WHOI-85-17

Programs for Computing Properties of Coastal-Trapped Waves and Wind-Driven Motions Over the Continental Shelf and Slope

by

Kenneth H. Brink

David C. Chapman

Woods Hole Oceanographic Institution
Woods Hole, Massachusetts 02543

April 1985

Technical Report

Funding was provided by the National Science Foundation under grant Number OCE 84-08563.

Approved for Distribution:

Robert C Beardsley

Robert C. Beardsley, Acting Chairman
Department of Physical Oceanography

Abstract

Documentation and listings are presented for a sequence of computer programs to be used for problems in continental shelf dynamics. Three of the programs are to be used for computing properties of free and forced coastal-trapped waves. A final program may be used to compute wind-driven fluctuations over the continental shelf and slope.

Table of Contents

		Page
1.	General Introduction	1
2.	Barotropic Shelf Waves	5
3.	Coastal-Trapped Waves with Stratification and Topography	16
4.	Near-Inertial Coastal-Trapped Waves with Stratification and Topography	25
5.	Wind-Driven Motions	31
	Word of Caution	43
	Acknowledgements	44
	References	45
	Program Listings	
	BTCSW	46
	BIGLOAD2	60
	CROSS	76
	BIGDRV2	81

CHAPTER 1

GENERAL INTRODUCTION

In a recent sequence of papers (Brink, 1982a,b; Chapman, 1983; Clarke and Brink, 1985) a number of computer programs have been described which compute properties of linear coastal-trapped waves and wind-driven motions over the continental shelf. These programs, since they allow rather arbitrary choices of topography, stratification, etc., may be of fairly general use to the oceanographic community. For this reason, listings and documentation for these algorithms have been assembled here in an accessible form.

Some definitions are common to all of the following routines. Specifically, we use the coordinate system shown in Figure 1, such that the coast (if present) lies at $x = 0$ and the ocean in the region $x > 0$. The alongshelf coordinate is y and the vertical coordinate is z (positive upwards), such that $z = 0$ at the ocean surface. The x, y and z velocity components are then u, v and w respectively. Depth-integrated u and v velocities are defined as U and V, respectively. Pressure and density are given as p and ρ, respectively. A few other commonly used variables are N^2, f, g, h, ω and ℓ, which represent the Brunt-Väisälä frequency squared, the Coriolis parameter, the acceleration due to gravity, the water depth, wave frequency and alongshelf wavenumber.

A few assumptions are common to all programs below. First, only linear problems are considered. Second, the water depth is always assumed to be a function of x only. Third, the Brunt-Väisälä frequency may vary in z only, and must be non-zero everywhere. The only exceptions are in computing barotropic continental shelf waves (program BTCSW, Chapter 2) where the problem is linearized and the Brunt-Väisälä frequency is not specified.

The general free-wave programs BTCSW and BIGLOAD2 (coastal-trapped waves with continuous stratification; Chapter 3) search for free-wave solutions using resonance iteration. The general approach is to assume that the dependent variables are sinusoidal in time and the alongshelf direction, e.g.

$$U(x,y,t) = \hat{U}(x) \exp[i(\omega t + \ell y)] \quad ,$$

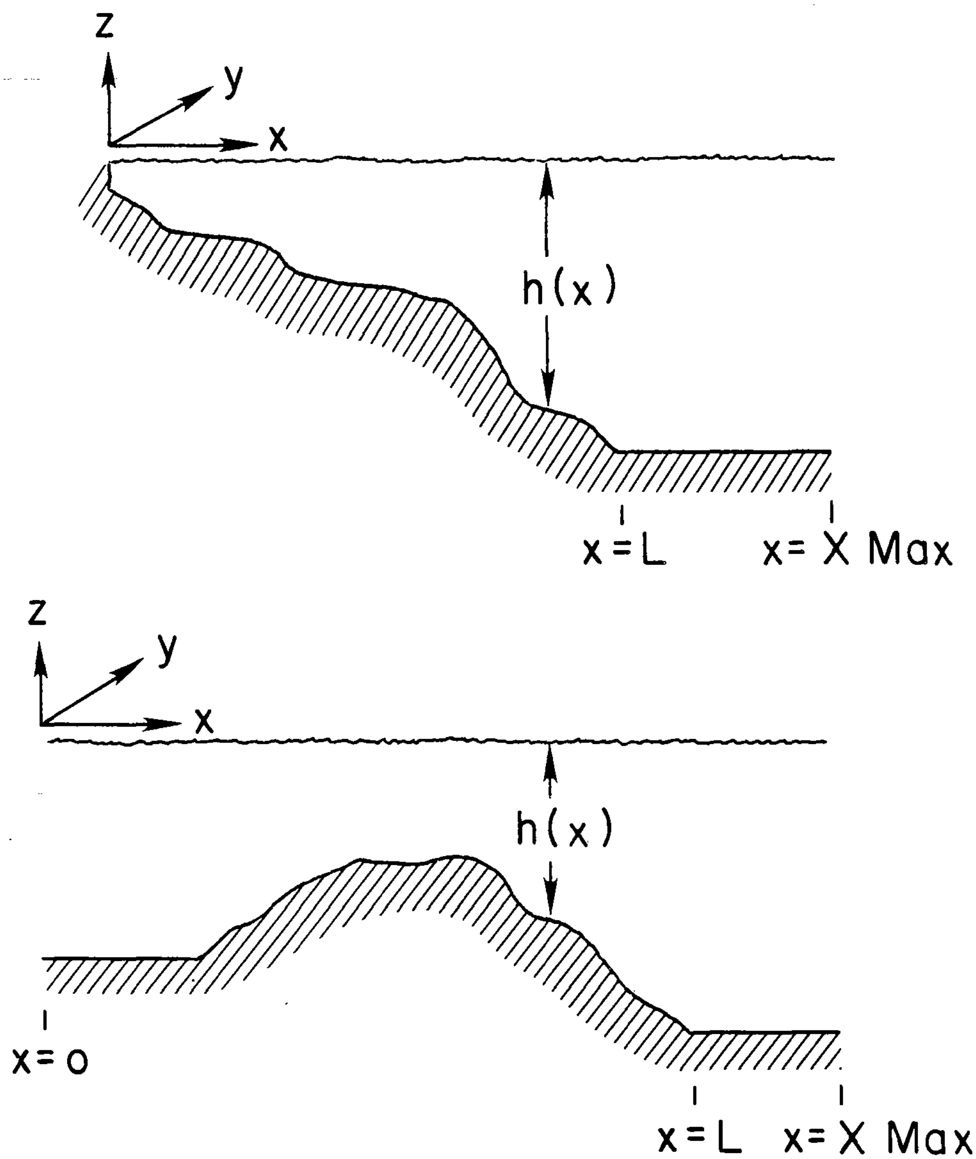

Figure 1: Topography and coordinate system definitions used in all programs: (upper) with a coast, (lower) without a coast.

reducing the problem to a two-dimensional eigenvalue problem in (ω, ℓ):

$$\mathcal{L}(\hat{U}(x;\omega,\ell)) = 0 \; .$$

This is solved for arbitrary forcing and a fixed ℓ. The frequency ω is then varied until the free-mode resonance is reached. Resonance is defined as the frequency at which the integrated field variable squared,

$$I_v = \int_0^\infty \hat{U}^2 dx$$

or

$$I_p = \int_0^\infty \int_{-h}^0 \hat{p}^2 \, dz \, dx \; ,$$

is at a maximum.

A few comments are in order about the workings of the programs. The units internal to all programs are cgs, although input values are often in convenient units (e.g. km for x). The input file is always number 5, and the output file number 6. All programs are self-contained except for BIGLOAD2, which requires the use of IMSL subroutine LEQT1B. This subroutine is used to solve the banded matrix equation by L-U decomposition.

The programs described below can be briefly summarized as follows:

1) BTCSW: barotropic continental shelf waves (e.g. Buchwald and Adams, 1968) and barotropic bank or trench waves (e.g. Brink, 1983; Mysak, LeBlond and Emery, 1979). Dispersion curves, modal structures, and wind coupling coefficients can be computed for arbitrary topography and mean alongshore flow.
2) BIGLOAD2: Coastal-trapped waves in the presence of continuous stratification (e.g. Wang and Mooers, 1976; Huthnance, 1978; Brink, 1982a,b). Dispersion curves (up to $\omega \cong 0.9f$), modal structures and wind coupling coefficients can be computed for arbitrary topography and (horizontally uniform) stratification.
3) CROSS: Finding flat-bottom baroclinic modes and where $\omega = f$ for general coastal-trapped waves (Chapman, 1983). The program allows arbitrary stratification and monotonic bottom topography.

4) BIGDRV2: Wind-driven motions over the continental margin (e.g. Clarke and Brink, 1985). The velocity, pressure and density fluctuations driven by a wind stress of the form $\underline{\hat{\tau}}(x) \exp[i(\omega t + \ell y)]$ can be computed for general topography, stratification and bottom friction.

Finally, the user should be aware that programs BTCSW and CROSS require very little CPU time to complete, whereas program BIGLOAD2 uses approximately one minute of CPU time for each point on a dispersion curve and program BIGDRV2 requires approximately one minute of CPU time to complete (both on a VAX 11/780).

CHAPTER 2
BAROTROPIC SHELF WAVES
Documentation for BTCSW

A. Introduction

This program computes modal structures and dispersion curves for free barotropic shelf waves. It can also compute bottom friction and wind coupling coefficients as in Brink and Allen (1978). Either a free surface or a rigid lid may be used, and a stable mean alongshelf flow can also be included. A variety of boundary conditions are available as options.

B. Formulation

For a linearized, inviscid barotropic ocean, the depth-integrated equations of motion are:

$$\epsilon U_t + \epsilon v_o U_y - fV = -gh\zeta_x , \tag{2.1a}$$

$$V_t + v_o V_y + U v_{ox} + fU = -gh\zeta_y , \tag{2.1b}$$

$$\delta\zeta_t + v_o\zeta_y + U_x + V_y = 0 , \tag{2.1c}$$

where the onshore, and alongshelf directions are x and y, respectively, and there are no alongshelf variations in the mean flow $v_o(x)$ or in the depth h. U and V are the depth-integrated velocities in the x and y directions. The free surface elevation is ζ, and subscripts x, y and t represent partial differentiation. The constants g and f are the acceleration due to gravity and the Coriolis parameter. The variables ϵ and δ are defined as follows:

$\epsilon = 0$ for the long-wave approximation,
$\epsilon = 1$ for general frequencies and wavenumbers,
$\delta = 0$ for the rigid-lid approximation,
and $\delta = 1$ for a free surface.

With the assumption that U, V and ζ vary as $\exp[i(\omega t + \ell y)]$, (2.1) become

$$i\omega'\epsilon U - fV = -gh\zeta_x ,$$

$$i\omega' V + f'U = -i\ell gh\zeta ,$$

$$i\omega'\delta\zeta + U_x + i\ell V = 0 ,$$

where $\omega' = \omega + \ell v_o$

and $f' = f + v_{ox}$.

These can be reduced to either:

$$\begin{aligned} 0 = {} & U_{xx}[\delta \frac{\omega'^2}{g} - h\ell^2]h \\ & + U_x[h_x \ell^2 - \frac{2\omega' \ell v_{ox}\delta}{g}]h \\ & + U[-\frac{\delta^2\omega'^2}{g^2}(ff' - \epsilon\omega'^2) + \epsilon h^2 \ell^4 \\ & \quad + \frac{\delta}{g} h \ell^2 (ff' - 2\omega'^2\epsilon + 2f'v_{ox}) \\ & \quad - \frac{h\ell^3 f' h_x}{\omega'} - h\frac{\ell}{\omega'} v_{oxx}(\delta\frac{\omega'^2}{g} - h\ell^2)] \end{aligned} \tag{2.2}$$

or

$$\begin{aligned} 0 = {} & \zeta_{xx}[ff' - \epsilon\omega'^2]h \\ & + \zeta_x[(ff' - \omega'^2\epsilon)h_x - h(f v_{oxx} - 2\omega' \ell v_{ox}\epsilon)] \\ & + \zeta[-\frac{\delta}{g}(ff' - \epsilon\omega'^2)^2 + h_x \frac{f\ell}{\omega'}(ff' - \epsilon\omega'^2) \\ & \quad - h\frac{f\ell}{\omega'}(fv_{oxx} - 2\epsilon\omega' \ell v_{ox}) \\ & \quad - h\ell^2\epsilon(ff' - \omega'^2\epsilon)] . \end{aligned} \tag{2.3}$$

Each of these equations presents a practical difficulty. The ζ equation (2.3) possesses a spurious solution. For example, when $v_0 = 0$ this solution has $\omega = f$, and $\zeta = \zeta_0 \exp(-\ell x)$. (See Pedlosky, 1979, pp. 79–81 for an explanation.) This spurious mode may, in turn, affect the true solutions. The U equation (2.2) does not possess a spurious mode, but can lead to numerical difficulties for very shallow water (e.g. solving for a laboratory case where $h < 1$ m everywhere). In general, it is preferable to use the U equation, and to check it against the results of the ζ equation. The program allows the choice of the U or ζ equation.

C. Program Input

As explained below, the user provides a bottom topographic profile, a mean flow profile (if desired), and choices for boundary conditions. The program returns modal structures for U and ζ, and frequencies for the prescribed wavenumbers. All outputs are in either cgs or arbitrary units, although inputs are in convenient units. Two geometries are possible (Figure 1, p. 2). The first case (Figure 1a) contains a coastal barrier, while the second case (Figure 1b) does not. The second case is useful for bank- or trench-trapped waves. Note that $\omega > 0$ is assumed, so that waves propagating in the positive y direction (opposite to standard shelf waves in the northern hemisphere) must be found using $\ell < 0$.

The following presentation of input parameters describes the user options. A compact list of parameters is given in section 2E. All data are read from file 5.

line 1: IMDM NN

IMDM is the number of cases to be studied. If IMDM $\neq$ 1, all of the other lines of input must be repeated for each case. This is useful if, for example, several geometries are to be studied in one run.

NN is the number of grid points in the x direction. Presently, NN $\leq$ 100, but this could be easily changed by the user.

line 2: NITM ISD EPS DEL

These are all parameters used in the search for the resonant frequency.

NITM is the maximum number of iterations allowed for finding a resonant frequency (typically 20-40).

ISD directs the frequency search.

For ISD = 0, the program searches for the free-wave frequency closest to the initial guesses.

For ISD = 1, the program searches only towards lower frequencies.

For ISD = -1, the program searches only towards higher frequencies.

EPS is the nominal fractional accuracy desired for ω. This is always less than the true error range within which ω is known. Typically, EPS = 0.005 (0.5 percent accuracy).

DEL is the fractional step size used for initially searching for ω. Typically, DEL = 0.05 (5 percent).

line 3: IUP ILLW

IUP provides the choice of searching with the U or ζ equation (see section 2B).

IUP = 0 specifies a search using U.

IUP = 1 specifies a search using ζ.

If some other value is given, the program defaults to IUP = 0.

ILLW allows the option of making the long-wave approximation exactly.

ILLW = 0 for long waves ($\varepsilon = 0$ in section 2B).

ILLW = 1 for the general case ($\varepsilon = 1$ in section 2B).

If some other value is given, the program defaults to ILLW = 0.

Also, if ILLW = 0, NCALM (see below) is set to 1, since the waves will be nondispersive.

line 4: IDD1 IDD3 IDD4

These parameters select the boundary conditions.

IDD1 = 0 for a rigid lid ($\delta = 0$ in section 2B).

IDD1 = 1 for a free surface ($\delta = 1$ in section 2B).

Other values set the default of IDD1 = 0.

IDD3 = 0; the boundary condition at x = XMAX is $U_x = 0$. This is not the "real" condition, but is used for comparison with the stratified wave program (Chapter 3).

IDD3 = 1; the boundary condition at x = XMAX is $U = 0$. This simulates a channel problem.

IDD3 = 2 sets up the real, exponentially decaying condition at x = XMAX. This is the preferred condition, <u>but</u> it is only valid if h and v_0 are constant near x = XMAX.

If another choice is made for IDD3, the program reverts to IDD3 = 0.

IDD4 = 0, the boundary condition at $x = 0$ is $U_x = 0$. This may be useful for bank or trench waves.

IDD4 = 1 sets $U = 0$ at $x = 0$. This is the desired condition for shelf waves.

IDD4 = 2 uses the exponential decay condition at $x = 0$. This is again the preferred condition for bank or trench waves, <u>but</u> it is only valid for h and v_0 constant at $x = 0$ (geometry of Figure 1b).

Other values of IDD4 cause the program to revert to IDD4 = 1, the shelf wave case.

line 5: NCALM ILW

NCALM is the maximum number of (ω, ℓ) pairs to be calculated for a given dispersion curve.

ILW provides an option on calculating parameters valid for the long-wave limit. These will only be computed for the first (ω, ℓ) pair.

If ILW = 0, then no long-wave parameters are computed.

If ILW ≠ 0, then the "streamfunction" $\phi_n(x)$, wind-coupling coefficient b_n and bottom drag coefficient a_{nn} are computed. The definitions follow from Brink and Allen (1978).

For computation and conceptual reasons, these parameters will not be computed if either IDD4 ≠ 1 or if $h(0) = 0$, even if ILW = 1. ILLW need not be set to 0.

line 6: RLF DRL

These parameters define the wavenumbers for which ω is calculated. The wavenumbers used in the program will be:

$$\ell = (\text{RLF} + (n - 1)\ \text{DRL}) \times 10^{-8} \text{cm}^{-1}$$

when n represents the number of the (ω, ℓ) pair on the dispersion curve. n ranges from 1 to NCALM (see line 5).

For example, if RLF = 0.5 and DRL = 1.0, then the first wavenumber to be computed is $\ell = 0.5 \times 10^{-8}cm^{-1}$ and the others will be $(1.5, 2.5, 3.5, \ldots) \times 10^{-8}cm^{-1}$.

line 7: IPC

If IPC $\neq$ 0, then the program prints out modal structures as well as search information for each (ω, ℓ) pair.

If IPC = 0, then the modal structure is printed only for the first (ω, ℓ) pair.

line 8: F XMAX

F is the Coriolis parameter, which is multiplied by 10^{-5} within the program. Thus, F = 7.5 represents $f = 7.5 \times 10^{-5}s^{-1}$.

XMAX is the distance (in km) from x = 0 to the offshore boundary of the grid (Figure 1). Typically, XMAX ~ 2L, so that about one half of the domain has a flat bottom.

line 9: NRX

This is the number of [x,h(x)] pairs to be input to define the bottom topography.

line 10 and following: X H

These are pairs of offshore distance (x) in km and depth (h) in m. These can be arbitrarily spaced, and the information is linearly interpolated to the grid points. The first pair must have x = 0. For x > (the last x value read), depth is set to the last h value read.

NRD

This is the number of $[x, v_0(x)]$ pairs to be input. If NRD = 0, the program sets $v_0 = 0$ everywhere.

X V

These are the NRD pairs of offshore distance (x) in km and mean alongshelf velocity (v_0) in cm/s. These can be arbitrarily spaced. For x < (the first x value read), the programs sets $v_0 = 0$. For x > (the last x value read), the program sets v_0 equal to the last v_0 value read.

WW(1) WW(2) WW(3)

These are three initial guesses at the free-wave frequency ω for the first value of $\ell (= RLF \times 10^{-8}cm^{-1})$. The program multiplies WW(I) by 10^{-5}, so WW(1) = 0.5 corresponds to $\omega = 0.5 \times 10^{-5}s^{-1}$.

NW

This is the number of x (in km) and friction weight function (WF, non-dimensional) pairs to be input. This is useful for x dependent bottom drag, i.e.

$$E_0^{1/2} = E' WF(x) ,$$

where E_0 is the Ekman number, E' is the Ekman number at $x = 0$, and $WF(x)$ a weighting function such that $WF(0) = 1$. If NW = 0, then $WF(x) = 1$ for all x as in Brink and Allen (1978). If WF varies, then

$$a_{nn} = \int_0^L WF(x)(\phi_{nx}(x))^2 \, dx ,$$

where ϕ_n is the streamfunction modal structure.

X WF

These are [x (in km), WF (non-dimensional)] pairs to be input. The first pair must start at $x = 0$. For $x >$ (last x value read), WF is set to the last value read.

D. <u>General Comments</u>

i.) The program will work with $h = 0$ at $x = 0$ only in the U equation mode. Thus, if $h(0) = 0$, use IUP = 0. Alternatively, $h(0)$ can be very small with either IUP = 0 or 1.

ii.) When the ζ equation is being used (e.g. IUP = 1), there is a check for small diagonal elements in the finite-difference matrix equation. If a diagonal element is less than 10^{-36}, a message is printed and the solution is omitted.

iii.) As a check of the U_x boundary condition against the "real" boundary condition, calculations were run for XMAX = 2L, no mean flow and $n = 1, 2$. The worst error in ω was 3.6 percent for $n = 1$, and the error decreased for large ℓ. The $n = 2$ long-wave coefficients (a_{22} and b_2) varied substantially, however. The error in b_2 was about 50 percent.

iv.) Identifying modes. The Kelvin wave mode will have no zero crossings of ζ. The first shelf wave mode will have 1 zero crossing, the second 2, etc. The first shelf wave mode will have no sign changes in U, although $U = 0$ at $x = 0$. The second mode has one zero crossing, etc.

v.) When $v_o \neq 0$, the program checks for critical layers, and prints out the number of critical layers in the solution. Further, the program checks to see if the necessary condition for barotropic instability is satisfied. That is, if

$$\left(\frac{f + v_{ox}}{h}\right)_x$$

changes sign, a warning is given.

E. <u>Input Summary</u>

The input file should include:

IMDM	NN		
NITM	ISD	EPS	DEL
IUP	ILLW		
IDD1	IDD3	IDD4	
NCALM	ILW		
RLF	DRL		
IPC			
F	XMAX		
NRX			
X	H	NRX times	
NRD			
X	V	NRD times	
WW(1)	WW(2)	WW(3)	
NW			
X	WF	NW times	

F. <u>Program Output</u>

The output (file 6) includes statements about which boundary conditions were used, and listings of $h(x)$, $v_o(x)$, $U_n(x)$, $\zeta_n(x)$, WF(x) and $\phi_n(x)$. The functions of $x(h, v_o, U, \zeta, WF, \phi)$ are listed at Δx increments across the page, beginning at x = 0 and proceeding to x = XMAX. Δx is given in the header information.

All units in the output are cgs, except for U and ζ which are in arbitrary units. $\phi(x)$ is normalized as

$$1 = \int_0^L \frac{h_x}{h^2} \phi_n^{\,2} \, dx \; ,$$

so that ϕ has units of $cm^{1/2}$.

The coefficients b_n and a_{nn} for ϕ_n are only strictly valid for $v_o = 0$, and for a rigid lid. Two different a_{nn}, b_n pairs are given. The first (streamfunction) set is as defined in Brink and Allen (1978). The second analogous set is defined for the long-wave problem in terms of pressure. This is useful if there is a free surface, since the streamfunction is invalid. In this case

$$p = \sum_n F_n(x) \, Y_n(y,t)$$

where the free-wave modal structures $F_n(x)$ are orthogonal by

$$\delta_{nm} = (hF_nF_m)\Big|_{x=0} + \int_0^\infty h_x F_n F_m dx \; ,$$

and Y_n obeys

$$b_n' \, \tau_o^{\,y} = Y_{ny} - \frac{1}{c_n} Y_{nt} - r_o \sum_m a_{nm}' \, Y_m \; .$$

The program prints out b_n', a_{nn}' and the pressure normalized as above. The bottom stress is taken to have the form

$$\tau_B^{\,y} = \rho r_o WF(x) v \; ,$$

where WF is as above, r_o is a bottom resistance coefficient in cm s^{-1}, and ρ the fluid density.

G. An Example

Input File:

```
1       100
20      0       0.001   0.05
0       1
0       2       1
1       1
1.0     1.0
0
10.0    400.
3
0.      10.
100.    150.
200.    4000.
3
0.      0.
50.     100.
100.    0.
0.5     0.52    0.54
0
```

The result is, after 14 iterations, $\omega = 0.6867 \times 10^{-5} s^{-1}$, $a_{11} = 0.19865 \times 10^{-7} cm^{-1}$ and $b_1 = 0.1428 \times 10^{-1} cm^{-1/2}$. This is the n = 1 mode.

CHAPTER 3

COASTAL-TRAPPED WAVES WITH STRATIFICATION AND TOPOGRAPHY

Documentation for BIGLOAD2

A. Introduction

This program calculates free-wave dispersion curves (ω, ℓ pairs) by resonance iteration, given input parameters including arbitrary bottom topography and stratification. Options include the choice of a free-surface or a rigid-lid boundary condition, and the inclusion of the component of planetary β perpendicular to the coast.

