,iy),m,n,dx,dy
      common/con1/q(2,iy,ncomp),p(2,iy,ncomp),sig(2,iy,ncomp)
      common/con2/k(2,iy,ncomp),kb(2,ncomp),w(2,iy,nnii),dd(2,iy,ncomp)
      common/nlin/an,anl,ntype
      common/wav1/iwave,nfreqs,freqs(ncomp),nwavs,istore(nnii)
     1,nii,fpeak
      common/wav2/amp(ncomp,nnd),dir(ncomp,nnd),tide
      common/comp/a(2,iy,nnii),psibar(ncomp),az(iy,nnii),h13(iy)
      dimension xxk(ncomp),s(iy),th(iy,nnii),sxy(iy),sxx(iy),syy(iy)
      complex a,w,az
      real k,kb
      g=9.80621
      rho=1000.
      pi=3.1415927
      eps=1.0e-05
      nii=nwavs*nfreqs
      if(nii.eq.1) then
      fpeak=freqs(1)
      iopt=0
      else
      read(iun(5),*) fpeak
      read(iun(5),*) iopt
      if(iopt.eq.1) then
      read(iun(5),*) ifcomp,idcomp
      icomp=nwavs*(ifcomp-1)+idcomp
```

```
      open(unit=36,file='specamp.dat')
      write(36,*) nr,mr
      write(36,*)(yr(jr)/dconv(iu),jr=1,nr)
      else
      iopt=0
      endif
      endif
c*---------------------------------------------------------------------
c*    specify initial nonlinear parameters once
c*---------------------------------------------------------------------
      if(ntype.eq.0) an=0.
      if(ntype.ne.0) an=1.
      if(ntype.ne.2) anl=0.
      if(ntype.eq.2) anl=1.
      dref=dr(1,1)+tide
      uu=ur(1,1)
      call wnum(dref,uu,freqs,xxk,eps,nfreqs)
      do 30 ifreq=1,nfreqs
      k(1,1,ifreq)=xxk(ifreq)
   30 continue
c*--------------------------------------------------------------------
c*    set-up initial condition
c*--------------------------------------------------------------------
      do 2 ii=1,nii
      do 3 j=1,n
      do 4 i=1,2
      a(i,j,ii)=cmplx(0.,0.)
    4 continue
    3 continue
    2 continue
      do 5 j=1,n
      do 6 ifreq=1,nfreqs
      do 7 iwavs=1,nwavs
      ii=nwavs*(ifreq-1)+iwavs
      a(1,j,ii)=amp(ifreq,iwavs)*cexp(cmplx(0.,k(1,1,ifreq)
     1*sin(dir(ifreq,iwavs))*y(j)))
      istore(ii)=ifreq
      th(j,ii)=dir(ifreq,iwavs)
    7 continue
    6 continue
    5 continue
c*--------------------------------------------------------------------
c*    compute significant waveheight hs for initial condition
c*--------------------------------------------------------------------
      do 13 j=1,n
      s(j)=0.
      do 14 ii=1,nii
      s(j)=s(j)+((cabs(a(1,j,ii)))**2.)
   14 continue
      h13(j)=sqrt(8.*s(j))
   13 continue
c*--------------------------------------------------------------------
c*    initial condition set up
c*    now execute model for each grid block
```

```
c*-------------------------------------------------------------------
      do 20 ii=1,nii
      psibar(ii)=0.
   20 continue
      do 100 ir=1,(mr-1)
c*-------------------------------------------------------------------
c*    establish interpolated grid block for segment ir
c*-------------------------------------------------------------------
      call grid(ir)
c*-------------------------------------------------------------------
c*    write initial values on iun(3)
c*-------------------------------------------------------------------
      if(ir.eq.1) then
      write(iun(3),*) x(1)/dconv(iu)
      write(iun(3),*)(d(1,j)/dconv(iu),j=1,n,nd)
      write(iun(3),*)(h13(j)/dconv(iu),j=1,n,nd)
      if(iopt.eq.1) then
      write(36,*) x(1)/dconv(iu),psibar(icomp)
      write(36,*)(d(1,j)/dconv(iu),j=1,n,nd)
      write(36,*)(a(1,j,icomp)/dconv(iu),j=1,n,nd)
      endif
      endif
      do 11 ii=1,nii
      do 12 j=1,n
      az(j,ii)=a(1,j,ii)
   12 continue
   11 continue
c*-------------------------------------------------------------------
c*    calculate constants for first row in each grid block
c*-------------------------------------------------------------------
      call con(ir,1,1)
c*-------------------------------------------------------------------
c*    Line printer output for first row
c*-------------------------------------------------------------------
      if(ir.eq.1) then
      do 21 j=1,n
      sxx(j)=0.
      syy(j)=0.
      sxy(j)=0.
   21 continue
      do 25 j=1,n
      do 22 ii=1,nii
      sxx(j)=sxx(j)+(cabs(a(1,j,ii))**2)
     1*(q(1,j,istore(ii))*(1+(cos(th(j,ii))**2))-0.5)
      syy(j)=syy(j)+(cabs(a(1,j,ii))**2)
     1*(q(1,j,istore(ii))*(1+(sin(th(j,ii))**2))-0.5)
      sxy(j)=sxy(j)+q(1,j,istore(ii))
     1*(cabs(a(1,j,ii))**2)*sin(th(j,ii))
   22 continue
      sxx(j)=sxx(j)*rho*g/8.
      syy(j)=syy(j)*rho*g/8
      sxy(j)=sxy(j)*rho*g/16.
   25 continue
c*---------------------------------------------------------------------
```

```
c*    Line printer output
c*----------------------------------------------------------------------
      mm1=m-1
      write(*,205) ir,mm1
      write(*,202) x(1)/dconv(iu)
      write(*,206)
      write(*,203) (h13(j)/dconv(iu),j=1,n,nd)
c*--------------------------------------------------------------------
c*    Output of radiation stresses
c*--------------------------------------------------------------------
      write(22,*) nr,mr
      write(22,*) (sxx(j)*dconv2(iu),j=1,n,nd)
      write(22,*) (sxy(j)*dconv2(iu),j=1,n,nd)
      write(22,*) (syy(j)*dconv2(iu),j=1,n,nd)
      endif
c*--------------------------------------------------------------------
c*    for next row, evaluate constants
c*--------------------------------------------------------------------
      do 9 icount=2,m
      call con(ir,icount,2)
c*--------------------------------------------------------------------
c*    perform finite differencing calculations
c*--------------------------------------------------------------------
      call fdcalc(ir,icount)
c*--------------------------------------------------------------------
c*    roll back values to row 1
c*--------------------------------------------------------------------
      call rbcon
    9 continue
c*-----------------------------------------------------------------------
c*    write out reference grid data on disk file iun(3)
c*-----------------------------------------------------------------------
      write(iun(3),*) x(m)/dconv(iu)
      write(iun(3),*)(d(m,j)/dconv(iu),j=1,n,nd)
      write(iun(3),*)(h13(j)/dconv(iu),j=1,n,nd)
      if(iopt.eq.1) then
      write(36,*) x(m)/dconv(iu),psibar(icomp)
      write(36,*)(d(m,j)/dconv(iu),j=1,n,nd)
      write(36,*)(a(1,j,icomp)/dconv(iu),j=1,n,nd)
      endif
  134 continue
  100 continue

 202  format(' x=',f10.2)
 203  format(' ',12(f10.4))
 205  format(' grid row ir=',i3,',  ',i3,' x-direction subdivisions',
     1' used')
 206  format(' Significant Wave Height')
      return
      end
```

```
c*---------------------------------------------------------------------
c*
      subroutine grid(ir)
c*
c*    interpolate the depth and current grids for reference
c*    grid block ir.
c*----------------------------------------------------------------------
      parameter(ixr=251,iyr=275,ix=11,iy=275,ncomp=10,
     1nnd=20,nnii=200)
      common/ref1/mr,nr,ispace,nd,md(ixr),iu,dconv(2),if(3),icur,ibc,
     1dconv2(2)
      common/ref2/dr(ixr,iyr),ur(ixr,iyr),vr(ixr,iyr),iun(8),iinput,
     1ioutput
      common/ref3/dxr,dyr,xr(ixr),yr(iyr),x(ix),y(iy)
      common/ref4/isd(ixr,iyr)
      common/block1/d(ix,iy),u(ix,iy),v(ix,iy),m,n,dx,dy
      common/con2/k(2,iy,ncomp),kb(2,ncomp),w(2,iy,nnii),dd(2,iy,ncomp)
      common/wav1/iwave,nfreqs,freqs(ncomp),nwavs,istore(nnii)
     1,nii,fpeak
      common/wav2/amp(ncomp,nnd),dir(ncomp,nnd),tide
      dimension dref(iy),npts(ncomp),sumk(ncomp),xu(iy),xk(iy,ncomp)
      real k,kb
      complex w
      pi=3.1415927
      eps=1.0e-05
c*---------------------------------------------------------------
c*    perform y-interpolation on reference grid
c*---------------------------------------------------------------
c*    interpolate first row
c*---------------------------------------------------------------
      do 10 j=1,n,nd
      d(1,j)=dr(ir,((j-1)/nd+1))
      u(1,j)=ur(ir,((j-1)/nd+1))
      v(1,j)=vr(ir,((j-1)/nd+1))
   10 continue
      if(nd.gt.1) then
      do 12 jj=2,nr
      do 11 j=1,(nd-1)
      jjj=nd*(jj-2)+(j+1)
      d(1,jjj)=(dr(ir,jj)-dr(ir,jj-1))*y(jjj)/dyr
     1+(yr(jj)*dr(ir,jj-1)-yr(jj-1)*dr(ir,jj))/dyr
      u(1,jjj)=(ur(ir,jj)-ur(ir,jj-1))*y(jjj)/dyr
     1+(yr(jj)*ur(ir,jj-1)-yr(jj-1)*ur(ir,jj))/dyr
      v(1,jjj)=(vr(ir,jj)-vr(ir,jj-1))*y(jjj)/dyr
     1+(yr(jj)*vr(ir,jj-1)-yr(jj-1)*vr(ir,jj))/dyr
   11 continue
   12 continue
      endif
c*----------------------------------------------------------------
c*    set number of x points and define x values
c*----------------------------------------------------------------
```

