

THE NUMERICAL SOLUTION OF STEADY WATER WAVE PROBLEMS

J.D. FENTON

Department of Civil Engineering, University of Auckland, Private Bag, Auckland, New Zealand

(Received 24 April 1984; accepted 6 April 1985; received for publication in revised form 8 February 1988)

Abstract—A simplified method is given for the numerical solution of the problem of steadily progressing waves, so that overall parameters and local fluid velocities and pressures may be obtained accurately for geophysical calculations and for engineering design. The method reduces to the solution of a set of nonlinear equations by Newton's method, where all the necessary derivatives are evaluated numerically, making the coding rather simpler. A FORTRAN program is presented which can solve problems in either deep water or water of finite depth, where either the wave length or period is specified, and where either the mean current or the mean mass transport velocity is specified.

Key Words: Coastal engineering, Currents, Ocean engineering, Oceans, Offshore structures, Sediment transport, Steady waves, Water waves.

INTRODUCTION

In many geophysical and engineering problems it is necessary to solve the problem of waves which propagate steadily without change in water of constant mean depth, to obtain the fluid velocities and pressures, for example in sediment transport studies, or in the calculation of forces on relatively transparent marine structures. This problem may be a close approximation to the actual physical problem of interest, such as in the propagation of dominant swell waves, or it could be a first-order approximation, such as the shoaling of waves, where the approximation is made that as the waves enter the shallower water, at each depth they act as if the water everywhere had that depth, but throughout the shoaling process the initial wave period, mass flux, and energy flux are conserved.

The two main explicit theories based on a system of rational approximation are Stokes theory and cnoidal theory. The former makes the assumption that all variation in the horizontal can be represented by Fourier series, the coefficients being expressed as perturbation expansions in terms of a parameter which increases with wave height. These have been calculated analytically as far as fifth order, most recently by Fenton (1985), who obtained a solution in terms of the wave height itself. If the wave period is one of the given parameters of a steady wave problem, it is necessary to know the current on which the waves are riding, following Rienecker and Fenton (1981). Good agreement with experiment was observed for waves shorter than ten times the water depth. For waves in shallow water, Stokes theory is well known to break down, and it is more appropriate to use cnoidal theory, for which a solution to fifth order in terms of the wave height to water depth ratio has been given by Fenton (1979). Although accurate for long

waves, it was determined to diverge from experiment significantly for waves higher than one-half of the water depth. The theory, as with almost all presentations of steady wave theory, was for the special situation where the time mean current at any point is zero; in a later paper (Fenton, 1983) cnoidal theory to second order was presented for the more general situation where the current has a specified value. Also presented were some recent formulae for the elliptic functions involved in cnoidal theory, making their calculation rather easier.

A limitation to the use of both Stokes and cnoidal theories is that they are not accurate for all waves, Stokes theory breaks down in shallow water, cnoidal theory in deep water, and both for high waves. Instead of using perturbation expansions for the coefficients of the Fourier series as in Stokes theory, it is possible to determine the values of the Fourier coefficients numerically, by solving a system of nonlinear equations. This is the essential feature of the methods of Chappellear (1961), Dean (1965), Chaplin (1980), and Rienecker and Fenton (1981). Each of these assumed a Fourier expansion which automatically satisfied the field equation throughout the fluid and the boundary condition on the bottom. Chappellear used the velocity potential for the field variable and introduced a Fourier series for the surface elevation. By using instead the stream function for the field variable, and point values of the surface elevations, Dean obtained a rather simpler set of equations. In each of these approaches the solution of the equations proceeded by a method of successive corrections to an initial estimate such that the least-squares errors in the surface boundary conditions were minimized. Chaplin improved on these methods by using Schmidt orthogonalization, and got better agreement with numerical results from high-order Stokes expansions.

A method which is simpler to apply was given by

Rienecker and Fenton. In their approach, the non-linear equations were solved by Newton's method, the only numerical approximation made was the truncation of the Fourier series, and the method could be applied to waves in deeper water. It was determined to give accurate results when compared with experiment and high-order theory, and for practical applications, allowance was made for the specification of a mean mass flux or a mean current in the water.

In this paper, the method of Rienecker and Fenton is modified, and a computer program presented, so that it can be applied to rather more general situations, including waves on both deep water and water of finite depth, where either wave length or period can be specified, in addition to the previously mentioned selection of mean mass-flux velocity or a mean current velocity. Although each of these alternatives can be treated and trivially included by modifying the equations of Rienecker and Fenton, the modifications to any computer program written from their paper are somewhat complicated, because a change in any of the equations requires a number of changes in the calculations of the Jacobian matrix which lies at the heart of the method. In this paper, the equations are rewritten in terms of the wavenumber, and simple numerical differentiation used to obtain the matrix elements. This makes the coding simpler and shorter and gives greater flexibility to the program. The resulting FORTRAN program is presented, with instructions as to its operation. Operation of the program was simple and the method robust, through a wide range of wave parameters.