Note that this program uses an external (IMSL) subroutine in the solution procedure.

B. Formulation

The problem is formulated in the geometry of Figure 1a. Note that the depth at the coast $h(0)$ is non-zero, although it can be arbitrarily small.

The governing equations are

$$\left.\begin{aligned} \epsilon u_t - fv &= -\frac{1}{\rho_o} p_x \\ v_t + fu &= -\frac{1}{\rho_o} p_y \\ 0 &= -p_z - g\rho \\ u_x + v_y + w_z &= 0 \\ \text{and} \quad \rho_t + w\rho_{oz} &= 0 . \end{aligned}\right\} \quad (3.1)$$

The variables u, v and w are the velocity components in the x, y and z directions, respectively. The Coriolis parameter is f, the acceleration due to gravity is g, and the pressure is p. Density is defined by

$$\hat{\rho}(x,y,z,t) = \rho_0(z) + \rho(x,y,z,t) .$$

The Boussinesq approximation is made throughout. Finally, subscripts x, y, z and t represent partial differentiation. The quantity ε is set to either 0 (long-wave approximation) or 1 (general frequency and wavenumber).

All variables are taken to vary as $\exp[i(\omega t + \ell y)]$, so that equations (3.1) reduce to:

$$0 = p_{xx} + \frac{2f\beta}{(f^2-\varepsilon\omega^2)} p_x - p[-\varepsilon \ell^2 + \frac{\ell\beta}{\omega} - \frac{2f^2\beta\ell}{\omega(f^2-\varepsilon\omega^2)}] + (f^2-\varepsilon\omega^2)\left(\frac{p_z}{N^2}\right)_z$$

subject to

$$p_z + \delta \frac{N^2}{g} p = 0 \qquad \text{at} \quad z = 0$$

$$w + h_x u = 0 \qquad \text{at} \quad z = -h(x)$$

$$u = 0 \qquad \text{at} \quad x = 0$$

and

$$u_x = 0 \qquad \text{at} \quad x = XMAX.$$

where N is the Brunt-Väisälä frequency. The fourth boundary condition (Brink, 1982b) replaces the more desirable

$$p \text{ bounded} \qquad \text{as} \quad x \to \infty ,$$

which is not very practical on a finite difference grid. The parameter δ is either 0 (rigid-lid surface) or 1 (free surface) at the user's discretion. Note that only the component of β perpendicular to the coast has been included, so that $f = f_0 - \beta x$. This means that if the land is north of the ocean, then $\beta > 0$, while if the land is south of the ocean, then $\beta < 0$. The component of β parallel to the coast is not included because of the considerable complications involved.

The problem is solved by using the coordinate transformation

$$\theta = \frac{z}{h(x)} .$$

This maps the domain into a rectangle, where the problem is solved on a fixed 17 (vertical) by 25 (horizontal) point grid. Thus, vertical resolution is far better close to shore, in shallow water.

C. Program Input

The user must supply stratification, topography, the Coriolis parameter, and other information. The program then, after converging to a free wave solution, prints out frequency, wavenumber and the modal structure. All program outputs are either in arbitrary or cgs units.

The contents of the input file (file 5) are as follows.

line 1: EPS EST DD1.

EPS is the nominal fractional accuracy desired for the free-wave frequency, i.e. $\Delta\omega/\omega$. The program stops searching when its next frequency estimate agrees with the previous best estimate to this accuracy. Typically, EPS = 0.005.

EST is the fractional initial search increment for ω. Typically, EST = 0.05.

DD1 determines whether there is a rigid lid (DD1 = 0.) or a free surface (DD1 = 1.0). This corresponds to the δ in section 3B.

line 2: ICCM NCALM NITM ISD

ICCM is the number of dispersion curves to be calculated. If ICCM $\neq$ 1, all of the remaining lines of input must be repeated for each dispersion curve.

NCALM is the number of (ω, ℓ) pairs to be calculated along each dispersion curve.

NITM is the maximum number of iterations allowed for finding a single frequency on the dispersion curve. If NITM is exceeded, the program terminates.

ISD determines the direction of search for frequency.

If ISD = 0, the program searches for the free-wave frequency closest to the initial guesses.

If ISD = 1, the program searches only toward frequencies lower than the initial estimates.

If IDS = -1, the program searches only towards higher frequencies.

line 3: F XMAX

F is the Coriolis parameter, which is multiplied by $10^{-5}s^{-1}$ within the program. For example, F = 5, represents $f_0 = 5 \times 10^{-5}s^{-1}$.

XMAX is the offshore extent of the grid in km. Typically, XMAX ~ 2L (see Figure 1a).

line 4: BETA ILWW

BETA is the component of planetary β perpendicular to the coast (see section 3B) entered in units of $s^{-1}cm^{-1}$.

ILWW is ε of section 3B.

If ILWW = 0, the long-wave limit is taken exactly.

If ILWW = 1, the program runs for general frequency and wavenumber.

If ILWW is not equal to 0 or 1, the program terminates.

line 5: NCAL, WH(1)

For a new dispersion curve, NCAL = 1 and WH(1) is any number. When resuming an older curve which has been partially completed, NCAL = 2 and WH(1) is the frequency of the last ℓ of the previous run. This must correspond to RLF (see line 7). This information will allow better estimates at succeeding frequencies. Note that WH(1) is multiplied by $10^{-6}s^{-1}$, so that WH(1) = 0.5 corresponds to $\omega = 0.5 \times 10^{-6}s^{-1}$.

line 6: IDIAG

If IDIAG ≠ 0, then the v, u and ρ fields (as well as p) will be output for the first (ω, ℓ) pair on the dispersion curve.

If IDIAG = 0, then only v (and of course p) will be output.

Regardless of IDIAG, only p will be output for points after the first (ω, ℓ) pair.

line 7: RLF DRL

These parameters determine the wavenumbers for which ω is computed. Specifically,

$$\ell = (\text{RLF} + (n-1)\ \text{DRL}) \times 10^{-7}cm^{-1},$$

for n = 1, 2, 3,... NCALM.

line 8: WW(1) WW(2) WW(3)

These are three initial estimates of the free-wave frequency for the starting wavenumber. The program multiplies these values by $10^{-6}s^{-1}$, so a value of 0.5 corresponds to $\omega = 0.5 \times 10^{-6}s^{-1}$.

line 9: NRX

This is the number of [x, h(x)] pairs to be input. NRX $\geq$ 1 is required.

line 10 and following: X H

These are values of offshore distance (x) in km and water depth (h) in m. There must be NRX pairs, and the first pair must have x = 0. The spacing in x is arbitrary, and the program fills out the topography by linear interpolation. For values of x greater than the last value read, the program assigns the last depth read.

NR DZR ALPH

These are parameters used for reading the profile of N^2 (the Brunt-Väisälä frequency squared).

NR is the number of N^2 values to be read.

DZR is the vertical spacing of N^2 values in m.

ALPH describes the exponential tail on the N^2 profile. Often N^2 is not available from surface to bottom. In this case, an exponential extrapolation is used:

$$N^2 = N_0^2 \exp(\zeta_0 - \zeta)/\text{ALPH})$$

where

N_0^2 is the last N^2 value read,

ζ_0 is the depth of the last N^2 value read, and

ζ is the depth of the point, i.e. $\zeta = -z$.

ALPH is then the exponential length scale of N^2 decay, in km.

CMLT

This is a conversion factor by which the input N^2 are multiplied in order to get units of $(\text{rad/s})^2$. Specifically,

$$N^2(\text{rad}^2/\text{s}^2) = \text{CMLT} \times N^2(\text{user units})$$

following lines: N^2

These are the values of N^2 in user units, one per line. There must be NR regularly spaced values. The first N^2 value should be at z = 0, and N^2 should never equal zero.

NRR

This is the number of [x, r(x)] pairs to be input, where r(x) is a bottom resistance coefficient in cm s^{-1} defined by

$$\frac{1}{\rho_o} \underline{\tau}_B = r(x)\ \underline{v}(x, -h) .$$

This information is used in subroutine LGWH for computing the bottom drag coefficient. NRR ≥ 1 is required.

X R

These are the NRR pairs of offshore distance (x) in km and bottom resistance coefficient (r) in cm/s. The first x value read must be zero. The x spacing is arbitrary, and is filled out by linear interpolation. For values of x greater than the last value read, the last value of R will be used.

D. General Comments

i.) Identifying modes. Generally, the barotropic Kelvin wave ($n = 0$) will have no zero crossings in pressure. The first coastal-trapped wave ($n = 1$) will have one zero crossing, etc. Occasionally, isolated small pockets of reversed sign in p will exist, representing numerical error. These extraneous zero crossings are usually obvious when the modal structure is plotted.

ii.) The program does not generally work well when the shelf-slope width is small relative to the first internal Rossby radius of the deep-ocean. For such a case, the user should experiment to see if ω is stable with respect to small changes in XMAX.

iii.) Since the governing equation is formulated in terms of pressure, a spurious mode exists for $\beta = 0$ and $\omega = f$. It has

$$p = p_o e^{-\ell x} ,$$

with $p_z = 0$. (See Pedlosky, 1979, pp. 79-81 for more detail.) This mode makes the program's performance suspect near $\omega = f$.

iv.) For $\omega > f$, the inertia-gravity wave continuum is quantized by the offshore boundary condition, and the results are useless. The program will stop after three iterations if $\omega > f$ is sought.

v.) The program has trouble finding the barotropic Kelvin wave.

vi.) The program uses an external (IMSL) subroutine to solve the matrix equation.

E. Input Summary

EPS	EST	DD1	
ICCM	NCALM	NITM	ISD
F	XMAX		
BETA	ILWW		
NCAL	WH(1)		
IDIAG			
RLF	DRL		
WW(1)	WW(2)	WW(3)	
NRX			
X	H	NRX times	
NR	DZR	ALPH	
CMLT			
N^2		NR times	
NRR			
X	R	NRR times	

F. Program Output

The program first lists the boundary conditions chosen, and a few parameters, such as f and β.

Next, N^2 at x = XMAX is listed at grid point locations, starting at the bottom of the water column. (The first point is at z = -h, and the last at z = 0). The Δz can be found in the pressure listing.

Then, information about the frequency search is listed. For each iteration, ω, ℓ and $c = \omega/\ell$ are listed, along with

$$RI = \int p^2 dzdx ,$$

a measure of resonance, and IER, an IMSL error code. A message announces convergence.

The v (alongshelf velocity) field is listed, beginning at x = 0. Total depth (h) and depth increment (DZ) are given for each x. Then v is listed, beginning at z = -h. The v field is computed after p has been normalized so that

$$1 = \int_{-h}^{0} p^2 dz \Big|_{x=0} + \int_{0}^{\infty} h_x p^2 dx \Big|_{z=-h} .$$

The pressure field is also listed, and (optionally) u and ρ. All units are consistent so that if p were in dyne/cm^2, then v would be in cm/s.

After the first (ω, ℓ) point on a dispersion curve, only p will be listed, and in this case it is not normalized.

Immediately after the v printout, a_{nn} and b_n are listed. (See Brink, 1982a.) This is an improved version due to Clarke and Van Gorder (1985). Finally, at various points in the output, the contributions of u and v to wave kinetic energy, and of ρ and free-surface height to wave potential energy are given. These can be used to compute the diagnostic

$$R = \frac{\text{kinetic energy}}{\text{potential energy}} .$$

This quantity approaches 1 for a baroclinic Kelvin-like wave, and becomes large (> 10) for a barotropic shelf wave.

G. An Example

Input file:

```
0.005    0.05    0.
1        1       20      0
10.0     200.
0.       0
1        0.
0
0.1      0.5
3.0      3.1     3.2
2
0.       10.
100.     4000.
2        5000.   5.
1.0  E-06
1.375
1.375
1
0.0      0.05
```

This represents a uniform N^2 and a uniformly sloping shelf. After seven iterations, ω/ℓ = 315.05 cm/s for the n = 1 mode. The coupling coefficients are

$$b_n = -0.36614 \times 10^{-2} cm^{-1/2}$$

$$a_{nn} = 0.18207 \times 10^{-8}\ cm^{-1} .$$

Note that the sign of b_n can be positive or negative, depending on the sign of p.

This result can be compared to that obtained by Huthnance (1978) of ω/ℓ = 310 cm/s. Note that he had h(0) = 0.

CHAPTER 4

NEAR-INERTIAL COASTAL-TRAPPED WAVES WITH STRATIFICATION AND TOPOGRAPHY

Documentation for CROSS

A. Introduction

This program finds the wavenumbers (if any) at which the dispersion curves for free coastal-trapped waves approach $\omega = f$. Also determined is the lowest-order pressure field at $\omega = f$. Input parameters include arbitrary bottom topography and vertical stratification. Options include the choice of a free-surface or a rigid-lid boundary condition. The program is designed to be compatible with BIGLOAD2 (Chapter 3).

This program can also be used to find the vertical structures and phase speeds of flat-bottom baroclinic Kelvin waves for arbitrary vertical stratification.

B. Formulation

The solution procedure is based on the near-inertial analysis of Huthnance (1978, Section 6c, see also Chapman, 1983). For a coastal-trapped wave with frequency ω slightly less than f, i.e. $\omega = f(1-\gamma)$ where $\gamma << 1$, then the pressure may be assumed to take the form

$$p = [p_0(z) + \gamma p_1(x,z)]e^{-\ell x}$$

where x is positive offshore, z positive upwards and ℓ the alongshelf wavenumber. The topography is shown in Figure 1a. It can be shown that with these assumptions, the lowest order pressure field obeys

$$\frac{d}{dz}\left(\frac{f^2}{N^2(z)} e^{-2\ell\underline{\overline{X}}(z)} \frac{dp_0}{dz}\right) + \ell^2 e^{-2\ell\underline{\overline{X}}(z)} p_0 = 0 \qquad (4.1)$$

where $N^2(z)$ is the squared Brunt-Väisälä frequency, and $\underline{\overline{X}}(z)$ is the inverse topography defined by $z = -h(\underline{\overline{X}})$ where h is the depth. Note that the <u>topography must be monotonic</u> to be inverted uniquely. The program checks for this. Boundary conditions are

$$\frac{dp_o}{dz} = 0 \qquad \text{at} \quad z = -H \tag{4.2a}$$

$$\frac{dp_o}{dz} + \frac{\delta N^2(0)}{g} p_o = 0 \qquad \text{at} \quad z = 0 \tag{4.2b}$$

where H is the maximum depth at $L < x < XMAX$, g gravitational acceleration, and $\delta = 1$ for a free surface or $\delta = 0$ for a rigid lid. Thus, for known f, topography and stratification, equations (4.1, 4.2) can be solved to find the wavenumber(s) ℓ at which $\omega = f$.

Solutions are found by a shooting technique in which (4.1) is represented in finite difference form and (4.2b) is assumed satisfied. Then ℓ is varied until the integration of (4.1) from $z = 0$ to $z = -H$ results in a pressure distribution which satisfies (4.2a). The wavenumber ℓ is found to a relative fractional accuracy of 10^{-7}.

C. Program Input

The user must supply such information as stratification, topography, the Coriolis parameter, etc. All program outputs are either in arbitrary or cgs units.

The contents of the input file (file 5) are as follows. They are designed to be similar to the contents of the input file used with BIGLOAD2.

line 1: NV DD1

NV is the number of grid points (in the vertical) to be used in the solution. First, the topography is computed exactly as in BIGLOAD2 to obtain 25 depths. Then the topography between the coast and the flat bottom ($0 < x < L$) is filled with NV points by linear interpolation. The maximum NV is 101.

DD1 determines whether there is a rigid lid (DD1 = 0.0) or a free surface (DD1 = 1.0). This corresponds to δ in (4.2b).

line 2: F XMAX

F is the Coriolis parameter, which is multiplied by $10^{-5} s^{-1}$ within the program. Thus, F = 5.0 corresponds to $f = 5. \times 10^{-5} s^{-1}$.

XMAX is the offshore extent of the BIGLOAD2 grid in km. It is used here only to insure that the original 25 depths (before interpolation) are computed as in BIGLOAD2.

line 3: NRX

This is the number of [x, h(x)] pairs to be input. ($NRX \geq 1$).

line 4 and following: X H

These are values of offshore distance (x) in km and water depths (h) in m. There must be NRX pairs, and the first pair must have $x = 0$. The spacing in x is arbitrary, and the program fills out the topography by linear interpolation. For values of x greater than the last value read, the program assigns the last depth read.

NR DZR ALPH

These are parameters used for reading the profile of N^2 (the Brunt-Väisälä frequency squared).

NR is the number of N^2 values to be read.

DZR is the vertical spacing of N^2 values in m.

ALPH describes the exponential tail on the N^2 profile. Often N^2 is not available from surface to bottom. In this case, an exponential extrapolation is used:

$$N^2 = N_0^2 \exp((\zeta_0 - \zeta)/ALPH)$$

where

N_0^2 is the last N^2 value read,

ζ_0 is the depth of the last N^2 value read, and

ζ is the depth of the point, i.e. $\zeta = -z$.

ALPH is then the exponential length scale of N^2 decay, in km.

CMLT

This is a conversion factor by which the input N^2 are multiplied in order to get units of $(rad/s)^2$. Specifically,

$$N^2(rad^2/s^2) = CMLT \times N^2 \text{ (user units)}$$

following lines: N^2

These are the values of N^2 in user units. There must be NR regularly spaced values. The first N^2 value should be at $z = 0$, and N^2 should never equal zero.

NRS

This is the number of wavenumber searches to be made. For each search, an ($\ell_{minimum}$, $\ell_{maximum}$, $\Delta\ell$) set is read. This allows several searches for the same mode or searches for several modes.

following lines: X5 X6 X7

X5 is the minimum ℓ to start the search

X6 is the maximum ℓ to end the search

X7 is the $\Delta\ell$ used to locate the solution.

The program multiplies these values by 10^{-7} to obtain cm^{-1}. Thus, (X5, X6, X7) = (2., 3., .1) corresponds to $\ell_{min} = 2. \times 10^{-7} cm^{-1}$, $\ell_{max} = 3. \times 10^{-7} cm^{-1}$, $\Delta\ell = 0.1 \times 10^{-7} cm^{-1}$. The program then locates a root by shooting using $\ell = \ell_{min} + n\Delta\ell$ $(n = 0, 1, 2...)$ up to $\ell = \ell_{max}$. If a root is located, Newton's method is used to home in on it. If no root is found between ℓ_{min} and ℓ_{max}, then the search moves to the next choice for X5, X6, X7. There must be NRS sets of X5, X6, X7.

D. General Comments

i.) Identifying modes. The barotropic or Kelvin wave ($n = 0$) will have no zero crossings in pressure. It will exist only if a free surface is used (DD1 = 1.0). The first coastal-trapped wave ($n = 1$) will have one zero crossing, the second ($n = 2$) will have two, etc.

ii.) The program is probably best used when BIGLOAD2 predicts a dispersion curve which reaches $\omega \approx .9f$, above which BIGLOAD2 has problems. The BIGLOAD2 dispersion curve can be used to estimate the search parameters for CROSS. The same topography and stratification should be used in both. It may be dangerous to use CROSS if the dispersion curves are unknown, because they may never reach $\omega = f$ and CROSS will only report that no root was found in the specific interval (i.e. CROSS cannot tell whether or not a dispersion curve ever reaches $\omega = f$, only whether or not it reaches $\omega = f$ in the specified interval).

iii.) This program can find flat-bottom baroclinic Kelvin wave modal structures and phase speeds by using one depth and the desired stratification (NRX = 1, [x, h] = [0., H]). Since Kelvin waves are nondispersive, the phase speed at f is the same as at any other frequency.

E. Input Summary

NV	DD1		
F	XMAX		
NRX			
X	H		NRX times
NR	DZR	ALPH	
CMLT			
N^2			NR times
NRS			
X5	X6	X7	NRS times

F. Program Output

The program first lists the surface boundary condition chosen, and the parameters used (f, XMAX). Then Δz is listed followed by the inverse topography at $z = -(n\Delta z)$ where $n = 0, 1, 2...$ NV-1. Next N^2 is listed also at $z = -(n\Delta z)$.

For each search, if a solution is found, the wavenumber ℓ and phase speed (f/ℓ) are listed followed by the pressure structure $(p_o e^{-\ell x})$

normalized by $\left(\int_{-H}^{0} p_o^2 dz \right)^{1/2}$. The pressure is listed at $z = -(n + 1/2)\ \Delta z$ where n = 1,2,3...NV-1. That is, the pressures are given midway between the topography grid points.

G. An Example

```
 51        0.
 10.     200.
  2
  0.       1.
100.    4000.
  2     5000      5.
1.0 E-6
 56.25
 56.25
  2
  2.   3.   .1
  4.   5.   .1
```

This represents a uniformly sloping shelf with uniform stratification. The analytical solution of (4.1,4.2) is $\ell = \frac{n\pi}{L} [(\frac{NH}{fL})^2 - 1)]^{-1/2}$ (Huthnance, 1978, p. 83) from which $\ell_1 = 1.11 \times 10^{-7} cm^{-1}$, $\ell_2 = 2.22 \times 10^{-7} cm^{-1}$. The values predicted by CROSS are $\ell_1 = 1.11 \times 10^{-7} cm^{-1}$, $\ell_2 = 2.22 \times 10^{-7} cm^{-1}$.

CHAPTER 5

WIND-DRIVEN MOTIONS

Documentation for BIGDRV2

A. Introduction

This program computes the velocity, pressure and density response of stratified shelf and slope waters to a time and space harmonic wind stress. Options include using

a) rigid lid or free surface,

b) "long wave" or general parameters,

c) alongshelf or cross-shelf winds.

The cross-shelf distributions of bottom resistance coefficient and of wind stress are at the user's discretion.

B. Formulation

The interior region (away from surface and bottom boundary layers) is described by the linear, inviscid equations:

$$\varepsilon u_t - fv = \frac{-1}{\rho_o} p_x \tag{5.1a}$$

$$v_t + fu = \frac{-1}{\rho_o} p_y \tag{5.1b}$$

$$0 = - p_z - g\rho \tag{5.1c}$$

$$u_x + v_y + w_z = 0 \tag{5.1d}$$

$$0 = \rho_t + w\rho_{oz} \ . \tag{5.1e}$$

The variables u, v and w are the velocity components in the x, y and z directions, respectively. The Coriolis parameter is f, the acceleration due to gravity is g, and the pressure is p. Density is defined by

$$\hat{\rho}(x,y,z,t) = \rho_o(z) + \rho(x,y,z,t) \ .$$

The Boussinesq approximation is made throughout. Finally, subscripts x, y, z and t represent partial differentiation. The quantity ε is set to either 0 (long-wave approximation) or 1 (general frequency and wavenumber). Equations (5.1) can be reduced to a single field equation for pressure,

$$0 = p_{xxt} + \varepsilon p_{yyt} + \left(f^2 + \varepsilon \frac{\partial^2}{\partial t^2}\right)\left(\frac{p_z}{N^2}\right)_{zt} , \tag{5.2}$$

where N^2 is the Brunt-Väisälä frequency squared.

The problem is solved by assuming wind stress in the form of

$$\tau_o{}^y = T^y(x) \exp[i(\omega t + \ell y)] ,$$

or

$$\tau_o{}^x = T^x(x) \exp[i(\omega t + \ell y)] ,$$

and all of the variables (u, v, ρ, p) are assumed to have a similar y and t dependence. Given these assumptions, (5.2) reduces to

$$0 = p_{xx} - \ell^2 \varepsilon p + (f^2 - \varepsilon\omega^2)\left(\frac{p_z}{N^2}\right)_z . \tag{5.3}$$

The boundary conditions are

$$0 = w + h_x u + (f^2 - \omega^2)^{-1}[-(frv_B + i\omega r\varepsilon u_B)_x + \omega \ell \varepsilon r v_B + i \ell f \varepsilon r u_B] \tag{5.4a}$$

at $z = -h(x)$,

$$0 = -\rho_o w + iw\delta g^{-1} p + (f^2 - \varepsilon\omega^2)^{-1}[(iw\varepsilon T^x + fT^y)_x + \varepsilon \ell (-ifT^x - \omega T^y)] \tag{5.4b}$$

at $z = 0$,

$$0 = u_x \qquad \text{at } x = XMAX , \tag{5.4c}$$

and

$$0 = -i(\ell f_p + \omega p_x)h + f(T^y - \rho_o r v_B) + i\omega\varepsilon(T^x - \rho_o r u_B) \tag{5.4d}$$

at $x = 0$.