```
      if(ispace.eq.0) then
c*----------------------------------------------------------------
c*    ispace=0, program sets subdivisions
c*----------------------------------------------------------------------
      do 13 j=1,n
      dref(j)=d(1,j)+tide
      if(dref(j).lt.0.001) dref(j)=0.001
   13 continue
      do 27 j=1,n
      xu(j)=u(1,j)
   27 continue
      call vwnum(dref,xu,freqs,xk,eps,nfreqs,n)
      do 29 ifreq=1,nfreqs
      do 28 j=1,n
      k(1,j,ifreq)=xk(j,ifreq)
   28 continue
   29 continue
      do 14 ifreq=1,nfreqs
      npts(ifreq)=0
      sumk(ifreq)=0.
   14 continue
      do 15 ifreq=1,nfreqs
      do 16 j=1,n
      if(d(1,j).gt.0.05) then
        sumk(ifreq)=sumk(ifreq)+k(1,j,ifreq)
        npts(ifreq)=npts(ifreq)+1
      else
      endif
   16 continue
   15 continue
      do 17 ifreq=1,nfreqs
      if(npts(ifreq).eq.0) then
      kb(1,ifreq)=k(1,1,ifreq)
      else
      kb(1,ifreq)=sumk(ifreq)/float(npts(ifreq))
      endif
   17 continue
c*-------------------------------------------------------------------
c*    find maximum wavenumber
c*-------------------------------------------------------------------
      big=kb(1,1)
      do 18 ifreq=2,nfreqs
      if(kb(1,ifreq).gt.big) big=kb(1,ifreq)
   18 continue
      write(*,*)'kb=',big
c*-------------------------------------------------------------------
c*    compute spacing
c*-------------------------------------------------------------------
      alw=2.*pi/big
      anw=dxr/alw
      np=nint(10.*anw)
      if(np.lt.1) np=1
      md(ir)=min((ix-1),np)
      if(np.gt.(ix-1)) write(*,100) np,ir
```

```
      else
c*-------------------------------------------------------------------
c*    ispace=1, user specified subdivision
c*-------------------------------------------------------------------
      endif
      m=md(ir)+1
      dx=dxr/float(md(ir))
      do 19 i=1,m
      x(i)=xr(ir)+float(i-1)*dx
   19 continue
c*---------------------------------------------------------------
c*    interpolate values on m row
c*---------------------------------------------------------------
      do 20 j=1,n,nd
      d(m,j)=dr(ir+1,((j-1)/nd+1))
      u(m,j)=ur(ir+1,((j-1)/nd+1))
      v(m,j)=vr(ir+1,((j-1)/nd+1))
   20 continue
      if(nd.gt.1) then
      do 21 jj=2,nr
      do 22 j=1,(nd-1)
      jjj=nd*(jj-2)+(j+1)
      d(m,jjj)=(dr(ir+1,jj)-dr(ir+1,jj-1))*y(jjj)/dyr
     1+(yr(jj)*dr(ir+1,jj-1)-yr(jj-1)*dr(ir+1,jj))/dyr
      u(m,jjj)=(ur(ir+1,jj)-ur(ir+1,jj-1))*y(jjj)/dyr
     1+(yr(jj)*ur(ir+1,jj-1)-yr(jj-1)*ur(ir+1,jj))/dyr
      v(m,jjj)=(vr(ir+1,jj)-vr(ir+1,jj-1))*y(jjj)/dyr
     1+(yr(jj)*vr(ir+1,jj-1)-yr(jj-1)*vr(ir+1,jj))/dyr
   22 continue
   21 continue
      endif
c*--------------------------------------------------------------------
c*    interpolate values in x-direction
c*--------------------------------------------------------------------
      do 23 i=2,m-1
      do 24 j=1,n
      d(i,j)=(d(m,j)-d(1,j))*x(i)/dxr+(x(m)*d(1,j)-x(1)*d(m,j))/dxr
      u(i,j)=(u(m,j)-u(1,j))*x(i)/dxr+(x(m)*u(1,j)-x(1)*u(m,j))/dxr
      v(i,j)=(v(m,j)-v(1,j))*x(i)/dxr+(x(m)*v(1,j)-x(1)*v(m,j))/dxr
   24 continue
   23 continue
c*--------------------------------------------------------------------
c*    add in user specified grid subdivisions (read from unit iun(2))
c*---------------------------------------------------------------------
      do 30 jr=1,nr-1
      if(isd(ir,jr).eq.1) then
      js=nd*jr+(1-nd)
      jf=js+nd
      read(iun(2),101)((d(i,j),j=js,jf),i=1,m)
      if(icur.eq.1)then
      read(iun(2),101)((u(i,j),j=js,jf),i=1,m)
      read(iun(2),101)((v(i,j),j=js,jf),i=1,m)
      endif
      do 31 i=1,m
```

```
      do 32 j=js,jf
      d(i,j)=d(i,j)*dconv(iu)
      u(i,j)=u(i,j)*dconv(iu)
      v(i,j)=v(i,j)*dconv(iu)
 32   continue
 31   continue
      else
      end if
 30   continue
c*---------------------------------------------------------------------
c*    add tidal offset to all rows and establish thin film
c*---------------------------------------------------------------------
      do 33 i=1,m
      do 34 j=1,n
      d(i,j)=d(i,j)+tide
      if(d(i,j).lt.0.001) d(i,j)=0.001
 34   continue
 33   continue
c*--------------------------------------------------------------------
c*    interpolation complete, return to model
c*--------------------------------------------------------------------
      return
 100  format(' model tried to put more spaces (md=',i4,
     1') than allowed in grid block ',i3)
 101  format(16f8.4)
      end
```

```
c*----------------------------------------------------------------------
c*
      subroutine con(ir,icount,ij)
c*
c*    subroutine calculates constants for row ij in reference grid
c*    block ir.
c*----------------------------------------------------------------------
      parameter(ixr=251,iyr=275,ix=11,iy=275,ncomp=10,
     1nnd=20,nnii=200)
      common/ref1/mr,nr,ispace,nd,md(ixr),iu,dconv(2),if(3),icur,ibc,
     1dconv2(2)
      common/block1/d(ix,iy),u(ix,iy),v(ix,iy),m,n,dx,dy
      common/con1/q(2,iy,ncomp),p(2,iy,ncomp),sig(2,iy,ncomp)
      common/con2/k(2,iy,ncomp),kb(2,ncomp),w(2,iy,nnii),dd(2,iy,ncomp)
      common/wav1/iwave,nfreqs,freqs(ncomp),nwavs,istore(nnii)
     1,nii,fpeak
      common/comp/a(2,iy,nnii),psibar(ncomp),az(iy,nnii),h13(iy)
      dimension akd(2,iy,ncomp),xd(iy),xu(iy),xk(iy,ncomp)
     1,npts(ncomp),sumk(ncomp)
      complex a,w,az
      real k,kb
      eps=1.0e-05
c*----------------------------------------------------------------------
c*    calculate constants
c*----------------------------------------------------------------------
      do 1 j=1,n
      xd(j)=d(icount,j)
      xu(j)=u(icount,j)
    1 continue
      call vwnum(xd,xu,freqs,xk,eps,nfreqs,n)
      do 5 ifreq=1,nfreqs
      do 2 j=1,n
      k(ij,j,ifreq)=xk(j,ifreq)
    2 continue
    5 continue
      do 3 j=1,n
      do 4 ifreq=1,nfreqs
      sig(ij,j,ifreq)=freqs(ifreq)-k(ij,j,ifreq)*u(icount,j)
      akd(ij,j,ifreq)=k(ij,j,ifreq)*d(icount,j)
      q(ij,j,ifreq)=(1.+akd(ij,j,ifreq)/
     1(sinh(akd(ij,j,ifreq))*cosh(akd(ij,j,ifreq))))/2.
      p(ij,j,ifreq)=q(ij,j,ifreq)*9.80621*tanh(akd(ij,j,ifreq))
     1/k(ij,j,ifreq)
      dd(ij,j,ifreq)=(cosh(4.*akd(ij,j,ifreq))+8.-2.*
     1(tanh(akd(ij,j,ifreq))**2))/(8.*(sinh(akd(ij,j,ifreq))**4.))
    4 continue
    3 continue
c*----------------------------------------------------------------------
c*    calculate the dissipation term w
c*----------------------------------------------------------------------
      call diss(ir,icount,ij)
```

```
c*-----------------------------------------------------------------------
c*    calculate the mean kb on each row
c*-----------------------------------------------------------------------
      do 6 ifreq=1,nfreqs
      npts(ifreq)=0
      sumk(ifreq)=0.
    6 continue
      do 7 ifreq=1,nfreqs
      do 8 j=1,n
      if(d(icount,j).gt.0.05) then
      sumk(ifreq)=sumk(ifreq)+k(ij,j,ifreq)
      npts(ifreq)=npts(ifreq)+1
      else
      endif
    8 continue
      if(npts(ifreq).eq.0) then
      kb(ij,ifreq)=k(ij,1,ifreq)
      else
      kb(ij,ifreq)=sumk(ifreq)/float(npts(ifreq))
      endif
    7 continue
      return
      end
```