THE STEADY WAVE EQUATIONS

The problem to be solved is that of periodic waves of length λ and crest-to-trough wave height H propagating without change over a layer of water of mean depth d . A coordinate system (x, y) is considered, located under a crest at the mean water level and propagating with the wave; in this frame all motion is steady. Throughout this work the wavenumber $k = 2\pi/\lambda$ will be used to make the variables and equations dimensionless, together with gravitational acceleration g and fluid density ρ . It is assumed that the wave height H and water depth d are known, the latter possibly being infinite.

In some situations the wavelength λ may be specified, in which case the problem is determined uniquely. Otherwise it may be the apparent wave period τ in a frame through which the waves propagate which is known. In this situation it is necessary to know the current on which the waves are travelling, to quantify precisely the amount by which the wave period is Doppler shifted. In the program given in the Appendix, allowance is made for the specification of the Eulerian time-mean velocity c_E which the irrotational theory on which the method is based requires to be constant throughout the fluid, or the specification of

the vertically integrated mean transport (Stokes) velocity c_S .

The quantities specified in the data are:

(i) The ratio of wave height to water depth, H/d .
 (ii) Either the ratio of wave height to wavelength H/λ , or the dimensionless parameter expressing the ratio of wave height to period squared, $H/g\tau^2$.

(iii) In a frame through which the waves move, the magnitude of the current, expressed nondimensionally as either c_E/gH or $c_S/(gH)^{1/2}$. If calculations are required only in the frame moving with the wave, a notional value of either need only be given.

The system of equations is presented here, each being written in the form $f_i(\mathbf{z}) = 0$, where i is the reference number of the equation and \mathbf{z} is the vector of the variables of the problem $\mathbf{z} = (z_1, z_2, \dots)$. The variables and their reference numbers are shown in Table 1. The first equation relates the dimensionless water depth kd and wave height kH through the specified value of wave height to depth:

$$f_1(z_1, z_2) = kH - (H/d)kd = 0.$$

In the program, if the depth is infinite effectively, this is replaced by the dummy equation

$$f_1(z_1) = kd + 1 = 0,$$

and the depth set to the notional value $kd = z_1 = -1$. This is a meaningless equation and variable, but the inclusion of such a dummy equation where none is valid makes the coding simpler, considered to be the major goal here.

Either of two alternatives is used for the second equation, relating the dimensionless wave height kH to the value of H/λ if that is specified.

$$f_2 = kH - 2\pi(H/\lambda) = 0,$$

or to the value of $H/g\tau^2$ if that is given:

$$f_2 = kH - (H/g\tau^2)(\tau(gk)^{1/2})^2 = 0.$$

The next equation relates the wave speed c , relative to a frame in which the period is τ , to wavelength and period by the definition $c = \lambda/\tau$, which in dimensionless form becomes

$$f_3 = c(k/g)^{1/2}\tau(gk)^{1/2} - 2\pi = 0.$$

In the next two equations, fluid velocities in the steady frame of calculation are related to those in the physical frame of interest, through which the waves travel at speed c in the positive x direction. In the frame of the waves, the apparent flow is in the negative x direction, under the stationary waves. If the mean fluid speed in this frame is \bar{u} , then the mean fluid velocity is $-\bar{u}$. The time mean current in the physical frame is c_E , and it is simple to show that

$$f_4 = c_E(k/g)^{1/2} + \bar{u}(k/g)^{1/2} - c(k/g)^{1/2} = 0.$$

The mean mass-transport velocity c_S in the physical frame is given by $c_S d = cd - Q$, where Q is the

Table 1. Dimensionless variables of steady wave problem and their initial values as given by linear theory

Variable reference number j	Dimensionless variable z_j	Initial value from linear theory
1	kd	See text
2	kH	$kd \times (H/d)$
3	$\tau(gk)^{1/2}$	$2\pi/(\tanh kd)^{1/2}$
4	$c(k/g)^{1/2}$	$(\tanh kd)^{1/2}$
5	$c_E(k/g)^{1/2}$	0 or $(kH)^{1/2} \times c_E/(gH)^{1/2}$
6	$c_S(k/g)^{1/2}$	$(kH)^{1/2} \times c_S/(gH)^{1/2}$ or 0
7	$\bar{u}(k/g)^{1/2}$	$(\tanh kd)^{1/2}$
8	$q(k^3/g)^{1/2}$	0
9	rk/g	$0.5 \tanh kd$
10	$k\eta_0$	$0.5 kH$
11	$k\eta_1$	$0.5 kH \cos(1 \times \frac{\pi}{N})$
12	$k\eta_2$	$0.5 kH \cos(2 \times \frac{\pi}{N})$
...
$N+10$	$k\eta_N$	$-0.5 kH$
$N+11$	B_1	$0.5 kH (\tanh kd)^{-1/2}$
$N+12$	B_2	0
...
$2N+10$	B_N	0

volume flux per unit span in the computational frame. It is convenient to introduce q , the volume flux due to the waves, given by $q = \bar{u}d - Q$, to give

$$f_5 = c_S(k/g)^{1/2} + \bar{u}(k/g)^{1/2} - c(k/g)^{1/2} - \frac{q(k^3/g)^{1/2}}{kd} = 0.$$

For infinite depth, the program does not include the last term.