The variables u_B and v_B are the interior velocities evaluated at the bottom:

$$u_B = -i(f^2 - \varepsilon\omega^2)^{-1}(\ell fp + \omega p_x)\Big|_{z=-h} \tag{5.5a}$$

and

$$v_B = (f^2 - \varepsilon\omega^2)^{-1}(fp_x + \varepsilon\omega\ell p)\Big|_{z=-h} \,. \tag{5.5b}$$

The parameter δ is either 0 (rigid-lid surface) or 1 (free surface) at the user's discretion. Implicit in (5.4a,b) is the assumption that the surface and bottom frictional boundary layers are infinitesimally thin. The offshore boundary condition, (5.4c) has been shown to be reasonably accurate for free coastal-trapped waves (Brink, 1982b), and is applied here as well.

The coastal boundary condition (5.4d) has been justified by Clarke and Brink (1985). It states that the net onshore transport (interior plus Ekman) sums to zero, with the further assumption that $u_z \cong 0$ at $x = 0$. In practice, this appears to be reasonable. The work of Mitchum and Clarke (1985) suggests that the "coast" be placed such that

$$h(0) = \frac{6r(0)}{f} \,, \tag{5.6}$$

where $r(x)$ is defined by

$$\underline{\tau}_B = \rho_0 r \underline{v}_B \,.$$

The general problem defined by (5.3) and (5.4) reduces to that of Clarke and Brink (1985) when

$$\delta = 0$$

$$\varepsilon = 0$$

$$T^x = 0$$

$$T_x^y = 0 \,.$$

Note that the cross-shelf component of wind stress only enters when the long-wave assumption is not made. Our sensitivity studies suggest that the cross-shelf wind stress is rarely an effective driving agency except near resonance with a coastal-trapped wave.

C. Program Input

The user provides an N^2 profile, bottom topography information, choices of assumptions (e.g. rigid lid), the bottom resistance coefficient, wind stress profiles, f, ω and ℓ. The program returns v, u, ρ and p in the form of amplitude and phase, as well as diagnostic information.

A full explanation of input is given here, and a compact listing in section 5E. All data are read from file 5.

line 1: ICCM

This is the number of (ω, ℓ) pairs for which the program will run. All other parameters stay the same for each run.

line 2: F XMAX

F is the Coriolis parameter, multiplied by $10^{-5} s^{-1}$ within the program. For example, F = 5. represents $f = 5. \times 10^{-5} s^{-1}$.

XMAX is the offshore extent of the grid in km. Typically, XMAX should be about twice the shelf-slope width.

line 3: ILW IRL IXY

ILW determines whether the long-wave assumption is made. It is ε in section 5B.

If ILW = 1, general frequency and wavenumber.

If ILW = 0, long-wave limit.

If ILW is neither 0 nor 1, the program defaults to ILW = 0.

IRL determines whether the rigid-lid assumption is made. It is δ in section 5B.

If IRL = 1, free surface.

If IRL = 0, rigid lid.

If IRL is neither 0 nor 1, the program defaults to IRL = 0.

IXY determines which wind stress component is used.

If IXY = 1, cross-shelf (T^x) winds.

If IXY = 0, alongshelf (T^y) winds.

If IXY is neither 0 nor 1, the program defaults to IXY = 0.

If the user sets IXY = 1 and ILW = 0, the program automatically stops and prints out an error message.

line 4: NRX

This is the number of [x,h(x)] pairs to be input. $50 \geq \text{NRX} \geq 1$ is required.

line 5 and following: X H

These are the values of offshore distance (x) in km and water depth (h) in m. There must be NRX pairs, and the first pair must have x = 0. The spacing in x is arbitrary, and the program fills out the topography by linear interpolation. For values of x greater than the last value read, the program assigns the last depth read.

NR DZR ALPH

These are parameters used for reading the profile of N^2 (the Brunt-Väisälä frequency squared).

NR is the vertical spacing of N^2 values to be read.

DZR is the vertical spacing of N^2 values in m.

ALPH describes the exponential tail of the N^2 profile. Often N^2 is not available from surface to bottom. In this case, an exponential extrapolation is used:

$$N^2 = N_o^2 \exp((\zeta_o - \zeta)/\text{ALPH})$$

where

N_o^2 is the last N^2 value read,

ζ_o is the depth of the last N^2 value read, and

ζ is the depth of the point, i.e. $\zeta = -z$.

ALPH is then the exponential length scale of N^2 decay, in km.

CMLT

This is a conversion factor by which the input N^2 are multiplied in order to get units of $(\text{rad/s})^2$. Specifically,

$$N^2(\text{rad}^2/\text{s}^2) = \text{CMLT} \times N^2 \text{ (user units)}.$$

following lines: N^2

These are the values of N^2 in user units, one per line. There must be NR regularly spaced values. The first N^2 value should be at $z = 0$, and N^2 should never equal zero.

NF

This is the number of $[x,r(x)]$ pairs to be read. NF ≥ 1 is required. The format for reading the bottom resistance coefficient r is exactly like that for depth h.

following lines: X R

These are values of offshore distance (x) in km and of the resistance coefficient (r) in cm/s. There must be NF pairs, and the first pair must have $x = 0$. The spacing in x is arbitrary, and the program fills out r by linear interpolation. For values of x greater than the last value read, the program assigns the last r value read.

NW

This is the number of values of $T(x)$ to be read. Whether it is T^x or T^y depends upon the choice of IXY in line 3. If NW = 0, T = 1 dyne/cm^2 for all x.

following lines: X T

These are values of offshore distance (x) in km and wind stress amplitude in dyne/cm^2. If NW = 0, these lines should not be inserted. The first pair must be for $x = 0$. The x spacing is arbitrary, and the program fills out the wind stress by linear interpolation. For values of x greater than the last value read, the program assigns the last wind stress read.

following lines: W RL

These are the frequency, wavenumber (ω, ℓ) pairs for which the program runs. There should be ICCM lines. Units are s^{-1} and cm^{-1} respectively. The program includes no internal multiplications for these parameters.

D. <u>General Comments</u>

i.) Using $r = 0$ results in a divide by zero. Thus, inviscid problems should not be attempted.

ii.) Using $\ell = 0$ causes no problem until the program is about to print the last values of pressure. An error message will result, but there is nothing wrong with the program's output, which is virtually complete.

iii.) No external subroutines are required.

iv.) The program uses the same 25 x 17 stretched grid as in BIGLOAD2.

E. Input Summary

ICCM

F XMAX

ILW IRL IXY

NRX

X H — NRX times

NR DZR ALPH

CMLT

N^2 — NR times

NF

X R — NF times

NW

X T — NW times

W RL — ICCM times

F. Program Output

The program first lists f and XMAX in s^{-1} and cm respectively. The assumptions chosen on line 3 of input are then stated.

Input functions are then listed:

i.) N^2 $(rad/s)^2$ at x = XMAX, beginning at z = -h up to z = 0 in increments of Δz at x = XMAX (i.e. h(XMAX)/16).

ii.) r(x) (cm/s), beginning at x = 0 out to x = XMAX in increments of Δx (i.e. XMAX/24).

iii.) T(x) (dyne/cm^2) in the same format as r(x).

Following this, the program prints out ω (s^{-1}) and ℓ (cm^{-1}), and the results for this particular input pair. All field variables (v, u, ρ, p) are listed as amplitude and phase at each grid point, beginning at the bottom for each x. For each x, water depth h (cm) and Δx (cm) are also given. The phase is negative for wind leading the response. The field variables are:

iv.) v (cm/s), followed by the v contribution to kinetic energy per unit length of coast (erg/cm), and the alongshelf bottom stress beginning at x = 0 (dyne/cm^2).

v.) u (cm/s), followed by the u contribution to kinetic energy per unit length of coast (erg/cm).

vi.) ρ (σ_t units), followed by the ρ contribution to fluctuating potential energy per unit length of coast (erg/cm). This is followed immediately by the free-surface height contribution to fluctuating potential energy. The free-surface contribution is set to zero if a rigid lid is imposed. At this point, the total (kinetic plus potential) fluctuating energy per unit length of coast (erg/cm) is given, along with the ratio of kinetic to potential energy R (Brink, 1982b). For R $\gtrsim$ 10, the response is generally highly barotropic, and for R $\lesssim$ 2, it can be regarded as very baroclinic.

vii.) p (dyne/cm^2).

G. <u>An Example</u>

Input File:

```
1
10.0    200.
1       1         0
2
0.      30.
100.    4000.
2       5000.     5.
1.0  E-06
1.375
1.375
1
0.0     0.05
0
1.0 E-05          2.0 E-08
```

The resulting output has the following energy components:

v → 2.94×10^{14} erg/cm

u → 0.08×10^{14} erg/cm

ρ → 0.31×10^{14} erg/cm

p → 0.45×10^{12} erg/cm

and R = 9.6.

The alongshelf velocity (Figure 2) is uniform in depth at x = 0 at 34.2 cm/s and a phase of -9°. The v maximum is at the surface at x = 8.33 km (85, -48°).

The cross-shelf velocity is depth-independent at x = 0 (2.5, -21°), and has a maximum at the surface at x = 8.33 km (6.6, -136°).

The maximum in density is at the bottom at x = 8.33 km with $\rho = 0.032\ \sigma_t$ and a phase of -39°. Density goes nearly to zero at the surface, and its phase is consequently unreliable there. When a rigid lid is used and NW = 0, density fluctuations are zero at the free surface in the long-wave limit.

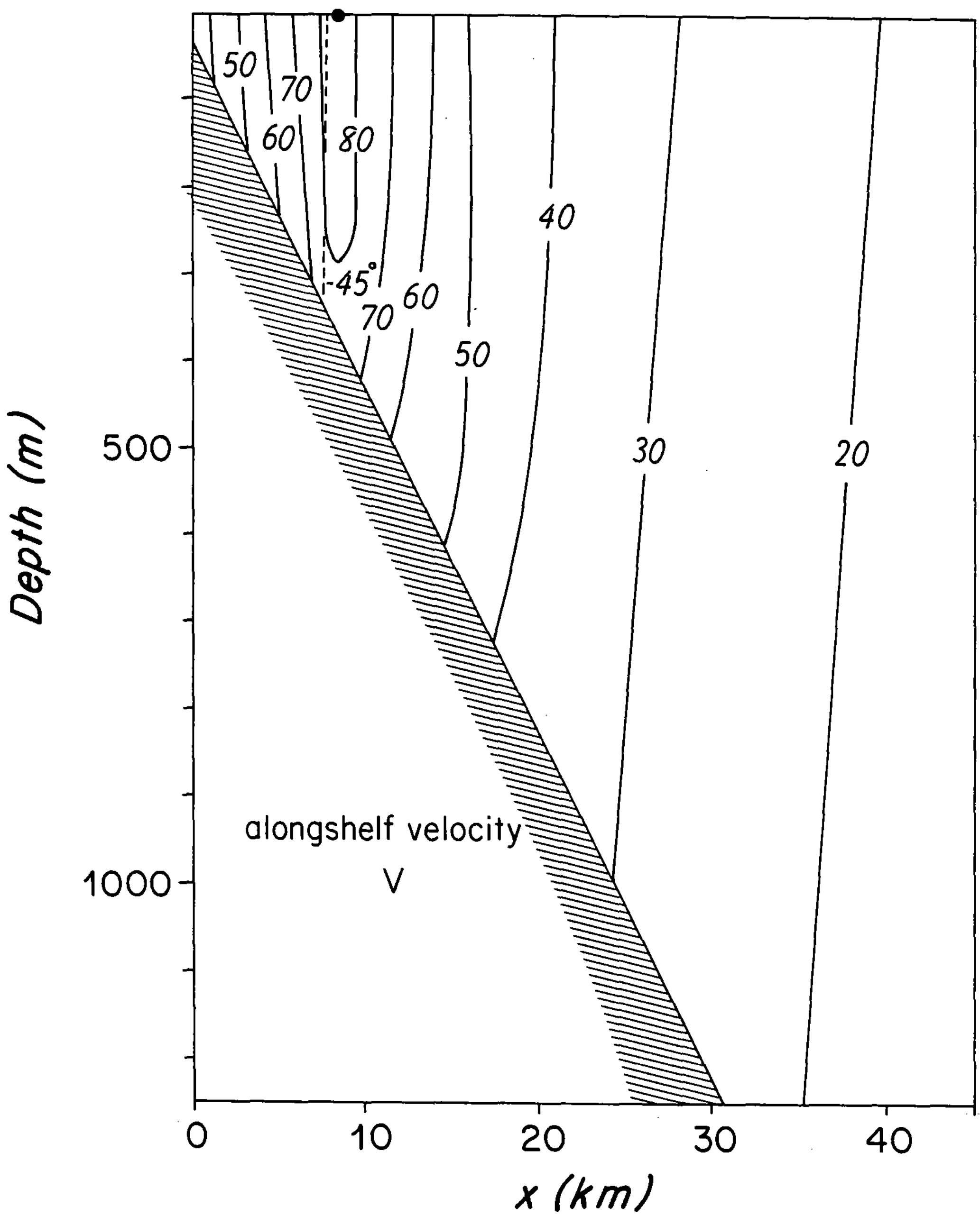

Figure 2: Alongshelf velocity for the example in section 5G. Amplitude (cm/s) is shown in solid lines and phase is shown by dashed contours. Only the upper 1250 m is shown.

The pressure is depth-independent and at a maximum at x = 0 (0.190 x 10^5 dyne/cm^2, phase = 131°). To get sea level, divide by g = 981 cm/s^2 to obtain 19 cm.

All fields become weaker far offshore and at great depth. The v and p fields have a roughly 180° phase change far offshore. The strength and structure of response vary radically near (ω, ℓ) resonances with free coastal-trapped waves.

Word of Caution

We have performed what we feel are extensive tests with all of the programs contained herein. However, we cannot guarantee that the programs will give sensible results in all situations. That is, it may be possible to find parameter combinations for which a program will complete the run, but the computed results will not make physical sense. Therefore, we cannot be responsible for the ways in which the programs are applied. On the other hand, if actual programming bugs or inconsistencies appear which are not mentioned in this document, please contact us with the details.

Acknowledgements

We thank the many people who helped us with these programs, both in setting up the original efforts, and by their comments as users.

This work was supported by National Science Foundation (NSF) grant OCE84-08563.

REFERENCES

Brink, K. H., 1982a. The effect of bottom friction on low-frequency coastal trapped waves. Journal of Physical Oceanography, 12, 127-133.

Brink, K. H., 1982b. A comparison of long coastal trapped wave theory with observations off Peru. Journal of Physical Oceanography, 12, 897-913.

Brink, K. H., 1983. Low-frequency free wave and wind-driven motions over a submarine bank. Journal of Physical Oceanography, 13, 103-116.

Brink, K. H. and J. S. Allen, 1978. On the effect of bottom friction on barotropic motion over the continental shelf. Journal of Physical Oceanography, 8, 919-922.

Buchwald, V. T. and J. K. Adams, 1968. The propagation of continental shelf waves. Proceedings of the Royal Society of London, A305, 235-250.

Chapman, D. C., 1983. On the influence of stratification and continental shelf and slope topography on the dispersion of subinertial coastally trapped waves. Journal of Physical Oceanography, 13, 1641-1652.

Clarke, A. J. and K. H. Brink, 1985. The response of stratified, frictional shelf and slope waters to fluctuating large-scale low-frequency wind forcing. Journal of Physical Oceanography, in press.

Clarke, A. J. and S. Van Gorder, 1985. A method for estimating wind driven shelf and slope water flow. Submitted to Journal of Physical Oceanography.

Huthnance, J. M., 1978. On coastal trapped waves: analysis and numerical calculation by inverse iteration. Journal of Physical Oceanography, 8, 74-92.

Mitchum, G. T. and A. J. Clarke, 1985. A coastal boundary condition for large-scale, low-frequency flow. Submitted to Journal of Physical Oceanography.

Mysak, L. A., P. M. LeBlond and W. J. Emery, 1979. Trench waves. Journal of Physical Oceanography, 9, 1001-1013.

Pedlosky, J., 1979. Geophysical Fluid Dynamics. Springer-Verlag, New York, 624 pp.

Wang, D-P. and C. N. K. Mooers, 1976. Coastal trapped waves in a continuously stratified ocean. Journal of Physical Oceanography, 6, 853-863.