```
c*-----------------------------------------------------------------
      subroutine fdcalc(ir,icount)
c*
c*    perform the crank-nicolson finite-difference calculations
c*    on grid block ir.  method is the implicit-implicit iteration
c*    used by kirby and dalrymple(1983)
c*-----------------------------------------------------------------
      parameter(ixr=251,iyr=275,ix=11,iy=275,ncomp=10,
     1nnd=20,nnii=200)
      common/ref1/mr,nr,ispace,nd,md(ixr),iu,dconv(2),if(3),icur,ibc,
     1dconv2(2)
      common/ref2/dr(ixr,iyr),ur(ixr,iyr),vr(ixr,iyr),iun(8),iinput,
     1ioutput
      common/ref3/dxr,dyr,xr(ixr),yr(iyr),x(ix),y(iy)
      common/block1/d(ix,iy),u(ix,iy),v(ix,iy),m,n,dx,dy
      common/con1/q(2,iy,ncomp),p(2,iy,ncomp),sig(2,iy,ncomp)
      common/con2/k(2,iy,ncomp),kb(2,ncomp),w(2,iy,nnii),dd(2,iy,ncomp)
      common/wav1/iwave,nfreqs,freqs(ncomp),nwavs,istore(nnii)
     1,nii,fpeak
      common/wav2/amp(ncomp,nnd),dir(ncomp,nnd),tide
      common/comp/a(2,iy,nnii),psibar(ncomp),az(iy,nnii),h13(iy)
      common/nlin/an,anl,ntype
      real k,kb,ksth1,ksth2
      complex a,w,az
      complex first,second,third,fourth,fifth,sixth
      complex c1,c2,c3,cp1,cp2,cp3,ci,damp
      complex ac(iy,nnii),bc(iy,nnii),cc(iy,nnii),rhs(iy,nnii)
     1,sol(iy,nnii),rhso(iy,nnii)
      dimension ucp1(iy),uc(iy),vc(iy),vcp1(iy)
     1,dc(iy),dcp1(iy),ksth1(nnii),ksth2(nnii),s(iy)
      dimension thet(iy,nnii),sxy(iy),sxx(iy),syy(iy),urs(iy,nnii)
c*-----------------------------------------------------------------
c*    statement functions for crank-nicolson coefficients
c*-----------------------------------------------------------------
      cg(i,j,ifreq)=sqrt(p(i,j,ifreq)*q(i,j,ifreq))
c*-----------------------------------------------------------------
      pv(i,j,ifreq)=p(i,j,ifreq)-vc(j)*vc(j)
      pvp1(i,j,ifreq)=p(i+1,j,ifreq)-vcp1(j)*vcp1(j)
c*-----------------------------------------------------------------
      bet(i,j,ifreq)=-4.*(k(i+1,j,ifreq)-k(i,j,ifreq))
     1/(dx*((k(i+1,j,ifreq)+k(i,j,ifreq))**2))
     1-4.*(k(i+1,j,ifreq)*(p(i+1,j,ifreq)-ucp1(j)**2)
     1-k(i,j,ifreq)*(p(i,j,ifreq)-uc(j)**2))
     1/(dx*((k(i+1,j,ifreq)+k(i,j,ifreq))**2.)
     1*(p(i+1,j,ifreq)+p(i,j,ifreq)-(ucp1(j)**2.+uc(j)**2)))
c*-----------------------------------------------------------------
      dv(i,j,ifreq)=(cg(i+1,j,ifreq)+ucp1(j))/sig(i+1,j,ifreq)
     1-(cg(i,j,ifreq)+uc(j))/sig(i,j,ifreq)
     1+(-delta1)*dx*((vcp1(j+1)/sig(i+1,j+1,ifreq))
     1+(vc(j+1)/sig(i,j+1,ifreq))
     1-(vcp1(j-1)/sig(i+1,j-1,ifreq))
```

```
     1-(vc(j-1)/sig(i,j-1,ifreq)))/(2.*dy)
c*-----------------------------------------------------------------
      damp(i,j,ifreq)=2.*ci*cdamp*((cg(i+1,j,ifreq)+ucp1(j))
     1+(cg(i,j,ifreq)+uc(j)))/(dy*dy*(k(i+1,j,ifreq)**2
     1+k(i,j,ifreq)**2))
c*-----------------------------------------------------------------
      deltap(i,j,ifreq)=a1-b1*kb(i,ifreq)/k(i,j,ifreq)
c*-----------------------------------------------------------------
      hrms(j)=h13(j)/(sqrt(2.))
c*-----------------------------------------------------------------
      alpha1(j)=0.75*sqrt(pi)*(b**3)
     1*fpeak*(hrms(j)**5)/((gam**4)*((0.5*(dcp1(j)+dc(j)))**5))
c*-----------------------------------------------------------------
c     alpha2(j)=0.75*sqrt(pi)*(b**3)
c    1*fpeak*(hrms(j)**5)/((gam**4)*(dc(j)**5))
c*-----------------------------------------------------------------
      first(i,j,ifreq)=(cg(i+1,j,ifreq)+ucp1(j))
     1*cmplx(1.,dx*(kb(i+1,ifreq)-a0*k(i+1,j,ifreq)))
     1+cmplx(1.,0.)*(cg(i,j,ifreq)+uc(j)
     1+dv(i,j,ifreq)*(sig(i+1,j,ifreq)+sig(i,j,ifreq))/4.)
     1+2.*freqs(ifreq)*cmplx(0.,1.)*(-b1)*bet(i,j,ifreq)
     1*(ucp1(j)+uc(j))/sig(i+1,j,ifreq)
     1+4.*freqs(ifreq)*(-b1)*cmplx(0.,1.)*
     1(3.*(ucp1(j)-uc(j))/dx+(vcp1(j+1)+vc(j+1)-vcp1(j-1)-vc(j-1))
     1/(4.*dy))/(sig(i+1,j,ifreq)*(k(i+1,j,ifreq)+k(i,j,ifreq)))
     1+cmplx(-2.*(-b1)/(dy*dy*(k(i+1,j,ifreq)+k(i,j,ifreq)))
     1+b1*bet(i,j,ifreq)*dx/(2.*dy*dy),(-deltap(i,j,ifreq))*dx
     1/(2.*dy*dy))*(pvp1(i,j+1,ifreq)+2.*pvp1(i,j,ifreq)
     1+pvp1(i,j-1,ifreq))/sig(i+1,j,ifreq)
      cp1(i,j,ifreq)=first(i,j,ifreq)
     1-cmplx(1.,0.)*freqs(ifreq)*delta2*(3.*ucp1(j)+uc(j))
     1/(2.*sig(i+1,j,ifreq))+ci*freqs(ifreq)*(a0-1.)*k(i+1,j,ifreq)
     1*ucp1(j)*dx/sig(i+1,j,ifreq)+2.*ifilt*damp(i,j,ifreq)
     1+cmplx(1.,0.)*alpha1(j)*dx
c*------------------------------------------------------------------
      second(i,j,ifreq)=cmplx((-delta1)*dx*(vcp1(j)+vc(j))/(2.*dy)
     1+b1*u2*bet(i,j,ifreq)*(ucp1(j)*vcp1(j)+uc(j)*vc(j))
     1/(dy*sig(i+1,j+1,ifreq)),(-delta1*u2)*(ucp1(j+1)*vcp1(j+1)
     1+uc(j+1)*vc(j+1)+2.*ucp1(j)*vcp1(j))/(2.*dy*sig(i+1,j+1,ifreq))
     1+dx*(-b1)*bet(i,j,ifreq)*(sig(i+1,j,ifreq)*vcp1(j)
     1+sig(i,j,ifreq)*vc(j))/(2.*dy*sig(i+1,j+1,ifreq)))
     1+cmplx(2.*(-b1)/(dy*dy*(k(i+1,j,ifreq)+k(i,j,ifreq)))
     1+(-b1)*bet(i,j,ifreq)*dx/(2.*dy*dy),-(-deltap(i,j,ifreq)*dx)
     1/(2.*dy*dy))*(pvp1(i,j+1,ifreq)+pvp1(i,j,ifreq))
     1/sig(i+1,j+1,ifreq)
     1+4.*cmplx(0.,1.)*(-b1)*sig(i+1,j,ifreq)*vcp1(j)
     1/(dy*sig(i+1,j+1,ifreq)*(k(i+1,j,ifreq)+k(i,j,ifreq)))
      cp2(i,j,ifreq)=second(i,j,ifreq)
     1-ifilt*damp(i,j,ifreq)
c*------------------------------------------------------------------
      third(i,j,ifreq)=cmplx(-(-delta1)*dx*(vcp1(j)+vc(j))/(2.*dy)
     1+(-b1)*u2*bet(i,j,ifreq)*(ucp1(j)*vcp1(j)+uc(j)*vc(j))
     1/(dy*sig(i+1,j-1,ifreq)),-(-delta1)*u2*(ucp1(j-1)*vcp1(j-1)
     1+uc(j-1)*vc(j-1)+2.*ucp1(j)*vcp1(j))
```