Program BTCSW

Listing

```
      PROGRAM BTSW
      COMMON RH(400),RHX(400),A(1200),B(400),V(400),VX(400),VXX(400)
      DIMENSION BH(400),WW(3),RX(3),RLH(400),WH(400),CH(400)
      DIMENSION WF(400)
C
C            MAIN PROGRAM CALLS ALL OF THE PIECES
C
C            NN LESS THAN OR EQUAL 100
      IMD = 1
      READ(5,*) IMDM,NN
C            DD1 = 1.  FREE SURFACE
C            DD1 = 0.  RIGID LID
C            IDD3 = 1  U = 0 AT X = XMAX
C            IDD3 = 0  UX = 0 AT X = XMAX
C            IDD3 = 2 REAL B.C. AT X = XMAX FOR VX,HX = 0. THERE
C            IDD4 =1   U = 0 AT X = 0
C            IDD4 = 0  UX = 0 AT X = 0
C            IDD4 = 2  DECAY B.C. AT X = 0 FOR VX,HX = 0 THERE
C            IPC = 0  REDUCED PRINT OUT
C            IUP = 0 SEARCH IN U
C            IUP = 1 SEARCH IN P
C            ILLW = 0 STRICTLY LONG WAVE
C            ILLW = 1 GENERAL FREQUENCY AND WAVENUMBER
C                    REVISED 11/9/84
    2 READ(5,*) NITM,ISD,EPS,DEL
      READ(5,*) IUP,ILLW
      READ(5,*) IDD1,IDD3,IDD4
      DD1 = FLOAT(IDD1)
      NCAL = 1
      READ(5,*) NCALM,ILW
      READ(5,*) RLF,DRL
      READ(5,*) IPC
      DRL = DRL*1.0E-08
      RLF = RLF*1.0E-08
      READ(5,*) F,XMAX
      XMAX = XMAX*1.0E+05
      F = F*1.0E-05
      DX = XMAX/FLOAT(NN-1)
      WRITE(6,907) F,XMAX,DX
      IF (IUP.NE.0) GO TO 315
  312 WRITE(6,924)
      IUP = 0
      GO TO 320
  315 IF (IUP.NE.1) GO TO 312
      WRITE(6,925)
  320 IF (IDD4.NE.0) GO TO 305
      WRITE(6,921)
      GO TO 310
  305 IF (IDD4.EQ.2) GO TO 306
      IDD4 = 1
      WRITE(6,922)
      GO TO 310
  306 WRITE(6,928)
  310 IF (IDD1.EQ.1) GO TO 3
      DD1 = 0.
      WRITE(6,917)
      GO TO 4
    3 WRITE(6,918)
    4 IF (IDD3.EQ.1) GO TO 6
      IF (IDD3.EQ.2) GO TO 7
```

```
      IDD3 = 0
      WRITE(6,919)
      GO TO 8
6     WRITE(6,920)
      GO TO 8
7     WRITE(6,923)
8     IF (ILLW.EQ.1) GO TO 11
      ILLW = 0
      NCALM = 1
      WRITE(6,926)
      GO TO 12
11    WRITE(6,927)
12    REPS = FLOAT(ILLW)
      CALL DEP(NN,XMAX)
      CALL VRD(NN,XMAX,F)
      DO 10 I = 1,NCALM
10    RLH(I) = DRL*FLOAT(I-1) + RLF
      RL = RLH(NCAL)
      READ(5,*) (WW(J),J=1,3)
      DO 5 J = 1,3
5     WW(J) = WW(J)*1.0E-05
1     RB = 0.
      WRITE(6,908)
      DO 18 J = 1,3
      W = WW(J)
      IF (IUP.EQ.1) GO TO 330
      CALL MUTS(NN,XMAX,DD1,F,W,RL,IDD3,RI,IDD4,REPS)
      GO TO 9
330   CALL MATS(NN,XMAX,DD1,F,W,RL,IDD3,RI,IDD4,REPS)
9     CP = W/RL
      CALL CRLC(W,RL,NN,ICR)
      WRITE(6,905) W,RL,CP,RI,ICR
      RX(J) = RI
      IF ( RI.LT.RB) GO TO  18
      RB = RI
      ICRH = ICR
      WB = W
      DO 15 I = 1,NN
15    BH(I) = B(I)
18    CONTINUE
      NIT = 3
      IGP = 0
20    CALL NGW(WW,RX,WB,WN,ISUC,EPS,DEL,IN,ISD,IGP)
      IF (ISUC.EQ.1) GO TO 100
      IF (IUP.EQ.1) GO TO 340
      CALL MUTS(NN,XMAX,DD1,F,WN,RL,IDD3,RI,IDD4,REPS)
      GO TO 345
340   CALL MATS(NN,XMAX,DD1,F,WN,RL,IDD3,RI,IDD4,REPS)
345   IF (IN.NE.0) GO TO 29
      IF (RI.GT.RB) GO TO 21
      IF (WN.LT.WW(2)) GO TO 23
22    IN = 3
      GO TO 29
23    IN = 1
      GO TO 29
21    IF (WN.GT.WW(2)) GO TO 23
      GO TO 22
29    RX(IN) = RI
      WW(IN) = WN
      CP = WN/RL
```

```
      CALL CRLC(WN,RL,NN,ICR)
      WRITE(6,905) WN,RL,CP,RI,ICR
      IF ( RI.LT.RB) GO TO 30
      WB = WN
      ICRH = ICR
      RB = RI
      DO 25 I = 1,NN
   25 BH(I) = B(I)
   30 NIT = NIT + 1
      IF ( NIT.LT.NITM) GO TO 20
      WRITE(6,902) NIT
      GO TO 140
  100 DO 110 I = 1,NN
  110 BH(I) = BH(I)/RB
      CC = WB/RL
      IF ( ICRH.EQ.0) GO TO 115
      WRITE(6,911) ICRH
  115 WRITE(6,903) WB,RL,CC,EPS,DX,NIT
      IF (NCAL.EQ.1) GO TO 120
      IF (IPC.EQ.0) GO TO 150
  120 IF (IUP.EQ.1) GO TO 350
      WRITE(6,913)
      GO TO 360
  350 WRITE(6,912)
  360 WRITE(6,904) (BH(I),I=1,NN)
      IF (IUP.EQ.0) GO TO 135
      CALL MUTS(NN,XMAX,DD1,F,WB,RL,IDD3,RI,IDD4,REPS)
      DO 143 I = 1,NN
  143 B(I) = B(I)/RI
      WRITE(6,913)
      WRITE(6,904) (B(I),I = 1,NN)
  135 IF (RH(1).EQ.0.0) GO TO 150
      CALL MATS(NN,XMAX,DD1,F,WB,RL,IDD3,RI,IDD4,REPS)
      DO 130 I = 1,NN
  130 B(I) = B(I)/RI
      IF (IUP.EQ.1) GO TO 145
      WRITE(6,912)
      WRITE(6,904) (B(I),I=1,NN)
  145 IF (ILW.EQ.0) GO TO 150
      IF (NCAL.NE.1) GO TO 150
      IF ( RH(1).EQ.0.) GO TO 150
      IF (IDD4.NE.1) GO TO 150
      WRITE(6,908)
      CALL LGWV(NN,XMAX,F,WF,WB,RL)
  150 WH(NCAL) = WB
      CH(NCAL) =CC
      NCAL = NCAL + 1
      IF ( NCAL.GT.NCALM) GO TO 250
      RL = RLH(NCAL)
      IF (NCAL.GE.3) GO TO 200
      WW(2) = CC*RL
      GO TO 205
  200 I1 = NCAL - 2
      I2 = NCAL - 1
      CG = (WH(I2) - WH(I1))/(RLH(I2) - RLH(I1))
      WW(2)=  WH(I2) + CG*(RLH(NCAL)-RLH(I2))
  205 WW(1) = WW(2)*(1. - DEL)
      WW(3) = WW(2)*(1. +DEL)
      GO TO 1
  250 WRITE(6,909)
```

```
      DO 260 I = 1,NCALM
260   WRITE(6,910) WH(I),RLH(I),CH(I)
      IMD = IMD +1
      IF (IMD.LE.IMDM) GO TO 2
902   FORMAT(//'  USED UP',I3,' ITERATIONS'/)
903   FORMAT(//'  CONVERGED: W,L,C,EPS,DX,NIT =',5E15.5,I5)
904   FORMAT(10E13.5)
905   FORMAT(/'  W,RL,CP,RI,ICR =',4E15.5,I10)
907   FORMAT(//'  F,XMAX,DX =',3E15.5/)
908   FORMAT(///)
909   FORMAT(///'       W                  L                    C'/)
910   FORMAT(3E15.5)
911   FORMAT(/'  SOLUTION HAS ',I3,' CRITICAL LAYERS'/)
912   FORMAT(/'   ZETA'/)
913   FORMAT(/'    U'/)
917   FORMAT(/'  RIGID LID'/)
918   FORMAT(/'  FREE SURFACE'/)
919   FORMAT('  UX = 0 AT X = XMAX'/)
920   FORMAT('  U = 0 AT X = XMAX'/)
921   FORMAT('  UX = 0 AT X = 0'/)
922   FORMAT('  U = 0 AT X = 0'/)
923   FORMAT('  REAL B.C.AT X = XMAX')
924   FORMAT(/' SEARCH IN U')
925   FORMAT(/' SEARCH IN P')
926   FORMAT(' LONG WAVE EXACTLY')
927   FORMAT(' GENERAL FREQUENCY AND WAVENUMBER')
928   FORMAT(' DECAY CONDITION AT X = 0')
140   STOP
      END

      SUBROUTINE NGW(WW,RX,WB,WN,ISUC,EPS,DEL,IN,ISD,IGP)
      DIMENSION WW(3),RX(3)
C
C               SUBROUTINE TO GUESS THE NEXT W
C
5     IC = 0
      DO 10 I = 1,2
      I1 = I +1
      AA = ABS(WW(I1))
      BB = ABS(WW(I))
      IF (AA.GT.BB) GO TO 10
      IC = 1
      RI = RX(I)
      WX = WW(I)
      RX(I) = RX(I1)
      WW(I) = WW(I1)
      WW(I1) = WX
      RX(I1) = RI
10    CONTINUE
      IF (IC.NE.0) GO TO 5
      ISUC = 0
      IF ( RX(3).GT.RX(2)) GO TO 150
      IF (RX(1).GT.RX(2)) GO TO 160
      IL = 1
      IH = 3
      IF (RX(3).GT.RX(1)) GO TO 9
      IL = 3
      IH = 1
9     WB1 = (WW(1) +WW(2))/2.
      WB2 = (WW(2) + WW(3))/2.
```

```
           RW1 = (RX(2) - RX(1))/(WW(2) - WW(1))
           RW2 = (RX(3) - RX(2))/(WW(3) -WW(2))
           A = (RW2 -RW1)/(WB2 - WB1)
           B = RW1  - A*WB1
           WN = -B/A
           EP= (WN -WB)/WB
           EP = ABS(EP)
           IF ( EP.LT.EPS) GO TO 100
           IGP = IGP + 1
           IF (WN.LT.WW(1)) GO TO 15
           IF (WN.LT.WW(3)) GO TO 20
   15      WN = (WW(2) +WW(IH))/2.
   20      IN = 0
           GO TO 130
   100      ISUC = 1
           IF (IGP.NE.0) GO TO 130
           IGP = IGP + 1
           ISUC = 0
           WN = (WW(2) + WW(IL))/2.
           GO TO 20
   150     IF (RX(1).GT.RX(2)) GO TO 180
   155     IF (ISD.EQ.1) GO TO 165
           WN = WW(3)*(1. + DEL)
           WW(1) = WN
           RX(1) = 0.
           IN = 1
           GO TO 130
   160     IF (RX(3).GT.RX(2)) GO TO 180
   165     IF (ISD.EQ.-1) GO TO 155
           WN = WW(1)*(1. - DEL)
           WW(3) = WN
           RX(3) = 0.
           IN = 3
           GO TO 130
   180     IF (RX(3).GT.RX(1)) GO TO 155
           GO TO 165
   130     RETURN
           END

           SUBROUTINE CALC(NN,RX)
           COMMON RH(400),RHX(400),A(1200),B(400),V(400),VX(400),VXX(400)
           DOUBLE PRECISION RI,DSQRT
    C
    C                 SUBROUTINE TO CALCULATE THE INTEGRAL OF RESPONSE SQUARED
    C
           RI = (B(1)**2 + B(NN)**2)/2.
           NX = NN - 1
           DO 5 I = 2,NX
    5      RI = RI + B(I)**2
           RI = DSQRT(RI)
           RX = RI
           RETURN
           END

           SUBROUTINE DEP(NN,XMAX)
           COMMON RH(400),RHX(400),A(1200),B(400),V(400),VX(400),VXX(400)
           READ(5,*) NRX
    C
    C                 SUBROUTINE TO READ AND INTERPOLATE THE DEPTH PROFILE
    C                              RH = DEPTH
```

```
C                    RHX = X DERIVATIVE OF DEPTH
C
          DO 10 I = 1,NRX
          READ(5,*) A(I),B(I)
          A(I) = A(I)*1.0E+05
   10     B(I) = B(I)*100.
          DX = XMAX/FLOAT(NN-1)
          RH(1) = B(1)
          DO 20 N = 2,NN
          X = DX*FLOAT(N-1)
          IF (X.GT.A(NRX)) GO TO 15
          IC = 0
          DO 8 J = 2,NRX
          IF (IC.NE.0) GO TO 8
          I = J
          IF (X.GT.A(I)) GO TO 8
          IC = I
    8     CONTINUE
          IM = I -1
          AA = (B(I) -B(IM))/(A(I) - A(IM))
          XX = X - A(IM)
          RH(N) = B(IM) +AA*XX
          GO TO 20
   15     RH(N) = B(NRX)
   20     CONTINUE
          RHX(1) = (RH(2) - RH(1))/DX
          NM = NN -1
          RHX(NN) = (RH(NN) - RH(NM))/DX
          D2 = 2.*DX
          DO 30 N = 2,NM
          IP = N +1
          IM = N -1
   30     RHX(N) = (RH(IP) - RH(IM))/D2
          WRITE(6,903)
          WRITE(6,902) (RH(N),N=1,NN)
  902     FORMAT(10E13.5)
  903     FORMAT(//'  DEPTH IN CM'/)
          RETURN
          END

          SUBROUTINE LGWV(NN,XMAX,F,WFF,WB,RL)
          COMMON RH(400),RHX(400),A(1200),B(400),V(400),VX(400),VXX(400)
          DIMENSION WFF(400)
C
C               SUBROUTINE TO CALCULATE THE BN AND ANN IN THE LONG WAVE
C                           LIMIT
C
          DX = XMAX/FLOAT(NN -1)
          DXX = 2.*DX
          GF = 980./F
          CALL WFR(NN,XMAX,WFF)
          A(1) = 0.
          DO 100 N = 2,NN
          A(N) = (RHX(N)*B(N) + RHX(1)*B(1))/2.
          NX = N - 1
          IF (NX.LE.1) GO TO 90
          DO 20 I = 2,NX
   20     A(N) = A(N) + RHX(I)*B(I)
   90     A(N) = A(N)*DX
          A(N) = A(N) + RH(1)*B(1) -RH(N)*B(N)
```

```
  100 A(N) = -GF*A(N)
      NX = NN - 1
      WF = RHX(1)*(A(1)/RH(1))**2
      WF = WF + RHX(NN)*(A(NN)/RH(NN))**2
      IFZ = 0
      WF = WF/2.
      DO 150 N = 2,NX
      IF ( IFZ.NE.0) GO TO 150
      IF (RHX(N).NE.0.) GO TO 140
      IFZ = N
  140 WF = WF + RHX(N)*(A(N)/RH(N))**2
  150 CONTINUE
      WF = WF*DX
      WF = SQRT(WF*WF)
      WF = SQRT(WF)
      DO 160 N = 1,NN
  160 A(N) = A(N)/WF
      NH = IFZ - 1
      PX = (A(2) -A(1))/(DX*RH(1))
      WF = 0.5*PX*PX*WFF(1)
      BN = 0.5*RHX(NH)*A(NH)/(RH(NH)**2)
      NH = NH -1
      DO 180 N = 2,NH
      BN = BN + RHX(N)*A(N)/(RH(N)**2)
      I1 = N - 1
      I2 = N + 1
      PX = (A(I2) -A(I1))/DXX
      PX = PX/RH(N)
  180 WF = WF+ PX*PX*WFF(N)
      BN = BN*DX
      WF = WF*RH(1)*DX
      WRITE(6,905)
      WRITE(6,901) WF
      WRITE(6,903) BN
      WRITE(6,904)
      WRITE(6,902) (A(I),I=1,NN)
      C = WB/RL
      NX = NN - 1
      WF = RHX(1)*B(1)*B(1)*0.5
      DO 210 I = 2,NX
  210 WF = WF +RHX(I)*B(I)*B(I)
      WF = WF + 0.5*RHX(NN)*B(NN)*B(NN)
      WF = WF*DX
      WF = WF + RH(1)*B(1)*B(1)
      WF = SQRT(WF)
      DO 215 I = 1,NN
  215 B(I) = B(I)/WF
      BX = -F*B(1)/C
      WF = 0.5*WFF(1)*BX*BX
      DO 220 I = 2,NX
      IP = I + 1
      IM = I - 1
      BX = (B(IP) - B(IM))/DXX
  220 WF = WF + WFF(I)*BX*BX
      WF = WF*DX/F
      BN = B(1)
      WRITE(6,906)
      WRITE(6,907) WF
      WRITE(6,908) BN
      WRITE(6,909)
```

```
      WRITE(6,902) (B(I),I = 1,NN)
901   FORMAT(//'  ANN =',E15.5,' 1/CM')
902   FORMAT(10E13.5)
903   FORMAT(/'  BN =',E15.5,' CM-1/2'/)
904   FORMAT(//'  PHI ='/)
905   FORMAT(' USING STREAM FUNCTION')
906   FORMAT(//' USING PRESSURE')
907   FORMAT(' ANN = ',E15.5,' SEC/CM2')
908   FORMAT(' BN =',E15.5,' CM-1/2')
909   FORMAT(' NORMALIZED PRESSURE (CM-1/2)')
      RETURN
      END

      SUBROUTINE MUTS(NN,XMAX,DD1,F,W,RL,IDD3,RI,IDD4,REPS)
      COMMON RH(400),RHX(400),A(1200),B(400),V(400),VX(400),VXX(400).
      DOUBLE PRECISION A1,A2,A3,AL,DSQRT
C
C             SUBROUTINE TO SET UP AND SOLVE THE U EQUATION
C
      NU = NN
      IF (IDD3.NE.1) GO TO 2
      NU = NN - 1
2     IF (IDD4.NE.1) GO TO 4
      NU = NU - 1
4     DX = XMAX/FLOAT(NN-1)
5     GG = DD1/980.
      RLL = RL*RL
      DXX = DX*DX
      DDD = DX/2.
      DO 100 N = 1,NU
      N1 = N
      IF (IDD4.NE.1) GO TO 8
      N1 = N + 1
8     B(N) = 0.
      WP = W + RL*V(N1)
      W2 = WP*WP
      RR = RH(N1)
      RX = RHX(N1)
      WX = RL*VX(N1)
      FP = F + VX(N1)
      A1 = RR*(GG*W2 - RR*RLL)
      A2 = RR*(RX*RLL -2.0*WP*WX*GG)
      A3 = -GG*GG*W2*(F*FP -REPS*W2)
      A3 = A3 + REPS*RR*RR*RLL*RLL
      A3 = A3 + GG*RR*RLL*(-2.0*W2*REPS + F*FP + 2.0*FP*VX(N1))
      A3 = A3 - RR*RLL*RL*FP*RX/WP
      A3 = A3 - RR*RL*VXX(N1)*(GG*W2 - RR*RLL)/WP
      N1 = N + NU
      N2 = N + 2*NU
      A(N) = -2.*A1 + A3*DXX
      A(N1) = A1 -A2*DDD
      A(N2) = A1 + A2*DDD
      IF (N.EQ.1) GO TO 20
      IF (N.EQ.NU) GO TO 30
      GO TO 100
20    IF (IDD4.NE.0) GO TO 25
      A(N2) = A(N1) + A(N2)
      A(N1) = 0.
      GO TO 100
25    IF (IDD4.EQ.1) GO TO 28
```

```
      WRITE(6,*) A1,A2,A3,REPS,GG,RR,W2,RLL,FP,F,RX,WP,VX(N1),VXX(N1)
      AL = DSQRT(-A3/A1)
      A(N2) = A(N2) + A(N1)
      A(N) = A(N) -2.0*DX*AL*A(N1)
 28   A(N1) = 0.
      GO TO 100
 30   IF (IDD3.NE.0) GO TO 35
      A(N1) = A(N1) + A(N2)
      A(N2) = 0.
      GO TO 100
 35   IF (IDD3.EQ.1) GO TO 36
      AL = DSQRT(-A3/A1)
      A(N1) = A(N1) +A(N2)
      A(N) = A(N) -2.0*DX*AL*A(N2)
 36   A(N2) = 0.0
 100  CONTINUE
      B(5) = 1.
      CALL TRI(NU)
      IF (IDD4.NE.1) GO TO 140
      DO 120 I = 1,NU
      N = NU -I +1
      N1 = N + 1
 120  B(N1) = B(N)
      B(1) = 0.
 140  IF (IDD3.NE.1) GO TO 150
      B(NN) = 0.
 150  CALL CALC(NN,RI)
      RETURN
      END

      SUBROUTINE VRD(NN,XMAX,F)
      COMMON RH(400),RHX(400),A(1200),B(400),V(400),VX(400),VXX(400)
 C
 C              SUBROUTINE TO READ AND INTERPOLATE THE MEAN V PROFILE
 C                        V = ALONGSHORE VELOCITY
 C                        VX = X DERIVATIVE OF V
 C                        VXX = SECOND X DERIVATIVE OF V
 C
      DX = XMAX/FLOAT(NN-1)
      READ(5,*) NRD
      IF (NRD.EQ.0) GO TO  200
      DO 5 I = 1,NRD
      READ(5,*) A(I),B(I)
 C              A = DISTANCE FROM SHORE IN KM
 C              B = V IN CM/SEC
 5    A(I) = A(I)*1.0E+05
 10   DO 100 N = 1,NN
      X = DX*FLOAT(N-1)
      IF (X.GT.A(NRD)) GO TO 90
      IF (X.LT.A(1)) GO TO 80
      J = 0
      DO 20 I = 2,NRD
      IF (J.NE.0) GO TO 20
      IF (X.GT.A(I)) GO TO 20
      J = I
 20   CONTINUE
      JJ = J - 1
      XX = A(J) - X
      XY = A(J) - A(JJ)
      V(N) = B(J) -XX*(B(J) -B(JJ))/XY
```

```
      GO TO 100
   80 V(N) =  0.
      GO TO 100
   90 V(N) = B(NRD)
  100 CONTINUE
      VX(1) = (V(2) - V(1))/DX
      NQ = NN - 1
      VX(NN) = (V(NN) - V(NQ))/DX
      DXX = DX*DX
      DZ = 2.*DX
      VXX(1) = 0.
      VXX(NN) = 0.
      DO 120 N = 2,NQ
      I1 = N - 1
      I2 = N + 1
      VX(N) = (V(I2) - V(I1))/DZ
  120 VXX(N) = (V(I2) - 2.*V(N) + V(I1))/DXX
      WRITE(6,904)
      WRITE(6,903) (V(N),N=1,NN)
      IST = 0
      DO 150 N = 1,NN
      Q = -RHX(N)*(F+VX(N)) + RH(N)*VXX(N)
      A(N) = Q
      IF (IST.NE.0) GO TO 150
      IF (N.NE.1) GO TO 130
      PQ = Q
      GO TO 150
  130 IF (PQ.EQ.0.) GO TO 135
      RQ = Q/PQ
      IF (RQ.LT.0.) GO TO 140
      IF (Q.EQ.0.) GO TO 150
  135 PQ = Q
      GO TO 150
  140 IST = 1
  150 CONTINUE
      IF (IST.EQ.0) GO TO 250
      WRITE(6,905)
      WRITE(6,906)
      WRITE(6,905)
      WRITE(6,907)
      WRITE(6,903) (A(N),N=1,NN)
      GO TO 250
  200 DO 220 N = 1,NN
      V(N) = 0.
      VX(N) = 0.
  220 VXX(N) = 0.
  903 FORMAT(10E13.5)
  904 FORMAT(//'  V IN CM/SEC'/)
  905 FORMAT(/' ****************************************************'/)
  906 FORMAT('  POSSIBILITY OF UNSTABLE MODES')
  907 FORMAT(//'  PBARX'/)
  250 RETURN
      END

      SUBROUTINE MATS(NN,XMAX,DD1,F,W,RL,IDD3,RI,IDD4,REPS)
      COMMON RH(400),RHX(400),A(1200),B(400),V(400),VX(400),VXX(400)
      DOUBLE PRECISION A1,A2,A3,B1,B2,B3,BB1,BB2,BX,DSQRT,AL
C
C             SUBROUTINE TO SET UP AND SOLVE THE PRESSURE EQUATION
C
```

```
      DX = XMAX/FLOAT(NN - 1)
      GG = DD1/980.
      RLL = RL*RL*REPS
      DXX = DX*DX
      DDX = 2.*DX
      DDD = DX/2.
      DO 100 N = 1,NN
      B(N) = 0.
      WP = W + RL*V(N)
      FP = F + VX(N)
      FW = F*FP - WP*WP*REPS
      RR = RH(N)
      RX = RHX(N)
      WX = RL*VX(N)
      FLW = F*RL/WP
      A1 = FW*RR
      A2 = FW*RX - RR*(F*VXX(N) -2.0*WP*WX*REPS)
      A3 = -GG*FW*FW +RX*FLW*FW -RR*FLW*(F*VXX(N) -2.*WP*WX*REPS)
      A3 = A3 -RR*RLL*FW
      N1 = N + NN
      N2 = N + 2*NN
      A(N) = -2.*A1 + A3*DXX
      A(N1) = A1 - A2*DDD
      A(N2) = A1 + A2*DDD
      IF (N.EQ.1) GO TO 20
      IF (N.EQ.NN) GO TO 30
      GO TO 100
   20 IF (IDD4.NE.0) GO TO 25
      B1 = RL*RR*FP
      B2 = RLL*WP*RR +GG*WP*FW
      A(N) = A(N) + 2.0*DX*B2*A(N1)/B1
      A(N2) = A(N2) + A(N1)
      A(N1) = 0.
      GO TO 100
   25 IF (IDD4.NE.1) GO TO 27
      A(N2) = A(N2) + A(N1)
      A(N) = A(N) + DDX*FLW*A(N1)
      A(N1) = 0.
      GO TO 100
   27 AL = DSQRT(-A3/A1)
      A(N2) = A(N2) + A(N1)
      A(N) = A(N) - 2.0*DX*AL*A(N1)
      A(N1) = 0.
      GO TO 100
   30 IF (IDD3.EQ.1) GO TO 35
      IF (IDD3.EQ.2) GO TO 32
      B1 = -RR*WP*FW
      B2 = -FW*(RR*RL*F + RX*WP + RR*WX)
      B2 = B2 + RR*WP*(F*VXX(N) -2.0*WP*WX*REPS)
      B3 = RL*F*(-RX*FW + RR*(F*VXX(N) -2.*WP*WX*REPS))
      BX = 1. + DDD*B2/B1
      BB1 = -(-2. +B3*DXX/B1)/BX
      BB2 = -(1. -B2*DDD/B1)/BX
      A(N) = A(N) + A(N2)*BB1
      A(N1) = A(N1) + A(N2)*BB2
      A(N2) = 0.
      GO TO 100
   32 AL = DSQRT(-A3/A1)
      A(N1) = A(N1) + A(N2)
      A(N) = A(N) -2.0*DX*AL*A(N2)
```

```
                 A(N2) = 0.
                 GO TO 100
      35         A(N1) = A(N1) + A(N2)
                 A(N) = A(N) -DDX*FLW*A(N2)
                 A(N2) = 0.
      100        CONTINUE
                 B(5) = 0.00001
                 EQQ = 1.0E-36
                 IKK = 0
                 DO 105 I = 1,NN
                 AQ = ABS(A(I))
                 IF (AQ.GT.EQQ) GO TO 105
                 IKK = 1
      105        CONTINUE
                 IF (IKK.EQ.1) GO TO 115
                 CALL TRI(NN)
                 CALL CALC(NN,RI)
                 GO TO 110
      115        WRITE(6,901)
      901        FORMAT(' NUMERICAL PROBLEM: SMALL DIAGONAL ELEMENT')
      110        RETURN
                 END

                 SUBROUTINE CRLC(W,RL,NN,ICR)
                 COMMON RH(400),RHX(400),A(1200),B(400),V(400),VX(400),VXX(400)
      C
      C                  SUBROUTINE TO CHECK FOR CRITICAL LAYERS
      C
                 ICR = 0
                 DO 100 N = 1,NN
                 WP = W + RL*V(N)
                 IF (N.NE.1) GO TO 30
                 PQ = WP
                 GO TO 100
      30         PQ = WP/PQ
                 IF (PQ.GT.0.) GO TO 40
                 ICR = ICR +1
      40         PQ = WP
      100        CONTINUE
                 RETURN
                 END

                 SUBROUTINE TRI(N)
                 COMMON RH(400),RHX(400),A(1200),B(400),V(400),VX(400),VXX(400)
                 DOUBLE PRECISION AA,BB
      C
      C
      C                  SUBROUTINE TO SOLVE A TRIDIAGONAL MATRIX BY GAUSSIAN
      C                          ELIMINATION
      C
      C                  STORE MAIN DIAGONAL FIRST
      C                  0,LOWER DIAGONAL
      C                  UPPER DIAGONAL,0
                 N3 = N +1
                 N2 = 2*N
                 NN = N - 1
                 DO 5 I = 1,NN
                 II = N3 + I
                 AA = A(II)/A(I)
                 I2 = I + 1
```

```
      I3 =N2 + I
      BB = A(I2) -AA*A(I3)
      A(I2) = BB
      BB = B(I2) -AA*B(I)
 5    B(I2) = BB
      DO 10 IJ = 1,NN
      I = N3 - IJ
      B(I) = B(I)/A(I)
      II = I + N2 -1
      IP = I -1
      BB = B(IP) -A(II)*B(I)
 10   B(IP) = BB
      B(1) = B(1)/A(1)
      RETURN
      END

      SUBROUTINE WFR(NN,XMAX,WF)
      COMMON RH(400),RHX(400),A(1200),B(400),V(400),VX(400),VXX(400)
      DIMENSION WF(400)
C
C              SUBROUTINE TO READ AND INTERPOLATE FRICTIONAL WEIGHT
C                        FUNCTIONS, WF
C
      READ(5,*) NW
      IF (NW.EQ.0) GO TO 100
      DO 5 I = 1,NW
      II = 100 + I
      READ(5,*) A(I),A(II)
 5    A(I) = A(I)*1.0E+05
      DX = XMAX/FLOAT(NN-1)
      WF(1) = A(101)
      NMM = 100 + NW
      WFM = A(NMM)
      DO 50 N = 2,NN
      X = DX*FLOAT(N-1)
      IF (X.GT.A(NW)) GO TO 48
      IC = 0
      DO 46 J = 2,NW
      IF (IC.NE.0) GO TO 46
      I = J
      IF (X.GT.A(I)) GO TO 46
      IC = I
 46   CONTINUE
      II = 100 + I
      IIM = II - 1
      IM = I -1
      WX = (A(II) - A(IIM))/(A(I) -A(IM))
      XX =X -A(IM)
      WF(N) = A(IIM) + WX*XX
      GO TO 50
 48   WF(N) = WFM
 50   CONTINUE
      GO TO 200
 100  DO 150 I = 1,NN
 150  WF(I) = 1.0
 200  WRITE(6,901)
      WRITE(6,902) (WF(I),I=1,NN)
 901  FORMAT(/' FRICTION WEIGHT FUNCTION')
 902  FORMAT(10E12.5)
      RETURN
      END
```

Program BIGLOAD2

Listing

```
      PROGRAM HYBD
      COMMON F,DX,DT,NN,MM,NM,NMX
      DOUBLE PRECISION AX,BX,XL
      COMMON AX(425,53),BX(425),XL(11475)
      COMMON RH(25),RHX(25),RHXX(25),BV(25,17),BVZ(25,17)
      DIMENSION WW(3), RII(3)
      DIMENSION R(25)
      DIMENSION BXB(425)
      DIMENSION WH(10), CH(10),RLH(10)
C
C             MAIN PROGRAM TO CALL ALL OF THE OTHER PIECES
C
C             DIMENSION(XL) = NM*(NN+2)
C             DIM(AX) = NM,2NN+3
C
C             REVISED APRIL, 1985
      READ(5,*) EPS,EST,DD1
      READ(5,*) ICCM,NCALM,NITM,ISD
      ICCC =1
    3 READ(5,*) F,XMAX
      XMAX = XMAX*1.0E+05
      F =F*1.0E-05
      WRITE(6,907) F,XMAX
      READ(5,*) BETA,ILWW
      IF (ILWW.EQ.1) GO TO 4
      IF (ILWW.NE.0) GO TO 140
      NCALM = 1
      WRITE(6,910)
      GO TO 5
    4 WRITE(6,911)
    5 DDLW = FLOAT(ILWW)
      WRITE(6,909) BETA
      READ(5,*) NCAL,WH(1)
      READ(5,*) IDIAG
      READ(5,*) RLF,DRL
      WH(1) = WH(1)*1.0E-06
      RLF = RLF*1.0E-07
      DRL = DRL*1.0E-07
      WRITE(6,905) EPS,EST,DD1
      MM = 17
      NN = 25
      NM = NN*MM
      NMX = 2*NN +3
      DX = XMAX/FLOAT(NN-1)
      DT = 1./FLOAT(MM-1)
      DO 50 I = 1,NCALM
   50 RLH(I) = RLF + DRL*FLOAT(I-1)
      RL = RLH(NCAL)
      READ(5,*) (WW(J),J=1,3)
      DO 6 J = 1,3
    6 WW(J) = WW(J)*1.0E-06
      CALL DEP
      CALL NSQ
    1 RIB = 0.
      I =1
   15 W = WW(I)
      CALL MATS(RL,W,DD1,RI,IER,BETA,DDLW)
      IF (IER.NE.0) GO TO 140
      RII(I) = RI
      IF (RI.LT.RIB) GO TO 18
```

```
              RIB = RI
              WB = W
              IB = I
              DO 10 IJ = 1,NM
   10         BXB(IJ) = BX(IJ)
   18         I = I+1
              IF (I.EQ.2) GO TO 15
              IF (I.EQ.4) GO TO 19
              IF (RII(1).LT.RII(2)) GO TO 15
              WW(3) = WW(1)*(1. -EST)
              GO TO 15
   19         NIT = 3
              IGP = 0
   20         CALL NGSW(WW,RII,WB,WN,ISUC,EPS,IN,IGP,EST,ISD)
              IF (WN.GT.F) GO TO 140
              IF (ISUC.EQ.2) GO TO 140
              IF (ISUC.EQ.1) GO TO 100
              CALL MATS(RL,WN,DD1,RI,IER,BETA,DDLW)
              IF (IER.NE.0) GO TO 140
              IF (IN.NE.0) GO TO 24
              IF (RI.GT.RIB) GO TO 21
              IF (WN.LT.WW(2)) GO TO 23
   22         IN = 3
              GO TO 24
   23         IN = 1
              GO TO 24
   21         IF (WN.LT.WW(2)) GO TO 22
              GO TO 23
   24         WW(IN) = WN
              RII(IN) = RI
              IF (RI.LT.RIB) GO TO 30
              WB = WN
              RIB = RI
              DO 25 I = 1,NM
   25         BXB(I) = BX(I)
   30         NIT = NIT +1
              IF(NIT.LT.NITM) GO TO 20
              WRITE(6,902) NIT
              GO TO 140
  100         RN = SQRT(RIB)
              DO 110 I = 1,NM
  110         BX(I) = BXB(I)/RN
              CC = WB/RL
              WRITE(6,903) WB,RL,CC,EPS,NIT
              IF (NCAL.NE.1) GO TO 125
              CALL LGWH(WB,RL,DD1,R,BETA,DDLW)
              IF (IDIAG.EQ.0) GO TO 125
              CALL UCAL(WB,RL,BETA,DDLW)
              CALL RHOC
  125         WRITE(6,908)
              DO 130 N = 1,NN
              X = DX*FLOAT(N-1)
              DZ = RH(N)*DT
              WRITE(6,904) X,RH(N),DZ
              ML = 1 +MM*(N-1)
              MH = MM*N
              WRITE(6,901) (BX(M),M=ML,MH)
  130         CONTINUE
              CALL DIAG(WB,RL,DDLW)
              WH(NCAL) = WB
```

```
      CH(NCAL) = CC
      NCAL = NCAL +1
      IF (NCAL.GT.NCALM) GO TO 140
      RL = RLH(NCAL)
      IF (NCAL.GE.3) GO TO 200
      WW(2) = CC*RL
      GO TO 205
  200 I1 = NCAL - 2
      I2 = NCAL - 1
      CG = (WH(I2) -WH(I1))/(RLH(I2) - RLH(I1))
      WW(2) = WH(I2) + CG*(RLH(NCAL) -RLH(I2))
  205 WW(1) = WW(2)*(1.0 - EST)
      WW(3) = WW(2)*(1.0 + EST)
      IF (WW(1).LT.0.) GO TO 140
      IF (WW(3).GT.F) GO TO 140
      GO TO 1
  140 ICCC = ICCC +1
      IF (ICCC.LE.ICCM) GO TO 3
  901 FORMAT(2X,10E12.5)
  902 FORMAT(//'  USED UP ',I5,'  ITERATIONS'//)
  903 FORMAT(///'   CONVERGED:  W,L,C,EPS,NIT =',4E15.5,I10//)
  904 FORMAT(/'  X,H,DZ =',3E15.5)
  905 FORMAT(/'  EPS,EST,DD1 =', 2E15.5,F10.2/)
  906 FORMAT(3F10.5)
  907 FORMAT(//'  F,XMAX =',2E15.5//)
  908 FORMAT(//'  PRESSURE'/)
   909 FORMAT(/' BETA= ',E15.5)
  910 FORMAT(' LONG WAVE LIMIT')
  911 FORMAT(' GENERAL FREQUENCY AND WAVENUMBER')
      STOP
      END

      SUBROUTINE NGSW(WW,RII,WB,WN,ISUC,EPS,IN,IGP,EST,ISD)
      COMMON F,DX,DT,NN,MM,NM,NMX
      DOUBLE PRECISION AX,BX,XL
      COMMON AX(425,53),BX(425),XL(11475)
      COMMON RH(25),RHX(25),RHXX(25),BV(25,17),BVZ(25,17)
      DIMENSION  WW(3),RII(3)
C
C              SUBROUTINE TO COMPUTE THE NEXT GUESS AT W
C
C              ISD = 0  NORMAL
C              ISD = 1  SEARCH DOWN
C              ISD = -1 SEARCH UP
      DEL = EST
    5 IC = 0
      DO 10 I = 1,2
      I1 = 1 +I
      IF (WW(I1).GT.WW(I)) GO TO 10
      IC = 1
      RX = RII(I)
      WX = WW(I)
      WW(I) = WW(I1)
      WW(I1) = WX
      RII(I) = RII(I1)
      RII(I1) =RX
   10 CONTINUE
      IF (IC.NE.0) GO TO 5
      IF (RII(3).GT.RII(2)) GO TO 150
      IF (RII(1).GT.RII(2)) GO TO 160
```

```
      ISD = 0
      WB1 = (WW(1) +WW(2))/2.
      WB2 = (WW(2) + WW(3))/2.
      RW1 = (RII(2) - RII(1))/(WW(2) -WW(1))
      RW2 = ( RII(3) - RII(2))/(WW(3) - WW(2))
      A = (RW2 -RW1)/(WB2 -WB1)
      B = RW1 - A*WB1
      WN = -B/A
      IF (WN.LE.WW(1)) GO TO 50
      IF (WN.GT.WW(3)) GO TO 45
      GO TO 20
 50   WN = WB1
      GO TO 20
 45   WN = WB2
 20   EP = WN - WB
      EP = ABS(EP)
      EP = EP/WB
      IF (EP.LE.EPS) GO TO 100
      IGP = IGP +1
      ISUC = 0
      IN = 0
      GO TO 120
 100  ISUC = 1
      IF (IGP.NE.0) GO TO 120
      IGP = IGP +1
      ISUC = 0
      IF (RII(3).GT.RII(1)) GO TO 110
      WN = WB2
      GO TO 20
 110  WN = WB1
      GO TO 20
 150  IF (RII(1).GT.RII(2)) GO TO 180
 155  IF (ISD.EQ.1) GO TO 165
      WN = WW(3)*(1. + DEL)
      ISUC = 0
      IGP = 0
      IN = 1
      GO TO 120
 160  IF (RII(3).GT.RII(2)) GO TO 180
 165  IF (ISD.EQ.-1) GO TO 155
      WN = WW(1)*(1. - DEL)
      ISUC = 0
      IGP = 0
      IN = 3
      GO TO 120
 180  IF (RII(3).GT.RII(1)) GO TO 155
      GO TO 165
 120  IF (WN.LE.0.) GO TO 125
      GO TO 130
 125  ISUC = 2
 130  RETURN
      END

      SUBROUTINE NSQ
      COMMON F,DX,DT,NN,MM,NM,NMX
      DOUBLE PRECISION AX,BX,XL
      COMMON AX(425,53),BX(425),XL(11475)
      COMMON RH(25),RHX(25),RHXX(25),BV(25,17),BVZ(25,17)
C
C              SUBROUTINE TO READ AND INTERPOLATE BUOYANCY FREQUENCY
```

```
C                             SQUARED PROFILE BV, AND Z DERIVATIVE BVZ
C
            READ(5,*) NR,DZR,ALPH
            ALPH = ALPH*1.0E+05
            DZR = DZR*100.
            MX = MM -1
            READ(5,*) CMLT
            DO 20 I = 1,NR
            READ(5,*) XL(I)
   20       XL(I) = XL(I)*CMLT
            TF = XL(NR)
            ZZZ = DZR*FLOAT(NR-1)
            DZI = RH(NN) - ZZZ
            NL = NR +1
            NH = 600
            DO 30 I = NL,NH
            Z = DZR*FLOAT(I-1)
   30       XL(I) = TF*EXP((ZZZ -Z)/ALPH)
            DO 200 N = 1,NN
            DD = RH(N)
            DZ = DT*DD
            IF (DZ.GT.DZR) GO TO 110
            BV(N,MM) = XL(1)
            DO 50 M = 1,MX
            Z = DD - DZ*FLOAT(M-1)
            IBXL = 1 + IFIX(Z/DZR)
            IBXH = IBXL +1
            ZS = DZR*FLOAT(IBXL -1)
   50       BV(N,M) = XL(IBXL) +(Z-ZS)*(XL(IBXH)-XL(IBXL))/DZR
            Z = DD - DZ
            IBXL = 1 + IFIX(Z/DZR)
            IBXH = IBXL + 1
            ZS = DZR*FLOAT(IBXL-1)
            AQ = XL(IBXL) +(Z-ZS)*(XL(IBXH)-XL(IBXL))/DZR
            GO TO 145
  110       ZD = DZ/2.
            XBB = 0.
            NAVG = 0
            DO 120 I = 1,NR
            ZC = DZR*FLOAT(I-1)
            IF (ZC.GT.ZD) GO TO 120
            XBB = XBB + XL(I)
            NAVG = NAVG +1
  120       CONTINUE
            BV(N,MM) = XBB/FLOAT(NAVG)
            DO 140 MQ = 1,MM
            M = MQ -1
            Z = DD - DZ*FLOAT(M-1)
            ZS = Z - DZ/2.
            ZD = ZS + DZ
  125       XBB = 0.
            NAVG = 0
            DO 130 I = 1,NH
            ZC = DZR*FLOAT(I-1)
            IF ( ZC.LT.ZS) GO TO 130
            IF (ZC.GT.ZD) GO TO 130
            XBB = XBB + XL(I)
            NAVG = NAVG + 1
  130       CONTINUE
            IF (NAVG.NE.0) GO TO 135
```

```
      IBXL = 1 + IFIX(Z/DZR)
      IBXH = IBXL + 1
      ZSS = DZR*FLOAT(IBXL-1)
      IF (M.EQ.0) GO TO 133
      BV(N,M) = XL(IBXL) +(Z-ZSS)*(XL(IBXH)-XL(IBXL))/DZR
      GO TO 140
133   AQ = XL(IBXL) +(Z-ZSS)*(XL(IBXH)-XL(IBXL))/DZR
      GO TO 140
135   IF (M.EQ.0) GO TO 138
      BV(N,M) = XBB/FLOAT(NAVG)
      GO TO 140
138   AQ = XBB/FLOAT(NAVG)
140   CONTINUE
145   DO 150 M = 2,MX
      IP = M +1
      IM = M -1
150   BVZ(N,M) = (BV(N,IP) - BV(N,IM))/(2.*DZ)
      BVZ(N,1) = (BV(N,2) -AQ)/(2.0*DZ)
      BVZ(N,MM) = (BV(N,MM) - BV(N,MX))/DZ
200   CONTINUE
      WRITE(6,901)
      WRITE(6,902) (BV(NN,J),J=1,MM)
901   FORMAT(//'   NSQUARED AT XMAX '/)
902   FORMAT(2X,10E12.5)
250   RETURN
      END

      SUBROUTINE MATS(RL,W,DD1,RI,IER,BETA,DDLW)
      COMMON F,DX,DT,NN,MM,NM,NMX
      DOUBLE PRECISION AX,BX,XL
      COMMON AX(425,53),BX(425),XL(11475)
      COMMON RH(25),RHX(25),RHXX(25),BV(25,17),BVZ(25,17)
C
C             SUBROUTINE TO SET UP AND SOLVE THE PRESSURE EQUATION
C
C             DD1 = 0  RIGID LID
C             DD1 = 1  FREE SURFACE
      GG = 1./980.
      DXDT = DX/(2.*DT)
      DXX = DX*DX
      DDD = DXX/(DT*DT)
      RLL = RL*RL*DDLW
      DDX = DX*DXDT
      DQ = 2.0*DX
      FLW = F*RL/W
      BLW=BETA*RL/W
      CCC1 = W*(F*F - DDLW*W*W)/DXX
      CCC2 = 0.5*F*(2.0*BETA*W + RL*(F*F - DDLW*W*W))/DX
      CCC3 = BETA*RL*(F*F + DDLW*W*W)
      AA5 = 2.0*F*BETA/(F*F -W*W*DDLW)
      RLQ = DDLW*RLL - BLW*(F*F + DDLW*W*W)/(F*F -DDLW*W*W)
      DO 5 I = 1,NM
5     BX(I) = 0.
      DO 10 J = 1,NMX
      DO 10 I = 1,NM
10    AX(I,J) = 0.
      BX(39) = 1.
      J = NN + 2
      IP1 = J + 1
      IM1 = J - 1
```

```
      IPM1 = 2*NN +1
      IPM = 2*NN + 2
      IPMM = 2*NN + 3
      DO 100 M = 1,MM
      DO 90 N = 1,NN
      I = N + NN*(M-1)
      CALL AS(A1,A2,A3,W,N,M,C1,C2,C3,DDLW)
      AX(I,1) = A2*DXDT
      AX(I,IM1) = 1.-AA5*DX/2.0
      AX(I,IPM1) = -A2*DXDT
      AX(I,2) = A1*DDD - DDX*(A3+A2*AA5)
      AX(I,J) = -2. -2.*DDD*A1 - RLQ*DXX
      AX(I,IPM) = A1*DDD + DDX*(A3+A2*AA5)
      AX(I,3) = -A2*DXDT
      AX(I,IP1) = 1.+AA5*DX/2.0
      AX(I,IPMM) = A2*DXDT
      IF ( N.EQ.1) GO TO 20
      IF (N.EQ.NN) GO TO 30
      IF (M.EQ.1) GO TO 40
      IF (M.EQ.MM) GO TO 50
      GO TO 90
20    AX(I,1) = 0.
      AX(I,IM1) = 0.
      AX(I,IPM1) = 0.
      AX(I,3) = 0.
      AX(I,IPMM) = 0.
      AX(I,IP1) = 2.
      TH = -1. + DT*FLOAT(M-1)
      B1 = -TH*RHX(N)/RH(N)
      AX(I,2) = AX(I,2) -A2*B1*DDD + A2*FLW*2.0*DDX - B1*2.0*DXDT
      AX(I,J) = AX(I,J) + 2.*DX*FLW + 2.*A2*B1*DDD
      AX(I,IPM) =AX(I,IPM)-A2*B1*DDD-A2*FLW*2.0*DDX + B1*2.0*DXDT
      IF (M.NE.MM) GO TO 25
      D5 = DD1*RH(N)*BV(N,M)*GG
      AX(I,2) = AX(I,2) + AX(I,IPM)
      AX(I,J) = AX(I,J) - 2.0*DT*D5*AX(I,IPM)
      AX(I,IPM) = 0.
      GO TO 90
25    IF (M.NE.1) GO TO 90
      AX(I,IPM) = AX(I,IPM) + AX(I,2)
      AX(I,2) = 0.
      GO TO 90
30    DZ = DT*RH(N)
      AX(I,3) = 0.
      AX(I,IPMM) = 0.
      AX(I,1) = 0.
      AX(I,3) = 0.
      AX(I,J) = AX(I,J) +AX(I,IP1)*(2.0*CCC1-CCC3)/(CCC1+CCC2)
      AX(I,IM1) = AX(I,IM1) +AX(I,IP1)*(CCC2-CCC1)/(CCC1+CCC2)
      AX(I,IP1) = 0.
      IF (M.NE.MM) GO TO 35
      AX(I,2) = AX(I,2) +AX(I,IPM)
      AX(I,J) = AX(I,J) -2.0*DZ*DD1*GG*BV(N,M)*AX(I,IPM)
      AX(I,IPM) = 0.
      GO TO 90
35    IF (M.NE.1) GO TO 90
      AX(I,IPM) = AX(I,IPM) + AX(I,2)
      AX(I,2) = 0.
      GO TO 90
40    D1 = 2.*C2*DT/(C1 +C2*C3)
```

```
      AX(I,IPM) = AX(I,IPM) + AX(I,2)
      AX(I,IP1) = AX(I,IP1) + D1*AX(I,2)/DQ
      AX(I,IM1) = AX(I,IM1) -D1*AX(I,2)/DQ
      AX(I,J) = AX(I,J) +D1*FLW*AX(I,2)
      NNN = N +1
      IJJP = J + 2
      CALL AS(A1,A2,A3,W,NNN,M,C1,C2,C3,DDLW)
      D1 = 2.0*C2*DT/(C1 + C2*C3)
      AX(I,IPMM) = AX(I,IPMM) + AX(I,3)
      AX(I,IJJP) = AX(I,IJJP) + D1*AX(I,3)/DQ
      AX(I,J) = AX(I,J) -D1*AX(I,3)/DQ
      AX(I,IP1) = AX(I,IP1) + D1*FLW*AX(I,3)
      IF (N.EQ.2) GO TO 45
      IJJM = J - 2
      NNN = N - 1
      CALL AS(A1,A2,A3,W,NNN,M,C1,C2,C3,DDLW)
      D1 = 2.0*C2*DT/(C1 + C2*C3)
      AX(I,IPM1) = AX(I,IPM1) + AX(I,1)
      AX(I,J) = AX(I,J) + D1*AX(I,1)/DQ
      AX(I,IJJM) = AX(I,IJJM) -D1*AX(I,1)/DQ
      AX(I,IM1) = AX(I,IM1) + D1*FLW*AX(I,1)
      GO TO 48
45    AX(I,IPM1) = AX(I,IPM1) + AX(I,1)
48    AX(I,1) = 0.
      AX(I,2) = 0.
      AX(I,3) = 0.
      GO TO 90
50    AX(I,1) = 0.
      AX(I,3) = 0.
      D5 = RH(N)*DD1*BV(N,M)*GG
      AX(I,2) = AX(I,2) + AX(I,IPM)
      AX(I,J) = AX(I,J) - D5*2.0*DT*AX(I,IPM)
      AX(I,IM1) = AX(I,IM1) -D5*2.0*DT*AX(I,IPM1)
      AX(I,IP1) = AX(I,IP1) -D5*DT*2.0*AX(I,IPMM)
      AX(I,IPM1) = 0.0
      AX(I,IPMM) = 0.0
      AX(I,IPM) = 0.0
90    CONTINUE
100   CONTINUE
      NDD = NN +1
      NNS = NM
      CALL LEQT1B(AX,NM,NDD,NDD,NM,BX,1,NM,0,XL,IER)
C            LEQT1B IS AN IMSL ROUTINE
      CALL CALI(RI)
      CC = W/RL
      WRITE(6,901) W,RL,CC,RI,IER
901   FORMAT(/'  W,L,C,RI,IER =',4E15.5,I10)
      RETURN
      END

      SUBROUTINE AS(A1,A2,A3,W,N,M,C1,C2,C3,DDLW)
      COMMON F,DX,DT,NN,MM,NM,NMX
      DOUBLE PRECISION AX,BX,XL
      COMMON AX(425,53),BX(425),XL(11475)
      COMMON RH(25),RHX(25),RHXX(25),BV(25,17),BVZ(25,17)
C
C            SUBROUTINE TO COMPUTE COEFFICIENTS FOR MATRIX
C
      F2 = F*F
      W2 = W*W*DDLW
```

```
      TZ = 1./RH(N)
      TH = -1. +DT*FLOAT(M-1)
      TX = -RHX(N)*TH*TZ
      AA = (RHX(N)/RH(N))**2
      TXX = (2.*AA -RHXX(N)/RH(N))*TH
      BW = BV(N,M)
      BWW = BW*BW
      A1 = TX*TX + TZ*TZ*(F2-W2)/BW
      A2 = TX
      A3 = TXX -TZ*BVZ(N,M)*(F2-W2)/BWW
      C1 = TZ
      C2 = RHX(N)*BV(N,1)/(F2-W2)
      C3 = C1*RHX(N)
      RETURN
      END

      SUBROUTINE DEP
      COMMON F,DX,DT,NN,MM,NM,NMX
      DOUBLE PRECISION AX,BX,XL
      COMMON AX(425,53),BX(425),XL(11475)
      COMMON RH(25),RHX(25),RHXX(25),BV(25,17),BVZ(25,17)
C
C              SUBROUTINE TO READ AND INTERPOLATE DEPTH PROFILE
C                        RH = DEPTH
C                        RHX = X DERIVATIVE OF DEPTH
C                        RHXX = SECOND X DERIVATIVE OF DEPTH
C
      READ(5,*) NRX
      DO 5 I =1,NRX
      READ(5,*) XL(I),BX(I)
      BX(I) = BX(I)*100.
    5 XL(I) = XL(I)*1.0E+05
      RHMA = BX(NRX)
      RH(1) = BX(1)
      DO 20 N = 2,NN
      X = DX*FLOAT(N-1)
      IF (X.GT.XL(NRX)) GO TO 15
      IC = 0
      DO 8 J = 2,NRX
      IF (IC.NE.0) GO TO 8
      I = J
      IF (X.GT.XL(I)) GO TO 8
      IC = I
    8 CONTINUE
      IM = I-1
      RHX(N) = (BX(I) -BX(IM))/(XL(I)-XL(IM))
      XX = X - XL(IM)
      RH(N) = BX(IM) + RHX(N)*XX
      GO TO 20
   15 RH(N) = RHMA
   20 CONTINUE
      RHX(1) = (RH(2) - RH(1))/DX
      RHXX(1) = 0.
      D2 = 2.*DX
      DXX = DX*DX
      NX = NN -1
      DO 30 N = 2,NX
      IP = N +1
      IM = N -1
      RHX(N) =(RH(IP) - RH(IM))/D2
```

```
 30    RHXX(N) = (RH(IP) -2.*RH(N) + RH(IM))/DXX
       RHX(NN) = 0.
       RHXX(NN) = 0.
       RETURN
       END

       SUBROUTINE CALI(RIX)
       COMMON F,DX,DT,NN,MM,NM,NMX
       DOUBLE PRECISION AX,BX,XL
       COMMON AX(425,53),BX(425),XL(11475)
       COMMON RH(25),RHX(25),RHXX(25),BV(25,17),BVZ(25,17)
       DOUBLE PRECISION RI,RZ
 C
 C                SUBROUTINE TO CALCULATE THE INTEGRAL OF RESPONSE SQUARED
 C
       DO 2 M = 1,MM
       DO 2 N = 1,NN
        IN = N + NN*(M -1)
       II = M + MM*(N-1)
 2     XL(II) = BX(IN)
       DO 3 I = 1,NM
 3     BX(I) = XL(I)
       RI = 0.
       RR = 0.
       RZ = 0.
       DO 50 N = 1,NN
       IF (N.EQ.1) GO TO 5
       IF (N.NE.NN) GO TO 10
 5     DXX = DX/2.
       GO TO 15
 10    DXX = DX
 15    DO 50 M = 1,MM
       DZ = DT*RH(N)
       IF (M.EQ.1) GO TO 25
       IF (M.NE.MM) GO TO 30
 25    DZZ = DZ/2.
       GO TO 35
 30    DZZ = DZ
 35    I = (N-1)*MM +M
       RZ = RZ + DZZ*DXX
       RI = RI + DZZ*DXX*BX(I)**2
 50    CONTINUE
       RI = RI/RZ
       RIX = RI
       RETURN
       END

       SUBROUTINE DIAG(W,RL,DDLW)
       COMMON F,DX,DT,NN,MM,NM,NMX
       DOUBLE PRECISION AX,BX,XL
       COMMON AX(425,53),BX(425),XL(11475)
       COMMON RH(25),RHX(25),RHXX(25),BV(25,17),BVZ(25,17)
 C
 C                SUBROUTINE TO COMPUTE MOMENTUM DIAGNOSTICS
 C
       N = 5
       I = N*MM
       IP = I + MM
       IM = I -MM
       PX = (BX(IP) -BX(IM))/(2.*DX)
```

```
      F2 = F*F - W*W*DDLW
      VT = W*(F*PX + DDLW*RL*W*BX(I))/F2
      FU = -F*(RL*F*BX(I) + W*PX)/F2
      PY = RL*BX(I)
      X = DX*FLOAT(N-1)
      WRITE(6,901) X
      WRITE(6,902) VT,FU,PY
901   FORMAT(//'  Y MOMENTUM TERMS AT Z = 0, X =',E15.5)
902   FORMAT(/'  VT,FU,PY =',3E15.5/)
      RETURN
      END

      SUBROUTINE LGWH(W,RL,DD1,R,BETA,DDLW)
      COMMON F,DX,DT,NN,MM,NM,NMX
      DOUBLE PRECISION AX,BX,XL
      COMMON AX(425,53),BX(425),XL(11475)
      COMMON RH(25),RHX(25),RHXX(25),BV(25,17),BVZ(25,17)
      DOUBLE PRECISION RD,RP,RI,XINT,XP
      DIMENSION R(25)
C
C                SUBROUTINE TO COMPUTE LONG WAVE PARAMETERS BN,ANN
C
      XINT = 0.
      RHO = 1.03
      F2 = F*F
      MX = MM - 1
C                NORMALIZE
      NX = NN -1
      RI = (BX(1)**2 + BX(MM)**2)/2.
      RD = (RHX(1)*BX(1)**2)/2.
      DO 10 I = 2,MX
10    RI = RI + BX(I)**2
      DO 20 I = 2,NX
      IJ = MM*(I-1) +1
20    RD = RD + RHX(I)*BX(IJ)**2
      RD = RD *DX
      RI = RI*RH(1)*DT
      RD = RD + RI
      RD = DSQRT(RD*RD)
      RD = DSQRT(RD)
      DO 30 I = 1,NM
30    BX(I) = BX(I)/RD
      CALL VCAL(W,RL,BETA,DDLW)
C                COMPUTE WIND COUPLING
40    BN = BX(MM)
C                NOW GET BOTTOM FRICTION
C                READ R(X)
      READ(5,*) NR
      DO 45 I = 1,NR
      READ(5,*) AX(I,1),AX(I,2)
45    AX(I,1) = AX(I,1)*1.0E+05
      R(1) = AX(1,2)
      RMA = AX(NR,2)
      DO 50 N = 2,NN
      X = DX*FLOAT(N-1)
      IF (X.GT.AX(NR,1)) GO TO 48
      IC = 0
      DO 46 J = 2,NR
      I = J
      IF (X.GT.AX(I,1)) GO TO 46
```

```
      IC = I
   46 CONTINUE
      IM = I -1
      RX = (AX(I,2) -AX(IM,2))/(AX(I,1)-AX(IM,1))
      XX = X - AX(IM,1)
      R(N) = AX(IM,2) + RX*XX
      GO TO 50
   48 R(N) = RMA
   50 CONTINUE
C                X=0 CONTRIBUTION
      RI = 0.5*R(1)*(RL*F*BX(1)/W)**2
C                X = L CONTRIBUTION
      I = 1 + MM*(NN-1)
      BXP = F*XL(I)
      RI = RI + 0.5*R(NN)*(BXP)**2
C                INTERMEDIATE X
      DO 150 N = 2,NX
      I = 1 + MM*(N-1)
      IP = I + MM
      IM = I - MM
      RX = 0.5*(BX(IP)-BX(IM))/DX
  150 RI = RI + R(N)*RX*F*XL(I)
      ANN = RI*DX/F
C
      WRITE(6,902) BN,ANN
      WRITE(6,901)
      WRITE(6,905) (R(I),I=1,NN)
      CC = 0.5/(980.*RHO)
      DO 220 N = 1,NN
      XX = DX
      IF (N.EQ.1) GO TO 210
      IF (N.NE.NN) GO TO 215
  210 XX = DX/2.
  215 I = N*MM
  220 XINT = XINT + CC*XX*BX(I)*BX(I)
      WRITE(6,904) XINT
  901 FORMAT(/' R =')
  902 FORMAT('   BN,ANN =',2E15.5///)
  903 FORMAT(2F10.5)
  904 FORMAT(//'  FREE SURFACE CONTRIBUTION TO PE =',E15.5/)
  905 FORMAT(10E12.5)
  500 RETURN
      END

      SUBROUTINE VCAL(W,RL,BETA,DDLW)
      COMMON F,DX,DT,NN,MM,NM,NMX
      DOUBLE PRECISION AX,BX,XL
      COMMON AX(425,53),BX(425),XL(11475)
      COMMON RH(25),RHX(25),RHXX(25),BV(25,17),BVZ(25,17)
      DOUBLE PRECISION XINT
C
C                SUBROUTINE TO COMPUTE THE V FIELD OF THE WAVE
C
      RHO = 0.515
      XINT = 0.
      WRITE(6,905)
      FW = F*F - DDLW*W*W
      WL = DDLW*W*RL
      RLW = RL/W
      FLW = F*RL/W
```

```
      MX = MM -1
      DXX = 2.*DX
      W2 = W*W*DDLW
      DO 50 N = 1,NN
      XX = DX
      X = DX*FLOAT(N-1)
      D = RH(N)
      DD = RHX(N)/D
      DZ = DT*D
      RHZ = RHX(N)**2
      IF (N.EQ.1) GO TO 10
      IF (N.NE.NN) GO TO 20
      GO TO 18
10    DO 15 M = 1,MM
      I = (N-1)*MM +M
15    XL(I) = -RLW*BX(I)
      XX = DX/2.
      GO TO 40
18    CC1 = W*FW/(DX*DX)
      CC2 = 0.5*(2.0*F*BETA*W +RL*F*FW)/DX
      CC3 = BETA*RL*(F*F +DDLW*W*W)
      DQ = 1./(F*DX)
      DO 19 M = 1,MM
      I = (N-1)*MM +M
      IM = I -MM
      XP = BX(I)*(2.0*CC1 -CC3)/(CC1+CC2)
      XP = XP + BX(IM)*(CC2-CC1)/(CC1+CC2)
      XPX = (XP - BX(IM))/(2.0*DX)
19    XL(I) = (F*XPX + WL*BX(I))/FW
      XX = DX/2.
      GO TO 40
20    I = (N-1)*MM +1
      IP = I + MM
      IM = I - MM
      CALL AS(A1,A2,A3,W,N,1,C1,C2,C3,DDLW)
      CQ = C2*C3/(C1+ C2*C3)
      XPX = (1.0 -CQ)*(BX(IP) -BX(IM))/DXX
      XPX = XPX -CQ*FLW*BX(I)
      XL(I) = (WL*BX(I) + F*XPX)/FW
      I = N*MM
      IP = I + MM
      IM = I -MM
      XPX = (BX(IP) - BX(IM))/DXX
      XL(I) =(WL*BX(I) + F*XPX)/FW
      DO 30 M = 2,MX
      I = (N-1)*MM +M
      IP = I + MM
      IM = I -MM
      I2 = I +1
      I1 = I -1
      XPX = (BX(IP) - BX(IM))/DXX
      T = -1. +DT*FLOAT(M-1)
      XPT = (BX(I2) -BX(I1))/(2.*DT)
      XPX = XPX - T*DD*XPT
30    XL(I) = (WL*BX(I) + F*XPX)/FW
40    DO 45 M = 1,MM
      I = (N-1)*MM + M
      ZZ = DZ
      IF (M.EQ.1) GO TO 42
      IF (M.NE.MM) GO TO 45
```

```
42      ZZ = DZ/2.
45      XINT = XINT + RHO*XL(I)*XL(I)*XX*ZZ
        WRITE(6,904) X,D,DZ
        IL = (N-1)*MM +1
        IH = N*MM
        WRITE(6,901) (XL(IJ),IJ=IL,IH)
50      CONTINUE
        WRITE(6,902) XINT
901     FORMAT(2X,10E12.5)
902     FORMAT(//'  V CONTRIBUTION TO KE =',E15.5/)
905     FORMAT(////'  V'//)
904     FORMAT(/'     X, H, DZ =',3E15.5)
        RETURN
        END

        SUBROUTINE RHOC
        COMMON F,DX,DT,NN,MM,NM,NMX
        DOUBLE PRECISION AX,BX,XL
        COMMON AX(425,53),BX(425),XL(11475)
        COMMON RH(25),RHX(25),RHXX(25),BV(25,17),BVZ(25,17)
        DOUBLE PRECISION XINT
C
C               SUBROUTINE TO COMPUTE THE DENSITY FIELD OF THE WAVE
C
        G2 = 980.*980./2.06
        XINT = 0.
        WRITE(6,903)
        G = 980.
        DO 50 N = 1,NN
        DXX = DX
        IF (N.EQ.1) GO TO 2
        IF (N.NE.NN) GO TO 5
2       DXX = DX/2.
5       X = DX*FLOAT(N-1)
        DZ = DT*RH(N)
        GDZ = G*DZ
        DO 40 M = 1,MM
        DZZ = DZ
        I = (N-1)*MM +M
        IF (M.EQ.1) GO TO 10
        IF (M.NE.MM) GO TO 20
        DZZ = DZ/2.
        XL(M) = BV(N,M)*BX(I)/(G*G)
        GO TO 35
10      IP = I + 1
        DZZ = DZ/2.
        XL(M) = -(BX(IP) - BX(I))/GDZ
        GO TO 35
20      IP = I +1
        IM = I -1
        XL(M) = -(BX(IP) -BX(IM))/(2.*GDZ)
35      XX = G2*XL(M)*XL(M)/BV(N,M)
        XINT = XINT + XX*DZZ*DXX
40      CONTINUE
        WRITE(6,901) X,RH(N),DZ
50      WRITE(6,902) (XL(M),M=1,MM)
        WRITE(6,904) XINT
901     FORMAT(/'  X,D,DZ =',3E15.5)
902     FORMAT(2X,10E12.5)
903     FORMAT(///'  RHO'/)
```

```
904   FORMAT(//'  RHO CONTRIBUTION TO PE =',E15.5/)
      RETURN
      END

      SUBROUTINE UCAL(W,RL,BETA,DDLW)
      COMMON F,DX,DT,NN,MM,NM,NMX
      DOUBLE PRECISION AX,BX,XL
      COMMON AX(425,53),BX(425),XL(11475)
      COMMON RH(25),RHX(25),RHXX(25),BV(25,17),BVZ(25,17)
      DOUBLE PRECISION XINT
C
C                 SUBROUTINE TO COMPUTE THE U FIELD OF THE WAVE
C
      RHO = 0.515
      XINT = 0.
      FW = F*F - DDLW*W*W
      FL = F*RL
      WRITE(6,903)
      DO 100 N = 1,NN
      DXX = DX
      X = DX*FLOAT(N-1)
      DZ = DT*RH(N)
      DO 5 I = 1,MM
5     XL(I) = 0.
      IF (N.EQ.1) GO TO 90
      IF (N.EQ.NN) GO TO 110
      DD = RHX(N)/RH(N)
      DO 85 M = 1,MM
      I = (N-1)*MM +M
      IP = I + MM
      IM = I -MM
      T = -1. + DT*FLOAT(M-1)
      IF (M.EQ.1) GO TO 10
      IF (M.EQ.MM) GO TO 15
      GO TO 20
10    I1 = I +1
      XPT = (BX(I1) - BX(I))/DT
      GO TO 25
15    XPT = 0.
      GO TO 25
20    I2 = I+1
      I1 = I-1
      XPT = (BX(I2) - BX(I1))/(2.*DT)
25    XPX = (BX(IP) - BX(IM))/(2.*DX)
      XPX = XPX -T*DD*XPT
85    XL(M) = -(W*XPX + FL*BX(I))/FW
      GO TO 90
110   CC1 = W*FW/(DX*DX)
      CC2 = 0.5*(2.0*F*BETA*W +RL*F*FW)/DX
      CC3 = BETA*RL*(F*F +DDLW*W*W)
      DXX = DX/2.
      DO 120 M = 1,MM
      I = (NN-1)*MM +M
      IM = I - MM
      XPIM = BX(I)*(2.0*CC1-CC3)/(CC1+CC2)
      XPIM = XPIM + BX(IM)*(CC2-CC1)/(CC1+CC2)
120   XL(M) =-(FL*BX(I) + W*0.5*(XPIM - BX(IM))/DX)/FW
90    DO 95 M = 1,MM
      DZZ = DZ
      IF (M.EQ.1) GO TO 92
```

```
      IF (M.NE.MM) GO TO 95
   92 DZZ = DZ/2.
   95 XINT = XINT + RHO*XL(M)*XL(M)*DXX*DZZ
      WRITE(6,901) X,RH(N),DZ
      WRITE(6,902) (XL(M),M=1,MM)
  100 CONTINUE
      WRITE(6,904) XINT
  901 FORMAT(/'  X,D,DZ =',3E15.5)
  902 FORMAT(2X,10E12.5)
  903 FORMAT(///'   U'/)
  904 FORMAT(//'  U CONTRIBUTION TO KE =',E15.5/)
      RETURN
      END
```