```
     1/(2.*dy*sig(i+1,j-1,ifreq))
     1-dx*(-b1)*bet(i,j,ifreq)*(sig(i+1,j,ifreq)*vcp1(j)
     1+sig(i,j,ifreq)*vc(j))/(2.*dy*sig(i+1,j-1,ifreq)))
     1+cmplx(2.*(-b1)/(dy*dy*(k(i+1,j,ifreq)+k(i,j,ifreq)))
     1-b1*bet(i,j,ifreq)*dx/(2.*dy*dy),-(-deltap(i,j,ifreq)*dx)
     1/(2.*dy*dy))*(pvp1(i,j,ifreq)+pvp1(i,j-1,ifreq))
     1/sig(i+1,j-1,ifreq)-4.*cmplx(0.,1.)*(-b1)*sig(i+1,j,ifreq)
     1*vcp1(j)/(dy*sig(i+1,j-1,ifreq)*(k(i+1,j,ifreq)+k(i,j,ifreq)))
      cp3(i,j,ifreq)=third(i,j,ifreq)
     1-ifilt*damp(i,j,ifreq)
c*--------------------------------------------------------------------
      fourth(i,j,ifreq)=cmplx(cg(i+1,j,ifreq)+ucp1(j)-
     1dv(i,j,ifreq)*(sig(i+1,j,ifreq)+sig(i,j,ifreq))/4.,0.)
     1+cmplx(1.,-dx*(kb(i,ifreq)-a0*k(i,j,ifreq)))
     1*(cg(i,j,ifreq)+uc(j))+2.*cmplx(0.,1.)*freqs(ifreq)*(-b1)
     1*bet(i,j,ifreq)*(ucp1(j)+uc(j))/sig(i,j,ifreq)
     1+4.*cmplx(0.,1.)*freqs(ifreq)*(-b1)*(3.*(ucp1(j)-uc(j))/dx
     1+(vcp1(j+1)+vc(j+1)-vcp1(j-1)-vc(j-1))
     1/(4.*dy))/(sig(i,j,ifreq)*(k(i+1,j,ifreq)+k(i,j,ifreq)))
     1+cmplx(-2.*(-b1)/(dy*dy*(k(i+1,j,ifreq)+k(i,j,ifreq)))
     1+(-b1)*bet(i,j,ifreq)*dx/(2.*dy*dy),-(-deltap(i,j,ifreq)*dx)
     1/(2.*dy*dy))*(pv(i,j+1,ifreq)
     1+2.*pv(i,j,ifreq)+pv(i,j-1,ifreq))/sig(i,j,ifreq)
      c1(i,j,ifreq)=fourth(i,j,ifreq)
     1-cmplx(1.,0.)*freqs(ifreq)*delta2*(3.*ucp1(j)+uc(j))
     1/(2.*sig(i,j,ifreq))-ci*freqs(ifreq)*(a0-1.)*k(i,j,ifreq)
     1*uc(j)*dx/sig(i,j,ifreq)+2.*ifilt*damp(i,j,ifreq)
     1-cmplx(1.,0.)*alpha1(j)*dx
c*---------------------------------------------------------------------
      fifth(i,j,ifreq)=cmplx(-(-delta1)*dx*(vcp1(j)+vc(j))/(2.*dy)+b1*u2
     1*bet(i,j,ifreq)*(ucp1(j)*vcp1(j)+uc(j)*vc(j))/
     1(dy*sig(i,j+1,ifreq)),(-delta1)*u2*(ucp1(j+1)*vcp1(j+1)
     1+uc(j+1)*vc(j+1)+2.*uc(j)*vc(j))/(2.*dy*sig(i,j+1,ifreq))
     1+4.*(-b1)*sig(i,j,ifreq)*vc(j)/(dy*(k(i+1,j,ifreq)
     1+k(i,j,ifreq))*sig(i,j+1,ifreq))
     1-dx*(-b1)*bet(i,j,ifreq)*(sig(i+1,j,ifreq)*vcp1(j)
     1+sig(i,j,ifreq)*vc(j))/(2.*dy*sig(i,j+1,ifreq)))
     1+cmplx(2.*(-b1)/(dy*dy*(k(i+1,j,ifreq)+k(i,j,ifreq)))+b1
     1*bet(i,j,ifreq)*dx/(2.*dy*dy),(-deltap(i,j,ifreq))*dx
     1/(2.*dy*dy))
     1*(pv(i,j+1,ifreq)+pv(i,j,ifreq))/sig(i,j+1,ifreq)
      c2(i,j,ifreq)=fifth(i,j,ifreq)
     1-ifilt*damp(i,j,ifreq)
c*---------------------------------------------------------------------
      sixth(i,j,ifreq)=cmplx((-delta1)*dx*(vcp1(j)+vc(j))/(2.*dy)
     1+(-b1)*u2*bet(i,j,ifreq)*(ucp1(j)*vcp1(j)+uc(j)*vc(j))
     1/(dy*sig(i,j-1,ifreq)),-(-delta1)*u2
     1*(ucp1(j-1)*vcp1(j-1)+uc(j-1)*vc(j-1)+2.*uc(j)*vc(j))
     1/(2.*dy*sig(i,j-1,ifreq))-4.*(-b1)*sig(i,j,ifreq)
     1*vc(j)/(dy*(k(i+1,j,ifreq)+k(i,j,ifreq))*sig(i,j-1,ifreq))
     1+dx*(-b1)*bet(i,j,ifreq)*(sig(i+1,j,ifreq)*vcp1(j)
     1+sig(i,j,ifreq)*vc(j))/(2.*dy*sig(i,j-1,ifreq)))
     1+cmplx(2.*(-b1)/(dy*dy*(k(i+1,j,ifreq)+k(i,j,ifreq)))-(-b1)
     1*bet(i,j,ifreq)*dx/(2.*dy*dy),(-deltap(i,j,ifreq))*dx
```

```
     1/(2.*dy*dy))*(pv(i,j,ifreq)+pv(i,j-1,ifreq))/sig(i,j-1,ifreq)
      c3(i,j,ifreq)=sixth(i,j,ifreq)
     1-ifilt*damp(i,j,ifreq)
c*--------------------------------------------------------------------
      f1(i,j,ifreq)=tanh(k(i,j,ifreq)*dc(j))**5.
      f1p1(i,j,ifreq)=tanh(k(i+1,j,ifreq)*dcp1(j))**5.
c*--------------------------------------------------------------------
      f2(i,j,ifreq)=(k(i,j,ifreq)*dc(j)
     1/sinh(k(i,j,ifreq)*dc(j)))**4.
      f2p1(i,j,ifreq)=(k(i+1,j,ifreq)*dcp1(j)
     1/sinh(k(i+1,j,ifreq)*dcp1(j)))**4.

c    Booij coefficients.

      a0=1.
      a1=-0.75
      b1=-0.25
c*---------------------------------------------------------------------
c*    constants
c*---------------------------------------------------------------------
      do 1 j=1,n
      dc(j)=d(icount-1,j)
      uc(j)=u(icount-1,j)
      vc(j)=v(icount-1,j)
      dcp1(j)=d(icount,j)
      ucp1(j)=u(icount,j)
      vcp1(j)=v(icount,j)
    1 continue
      u2=1.0
      b=1.
      gam=0.6
      g=9.80621
      pi=3.1415927
      rho=1000.
      ci=cmplx(0.,1.)
      cdamp=0.0025
      ifilt=1
      delta1=a1-b1
      delta2=1+2.*a1-2.*b1
      nii=nfreqs*nwavs
      it=0
c*---------------------------------------------------------------------
c*    setup r.h.s.
c*---------------------------------------------------------------------
      do 2 ii=1,nii
      rhs(1,ii)=cmplx(0.,0.)
      do 3 j=2,(n-1)
      rhso(j,ii)=c1(1,j,istore(ii))*a(1,j,ii)+c2(1,j,istore(ii))
     1*a(1,j+1,ii)+c3(1,j,istore(ii))*a(1,j-1,ii)
      rhs(j,ii)=rhso(j,ii)
     1-dx*w(1,j,ii)*a(1,j,ii)/2.-dx*cmplx(0.,1.)*an*anl
     1*sig(1,j,istore(ii))*k(1,j,istore(ii))*k(1,j,istore(ii))
     1*dd(1,j,istore(ii))*(cabs(a(1,j,ii))**2.)
     1*a(1,j,ii)/2.-dx*cmplx(0.,1.)*an*(1.-anl)
```

```
     1*sig(1,j,istore(ii))*((1.+f1(1,j,istore(ii))*k(1,j,istore(ii))
     1*k(1,j,istore(ii))*(cabs(a(1,j,ii))**2.)
     1*dd(1,j,istore(ii)))*tanh(k(1,j,istore(ii))*dc(j)
     1+f2(1,j,istore(ii))*k(1,j,istore(ii))*0.5*h13(j))
     1/tanh(k(1,j,istore(ii))*dc(j))-1.)*a(1,j,ii)/2.
    3 continue
      rhs(n,ii)=cmplx(0.,0.)
    2 continue
c*-----------------------------------------------------------------------
c*    return here for iterations
c*-----------------------------------------------------------------------
  20  it=it+1
      if(it.eq.1)iii=1
      if(it.eq.2)iii=2
c*-----------------------------------------------------------------------
c*    establish boundary conditions
c*-----------------------------------------------------------------------
      if(ibc.eq.1)then
      do 4 ii=1,nii
      ksth1(ii)=real((2.*(a(1,2,ii)-a(1,1,ii))/((a(1,2,ii)
     1+a(1,1,ii))*dy))*cmplx(0.,-1.))
      ksth2(ii)=real((2.*(a(1,n,ii)-a(1,n-1,ii))/((a(1,n,ii)
     1+a(1,n-1,ii))*dy))*cmplx(0.,-1.))
      bc(1,ii)=cmplx(1.,ksth1(ii)*dy/2.)
      cc(1,ii)=-cmplx(1.,-ksth1(ii)*dy/2.)
      bc(n,ii)=-cmplx(1.,-ksth2(ii)*dy/2.)
      ac(n,ii)=cmplx(1.,ksth2(ii)*dy/2.)
    4 continue
      else
      do 5 ii=1,nii
      bc(1,ii)=cmplx(1.,0.)
      cc(1,ii)=-bc(1,ii)
      bc(n,ii)=cmplx(1.,0.)
      ac(n,ii)=-bc(n,ii)
    5 continue
      endif
c*----------------------------------------------------------------------
c*    coefficients for forward row
c*----------------------------------------------------------------------
      do 6 ii=1,nii
      do 7 j=2,(n-1)
      ac(j,ii)=cp3(1,j,istore(ii))
      bc(j,ii)=cp1(1,j,istore(ii))+(dx/2.)*(w(2,j,ii))
     1+cmplx(0.,an*anl)*sig(2,j,istore(ii))*k(2,j,istore(ii))
     1*k(2,j,istore(ii))*dd(2,j,istore(ii))*(cabs(a(iii,j,ii))**2.)
     1*(dx/2.)+cmplx(0.,an*(1.-anl))
     1*sig(2,j,istore(ii))*(dx/2.)*((1.+f1p1(1,j,istore(ii))*
     1k(2,j,istore(ii))*k(2,j,istore(ii))*(cabs(a(iii,j,ii))**2.)
     1*dd(2,j,istore(ii)))*tanh(k(2,j,istore(ii))
     1*dcp1(j)+f2p1(1,j,istore(ii))*k(2,j,istore(ii))
     1*0.5*h13(j))/tanh(k(2,j,istore(ii))*dcp1(j))-1.)
      cc(j,ii)=cp2(1,j,istore(ii))
    7 continue
    6 continue
```