Program CROSS

Listing

```
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      EXTERNAL FNA
      COMMCN S(101),P(101),G(1C1),T(101),X(1C1),VS(1C1),NV,DZ,SURF
      DIMENSION RH(25)
C
C              MAIN PROGRAM TO CALL THE CTHER PIECES
C
C                    REVISED APRIL,1985
C
      READ(5,*)NV,DD1
      READ(5,*)F,XMAX
      XMAX=XMAX*1.E5
      F=F*1.E-5
      IF(DD1.EQ.C.)WRITE(6,9C8)
      IF(DD1.EQ.1.)WRITE(6,9C9)
      WRITE(6,907)F,XMAX
      NN=25
      DX=XMAX/(NN-1)
      CALL DEP(DX,RH,NN)
      DZ=RH(NN)/(NV-1)
      K1=1
      X(1)=C.0
      X1=0.0
      X2=0.0
      H1=C.C
      H2=RH(1)
      DO 1CC I=2,NV
      Z=FLCAT(I-1)*DZ
      IF(Z.GE.RH(NN))Z=RH(NN)
  160 IF(Z.LE.H2)GO TO 150
      K1=K1+1
      H1=RH(K1-1)
      H2=RH(K1)
      X1=DX*FLOAT(K1-2)
      X2=DX*FLOAT(K1-1)
      GO TC 160
  15C DH=H2-H1
      SX=(X2-X1)/DH
      BX=-(X2*H1-X1*H2)/DH
      X(I)=SX*Z+BX
  100 CONTINUE
      CALL NSQ(RH(NN),F,DZ,VS,NV)
      SURF=DD1*VS(1)*F*F/981.
      READ(5,*)NRS
      WRITE(6,901)DZ
      WRITE(6,902)(X(I),I=1,NV)
      WRITE(6,903)
      WRITE(6,902)(VS(I),I=1,NV)
      DO 2CC IR=1,NRS
      READ(5,*)XMIN,XMAX,XINC
      XMIN=XMIN*1.E-7
      XMAX=XMAX*1.E-7
      XINC=XINC*1.E-7
      CALL ROOT(XMIN,XMAX,XINC,FNA,X3,IRCOT)
      IF(IRCOT.EQ.0)GO TO 2CO
      DO 3CC I=2,NV
  3CC P(I)=P(I)*DEXP(-X3*(X(I)+X(I-1))*.5)
      PSUM=C.0
      DO 35C I=2,NV
  350 PSUM=PSUM+P(I)**2
```

```
      PSUM=DSQRT(DZ*PSUM)
      DO 360 I=2,NV
  360 P(I)=P(I)/PSUM
      CF=F/X3
      WRITE(6,904)X3,CF
      WRITE(6,905)
      WRITE(6,902)(P(I),I=2,NV)
  200 CONTINUE
  901 FORMAT(//'  INVERSE TOPOGRAPHY, DZ=',D15.5/)
  902 FORMAT(1X,10D12.5)
  903 FORMAT(//'  NSQUARED/FSQUARED'/)
  904 FORMAT(////'  DISPERSION CURVE CROSSES W=F AT L=',D15.5,5X,
     1  'PHASE SPEED=',D15.5/)
  905 FORMAT(/'  PRESSURE FIELD (SURFACE TO BOTTOM)'/)
  907 FORMAT(//'  F,XMAX=',2D15.5)
  908 FORMAT(//'   RIGID LID'/)
  909 FORMAT(//'   FREE SURFACE'/)
      STOP
      END
      SUBROUTINE DEP(DX,RH,NN)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION XL(101),BX(101),RH(NN)
C
C          SUBROUTINE TO READ AND INTERPOLATE DEPTH PROFILE
C                    RH= DEPTH
C
      READ(5,*)NRX
      DO 5 I=1,NRX
      READ(5,*)XL(I),BX(I)
      BX(I)=BX(I)*100.
    5 XL(I)=XL(I)*1.E5
      IF(NRX.EQ.1)GO TO 7
      DO 6 I=2,NRX
    6 IF(BX(I).LE.BX(I-1))GO TO 100
    7 RHMA=BX(NRX)
      XMA=XL(NRX)
      RH(1)=BX(1)
      DO 20 N=2,25
      X=DX*FLOAT(N-1)
      IF(X.GT.XMA)GO TO 15
      IC=0
      DO 8 J=2,NRX
      IF(IC.NE.0)GO TO 8
      I=J
      IF(X.GT.XL(I))GO TO 8
      IC=I
    8 CONTINUE
      IM=I-1
      RHX=(BX(I)-BX(IM))/(XL(I)-XL(IM))
      XX=X-XL(IM)
      RH(N)=BX(IM)+RHX*XX
      GO TO 20
   15 RH(N)=RHMA
   20 CONTINUE
      GO TO 200
  100 WRITE(6,901)
  901 FORMAT(/'  TOPOGRAPHY IS NOT MONOTONIC'//)
  200 RETURN
      END
      SUBROUTINE NSQ(RHMA,F,DZ,VS,NV)
```

```
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION XL(1000),VS(NV)
C
C             SUBROUTINE TO READ AND INTERPOLATE BUOYANCY FREQUENCY
C
      FSQ=F*F
      READ(5,*)NR,DZR,ALPH
      ALPH=ALPH*1.E5
      DZR=DZR*100.
      READ(5,*)CMLT
      DO 20 I=1,NR
      READ(5,*)XL(I)
   20 XL(I)=XL(I)*CMLT
      TF=XL(NR)
      ZZZ=DZR*FLOAT(NR-1)
      I=NR+1
   50 Z=DZR*FLOAT(I-1)
      XL(I)=TF*DEXP((ZZZ-Z)/ALPH)
      IF(Z.GT.RHMA)GO TO 100
      I=I+1
      GO TO 50
  100 K1=1
      VS(1)=XL(1)/FSQ
      H1=0.0
      H2=DZR
      DO 200 I=2,NV
      Z=FLOAT(I-1)*DZ
      IF(Z.GE.RHMA)Z=RHMA
  160 IF(Z.LE.H2)GO TO 150
      K1=K1+1
      H1=FLOAT(K1-1)*DZR
      H2=FLOAT(K1)*DZR
      GO TO 160
  150 DH=H2-H1
      SN=(XL(K1+1)-XL(K1))/DH
      BN=-(XL(K1+1)*H1-XL(K1)*H2)/DH
      VS(I)=(SN*Z+BN)/FSQ
  200 CONTINUE
      RETURN
      END
      REAL*8 FUNCTION FNA(W)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON S(101),P(101),G(101),T(101),X(101),VS(101),NV,DZ,SURF
C
C             SUBFUNCTION TO INTEGRATE FROM Z=0 TO Z=-H
C
      NVM=NV-2
      W2=W*W
      TW=-2.*W
      DO 100 I=1,NV
      EX=DEXP(TW*X(I))
      T(I)=W2*EX
  100 S(I)=EX/VS(I)
      P(NV)=1.0
      G(NV)=0.0
      DO 200 I=1,NVM
      J=NV-I
      K=J+1
      G(J)=G(K)-.5*DZ*(T(J)+T(K))*P(K)
  200 P(J)=P(K)+DZ*G(J)/S(J)
```

```
      G(1)=G(2)-.5*DZ*(T(1)+T(2))*P(2)
      FNA=C(1)+S(1)*SURF*.5*(3.*P(2)-P(3))
      RETURN
      END
      SUBROUTINE ROOT(X5,X6,X7,FNA,X3,IROOT)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
C           SUBROUTINE TO FIND ROOT OF FNA BETWEEN X5 AND X6
C
      IROOT=1
  220 IQ=INT((X6-X5)/X7)-1
      I1=0
  240 X1=X5+I1*X7
      X2=X1+X7
      Y1=FNA(X1)
      Y2=FNA(X2)
      IF(Y1*Y2.LT.0.0)GO TO 340
      I1=I1+1
      IF(I1.NE.IQ)GO TO 240
      WRITE(6,13)
      IROOT=0
   13 FORMAT(//'  NO ROOT FOUND'/)
      GO TO 400
  340 IF(X1.NE.X5)GO TO 370
      X7=X7/2.
      GO TO 220
  370 X2=X1
      X1=X1-0.1*X7
      Y1=FNA(X1)
      Y2=FNA(X2)
  420 X3=(Y2*X1-Y1*X2)/(Y2-Y1)
      IF(X3.GT.X5.AND.X3.LT.X6)GO TO 200
      WRITE(6,11)X3
   11 FORMAT(/'  PROJECTION OUT OF INTERVAL:',D15.5//)
  200 Y3=FNA(X3)
      IF(DABS((X3-X2)/X2).LT.1.E-7)GO TO 400
      X1=X2
      X2=X3
      Y1=Y2
      Y2=Y3
      GO TO 420
  400 RETURN
      END
```