```
c*---------------------------------------------------------------------
c*    update solution one step
c*---------------------------------------------------------------------
      call vtrida(1,n,ac,bc,cc,rhs,sol,nii)
      do 8 ii=1,nii
      do 9 j=1,n
      a(2,j,ii)=sol(j,ii)
      sol(j,ii)=cmplx(0.,0.)
    9 continue
    8 continue
      if(it.eq.1) go to 20
c*---------------------------------------------------------------------
c*    Check Ursell parameter for Stokes solution
c*---------------------------------------------------------------------
      if(ntype.eq.2)then
      do 23 j=1,n
      do 24 ii=1,nii
      urs(j,ii)=(cabs(a(2,j,ii))/dcp1(j))
     1/((k(2,j,istore(ii))*dcp1(j))**2)
      if(urs(j,ii).gt.0.5) write(*,204)urs(j,ii),icount,j,ii
  24  continue
  23  continue
      end if
c*---------------------------------------------------------------------
c*    calculate reference phase function
c*---------------------------------------------------------------------
      do 10 ifreq=1,nfreqs
      psibar(ifreq)=psibar(ifreq)+(kb(2,ifreq)+kb(1,ifreq))*dx/2.
   10 continue
c*---------------------------------------------------------------------
c*    calculate significant waveheight h13
c*---------------------------------------------------------------------
      do 25 j=1,n
      s(j)=0.
      do 26 ii=1,nii
      s(j)=s(j)+(cabs(a(2,j,ii))**2)
   26 continue
      h13(j)=sqrt(8.*s(j))
   25 continue
      if(icount.eq.m) then
c*-------------------------------------------------------------------------
c Calculate wave angles at reference grid rows.   Note:  angles are not well
c defined in a directional, multicomponent sea, or where waves become short
c crested.  This routine was heavily modified by Raul Medina, University of
c Cantabria.
c*-------------------------------------------------------------------------
      do 16 ii=1,nii
      do 15 j=1,n
      if(a(2,j,ii).eq.(0.,0.)) then
          akx2=0.
      else
          akx2=aimag(clog(a(2,j,ii)))
      endif
      if(a(1,j,ii).eq.(0.,0.)) then
```

```
          akx1=0.
      else
          akx1=aimag(clog(a(1,j,ii)))
      endif
      if(abs(akx2-akx1).gt.pi)then
      akx=sign((2.*pi-(abs(akx1)+abs(akx2)))/dx,akx1)
      else
      akx=(akx2-akx1)/dx
      endif
      if(j.ne.n) then
          if(a(2,j+1,ii).eq.(0.,0.)) then
              aky2=0.
          else
              aky2=aimag(clog(a(2,j+1,ii)))
          endif
          if(a(2,j,ii).eq.(0.,0.)) then
              aky1=0.
          else
              aky1=aimag(clog(a(2,j,ii)))
          endif
      else
          if(a(2,j,ii).eq.(0.,0.)) then
              aky2=0.
          else
              aky2=aimag(clog(a(2,j,ii)))
          endif
          if(a(2,j-1,ii).eq.(0.,0.))then
              aky1=0.
          else
              aky1=aimag(clog(a(2,j-1,ii)))
          endif
      endif
      if(abs(aky2-aky1).gt.pi)then
          aky=sign((2.*pi-(abs(aky1)+abs(aky2)))/dy,aky1)
      else
          aky=(aky2-aky1)/dy
      endif
      thet(j,ii)=atan2(aky,(akx+kb(2,istore(ii))))
 15   continue
 16   continue
c*-------------------------------------------------------------------
c*    Estimation of radiation stresses
c*-------------------------------------------------------------------
      do 17 j=1,n
      sxx(j)=0.
      syy(j)=0.
      sxy(j)=0.
   17 continue
      do 22 j=1,n
      do 18 ii=1,nii
      sxx(j)=sxx(j)+(cabs(a(2,j,ii))**2)
     1*(q(2,j,istore(ii))*(1+(cos(thet(j,ii))**2))-0.5)
      syy(j)=syy(j)+(cabs(a(2,j,ii))**2)
     1*(q(2,j,istore(ii))*(1+(sin(thet(j,ii))**2))-0.5)
```

```
      sxy(j)=sxy(j)+q(2,j,istore(ii))
     1*(cabs(a(2,j,ii))**2)*sin(thet(j,ii))
   18 continue
      sxx(j)=sxx(j)*rho*g/2.
      syy(j)=syy(j)*rho*g/2.
      sxy(j)=sxy(j)*rho*g/4.
   22 continue
c*----------------------------------------------------------------------
c*    Line printer output
c*----------------------------------------------------------------------
      mm1=m-1
      write(*,205) (ir+1),mm1
      write(*,202) x(m)/dconv(iu)
      write(*,206)
      write(*,203) (h13(j)/dconv(iu),j=1,n,nd)
c*----------------------------------------------------------------------
c*    Output of radiation stresses
c*----------------------------------------------------------------------
      write(22,*) (sxx(j)*dconv2(iu),j=1,n,nd)
      write(22,*) (sxy(j)*dconv2(iu),j=1,n,nd)
      write(22,*) (syy(j)*dconv2(iu),j=1,n,nd)
      endif
c*----------------------------------------------------------------------
c*    return control back to model
c*----------------------------------------------------------------------
      return
 202  format(' x=',f10.2)
 203  format(' ',12(f10.4))
 204  format(' '//' Warning: Ursell number =',f10.4,' encountered at','g
     1rid location',i6,',',i6/' during computation of wave component',i3,
     1'should be using Stokes-Hedges model
     1(ntype=1) due to shallow','water')
 205  format(' grid row ir=',i3,',  ',i3,' x-direction subdivisions',
     1' used')
 206  format(' Significant Wave Height')

      end
```

```
c*------------------------------------------------------------------
      subroutine rbcon
c*
c*    rolls back constants and solution to row 1
c*------------------------------------------------------------------
      parameter(ixr=251,iyr=275,ix=11,iy=275,ncomp=10,
     1nnd=20,nnii=200)
      common/ref1/mr,nr,ispace,nd,md(ixr),iu,dconv(2),if(3),icur,ibc,
     1dconv2(2)
      common/block1/d(ix,iy),u(ix,iy),v(ix,iy),m,n,dx,dy
      common/con1/q(2,iy,ncomp),p(2,iy,ncomp),sig(2,iy,ncomp)
      common/con2/k(2,iy,ncomp),kb(2,ncomp),w(2,iy,nnii),dd(2,iy,ncomp)
      common/wav1/iwave,nfreqs,freqs(ncomp),nwavs,istore(nnii)
     1,nii,fpeak
      common/comp/a(2,iy,nnii),psibar(ncomp),az(iy,nnii),h13(iy)
      dimension akd(2,iy,ncomp)
      complex a,w,az
      real k,kb
      do 1 j=1,n
      do 2 ifreq=1,nfreqs
      k(1,j,ifreq)=k(2,j,ifreq)
      sig(1,j,ifreq)=sig(2,j,ifreq)
      akd(1,j,ifreq)=akd(2,j,ifreq)
      q(1,j,ifreq)=q(2,j,ifreq)
      p(1,j,ifreq)=p(2,j,ifreq)
      dd(1,j,ifreq)=dd(2,j,ifreq)
      k(2,j,ifreq)=0.
      sig(2,j,ifreq)=0.
      akd(2,j,ifreq)=0.
      q(2,j,ifreq)=0.
      p(2,j,ifreq)=0.
      dd(2,j,ifreq)=0.
    2 continue
    1 continue
      do 3 ifreq=1,nfreqs
      kb(1,ifreq)=kb(2,ifreq)
      kb(2,ifreq)=0.
    3 continue
c*------------------------------------------------------------------
c*    roll back solution
c*------------------------------------------------------------------
      do 4 j=1,n
      do 5 ii=1,nii
      a(1,j,ii)=a(2,j,ii)
      a(2,j,ii)=cmplx(0.,0.)
    5 continue
    4 continue
      return
      end
```

```
c*-----------------------------------------------------------------
      subroutine wnum(d,u,freqs,kk,eps,nfreqs)
c*    note: first guess is still to be changed to fenton's
c*
c*    calculate wavenumber k according to the form
c*
c*        s*s-2*s*k*u+k*k*u*u=g*k*tanh(k*d)
c*
c*    where
c*
c*        d   = local water depth
c*        s   = absolute frequency
c*        g   = gravitational acceleration constant
c*        u   = x-component of ambient current
c*        eps = tolerance for iteration convergence
c*        i,j = indices in finite-difference grid
c*
c*    solution by newton-raphson iteration using eckart's
c*    approximation as a seed value.
c*-----------------------------------------------------------------
      parameter(ixr=251,iyr=275,ix=11,iy=275,ncomp=10,
     1nnd=20,nnii=200)
c     common/ref2/dr(ixr,iyr),ur(ixr,iyr),vr(ixr,iyr),iun(8),iinput,
c    1ioutput
      dimension freqs(ncomp),f(ncomp),fp(ncomp)
      real kk(ncomp),kn(ncomp)
      g=9.806
      pi=3.1415927
c*-----------------------------------------------------------------
c*    calculate first guess
c*-----------------------------------------------------------------
      do 1 ifreq=1,nfreqs
      kk(ifreq)=freqs(ifreq)*freqs(ifreq)/
     1(g*sqrt(tanh(freqs(ifreq)*freqs(ifreq)*d/g)))
    1 continue
c*-----------------------------------------------------------------
c*    iteration
c*-----------------------------------------------------------------
      do 2 ifreq=1,nfreqs
      do 3 i=1,40
      f(ifreq)=freqs(ifreq)**2.-2.*freqs(ifreq)*kk(ifreq)*u+
     1(kk(ifreq)*u)**2.-g*kk(ifreq)*tanh(kk(ifreq)*d)
      fp(ifreq)=-2.*freqs(ifreq)*u+2.*kk(ifreq)*(u**2.)-
     1g*tanh(kk(ifreq)*d)-g*kk(ifreq)*d/(cosh(kk(ifreq)*d)**2.)
      kn(ifreq)=kk(ifreq)-f(ifreq)/fp(ifreq)
      if((abs(kn(ifreq)-kk(ifreq))/kn(ifreq)).lt.eps) go to 4
      kk(ifreq)=kn(ifreq)
    3 continue
      write(*,*)'Wavenumber failed to converge for frequency component'
     1,ifreq,'on point (1,1)'
    4 kk(ifreq)=kn(ifreq)
```