Program BIGDRV2

Listing

```
      PROGRAM WDSTF
      COMMON F,DX,DT,NN,MM,NM,NMX
      COMMON RH(25),RHX(25),RHXX(25),BV(25,17),BVZ(25,17)
      COMMON XL(600)
      DOUBLE COMPLEX AX(425,53),BX(425),BXB(425),DCMPLX
      DIMENSION BBB(50),TX(25),TY(25),TYX(25),TXX(25)
      DIMENSION R(25),RX(25)
      DOUBLE PRECISION DREAL,DIMAG,DATAN2,A,B,AMT,THT,DSQRT
      DOUBLE COMPLEX FFC(25)
C
C             MAIN PROGRAM TO CALL ALL OF THE OTHER PIECES
C
C             DIM(AX) = NM,2NN+3
      READ(5,*) ICOM
      ICCC =1
      READ(5,*) F,XMAX
      FF = 180.0/3.14149
      XMAX = XMAX*1.0E+05
      F =F*1.0E-05
      WRITE(6,907) F,XMAX
      READ(5,*) ILW,IRL,IXY
      IF (ILW.EQ.1) GO TO 5
      ILW = 0
      WRITE(6,910)
      GO TO 10
 5    WRITE(6,911)
 10   DD2 = FLOAT(ILW)
      IF (IRL.EQ.1) GO TO 15
      IRL = 0
      WRITE(6,912)
      GO TO 25
 15   WRITE(6,913)
 25   DD1 = FLOAT(IRL)
      IF (IXY.EQ.1) GO TO 27
      IXY = 0
      WRITE(6,915)
      GO TO 29
 27   IF (ILW.EQ.0) GO TO 28
      WRITE(6,916)
      GO TO 29
 28   WRITE(6,917)
      GO TO 200
 29   MM = 17
      NN = 25
      NM = NN*MM
      NMX = 2*NN +3
      DX = XMAX/FLOAT(NN-1)
      DT = 1./FLOAT(MM-1)
      CALL DEP(BBB,RQPP)
      CALL NSQ
      CALL FRIC(R,RX)
      CALL WRD(TX,TY,TXX,TYX,IXY)
 3    READ(5,*) W,RL
      CALL MATS(RL,W,AX,BX,R,RX,DD1,DD2,TX,TY,TXX,TYX,FFC)
      DO 20 M = 1,MM
      DO 20 N = 1,NN
      IN = N + NN*(M-1)
      II = M + MM*(N-1)
 20   BXB(II) = BX(IN)
      DO 30 I = 1,NM
```

```
 30       BX(I) = BXB(I)
          WRITE(6,903) W,RL
          CALL VCAL(W,RL,BX,BXB,R,RX,VKE,DD2,TX,TY)
          CALL UCAL(W,RL,BX,BXB,R,UKE,DD2,TX,TY)
          CALL RHOC(BX,BXB,RPE,DD1,DD2,FFC)
          CALL PEC(BX,SPE,IRL)
          RRRR = (UKE + VKE)/(RPE + SPE)
          TE = UKE + VKE + RPE + SPE
          WRITE(6,914) TE
          WRITE(6,909) RRRR
 125      WRITE(6,908)
 126      DO 130 N = 1,NN
          X = DX*FLOAT(N-1)
          DZ = RH(N)*DT
          WRITE(6,904) X,RH(N),DZ
          ML = 1 +MM*(N-1)
          MH = MM*N
          DO 128 M = ML,MH
          A = DREAL(BX(M))
          B = DIMAG(BX(M))
          AMT = DSQRT(A*A + B*B)
          THT = FF*DATAN2(B,A)
 128      BX(M) = DCMPLX(AMT,THT)
          WRITE(6,901) (BX(M),M=ML,MH)
 130      CONTINUE
 140      ICCC = ICCC +1
          IF (ICCC.LE.ICCM) GO TO 3
 901      FORMAT(2X,E12.5,F9.2,E12.5,F9.2,E12.5,F9.2,E12.5,F9.2,E12.5,F9.2)
 903      FORMAT(////' W,L =',2E15.5)
 904      FORMAT(/'  X,H,DZ =',3E15.5)
 907      FORMAT(//'  F,XMAX =',2E15.5//)
 908      FORMAT(//'  PRESSURE'/)
 909      FORMAT(/' KE/PE =',E15.5)
 910      FORMAT(' LONG WAVE')
 911      FORMAT(' GENERAL FREQUENCY AND WAVELENGTH')
 912      FORMAT(' RIGID LID')
 913      FORMAT(' FREE SURFACE')
 914      FORMAT(' TOTAL ENERGY/LENGTH =',E15.5)
 915      FORMAT(' TAU Y DRIVING')
 916      FORMAT(' TAU X DRIVING')
 917      FORMAT(' ERROR: TAU X DRIVING IN THE LONG WAVE LIMIT')
 200      STOP
          END

          SUBROUTINE NSQ
          COMMON F,DX,DT,NN,MM,NM,NMX
          COMMON RH(25),RHX(25),RHXX(25),BV(25,17),BVZ(25,17)
          COMMON XL(600)
 C
 C
 C                  PROGRAM TO READ AND INTERPOLATE BUOYANCY FREQUENCY SQUARE
 C                            BV = BUOYANCY FREQUENCY SQUARED
 C                            BVZ = Z DERIVATIVE OF BV
 C
          READ(5,*) NR,DZR,ALPH
          ALPH = ALPH*1.0E+05
          DZR = DZR*100.
          MX = MM -1
          READ(5,*) CMLT
          DO 20 I = 1,NR
```

```
      READ(5,*) XL(I)
   20 XL(I) = XL(I)*CMLT
      TF = XL(NR)
      ZZZ = DZR*FLOAT(NR-1)
      DZI = RH(NN) - ZZZ
      NL = NR +1
      NH = 600
      DO 30 I = NL,NH
      Z = DZR*FLOAT(I-1)
   30 XL(I) = TF*EXP((ZZZ -Z)/ALPH)
      DO 200 N = 1,NN
      DD = RH(N)
      DZ = DT*DD
      IF (DZ.GT.DZR) GO TO 110
      BV(N,MM) = XL(1)
      DO 50 M = 1,MX
      Z = DD - DZ*FLOAT(M-1)
      IBXL = 1 + IFIX(Z/DZR)
      IBXH = IBXL +1
      ZS = DZR*FLOAT(IBXL -1)
   50 BV(N,M) = XL(IBXL) +(Z-ZS)*(XL(IBXH)-XL(IBXL))/DZR
      Z = DD - DZ
      IBXL = 1 + IFIX(Z/DZR)
      IBXH = IBXL + 1
      ZS = DZR*FLOAT(IBXL-1)
      AQ = XL(IBXL) +(Z-ZS)*(XL(IBXH)-XL(IBXL))/DZR
      GO TO 145
  110 ZD = DZ/2.
      XBB = 0.
      NAVG = 0
      DO 120 I = 1,NR
      ZC = DZR*FLOAT(I-1)
      IF (ZC.GT.ZD) GO TO 120
      XBB = XBB + XL(I)
      NAVG = NAVG +1
  120 CONTINUE
      BV(N,MM) = XBB/FLOAT(NAVG)
      DO 140 MQ = 1,MM
      M = MQ -1
      Z = DD - DZ*FLOAT(M-1)
      ZS = Z - DZ/2.
      ZD = ZS + DZ
  125 XBB = 0.
      NAVG = 0
      DO 130 I = 1,NH
      ZC = DZR*FLOAT(I-1)
      IF ( ZC.LT.ZS) GO TO 130
      IF (ZC.GT.ZD) GO TO 130
      XBB = XBB + XL(I)
      NAVG = NAVG + 1
  130 CONTINUE
      IF (NAVG.NE.0) GO TO 135
      IBXL = 1 + IFIX(Z/DZR)
      IBXH = IBXL + 1
      ZSS = DZR*FLOAT(IBXL-1)
      IF (M.EQ.0) GO TO 133
      BV(N,M) = XL(IBXL) +(Z-ZSS)*(XL(IBXH)-XL(IBXL))/DZR
      GO TO 140
  133 AQ = XL(IBXL) +(Z-ZSS)*(XL(IBXH)-XL(IBXL))/DZR
      GO TO 140
```

```
135   IF (M.EQ.0) GO TO 138
      BV(N,M) = XBB/FLOAT(NAVG)
      GO TO 140
138   AQ = XBB/FLOAT(NAVG)
140   CONTINUE
145   DO 150 M = 2,MX
      IP = M +1
      IM = M -1
150   BVZ(N,M) = (BV(N,IP) - BV(N,IM))/(2.*DZ)
      BVZ(N,1) = (BV(N,2) -AQ)/(2.0*DZ)
      BVZ(N,MM) = (BV(N,MM) - BV(N,MX))/DZ
200   CONTINUE
      WRITE(6,901)
      WRITE(6,902) (BV(NN,J),J=1,MM)
901   FORMAT(//'   NSQUARED AT XMAX '/)
902   FORMAT(2X,10E12.5)
250   RETURN
      END

      SUBROUTINE MATS(RL,W,AX,BX,R,RX,DD1,DD2,TX,TY,TXX,TYX,FFC)
      COMMON F,DX,DT,NN,MM,NM,NMX
      COMMON RH(25),RHX(25),RHXX(25),BV(25,17),BVZ(25,17)
      COMMON XL(600)
      DOUBLE COMPLEX AX(425,53),BX(425),AB,DCMPLX,B3,B5
      DOUBLE COMPLEX AA3,AA4,AA5,DQ,AA1,AA2,FFC(25)
      DOUBLE COMPLEX GAM,B4,ABB
      DIMENSION R(25),RX(25),TX(25),TY(25),TYX(25),TXX(25)
C
C
C                SUBROUTINE TO SET AND SOLVE THE P EQUATION
C
      GG = 1./980.
      CSS = GG*2.0*DT*DD1
      CALL FFCCAL(TX,TY,TXX,TYX,FFC,W,RL,DD2)
      DXDT = DX/(2.*DT)
      DXX = DX*DX
      DDD = DXX/(DT*DT)
      RLL = DD2*RL*RL
      DDX = DX*DXDT
      RQP = 2.0*DX
      DQ = DCMPLX(RQP,0.0)
      FLW = F*RL/W
3     DO 5 I = 1,NM
5     BX(I) = (0.0,0.0)
      DO 10 J = 1,NMX
      DO 10 I = 1,NM
10    AX(I,J) =(0.0,0.0)
      J = NN + 2
      IP1 = J + 1
      IM1 = J - 1
      IJJP = J + 2
      IJJM = J - 2
      IPM1 = 2*NN +1
      IPM = 2*NN + 2
      IPMM = 2*NN + 3
      DO 100 M = 1,MM
      DO 90 N = 1,NN
      I = N + NN*(M-1)
      CALL AS(A1,A2,A3,W,N,M,C1,C2,C3,DD2)
      RQP = A2*DXDT
```

```
      AB = DCMPLX(RQP,0.0)
      AX(I,1) = AB
      AX(I,IPM1) = -AB
      AX(I,3) = -AB
      AX(I,IPMM) = AB
      AX(I,IM1) = (1.0,0.0)
      AX(I,IP1) = (1.0,0.0)
      RQP = A1*DDD - A3*DDX
      AX(I,2) = DCMPLX(RQP,0.0)
      RQP = A1*DDD + A3*DDX
      AX(I,IPM) = DCMPLX(RQP,0.0)
      RQP = -2.0 -2.0*A1*DDD -RLL*DXX
      AX(I,J) = DCMPLX(RQP,0.0)
      IF ( N.EQ.1) GO TO 20
      IF (N.EQ.NN) GO TO 30
      IF (M.EQ.1) GO TO 40
      IF (M.EQ.MM) GO TO 50
      GO TO 90
20    RQP = 2.0*DX
      FW = F*F - DD2*W*W
      CHH = RL*F*RH(N)
      CH = R(N)*DD2*2.0*F*W*RL/FW
      AA1 = DCMPLX(CH,CHH)
      CH = R(N)*(F*F + DD2*W*W)/FW
      CHH = W*RH(N)
      AA2 = DCMPLX(CH,CHH)
      CH = F*TY(N)
      CHH = W*TX(N)*DD2
      ABB = DCMPLX(CH,CHH)
      ABB = 2.0*DX*ABB/AA2
      BX(I) = BX(I) + AX(I,IM1)*ABB
      BX(I) = BX(I) + AX(I,IPM1)*ABB
      BX(I) = BX(I) + AX(I,1)*ABB
      AX(I,IP1) = AX(I,IP1) + AX(I,IM1)
      AX(I,J) = AX(I,J) + RQP*AA1*AX(I,IM1)/AA2
      AX(I,IPM) = AX(I,IPM) + AX(I,IM1)*A2*DX/DT
      AX(I,2) = AX(I,2) - AX(I,IM1)*A2*DX/DT
      AX(I,IPMM) = AX(I,IPMM) + AX(I,IPM1)
      AX(I,IPM) = AX(I,IPM) + AX(I,IPM1)*RQP*(AA1/AA2 + A2/DT)
      AX(I,J) = AX(I,J) - AX(I,IPM1)*A2*RQP/DT
      AX(I,3) = AX(I,3) + AX(I,1)
      AX(I,2) = AX(I,2) + AX(I,1)*RQP*(AA1/AA2 - A2/DT)
      AX(I,J) = AX(I,J) + AX(I,1)*A2*RQP/DT
      AX(I,1) = (0.0,0.0)
      AX(I,IPM1) = (0.0,0.0)
      AX(I,IM1) =(0.0,0.0)
24    IF (M.NE.MM) GO TO 25
      AX(I,2) = AX(I,2) + AX(I,IPM)
      AX(I,3) = AX(I,3) + AX(I,IPMM)
      CZQ = RH(N)*BV(N,M)*CSS
      AX(I,IP1) = AX(I,IP1) - AX(I,IPMM)*CZQ
      AX(I,J) = AX(I,J) - AX(I,IPM)*CZQ
      BX(I) = BX(I) - AX(I,IPM)*2.0*RH(1)*DT*FFC(1)
      BX(I) = BX(I) - AX(I,IPMM)*2.0*RH(1)*DT*FFC(1)
      AX(I,IPM) = (0.0,0.0)
      AX(I,IPMM) = (0.0,0.0)
      GO TO 90
25    IF (M.NE.1) GO TO 90
      CH = R(N)*DD2*2.0*F*W*RL/FW
      CHH = F*RL*RH(N)
```

```
      B3 = DCMPLX(CH,CHH)
      CH = R(N)*(F*F + DD2*W*W)/FW
      CHH = W*RH(N)
      B4 = DCMPLX(CH,CHH)
      CH = F*TY(N)
      CHH = W*DD2*TX(N)
      B5 = DCMPLX(CH,CHH)
      CH = -2.0*W*RL*DD2*F*RX(N)/FW
      CHH = -RL*F*RHX(N)
      AA1 = DCMPLX(CH,CHH)
      CH = (F*F + DD2*W*W)*RX(N)/FW
      CHH = W*RHX(N)
      AA2 = DCMPLX(CH,CHH)
      AA1 = AA1 + AA2*B3/B4
      CH = -R(N)*(F*F +DD2*W*W)*BVZ(N,M)/(BV(N,M)*BV(N,M))
      CH = CH + 2.0*W*RL*DD2*F*R(N)*RHX(N)/FW
      CHH = -W*FW/BV(N,M)
      AA2 = DCMPLX(CH,CHH)
      CH = (F*F + DD2*W*W)*(-R(N)*RHX(N))/FW
      AA2 = AA2 + CH*B3/B4
      CH = R(N)*(F*F +DD2*W*W)/BV(N,M)
      AA3 = DCMPLX(CH,0.0)
      CH = -(F*F + DD2*W*W)*RX(N)/FW
      CHH = -W*RHX(N)
      AA4 = DCMPLX(CH,CHH)
      AA4 = AA4/B4
      DZ = DT*RH(N)
      ABB = (0.5*AA2 - AA3/DZ)/DZ
      AX(I,J) = AX(I,J)+ AX(I,2)*(AA1 -2.0*AA3/(DZ*DZ))/ABB
      AX(I,IPM) = AX(I,IPM) +AX(I,2)*(0.5*AA2 +AA3/DZ)/(DZ*ABB)
      BX(I) = BX(I) - AX(I,2)*AA4*B5/ABB
      RQP = 2.0*DX
      CALL AACAL(W,RL,R,RX,N,M,DD2,AA1,AA2,AA3,AA4,AA5)
      ABB = AA1*0.5/DT -AA4/(DT*DT)
      AX(I,J) = AX(I,J) + AX(I,3)*(-AA3/RQP + AA5/(DX*DT))/ABB
      B3 = AX(I,3)*(AA2 -2.0*AA4/(DT*DT) -AA5/(DX*DT))/ABB
      AX(I,IP1) = AX(I,IP1) + B3
      AX(I,IJJP) = AX(I,IJJP) + AX(I,3)*AA3/(RQP*ABB)
      AX(I,IPM) = AX(I,IPM) -AX(I,3)*AA5/(DX*DT*ABB)
      B3 = AX(I,3)*(AA1*0.5/DT + AA4/(DT*DT) + AA5/(DX*DT))/ABB
      AX(I,IPMM) = AX(I,IPMM) +B3
      AX(I,3) = (0.0,0.0)
      AX(I,2) = (0.0,0.0)
      GO TO 90
30    DZ = DT*RH(N)
      AX(I,IPM1) = (0.0,0.0)
      AX(I,3) = (0.0,0.0)
      AX(I,IPMM) = (0.0,0.0)
      AX(I,1) = (0.0,0.0)
      C1 = 1. + FLW*DX/2.
      C2 = 1. - FLW*DX/2.
      AX(I,J) = AX(I,J) +2.0*AX(I,IP1)/C1
      AX(I,IM1) = AX(I,IM1) -C2*AX(I,IP1)/C1
      AX(I,IP1) = (0.0,0.0)
      IF (M.NE.MM) GO TO 35
      AX(I,2) = AX(I,2) +AX(I,IPM)
      CZQ = RH(N)*BV(N,M)*CSS
      AX(I,J) = AX(I,J) - AX(I,IPM)*CZQ
      BX(I) = BX(I) -AX(I,IPM)*2.0*RH(N)*DT*FFC(N)
      AX(I,IPM) = (0.0,0.0)
```

```
          GO TO 90
   35     IF (M.NE.1) GO TO 90
          CA = R(N)*(F*F + DD2*W*W)/(F*F -DD2*W*W)
          CH = -CA*BVZ(N,M)/BV(N,M)
          CHH = -W
          ABB = DCMPLX(CH,CHH)
          ABB = ABB/CA
          AA1 = 1.0 -ABB*RH(N)*DT/2.0
          AA2 = 1.0 + ABB*DT*RH(N)/2.0
          AX(I,J) = AX(I,J) +AX(I,2)*2.0/AA1
          AX(I,IPM) = AX(I,IPM) -AX(I,2)*AA2/AA1
          AX(I,2) = (0.0,0.0)
          GO TO 90
   40     CALL AACAL(W,RL,R,RX,N,M,DD2,AA1,AA2,AA3,AA4,AA5)
          DTQ = 2.0*DT*DX
          RQP = 2.0*DX
          ABB = AA1*0.5/DT - AA4/(DT*DT)
          AX(I,IM1) = AX(I,IM1) + AX(I,2)*(-AA3/RQP + AA5/DTQ)/ABB
          AX(I,J) = AX(I,J) + AX(I,2)*(AA2 - AA4*2.0/(DT*DT))/ABB
          AX(I,IP1) = AX(I,IP1) + AX(I,2)*(AA3/RQP - AA5/DTQ)/ABB
          AX(I,IPM1) = AX(I,IPM1) + AX(I,2)*(-AA5/DTQ)/ABB
          AX(I,IPM) = AX(I,IPM) + AX(I,2)*(AA1*0.5/DT + AA4/(DT*DT))/ABB
          AX(I,IPMM) = AX(I,IPMM) + AX(I,2)*AA5/(DTQ*ABB)
          IF (N.EQ.2) GO TO 42
          AX(I,IJJM) = AX(I,IJJM) - AX(I,1)*AA3/(RQP*ABB)
          B3 = AX(I,1)*(AA2 - 2.0*AA4/(DT*DT) + AA5*2.0/DTQ)/ABB
          AX(I,IM1) = AX(I,IM1) + B3
          AX(I,J) = AX(I,J) + AX(I,1)*(AA3/RQP - AA5*2.0/DTQ)/ABB
          B3 = AX(I,1)*(AA1*0.5/DT + AA4/(DT*DT) - AA5*2.0/DTQ)/ABB
          AX(I,IPM1) = AX(I,IPM1) + B3
          AX(I,IPM) = AX(I,IPM) + AX(I,1)*AA5*2.0/(DTQ*ABB)
   41     AX(I,J) = AX(I,J) + AX(I,3)*(-AA3/RQP + AA5*2.0/DTQ)/ABB
          B3 = AX(I,3)*(AA2 - AA4*2.0/(DT*DT) -AA5*2.0/DTQ)/ABB
          AX(I,IP1) = AX(I,IP1) + B3
          AX(I,IJJP) = AX(I,IJJP) + AX(I,3)*AA3/(RQP*ABB)
          AX(I,IPM) = AX(I,IPM) - AX(I,3)*AA5*2.0/(DTQ*ABB)
          B3 = AX(I,3)*(AA1*0.5/DT +AA4/(DT*DT) + AA5*2.0/DTQ)/ABB
          AX(I,IPMM) = AX(I,IPMM) + B3
          AX(I,3) = (0.0,0.0)
          AX(I,1) = (0.0,0.0)
          AX(I,2) = (0.0,0.0)
          GO TO 90
   42     B3 = AX(I,1)*(AA2 -AA3/DX -2.0*AA4/(DT*DT) + AA5*2.0/DTQ)/ABB
          AX(I,IM1) = AX(I,IM1) + B3
          AX(I,J) = AX(I,J) + AX(I,1)*(AA3/DX - AA5*2.0/DTQ)/ABB
          B3 = AX(I,1)*(0.5*AA1/DT + AA4/(DT*DT) -AA5*2.0/DTQ)/ABB
          AX(I,IPM1) = AX(I,IPM1) + B3
          AX(I,IPM) = AX(I,IPM) + AX(I,1)*AA5*2.0/(DTQ*ABB)
          GO TO 41
   50     AX(I,1) = (0.0,0.0)
          AX(I,3) = (0.0,0.0)
          AX(I,IPM1) = (0.0,0.0)
          AX(I,IPMM) =(0.0,0.0)
          AX(I,2) = AX(I,2) + AX(I,IPM)
          CZQ = RH(N)*BV(N,M)*CSS
          AX(I,J) = AX(I,J) -AX(I,IPM)*CZQ
          BX(I) = BX(I) - AX(I,IPM)*2.0*RH(N)*DT*FFC(N)
          AX(I,IPM) = (0.0,0.0)
   90     CONTINUE
  100     CONTINUE
```

```
      CALL BANDG(AX,BX)
      RETURN
      END

      SUBROUTINE AS(A1,A2,A3,W,N,M,C1,C2,C3,DD2)
      COMMON F,DX,DT,NN,MM,NM,NMX
      COMMON RH(25),RHX(25),RHXX(25),BV(25,17),BVZ(25,17)
      COMMON XX(600)
C
C           SUBROUTINE TO CALCULATE COEFFICIENTS FOR MATS
C
      F2 = F*F
      W2 = DD2*W*W
      TZ = 1./RH(N)
      TH = -1. +DT*FLOAT(M-1)
      TX = -RHX(N)*TH*TZ
      AA = (RHX(N)/RH(N))**2
      TXX = (2.*AA -RHXX(N)/RH(N))*TH
      BW = BV(N,M)
      BWW = BW*BW
      A1 = TX*TX + TZ*TZ*(F2-W2)/BW
      A2 = TX
      A3 = TXX -TZ*BVZ(N,M)*(F2-W2)/BWW
      C1 = TZ
      C2 = RHX(N)*BV(N,1)/(F2-W2)
      C3 = C1*RHX(N)
      RETURN
      END

      SUBROUTINE DEP(BX,RQPP)
      COMMON F,DX,DT,NN,MM,NM,NMX
      COMMON RH(25),RHX(25),RHXX(25),BV(25,17),BVZ(25,17)
      COMMON XL(600)
      DIMENSION BX(50)
C
C           SUBROUTINE TO READ AND INTERPOLATE DEPTH PROFILE
C                     RH = DEPTH
C                     RHX = X DERIVATIVE OF RH
C                     RHXX = SECOND X DERIVATIVE OF RH
C
      READ(5,*) NRX
      DO 5 I =1,NRX
      READ(5,*) XL(I),BX(I)
      BX(I) = BX(I)*100.
    5 XL(I) = XL(I)*1.0E+05
      RHMA = BX(NRX)
      RQPP = XL(NRX)
      RH(1) = BX(1)
      DO 20 N = 2,NN
      X = DX*FLOAT(N-1)
      IF (X.GT.XL(NRX)) GO TO 15
      IC = 0
      DO 8 J = 2,NRX
      IF (IC.NE.0) GO TO 8
      I = J
      IF (X.GT.XL(I)) GO TO 8
      IC = I
    8 CONTINUE
      IM = I-1
      RHX(N) = (BX(I) -BX(IM))/(XL(I)-XL(IM))
```

```
      XX = X - XL(IM)
      RH(N) = BX(IM) + RHX(N)*XX
      GO TO 20
 15   RH(N) = RHMA
 20   CONTINUE
      RHX(1) = (RH(2) - RH(1))/DX
      RHXX(1) = 0.
      D2 = 2.*DX
      DXX = DX*DX
      NX = NN -1
      DO 30 N = 2,NX
      IP = N +1
      IM = N -1
      RHX(N) =(RH(IP) - RH(IM))/D2
 30   RHXX(N) = (RH(IP) -2.*RH(N) + RH(IM))/DXX
      RHX(NN) = 0.
      RHXX(NN) = 0.
      RETURN
      END

      SUBROUTINE VCAL(W,RL,BX,XL,R,RX,XINT,DD2,TX,TY)
      COMMON F,DX,DT,NN,MM,NM,NMX
      COMMON RH(25),RHX(25),RHXX(25),BV(25,17),BVZ(25,17)
      COMMON XB(600)
      DOUBLE COMPLEX BX(425),XL(425),XPX,XPT,DCMPLX,RMU,T3,B3,B5,AA1,AA2
      DOUBLE COMPLEX GAM,B4,AA3,AA4,AA5
      DOUBLE PRECISION DREAL,DIMAG,DATAN2,DSQRT,A,B
      DIMENSION R(25),RX(25),TX(25),TY(25)
      DOUBLE PRECISION XINT
C
C              SUBROUTINE TO CALCULATE V FROM P
C
      RHO = 0.5015
      FF = 180.0/3.14159
      XINT = 0.
      WRITE(6,905)
      FW = F*F - DD2*W*W
      WL= DD2*W*RL
      RLW = RL/W
      FLW = F*RL/W
      MX = MM -1
      DXX = 2.*DX
      W2 = DD2*W*W
      DO 50 N = 1,NN
      XX = DX
      X = DX*FLOAT(N-1)
      D = RH(N)
      DD = RHX(N)/D
      DZ = DT*D
      RHZ = RHX(N)**2
      IF (N.EQ.1) GO TO 2
      IF (N.NE.NN) GO TO 20
      GO TO 18
 2    CH = R(N)*(F*F + W2)/FW
      CHH = W*RH(N)
      GAM = DCMPLX(CH,CHH)
      CH = F*TY(N)
      CHH = W*TX(N)*DD2
      XPX = DCMPLX(CH,CHH)
      XPX = XPX/GAM
```

```
      XPT = XPX
      CH = R(N)*DD2*2.0*W*F*RL/FW
      CHH = RL*F*RH(N)
      B4 = DCMPLX(CH,CHH)
      DO 5 I = 1,MM
      XPX = XPT -B4*BX(I)/GAM
    5 XL(I) = (F*XPX + WL*BX(I))/FW
      XX = DX/2.0
      GO TO 40
   18 C1 =  1. + FLW*DX/2.
      C2 = 1. - FLW*DX/2.
      DQ = 1./(F*DX)
      DO 19 M = 1,MM
      I = (N-1)*MM +M
      IM = I -MM
      XPX = (BX(I)*2.0/C1 - BX(IM)*(1. + C2/C1))/DXX
   19 XL(I) = (F*XPX + WL*BX(I))/FW
      XX = DX/2.
      GO TO 40
   20 I = (N-1)*MM +1
      M = 1
      IP = I + MM
      IM = I - MM
      I2 = I + 1
      I1 = I - 1
      IQQ = I -MM + 1
      IQR = I + MM + 1
      CALL AACAL(W,RL,R,RX,N,M,DD2,AA1,AA2,AA3,AA4,AA5)
      DTT = 2.0*DT
      GAM = AA1/DTT - AA4/(DT*DT)
      B3 = BX(IM)*(-AA3/DXX +AA5/(DXX*DT))
      B3 = B3 +BX(I)*(AA2-2.0*AA4/(DT*DT))
      B3 = B3 + BX(IP)*(AA3/DXX -AA5/(DXX*DT))
      B3 = B3 + BX(IQQ)*(-AA5/(DXX*DT))
      B3 = B3 + BX(I2)*(AA1/DTT + AA4/(DT*DT))
      B3 = B3 + BX(IQR)*(AA5/(DTT*DX))
      B3 = B3/GAM
      XPX = (BX(IP) - BX(IM))/DXX
      T = -1.0
      XPT = (BX(I2)-B3)/DTT
      XPX = XPX - T*DD*XPT
      XL(I) = (F*XPX + WL*BX(I))/FW
      I = N*MM
      IP = I + MM
      IM = I -MM
      XPX = (BX(IP) - BX(IM))/DXX
      XL(I) =(WL*BX(I) + F*XPX)/FW
      DO 30 M = 2,MX
      I = (N-1)*MM +M
      IP = I + MM
      IM = I -MM
      I2 = I +1
      I1 = I -1
      XPX = (BX(IP) - BX(IM))/DXX
      T = -1. +DT*FLOAT(M-1)
      XPT = (BX(I2) -BX(I1))/(2.*DT)
      XPX = XPX - T*DD*XPT
   30 XL(I) = (WL*BX(I) + F*XPX)/FW
   40 DO 45 M = 1,MM
      I = (N-1)*MM + M
```

```
            ZZ = DZ
            A = DREAL(XL(I))
            B = DIMAG(XL(I))
            CQ = A*A + B*B
            IF (M.EQ.1) GO TO 42
            IF (M.NE.MM) GO TO 45
   42       ZZ = DZ/2.
   45       XINT = XINT + RHO*XX*ZZ*CQ
            WRITE(6,904) X,D,DZ
            IL = (N-1)*MM +1
            IH = N*MM
            DO 46 IJ = IL,IH
            A = DREAL(XL(IJ))
            B = DIMAG(XL(IJ))
            AMT = DSQRT(A*A + B*B)
            THT = FF*DATAN2(B,A)
   46       XL(IJ) = DCMPLX(AMT,THT)
            WRITE(6,901) (XL(IJ),IJ=IL,IH)
   50       CONTINUE
            WRITE(6,902) XINT
            DO 110 N = 1,NN
            IJ = 1 + MM*(N-1)
            A = DREAL(XL(IJ))
            B = DIMAG(XL(IJ))
            AMT = R(N)*A
            THT = B
  110       XL(N) = DCMPLX(AMT,THT)
            WRITE(6,906)
            WRITE(6,901)(XL(N),N=1,NN)
  901       FORMAT(2X,E12.5,F9.2,E12.5,F9.2,E12.5,F9.2,E12.5,F9.2,E12.5,F9.2)
  902       FORMAT(//'  V CONTRIBUTION TO KE =',E15.5/)
  905       FORMAT(////'  V'//)
  904       FORMAT(/'     X, H, DZ =',3E15.5)
  906       FORMAT(//'  Y BOTTOM STRESS (DYNES/CM2)='/)
            RETURN
            END

            SUBROUTINE RHOC(BX,XL,XINT,DD1,DD2,FFC)
            COMMON F,DX,DT,NN,MM,NM,NMX
            COMMON RH(25),RHX(25),RHXX(25),BV(25,17),BVZ(25,17)
            COMMON XQ(600)
            DOUBLE COMPLEX BX(425),XL(425),DCMPLX,FFC(25)
            DOUBLE PRECISION DREAL,DIMAG,DATAN2,DSQRT,A,B,CDABS
            DOUBLE PRECISION XINT
   C
   C                  SUBROUTINE TO CALCULATE DENSITY FROM P
   C
            G2 = 980.*980./2.06
            GRC = (1.0/980.0)**2
            FF = 180.0/3.14159
            XINT = 0.
            WRITE(6,903)
            G = 980.
            DO 50 N = 1,NN
            DXX = DX
            IF (N.EQ.1) GO TO 2
            IF (N.NE.NN) GO TO 5
    2       DXX = DX/2.
    5       X = DX*FLOAT(N-1)
            DZ = DT*RH(N)
```

```
      GDZ = G*DZ
      DO 40 M = 1,MM
      DZZ = DZ
      I = (N-1)*MM +M
      IF (M.EQ.1) GO TO 10
      IF (M.NE.MM) GO TO 20
      DZZ = DZ/2.
      XL(M) = DD1*BV(N,M)*GRC*BX(I) -FFC(N)/980.
      GO TO 35
10    IP = I + 1
      DZZ = DZ/2.
      XL(M) = -(BX(IP) - BX(I))/GDZ
      GO TO 35
20    IP = I +1
      IM = I -1
      XL(M) = -(BX(IP) -BX(IM))/(2.*GDZ)
35    XX = G2*CDABS(XL(M))*CDABS(XL(M))/BV(N,M)
      XINT = XINT + XX*DZZ*DXX
40    CONTINUE
      WRITE(6,901) X,RH(N),DZ
      DO 45 M = 1,MM
      A = DREAL(XL(M))
      B = DIMAG(XL(M))
      AMT = 1000.0*DSQRT(A*A + B*B)
      IF (AMT.NE.0.0) GO TO 42
41    THT = 0.0
      GO TO 45
42    THT = FF*DATAN2(B,A)
45    XL(M) = DCMPLX(AMT,THT)
50    WRITE(6,902) (XL(M),M=1,MM)
      WRITE(6,904) XINT
901   FORMAT(/'  X,D,DZ =',3E15.5)
902   FORMAT(2X,E12.5,F9.2,E12.5,F9.2,E12.5,F9.2,E12.5,F9.2,E12.5,F9.2
903   FORMAT(///'  RHO (SIGMA-T UNITS)'/)
904   FORMAT(//'  RHO CONTRIBUTION TO PE =',E15.5/)
      RETURN
      END

      SUBROUTINE UCAL(W,RL,BX,XL,R,XINT,DD2,TX,TY)
      COMMON F,DX,DT,NN,MM,NM,NMX
      COMMON RH(25),RHX(25),RHXX(25),BV(25,17),BVZ(25,17)
      COMMON XX(600)
      DIMENSION R(25),TX(25),TY(25)
      DOUBLE COMPLEX BX(425),XL(425),DCMPLX,XPT,XPX,XPIM,GAM,B4
      DOUBLE PRECISION XINT
      DOUBLE PRECISION CDABS,DREAL,DIMAG,DATAN2,DSQRT,A,B
C
C                SUBROUTINE TO CALCULATE U FROM P
C
      RHO = 0.5015
      FF = 180.0/3.14159
      XINT = 0.
      FW = F*F - DD2*W*W
      FL = F*RL
      WRITE(6,903)
      DO 100 N = 1,NN
      DXX = DX
      X = DX*FLOAT(N-1)
      DZ = DT*RH(N)
      IF (N.EQ.1) GO TO 86
```

```
      IF (N.EQ.NN) GO TO 110
      DD = RHX(N)/RH(N)
      DO 85 M = 1,MM
      I = (N-1)*MM +M
      IP = I + MM
      IM = I -MM
      T = -1. + DT*FLOAT(M-1)
      IF (M.EQ.1) GO TO 10
      IF (M.EQ.MM) GO TO 15
      GO TO 20
10    I1 = I +1
      XPT = (BX(I1) - BX(I))/DT
      GO TO 25
15    XPT = (0.0,0.0)
      GO TO 25
20    I2 = I+1
      I1 = I-1
      XPT = (BX(I2) - BX(I1))/(2.*DT)
25    XPX = (BX(IP) - BX(IM))/(2.*DX)
      XPX = XPX -T*DD*XPT
85    XL(M) = -(0.0,1.0)*(W*XPX + FL*BX(I))/FW
      GO TO 90
86    CH = R(N)*(F*F + DD2*W*W)/FW
      CHH = W*RH(N)
      GAM = DCMPLX(CH,CHH)
      CH = F*TY(N)
      CHH = W*TX(N)*DD2
      XPX = DCMPLX(CH,CHH)
      XPX = XPX/GAM
      XPT = XPX
      CH = R(N)*DD2*2.0*W*F*RL/FW
      CHH = RL*F*RH(N)
      B4 = DCMPLX(CH,CHH)
      DO 87 M = 1,MM
      XPX = XPT - B4*BX(M)/GAM
87    XL(M) = -(0.0,1.0)*(FL*BX(M) + W*XPX)/FW
      DXX = DX/2.0
      GO TO 90
110   FLW = 0.5*FL/W
      DXX = DX/2.
      DO 120 M = 1,MM
      I = (NN-1)*MM +M
      IM = I - MM
      XPIM = 2.*BX(I)/DX +BX(IM)*(FLW -1./DX)
      XPIM = XPIM/(FLW + 1./DX)
120   XL(M) = -(0.0,1.0)*(FL*BX(I) +W*0.5*(XPIM-BX(IM))/DX)/FW
90    DO 95 M = 1,MM
      DZZ = DZ
      IF (M.EQ.1) GO TO 92
      IF (M.NE.MM) GO TO 95
92    DZZ = DZ/2.
95    XINT = XINT + RHO*CDABS(XL(M))*CDABS(XL(M))*DXX*DZZ
      DO 98 M = 1,MM
      A = DREAL(XL(M))
      B = DIMAG(XL(M))
      AMT = CDABS(XL(M))
      THT = FF*DATAN2(B,A)
98    XL(M) = DCMPLX(AMT,THT)
      WRITE(6,901) X,RH(N),DZ
      WRITE(6,902) (XL(M),M=1,MM)
```

```
  100   CONTINUE
        WRITE(6,904) XINT
  901   FORMAT(/'  X,D,DZ =',3E15.5)
  902   FORMAT(2X,E12.5,F9.2,E12.5,F9.2,E12.5,F9.2,E12.5,F9.2,E12.5,F9.2(
  903   FORMAT(///'   U'/)
  904   FORMAT(//'  U CONTRIBUTION TO KE =',E15.5/)
        RETURN
        END

        SUBROUTINE BANDG(A,BB)
        COMMON F,DX,DT,NN,MM,NM,NMX
        COMMON RH(25),RHX(25),RHXX(25),BV(25,17),BVZ(25,17),XL(600)
        DOUBLE COMPLEX A(425,53),BB(425)
        DOUBLE COMPLEX R
  C
  C                  SUBROUTINE TO DO BAND GAUSSIAN ELIMINATION
  C
         MBW = NN + 1
        NH = NM -1
        MDIAG = MBW + 1
        DO 50 N = 1,NH
        IP = N + 1
        MH = N + MBW
        IF (MH.LE.NM) GO TO 5
        MH = NM
  5     DO 50 IR = IP,MH
        ICD = MBW + N + 1 -IR
        R = A(IR,ICD)/A(N,MDIAG)
        BB(IR) = BB(IR) -R*BB(N)
        DO 50 IC = IP,MH
        ICD = MBW + IC + 1 - IR
        ICB = MBW + IC + 1 -N
        A(IR,ICD) = A(IR,ICD) - R*A(N,ICB)
  50    CONTINUE
        DO 100 I = 1,NH
        N = NM - I + 1
        BB(N) = BB(N)/A(N,MDIAG)
        IL = N - MBW
        IH= N - 1
        IF (IL.GE.1) GO TO 60
        IL = 1
  60    DO 100 IR = IL,IH
        ICD = MBW + N + 1 - IR
        BB(IR) = BB(IR) -A(IR,ICD)*BB(N)
  100   CONTINUE
        BB(1) = BB(1)/A(1,MDIAG)
        RETURN
        END

        SUBROUTINE FRIC(R,RX)
        COMMON F,DX,DT,NN,MM,NM,NMX
        COMMON RH(25),RHX(25),RHXX(25),BV(25,17),BVZ(25,17),XL(600)
        DIMENSION R(25),RX(25),A(25)
  C
  C                  SUBROUTINE TO READ AND INTERPOLATE BOTTOM RESISTANCE
  C                           COEFFICIENT R
  C
  C                           RX = X DERIVATIVE OF R
  C
        READ(5,*) NF
```

```
      DO 5 I = 1,NF
      READ(5,*) A(I),XL(I)
    5 A(I) = A(I)*1.0E+05
      RMA = XL(NF)
      R(1) = XL(1)
      DO 20 N = 2,NN
      X = DX*FLOAT(N-1)
      IF (X.GT.A(NF)) GO TO 15
      IC = 0
      DO 8 J = 2,NF
      IF (IC.NE.0) GO TO 8
      I = J
      IF (X.GT.A(I)) GO TO 8
      IC = I
    8 CONTINUE
      IM = I - 1
      RQ = (XL(I) - XL(IM))/(A(I) -A(IM))
      XX = X - A(IM)
      R(N) = XL(IM) + XX*RQ
      GO TO 20
   15 R(N) = RMA
   20 CONTINUE
      RX(1) = (R(2) - R(1))/DX
      D2 = 2.0*DX
      NX = NN - 1
      DO 30 N = 2,NX
      IP = N + 1
      IM = N - 1
   30 RX(N) = (R(IP) - R(IM))/D2
      RX(NN) = 0.
      WRITE(6,901)
      WRITE(6,902) (R(I),I = 1,NN)
      WRITE(6,903)
  901 FORMAT(//'  R(CM/SEC)'/)
  902 FORMAT(10E12.5)
  903 FORMAT(///)
      RETURN
      END

      SUBROUTINE FFCCAL(TX,TY,TXX,TYX,FFC,W,RL,DD2)
      COMMON F,DX,DT,NN,MM,NM,NMX
      COMMON RH(25),RHX(25),RHXX(25),BV(25,17),BVZ(25,17),XL(600)
      DIMENSION TX(25),TY(25),TXX(25),TYX(25)
      DOUBLE COMPLEX FFC(25),DCMPLX
C
C             SUBROUTINE TO CALCULATE WIND FORCING TERMS
C
      FW = W*(F*F - DD2*W*W)
      FW = 1.0/FW
      M = MM
      DO 20 N = 1,NN
      CC = BV(N,M)*FW
      CR = DD2*(-W*TXX(N) + RL*F*TX(N))
      CI = F*TYX(N) -DD2*RL*W*TY(N)
      FFC(N) = DCMPLX(CR,CI)
   20 FFC(N) = CC*FFC(N)
      RETURN
      END

      SUBROUTINE PEC(BX,SPE,IRL)
```

```
      COMMON F,DX,DT,NN,MM,NM,NMX
      COMMON RH(25),RHX(25),RHXX(25),BV(25,17),BVZ(25,17),XL(600)
      DOUBLE COMPLEX BX(425)
      DOUBLE PRECISION XINT,CDABS
C
C               SUBROUTINE TO CALCULATE THE POTENTIAL ENERGY ASSOCIATED
C                       WITH FREE SURFACE ELEVATION
C
      IF (IRL.EQ.0) GO TO 50
      GG = 1.0/(2.0*980.*1.03)
      XINT = 0.
      DO 20 N = 1,NN
      I = N*MM
      XX = DX
      IF (N.EQ.1) GO TO 5
      IF (N.EQ.NN) GO TO 5
      GO TO 20
5     XX = DX/2.
20    XINT = XINT + CDABS(BX(I))*CDABS(BX(I))*XX
      SPE = GG*XINT
      GO TO 60
50    SPE = 0.
60    WRITE(6,901) SPE
901   FORMAT(/' FREE SURFACE POTENTIAL ENERGY =',E15.5)
      RETURN
      END

      SUBROUTINE WRD(TX,TY,TXX,TYX,IXY)
      COMMON F,DX,DT,NN,MM,NM,NMX
      COMMON RH(25),RHX(25),RHXX(25),BV(25,17),BVZ(25,17),XL(600)
      DIMENSION TX(25),TY(25),TXX(25),TYX(25)
C
C               SUBROUTINE TO READ AND INTERPOLATE WIND STRESS PROFILES
C
C                       TX = X WIND STRESS
C                       TXX = X DERIVATIVE OF TX
C                       TY = Y WIND STRESS
C                       TYX = X DERIVATIVE OF TY
C
      READ(5,*) NRD
      IF (NRD.EQ.0) GO TO 100
      DO 5 I = 1,NRD
      READ(5,*) TXX(I),XL(I)
5     TXX(I) = TXX(I)*1.0E+05
      RMA = XL(NRD)
      TX(1) = XL(1)
      DO 20 N = 2,NN
      X = DX*FLOAT(N-1)
      IF (X.GT.TXX(NRD)) GO TO 15
      IC = 0
      DO 8 J = 2,NRD
      IF (IC.NE.0) GO TO 8
      I = J
      IF (X.GT.TXX(I)) GO TO 8
      IC = I
8     CONTINUE
      IM = I -1
      RQ = (XL(I) -XL(IM))/(TXX(I) -TXX(IM))
      XX = X - TXX(IM)
      TX(N) = XL(IM) + XX*RQ
```

```
      GO TO 20
15    TX(N) = RMA
20    CONTINUE
      GO TO 110
100   DO 105 I = 1,NN
105   TX(I) = 1.0
110   TXX(1) = (TX(2) - TX(1))/DX
      D2 = 2.0*DX
      NX = NN - 1
      DO 30 N = 2,NX
      IP = N + 1
      IM = N - 1
30    TXX(N) = (TX(IP) - TX(IM))/D2
      TXX(NN) = 0.
      IF (IXY.EQ.0) GO TO 140
      WRITE(6,901)
      GO TO 150
140   WRITE(6,902)
      DO 145 I = 1,NN
      TY(I) = TX(I)
      TYX(I) = TXX(I)
      TX(I) = 0.0
145   TXX(I) = 0.
      WRITE(6,903) (TY(I),I = 1,NN)
      GO TO 160
150   DO 155 I = 1,NN
      TY(I) = 0.
155   TYX(I) = 0.
      WRITE(6,903) (TX(I),I = 1,NN)
901   FORMAT(' TAUX (DYNE/CM2)')
902   FORMAT(' TAUY (DYNE/CM2)')
903   FORMAT(10E12.5)
160   RETURN
      END

      SUBROUTINE AACAL(W,RL,R,RX,N,M,DD2,AA1,AA2,AA3,AA4,AA5)
      COMMON F,DX,DT,NN,MM,NM,NMX
      COMMON RH(25),RHX(25),RHXX(25),BV(25,17),BVZ(25,17),XL(600)
      DIMENSION R(25),RX(25)
      DOUBLE COMPLEX AA1,AA2,AA3,AA4,AA5,DCMPLX
C
C                SUBROUTINE TO CALCULATE COEFFICIENTS FOR MATS
C
      CALL AS(A1,A2,A3,W,N,M,C1,C2,C3,DD2)
      CB = F*F + DD2*W*W
      CC = F*F - DD2*W*W
      CA = BV(N,M)/(CC**2)
      CH = (-RX(N)*C3-R(N)*C3*C3)*CB-R(N)*CC*CB*BVZ(N,M)*C1/(BV(N,M)**2)
      CH = -CA*(CH + 2.0*DD2*W*RL*F*R(N)*C3)
      CHH = W*(C1 + C2*C3)
      AA1 = DCMPLX(CH,CHH)
      CH = CA*2.0*DD2*W*RL*F*RX(N)
      CHH = C2*F*RL
      AA2 = DCMPLX(CH,CHH)
      CH = CA*CB*RX(N)
      CHH = W*C2
      AA3 = DCMPLX(CH,CHH)
      CH = CB*R(N)*C3*C3 + R(N)*CB*CC*C1*C1/BV(N,M)
      CH = -CA*CH
      CHH = 0.
```

```
      AA4 = DCMPLX(CH,CHH)
      CH = -CA*CB*R(N)*C3
      AA5 = DCMPLX(CH,CHH)
      RETURN
      END
```

50272-101

REPORT DOCUMENTATION PAGE	1. REPORT NO. WHOI-85-17	2.	3. Recipient's Accession No.

4. Title and Subtitle	5. Report Date
Programs for Computing Properties of Coastal-Trapped Waves and Wind-Driven Motions Over the Continental Shelf and Slope	April 1985
	6.
7. Author(s) Kenneth H. Brink, David C. Chapman	8. Performing Organization Rept. No. WHOI-85-17
9. Performing Organization Name and Address Woods Hole Oceanographic Institution Woods Hole, Massachusetts 02543	10. Project/Task/Work Unit No.
	11. Contract(C) or Grant(G) No. (C) (G) OCE 84-08563
12. Sponsoring Organization Name and Address National Science Foundation	13. Type of Report & Period Covered Technical
	14.

15. Supplementary Notes

This report should be cited as: Woods Hole Oceanog. Inst. Tech. Rept. WHOI-85-17.

16. Abstract (Limit: 200 words)

Documentation and listings are presented for a sequence of computer programs to be used for problems in continental shelf dynamics. Three of the programs are to be used for computing properties of free and forced coastal-trapped waves. A final program may be used to compute wind-driven fluctuations over the continental shelf and slope.

17. Document Analysis a. Descriptors

1. coastal-trapped waves
2. wind-driven shelf currents
3. continental shelf

b. Identifiers/Open-Ended Terms

c. COSATI Field/Group

18. Availability Statement	19. Security Class (This Report)	21. No. of Pages
Approved for publication; distribution unlimited.	UNCLASSIFIED	98
	20. Security Class (This Page)	22. Price

(See ANSI-Z39.18) See Instructions on Reverse

OPTIONAL FORM 272 (4-77)
(Formerly NTIS-35)
Department of Commerce

DOCUMENT LIBRARY

April 9, 1985

DISTRIBUTION LIST FOR TECHNICAL REPORT EXCHANGE

Institute of Marine Sciences Library
University of Alaska
O'Neill Building
905 Koyukuk Ave., North
Fairbanks, AK

Attn: Stella Sanchez-Wade
Documents Section
Scripps Institution of Oceanography
Library, Mail Code C-075C
La Jolla, CA 92093

Hancock Library of Biology & Oceanography
Alan Hancock Laboratory
University of Southern California
University Park
Los Angeles, CA 90089-0371

Gifts & Exchanges
Library
Bedford Institute of Oceanography
P.O. Box 1006
Dartmouth, NS, B2Y 4A2, CANADA

Office of the International
Ice Patrol
c/o Coast Guard R & D Center
Avery Point
Groton, CT 06340

Library
Physical Oceanographic Laboratory
Nova University
8000 N. Ocean Drive
Dania, FL 33304

NOAA/EDIS Miami Library Center
4301 Rickenbacker Causeway
Miami, FL 33149

Library
Skidaway Institute of Oceanography
P.O. Box 13687
Savannah, GA 31416

Institute of Geophysics
University of Hawaii
Library Room 252
2525 Correa Road
Honolulu, HI 96822

Library
Chesapeake Bay Institute
4800 Atwell Road
Shady Side, MD 20876

MIT Libraries
Serial Journal Room 14E-210
Cambridge, MA 02139

Director, Ralph M. Parsons Laboratory
Room 48-311
MIT
Cambridge, MA 02139

Marine Resources Information Center
Bldg. E38-320
MIT
Cambridge, MA 02139

Library
Lamont-Doherty Geological Observatory
Colombia University
Palisades, NY 10964

Library
Serials Department
Oregon State University
Corvallis, OR 97331

Pell Marine Science Library
University of Rhode Island
Narragansett Bay Campus
Narragansett, RI 02882

Working Collection
Texas A&M University
Dept. of Oceanography
College Station, TX 77843

Library
Virginia Institute of Marine Science
Gloucester Point, VA 23062

Fisheries-Oceanography Library
151 Oceanography Teaching Bldg.
University of Washington
Seattle, WA 98195

Library
R.S.M.A.S.
University of Miami
4600 Rickenbacker Causeway
Miami, FL 33149