```
 2 continue
   return
   end
```

```
c*-----------------------------------------------------------------
      subroutine vwnum(dt,u,freqs,k,eps,nfreqs,n)
c*    note: first guess is still to be changed to fentan's
c*
c*    vectorized wavenumber calculations. variable definitions
c*    same as in subroutine wnum.
c*-----------------------------------------------------------------
      parameter(ixr=251,iyr=275,ix=11,iy=275,ncomp=10,
     1nnd=20,nnii=200)
c     common/ref2/dr(ixr,iyr),ur(ixr,iyr),vr(ixr,iyr),iun(8),iinput,
c    1ioutput
      dimension dt(iy),u(iy),freqs(ncomp),f(iy,ncomp),
     1fp(iy,ncomp)
      real k(iy,ncomp),kn(iy,ncomp)
      g=9.806
      pi=3.1415927
c*-----------------------------------------------------------------
c*    calculate first guess
c*-----------------------------------------------------------------
      do 1 ifreq=1,nfreqs
      do 2 j=1,n
      k(j,ifreq)=freqs(ifreq)*freqs(ifreq)/
     1(g*sqrt(tanh(freqs(ifreq)*freqs(ifreq)*dt(j)/g)))
    2 continue
    1 continue
c*-----------------------------------------------------------------
c*    iteration
c*-----------------------------------------------------------------
      do 4 ifreq=1,nfreqs
      do 5 j=1,n
      do 3 i=1,40
      f(j,ifreq)=freqs(ifreq)**2-2.*freqs(ifreq)*k(j,ifreq)*u(j)+
     1(k(j,ifreq)*u(j))**2-g*k(j,ifreq)*tanh(k(j,ifreq)*dt(j))
      fp(j,ifreq)=-2.*freqs(ifreq)*u(j)+2.*k(j,ifreq)*(u(j)**2)-
     1g*tanh(k(j,ifreq)*dt(j))-g*k(j,ifreq)*dt(j)
     1/(cosh(k(j,ifreq)*dt(j))**2.)
      kn(j,ifreq)=k(j,ifreq)-f(j,ifreq)/fp(j,ifreq)
      if((abs(kn(j,ifreq)-k(j,ifreq))/kn(j,ifreq)).lt.eps) go to 10
      k(j,ifreq)=kn(j,ifreq)
    3 continue
      write(*,*)'Wavenumber failed to converge for frequency component'
     1,ifreq,'on column',j
   10 k(j,ifreq)=kn(j,ifreq)
    5 continue
    4 continue
      return
      end
```

```
c*-----------------------------------------------------------------------
c*
      subroutine vtrida(if,l,a,b,c,d,v,mm)
c*
c*    vectorized tridiagonal matrix solution by double sweep algorithm.
c*    present subroutine adopted from the subroutine described in:
c*
c*        carnahan, luther and wilkes, applied numerical
c*        methods, wiley, 1969
c*
c*    modified to handle complex array coefficients and solution
c*    values.  input and output are
c*
c*        a,b,c  = coefficients of row in tridiagonal matrix
c*        d      = right hand side vector of matrix equation
c*        v      = solution vector
c*        if,l   = beginning and end indices of positions in the
c*                 dimensioned range of the column vector to be
c*                 considered.

c*-----------------------------------------------------------------------
      parameter(iy=275,nnii=200)
      complex a(iy,nnii),b(iy,nnii),c(iy,nnii),d(iy,nnii)
     1,v(iy,nnii),beta(iy,nnii),gamma(iy,nnii)
c*-----------------------------------------------------------------------
c*    compute intermediate vectors beta and gamma
c*-----------------------------------------------------------------------
      do 1 j=1,mm
      beta(if,j)=b(if,j)
      gamma(if,j)=d(if,j)/beta(if,j)
    1 continue
      ifp1=if+1
      do 2 i=ifp1,l
      do 3 j=1,mm
      beta(i,j)=b(i,j)-a(i,j)*c(i-1,j)/beta(i-1,j)
      gamma(i,j)=(d(i,j)-a(i,j)*gamma(i-1,j))/beta(i,j)
    3 continue
    2 continue
c*-----------------------------------------------------------------------
c*    compute solution vector v
c*-----------------------------------------------------------------------
      do 4 j=1,mm
      v(l,j)=gamma(l,j)
    4 continue
      last=l-if
      do 5 k=1,last
      i=l-k
      do 6 j=1,mm
      v(i,j)=gamma(i,j)-c(i,j)*v(i+1,j)/beta(i,j)
    6 continue
    5 continue
```

```
return
end
```

```
c*-----------------------------------------------------------------
c*
      subroutine diss(ir,icount,i)
c*
c*    subroutine calculates the dissipation at a single grid point
c*    based on values of the switch iw at that point.
c*-----------------------------------------------------------------
      parameter(ixr=251,iyr=275,ix=11,iy=275,ncomp=10,
     1nnd=20,nnii=200)
      common/ref1/mr,nr,ispace,nd,md(ixr),iu,dconv(2),if(3),icur,ibc,
     1dconv2(2)
      common/con1/q(2,iy,ncomp),p(2,iy,ncomp),sig(2,iy,ncomp)
      common/con2/k(2,iy,ncomp),kb(2,ncomp),w(2,iy,nnii),dd(2,iy,ncomp)
      common/block1/d(ix,iy),u(ix,iy),v(ix,iy),m,n,dx,dy
      common/wav1/iwave,nfreqs,freqs(ncomp),nwavs,istore(nnii)
     1,nii,fpeak
      common/comp/a(2,iy,nnii),psibar(ncomp),az(iy,nnii),h13(iy)
      real k,kb,nu,cp,kd(iy,nnii)
      complex w,a,az
      sq(i,j,ifreq)=sqrt(nu/(2.*sig(i,j,ifreq)))
      nu=1.3e-06
      cp=4.5e-11
      g=9.80621
      pi=3.1415927
c*-----------------------------------------------------------------
c*    value of f here is value assuming tau=(f/8)*u**2
c*    f=4.*fw; fw=wave friction factor
      f=0.01*4.0
c*-----------------------------------------------------------------
      do 1 j=1,n
      do 2 ii=1,nii
      w(i,j,ii)=cmplx(0.,0.)
      kd(j,ii)=k(i,j,ii)*d(icount,j)
c*------------------------------------------------------------------
c*    if if(1) = 1, use turbulent boundary layer damping
c*------------------------------------------------------------------
      if(if(1).eq.1) w(i,j,ii)=2.*f*cabs(az(j,ii))
     1*sig(i,j,istore(ii))*k(i,j,istore(ii))/(sinh(2.*kd(j,ii))
     1*sinh(kd(j,ii))*3.*pi)
c*----------------------------------------------------------------
c*    if(if(2) = 1, add porous bottom damping
c*----------------------------------------------------------------
      if(if(2).eq.1) w(i,j,ii)=w(i,j,ii)+(g*k(i,j,istore(ii))
     1*cp/(nu*(cosh(kd(j,ii))**2)))*cmplx(1.,0.)
c*----------------------------------------------------------------
c*    if if(3) = 1, add boundary layer damping
c*---------------------------------------------------------------
      if(if(3).eq.1) w(i,j,ii)=w(i,j,ii)+2.*k(i,j,istore(ii))
     1*sig(i,j,istore(ii))*sq(i,j,istore(ii))
     1*(1.+(cosh(kd(j,ii))**2))*cmplx(1.,-1.)/sinh(2.*kd(j,ii))
    2 continue
```

```
 1 continue
   return
   end
```

6 Appendix B: PLOTAMP Program Listing

The following section contains the listing of the program **PLOTAMP** as an aid to read the stored output file *outdat.dat*. This program was used to create contour plots of normalized significant wave height in the example cases shown in this manual.

```
c*-----------------------------------------------------------------
c*    plotamp.f
c*
c*    Plot significant wave height field constructed from output
c*    in 'outdat.dat'  from a REF/DIF S model.
c*-----------------------------------------------------------------
      dimension wave(251,301),h(251,301),x(251),y(301)
      dimension hs(251,301)
      open(10,file='outdat.dat')
      read(10,*)n,m
      write(*,*)' grid dimensions are ',m,' by ',n
      read(10,*)(y(j),j=1,n)
      do 1 i=1,m
      read(10,*)x(i)
      read(10,*)(h(i,j),j=1,n)
      read(10,*)(hs(i,j),j=1,n)
 1    continue
      close(10)
      h0=hs(1,1)
      hh=h(1,1)
c*------------------------------------------------------------------
c*    construct normalized wave height field
c*------------------------------------------------------------------
      do 3 i=1,m
      do 2 j=1,n
      wave(i,j)=hs(i,j)/h0
      h(i,j)=h(i,j)/hh
 2    continue
 3    continue
c*------------------------------------------------------------------
c*    enter corner points for plotted grid
c*------------------------------------------------------------------
      write(*,*)' enter imin,imax,jmin,jmax'
      read(*,*)imin,imax,jmin,jmax
      mnew=(imax-imin)+1
      nnew=(jmax-jmin)+1
      do 5 i=1,mnew
      ii=(i+imin)-1
      do 4 j=1,nnew
      jj=(j+jmin)-1
      wave(i,j)=wave(ii,jj)
      h(i,j)=h(ii,jj)
 4    continue
 5    continue
c*------------------------------------------------------------------
c*    contour the normalized wave height field
c*------------------------------------------------------------------
      call gopks(6,idum)
      call gopwk(1,2,1)
      call gacwk(1)
      write(*,*)' enter contour interval for depth'
```

```
read(*,*)cont
call conrec(h,251,mnew,nnew,0.,3.,cont,0,-1,200)
write(*,*)' enter contour interval for amplitude'
read(*,*)cont
call conrec(wave,251,mnew,nnew,0.,3.,cont,0,-1,0)
call frame
call gdawk(1)
call gclwk(1)
call gclks
stop
end
```

7 Appendix C: *fweb* Documentation of the SPECGEN Program Listing

The following section contains a document which provides a heavily anotated program listing of **SPECGEN**, which is the preprocessing program used to create most of the input files to **REF/DIF S** for the example cases in this manual.

1. Data Generation for REF/DIF S: Introduction and Program Input.

This program generates the input file *indat.dat* for **REF/DIF S** Version 1.0, which models refraction and diffraction of two-dimensional spectra on arbitrary bathymetry.

The two-dimensional spectrum is obtained by generating a TMA spectrum which is used in conjunction with an angular spreading function. The resulting two-dimensional spectrum is divided into discrete components. The generated input file can be used with **REF/DIF S** as well as **REF/DIF 1**.

The input to **SPECGEN** is read in free format from the file *specgen.dat* which contains information about the domain and options desired to be used in **REF/DIF S** as well as information about the desired spectra. The first part of information is not used in the computational part of **SPECGEN** but is just past to the input file of the model **REF/DIF S**, whereas information about the spectra is used to generate and discretize the spectra. This results in discrete wave components to be input into **REF/DIF S**.

The input file *specgen.dat* is set up in the following manner:

The parameters in this first part are not related to the operation of **SPECGEN**, but will control the operation of the model **REF/DIF S**

1. $iun(1), iun(2), iun(3)$

   Logical device numbers of reference grid data, user-specified subdivision data and reference grid output.

2. mr, nr : Reference grid dimensions.

3. $iu, ntype, icur, ibc$:

   iu is the switch for physical units; $iu = 1$, MKS; $iu = 2$, English.

   $ntype$ is switch for nonlinearity; $ntype = 0$, linear model: $ntype = 1$, composite model; $ntype = 2$, Stokes wave model.

   $icur$ is switch for input current data. $icur = 0$, nocurrents input; $icur = 1$, currents input.

   ibc is the boundary condition switch. $ibc = 0$, closed boundaries; $ibc = 1$, open boundaries.

4. dxr, dyr, dt : Reference grid x-spacing and y-spacing and depth tolerance value.

5. $ispace, nd$:

ispace is switch controlling subdivisions. $ispace = 0$, program attempts its own x subdivisions; $ispace = 1$, user specifies x subdivisions.

nd is the number of y subdivisions.

If $ispace = 1$, the x subdivisions are inserted here.

$$(md(ir), ir = 1, mr - 1)$$

6. $iff(1), iff(2), iff(3)$: Dissipation switches.

   $iff(1) = 1$, turn on turbulent boundary layer; $iff(2) = 1$, turn on porous bottom damping; $iff(3) = 1$, turn on laminar boundary layer.

   No damping if all values are zero.

7. *isp* : Switch for user-specified subgrid specifications.

   $isp = 0$, no subgrids to be read; $isp = 1$, subgrids will be read.

8. *tide* : Tidal offset.

9. *iopt* : *iopt* specifies whether amplitudes values of a specific components are to be stored in file *specamp.dat*.

   $iopt = 1$ turns the option on;

   $iopt = 0$ skips the option. The default value is zero.

   If $iopt = 1$ the desired wave component is specified here:

   $ifcomp, idcomp$: The amplitude output is to be done for frequency component $ifcomp$ and directional component $idcomp$.

   This line concludes the first part of the input file. The following lines of input specify the input spectra.

10. *depth* : Water depth in (m)

11. f_m : Peak frequency of frequency spectrum in (Hz)

12. f_{max} : Maximum frequency that is desired to be resolved in (Hz)

13. H_s : Significant wave height in (m)

14. γ : Peak enhancement factor

15. $nebin$: Desired number of frequency bins

16. θ_m : Mean direction in ($^\circ$)

17. σ_m : Directional spreading parameter in ($^\circ$)

18. $ndbin$: Desired number of directional bins

2. **The Code.**

program *specgen*

3. The definition and input sections of the code are as follows.

⟨ Common Declarations 15 ⟩
dimension $f(200000)$, $tma(200000)$, $f\!f(200000)$, $d(200000)$, $tt(200000)$, $thet(200000)$

⟨ Open Statements 16 ⟩
call *readin*

4. Additional parameters are defined and conversions are performed.

$depth = depth*100.$
$hs = hs*100.$
$pi = 4.*\mathsf{atan}(1.)$
$g = 980.621$
$l = 20$
$deltaf = 0.025$
$fmin = deltaf$
$nn = nint(fmax/deltaf)$
$deltat = pi*4.5/180.$
$thetm = pi*thetm/180.$
$sigm = pi*sigm/180.$

5. Construction of frequency spectrum array.

$alpha = 1.$
do 1 $i = 1, nn$
  $f(i) = deltaf*\mathsf{float}(i)$
  $tma(i) = sptma(f(i))$
1: **continue**
write $(*,*)$ `'Frequency␣spectrum␣computed'`

6. Computation of area under frequency spectrum and proper value of α.

call $integ(1, 1, tma, 1, nn, deltaf, am)$

write $(*, *)$ $alpha$

$alpha = (hs^2/2.)/(8.*am)$

write $(*, *)$ $alpha$

do 33 $i = 1, nn$

  $tma(i) = alpha*tma(i)$

33: **continue**

call $integ(1, 1, tma, 1, nn, deltaf, am)$

7. Determination of cut-off points.

$ij = 1$

call $integ(2, 1, tma, ij, nn, deltaf, am)$

$fsave = f(ij)$

$nn = nn - (ij - 1)$

do 6 $i = 1, nn$

  $f(i) = fsave + \mathsf{float}(i - 1)*deltaf$

  $tma(i) = sptma(f(i))$

  write $(20, *)$ $f(i)$

  write $(30, *)$ $tma(i)$

6: **continue**

$a = am/\mathsf{float}(nebin)$

8. Integration for bins.

call $bins(1, deltaf, nn, nebin, f, tma, a, ff)$

9. Construction of directional spectrum array.

$nn = 81$

do $3\ i = 1, nn$

$\quad thet(i) = -(pi/2.) + \mathsf{float}(i-1)*deltat$

3: **continue**

do $4\ i = 1, nn$

$\quad d(i) = spd(thet(i))$

4: **continue**

write $(*, *)$ `'Directional␣spectrum␣computed'`

10. Computation of area under directional spectrum.

call $integ(1, 2, d, 1, nn, deltat, ad)$

11. Determination of cutoff points.

$ij = 1$

call $integ(2, 2, d, ij, nn, deltat, ad)$

$tsave = thet(ij)$

$nn = nn - (ij - 1)$

do $8\ i = 1, nn$

$\quad thet(i) = tsave + \mathsf{float}(i-1)*deltat$

$\quad d(i) = spd(thet(i))$

$\quad$ **write** $(40, *)\ thet(i)$

$\quad$ **write** $(50, *)\ d(i)$

8: **continue**

$aa = ad/\mathsf{float}(ndbin)$

12. Determination of amplitudes of components.

$amp = \mathsf{sqrt}(2.*a*aa)/100.$

13. Integration for bins.

call $bins(2, deltat, nn, ndbin, thet, d, aa, tt)$

14. Assure symmetry of directional spectrum.

$ndd = ndbin/2 + 1$

do 15 $ii = 1, ndd$

$dif = \mathsf{abs}(\mathsf{abs}(tt(ii)) - \mathsf{abs}(tt(ndbin - ii + 2)))$

if $(dif > 1. \cdot 10^{-08})$ **then**

$tt(ndbin - ii + 2) = (\mathsf{abs}(tt(ii)) + \mathsf{abs}(tt(ndbin - ii + 2)))/2.$

$tt(ii) = -tt(ndbin - ii + 2)$

endif

15: **continue**

$tt(ndd) = 0.$

⟨ Output Statements 17 ⟩

end

15. Common and Statements.

⟨ Common Declarations 15 ⟩ ≡

common /*blk1*/ *fpeak*, *alpha*, *gamma*, *depth*, *g*, *fmin*, *fmax*

common /*blk2*/ *sigm*, *thetm*, *l*

common /*blk3*/ *nn*, *deltaf*, *nebin*, *deltat*, *ndbin*, *hs*

common /*blk4*/ *iun*(3), *mr*, *nr*, *iu*, *ntype*, *icur*, *ibc*, *dxr*, *dyr*, *dt*, *ispace*, *nd*, *iff*(3), *isp*, *tide*, *iopt*, *ifcomp*, *idcomp*

common /*blk5*/ *iinput*, *ioutput*, *iwave*

This code is used in sections 3 and 18.

16. Open Statements.

⟨ Open Statements 16 ⟩ ≡

open (10, *file* = `'specgen.dat'`)

open (20, *file* = `'f.dat'`)

open (30, *file* = `'ftma.dat'`)

open (40, *file* = `'thet.dat'`)

open (50, *file* = `'d.dat'`)

open (60, *file* = `'ft.dat'`)

open (70, *file* = `'indat.dat'`)

This code is used in section 3.

17. Output Statements.

⟨ Output Statements 17 ⟩ ≡

```
      // Output into "f.dat"
    do 2 ii = 1, nebin
      ff(ii) = 0.5*(ff(ii) + ff(ii + 1))
      ff(ii) = 1./ff(ii)
  2: continue
    write (60, *) (ff(ii), ii = 1, nebin)
    do 5 ii = 1, ndbin
      tt(ii) = 0.5*(tt(ii) + tt(ii + 1))
      tt(ii) = 180.*(tt(ii)/pi)
  5: continue
    write (60, *) (tt(ii), ii = 1, ndbin)

      // Output as "indat.dat".

    write (70, *) (iun(i), i = 1, 3)
    write (70, *) mr, nr
    write (70, *) iu, ntype, icur, ibc
    write (70, *) dxr, dyr, dt
    write (70, *) ispace, nd
    write (70, *) (iff(i), i = 1, 3)
    write (70, *) isp
    write (70, *) iinput, ioutput
    write (70, *) iwave, nebin
    do 10 ii = 1, nebin
      write (70, *) ff(ii), tide
      write (70, *) ndbin
      do 11 iii = 1, ndbin
        write (70, *) amp, tt(iii)
   11: continue
 10: continue
    write (70, *) fpeak
    write (70, *) iopt
    if (iopt ≡ 1) write (70, *) ifcomp, idcomp
```

This code is used in section 14.

18. Routine to Read Input.

This routine reads input from file *specgen.dat*.

subroutine *readin*

$\langle$ Common Declarations 15 $\rangle$

// Read or specify parameters for 'indat.dat'

read $(10,*)$ $(iun(i), i = 1, 3)$

read $(10,*)$ mr, nr

read $(10,*)$ iu, $ntype$, $icur$, ibc

read $(10,*)$ dxr, dyr, dt

read $(10,*)$ $ispace$, nd

read $(10,*)$ $(iff(i), i = 1, 3)$

read $(10,*)$ isp

read $(10,*)$ $tide$

read $(10,*)$ $iopt$

if $(iopt \equiv 1)$ **then**

read $(10,*)$ $ifcomp$, $idcomp$

else

$iopt = 0$

endif

$iinput = 1$

$ioutput = 1$

$iwave = 1$

// Read parameters for setup and discretezation of spectra

read $(10,*)$ $depth$

read $(10,*)$ $fpeak$

read $(10,*)$ $fmax$

read $(10,*)$ hs

read $(10,*)$ $gamma$

read $(10,*)$ $nebin$

read $(10,*)$ $thetm$

read $(10,*)$ $sigm$

read $(10,*)$ $ndbin$

return

end

19. Function to Generate TMA Spectrum.

This routine, originally written by Vincent and Briggs, calculates a TMA spectrum using input parameters provided by the user in the second part of the file *specgen.dat*.

The TMA spectrum is given by the energy density $E(f)$ for frequency f,

$$E(f) = \frac{\alpha g^2}{(2\pi)^4 f^5} \exp\left\{-1.25\left(\frac{f_m}{f}\right)^4 + (\ln\gamma)\exp\left[\frac{-(f-f_m)^2}{2\sigma^2 f_m^2}\right]\right\}\phi(f,h) \tag{44}$$

The parameters α, f_m and γ were defined while describing the input. Furthermore $\sigma =$ Shape parameter defined in the program by,

$$\sigma = \begin{cases} \sigma_a = 0.07 & \text{if } f < f_m \\ \sigma_b = 0.09 & \text{if } f \geq f_m \end{cases} \tag{45}$$

The factor $\phi(f,h)$ incorporates the effect of the depth h and is computed following Hughes (1984) by,

$$\phi = \begin{cases} 0.5(\omega_h)^2 & \text{if } \omega_h < 1 \\ 1 - 0.5(2-\omega_h)^2 & \text{if } 1 \leq \omega_h \leq 2 \\ 1 & \text{if } \omega_h > 2 \end{cases} \tag{46}$$

where,

$$\omega_h = 2\pi f\sqrt{\frac{h}{g}}$$

Broad or narrow frequency spectra can be obtained assigning the values 2 and 20 to the parameter γ, respectively. The code follows.

```
function sptma(ff)

common /blk1/ fpeak, alpha, gamma, depth, g, fmin, fmax

sa = 0.07
sb = 0.09
twopi = 8.*atan(1.)
if (ff < fmin) then
   sptma = 0.
   return
endif
if (ff > fmax) then
   sptma = 0.

   return
endif
```

20. Calculation of four factors:

- High frequency tail equilibrium range factor (*p1*)
- Low frequency tail factor (Pierson-Moskowitz) (*p2*)
- Jonswap factor (*p3*)
- TMA spectrum factor (*phi*)

$p1 = (alpha*g*g)/(1558.55*ff^5)$
$z2 = -1.25*(ff/fpeak)^{-4}$
if $(z2 < -28.)\ z2 = -28$
$p2 = \mathsf{exp}(z2)$
$ss = sa$
if $(ff > fpeak)\ ss = sb$
$bb = (ff - fpeak)/(ss*fpeak)$
$bb = .5*bb^2$
if $(bb > 28)\ bb = 28$
$p3 = gamma^{\mathsf{exp}(-bb)}$
$omh = twopi*ff*\mathsf{sqrt}(depth/g)$
if $(omh \leq 1.)\ phi = omh^2/2.$
if $(omh > 1.)\ phi = 1. - 0.5*(2. - omh)^2$
if $(omh \geq 2.)\ phi = 1.$

21. TMA Spectrum.

$sptma = p1*p2*p3*phi$

return

end

22. Function to Calculate Directional Spectrum.

A directional spectrum $D(\theta)$ was used in conjunction with the frequency spectrum to create a two-dimensional spectrum. The angular spectrum is defined by,

$$D(\theta) = \frac{1}{2\pi} + \frac{1}{\pi} \sum_{j=1}^{J} \exp\left[-\frac{(j\sigma_m)^2}{2}\right] \cos j(\theta - \theta_m) \tag{47}$$

where,

θ_m = mean wave direction = 0^0.

J = number of terms in the series chosen as 20 in the numerical calculations.

σ_m is chosen to be 10^0 or 30^0 giving a narrow or broad directional spreading, respectively. All three parameters are input. The code follows.

```
function spd(thet)
  common /blk2/ sigm, thetm, l
  pi = 4.*atan(1.)
  sum = 0.
  do 1 i = 1,l
    sum = sum + exp(-0.5*(i*sigm)^2)*cos(i*(thet - thetm))
1: continue
  spd = (1./(2.*pi)) + (1./pi)*sum
  return
end
```

23. Routine to Calculate Areas under Spectra.

This routine calulates the area under a function *fun(i)* using Simpson's Rule with a step size of *df.* The routine performs two different functions.

24. Computation of Total Area.

If option 1 is chosen for the variable *opt* in the call statement the total area under the curve of the function *fun(i)* is computed and returned to the calling routine.

subroutine *integ*(*opt*,*ii*,*fun*,*isave*,*n*,*df*,*area*)

dimension $fun(n)$
integer *opt*

if ($opt \equiv 1$) **then**
  $s1 = fun(2)$
  $s2 = fun(3)$
  do 1 $i = 1, n/2 - 2$
    $s1 = s1 + fun(2 + 2*i)$
    $s2 = s2 + fun(3 + 2*i)$
1: **continue**
  $area = (df/3.)*(fun(1) + 4.*s1 + 2.*s2 + fun(n))$
  return

25. Determination of Cut-off Points.

If option 2 is chosen the left tail of the function *fun(i)* containing 0.25% of the total area is cut off. If *fun(i)* is representing the frequency spectrum the right tail containing 1% of the total area is cut off.For the directional spectrum this value is 0.25%. The new values of the starting and ending locations of the array *fun(i)* are returned to the calling routine.

```
      elseif (opt ≡ 2) then
      asave = area
      area = 0.
      icount = 1
      do 2 i = isave, n − 2, 2
         area = area + (df/3.)*(fun(i) + 4.*fun(i + 1) + fun(i + 2))
         if (icount ≡ 1) then
            if (area > (0.0025*asave)) then
               icount = icount + 1
               isave = i
            endif
         endif
         if (ii ≡ 1 ∧ area > (0.99*asave)) go to 10
         if (ii ≡ 2 ∧ area > (0.9975*asave)) go to 10
         if (i ≡ (n − 2)) write (*, *) 'Doing last array member'
   2: continue
  10: n = i + 2
      return
    endif

    end
```

26. Routine to Divide Spectra into Equal-Energy Bins.

This routine integrates a given function $f(x)$ until the desired energy value, i.e. area value is reached. If the resolution of the given disscrete function is not high enough, twice the resolution is introduced and the spectrum for the new resolution computed. Integration is started again at the bin where convergence was not achieved. Integration is performed by Simpson's Rule.

The program structure is as follows.

subroutine $bins(ni,dx,n,nbin,x,f,area,xx)$

common $/blk1/$ $fpeak$, $alpha$, $gamma$, $depth$, g, $fmin$, $fmax$

common $/blk2/$ $sigm$, $thetm$, l

dimension $x(200000)$, $f(200000)$, $xx(200000)$

real in, ins, inb

$eps = 1. \cdot 10^{-03}$

$in = 0.$

$isave = 1$

$dxsave = dx$

$ii = 1$

$xx(1) = x(1)$

27. Main Integration Loop.

This loop integrates the function. If convergence is achieved the integration for that bin is completed. Integration is started again for the next bin.

If the integral value is too small at one step and too big at the other, the resolution is increased by simply dividing Δx by 2 and setting up the spectrum again with the new, finer resolution. Integration is started again at the last point where the integral alue was smaller than the desired value.

```
80: do 2 i = isave, n - 2, 2
      in = in + (dx/3.)*(f(i) + 4.*f(i + 1) + f(i + 2))
        // Check for convergence
      if (abs(in - float(ii)*area) < eps) then
        write (*, *) 'Bin␣', ii, '␣completed.'
        xx(ii + 1) = x(i + 2)
        ii = ii + 1
        isave = i + 2
        go to 80
      endif
      if (in < (float(ii)*area)) then
        ins = in
      elseif (in > (float(ii)*area)) then
        inb = in
        dx = dx/2.
        n = 2*n - 1
        isave = 2*i - 1
        xr = x(i)
        in = ins
        if (ni ≡ 1) then
          do 4 j = isave, n
            x(j) = dx*float(j - 1) + dxsave
            f(j) = sptma(x(j))
        4: continue
        elseif (ni ≡ 2) then
          do 5 j = isave, n
            x(j) = xr + float(j - isave)*dx
            f(j) = spd(x(j))
```

```
      5: continue
      endif
      go to 80
    endif
2: continue
  xx(nbin + 1) = x(n)
  return
  end
```

8 Appendix D: Answers to Frequently Asked Questions

The following section contains answers to frequently asked questions.

1. What is the maximum current this model can handle?

   The program can handle currents that are of the same order as the wave speed. However, opposing currents which are strong enough to stop the waves are not handled properly. This extension will be added in the future.

2. In which situations should the user specify totally reflecting lateral boundaries ($ibc = 0$), and when should partially transmitting lateral boundaries ($ibc = 1$) be used?

   It is common practice to use the closed boundary condition as the amount of reflection can be identified and the domain width can be chosen so that there are no reflections in the region of interest. The final run may be made using partially transmitting boundaries, but it should be noted that there will still be a certain amount of reflected waves present in the domain.

3. What is the difference between a user-specified subdivisions and user-specified subgrid?

   User-specified subdivisions and user-specified subgrids are two completely independent options. The user has to specify the desired number of subdivisions in the y-direction nd. However, the user has an option between specifying the number of subdivisions in the x-direction $md(ir)$ or having the program specify those. If the program specifies $md(ir)$, the switch *ispace* has to be set to zero and the program performs interpolations of the depth and velocity grids. If the user choses to specify the $md(ir)$ values himself, the switch *ispace* has to be set to unity. In this case, the user again has a choice: The user can specify $isp = 0$, then the program will perform interpolations. The user can set $isp = 1$, in which case he/she has to specify a subgrid in the file *subdat.dat*.

4. What are the guidelines for the specification of the number of subdivisions $md(ir)$ in the x-direction?

   It is recommended that the number of subdivisions $md(ir)$ be determined by the program. However, if the user wants to specify $md(ir)$'s, the user should make sure that the subdivided grid is at least as fine as the program would have determined. In order to achieve this, the user can first run the program by letting it pick its own subdivisions. Then the user can chose his/her subdivision using the programs subdivisions as a guideline.

5. What are the guidelines for the choice of the number of subdivisions nd in the y-direction ?

   nd should be chosen such that the final subdivided reference grid cells have increments in the x- and y-directions that are fairly close in size.

6. Where is the origin of the grid ?

   The origin of the domain is always chosen to be the lower right-hand corner of the domain with the x-axis pointing in the propagation direction and the y-axis pointing towards the left.