Equation six incorporates the specified value of either $c_E/(gH)^{1/2}$, presumed obtained from current meter measurements or $c_S/(gH)^{1/2}$ if that is known (such as in a closed wave tank, when it is zero). That is,

$$f_6 = c_x(k/g)^{1/2} - \frac{c_x}{(gH)^{1/2}} (kH)^{1/2} = 0,$$

where the c_x is either c_E or c_S .

The nonlinear surface boundary conditions are to be satisfied at each of $N + 1$ points on the surface, equispaced horizontally between the crest and the trough. The elevation $\eta(x)$ of the surface above the mean at these points is denoted by $\eta_m = \eta(x_m)$, $m = 0(1)N$, η_0 the elevation at the crest, η_N that at the trough. The mean value of $\eta(x)$ is zero, and in terms of the point values this is satisfied if

$$f_7 = k\eta_0 + k\eta_N + 2 \sum_{m=1}^{N-1} k\eta_m = 0,$$

this trapezoidal type of sum being obtained from an N -term Fourier interpolation of the point values and subsequent integration. The error of this approximation is much smaller than that of the trapezoidal rule for functions which are not periodic, and is of the same accuracy as that due to the truncation of the Fourier series for ψ after N terms, introduced next. The requirement that the crest and trough elevations differ by H is given by

$$f_8 = k\eta_0 - k\eta_N - kH = 0.$$

All of the preceding equations have been simple geometric or kinematic ones. Now the dynamics and kinematics of the flow field itself are introduced. The fluid is assumed to be incompressible, such that a stream function $\psi(x, y)$ exists, the fluid velocity $(u, v) = (\partial\psi/\partial y, -\partial\psi/\partial x)$, and the motion is assumed to be irrotational such that $\partial u/\partial y - \partial v/\partial x = 0$. A series for ψ which corresponds to motion being periodic in x , on a flow of mean velocity $-\bar{u}$ and which satisfies these equations is

$$\psi(x, y) = -\bar{u}(d + y)$$

$$+ \left(\frac{g}{k^3}\right)^{1/2} \sum_{j=1}^N B_j \frac{\sinh jk(d+y)}{\cosh jkd} \cos jkx,$$

where the B_j are dimensionless constants. This expression also satisfies the boundary condition on the bottom, that $\psi = 0$ when $y = -d$. The free surface is a streamline on which $\psi = -Q = q - \bar{u}d$ is constant. This kinematic free-surface boundary condition is to be satisfied at each of the $N + 1$ points, $kx_m = m\pi/N$:

$$f_{m+9} = -q(k^2/g)^{1/2} - k\eta_m \bar{u}(k/g)^{1/2} + \sum_{j=1}^N B_j \left[\frac{\sinh j(kd + k\eta_m)}{\cosh jkd} \right] \cos \frac{j m \pi}{N} = 0$$

for $m = 0(1)N$. For the case of infinitely deep water, the term in square brackets is replaced by $[\exp(jk\eta_m)]$.

The remaining boundary condition is that the pressure is constant on the surface. It follows from Bernoulli's equation that

$$\frac{1}{2} \left[\left(\frac{\partial \psi}{\partial x}(x, \eta(x)) \right)^2 + \left(\frac{\partial \psi}{\partial y}(x, \eta(x)) \right)^2 \right] + g(\eta(x) + d) = R,$$

where R is a constant. Introducing $r = R - gd$ so that deep water can be treated, and substituting the series for ψ gives

$$f_{N+10+m} = \frac{1}{2} \left(-\bar{u}(k/g)^{1/2} + \sum_{j=1}^N j B_j \left[\frac{\cosh j(kd + k\eta_m)}{\cosh jkd} \right] \cos \frac{j m \pi}{N} \right)^2 \times \frac{1}{2} \left(\sum_{j=1}^N j B_j \left[\frac{\sinh j(kd + k\eta_m)}{\cosh jkd} \right] \times \sin \frac{j m \pi}{N} \right)^2 + k\eta_m - \frac{rk}{g} = 0,$$

for $m = 0(1)N$. For deep water, each term in square brackets is replaced by $[\exp(jk\eta_m)]$. This completes the set of $2N + 10$ nonlinear equations in $2N + 10$ variables.

SOLUTION OF THE EQUATIONS

The system of equations can be written

$$\mathbf{f}(\mathbf{z}) = \{f_i(\mathbf{z}), i = 1(1)2N + 10\} = \mathbf{0},$$

and this can be solved iteratively by Newton's method. If an approximate solution after n iterations is $\mathbf{z}^{(n)}$, to give a better approximation $\mathbf{z}^{(n+1)}$ the method requires solution of the linear matrix equation

$$\left[\frac{\partial f_i}{\partial z_j} \right]^{(n)} (\mathbf{z}^{(n+1)} - \mathbf{z}^{(n)}) = -\mathbf{f}(\mathbf{z}^{(n)})$$

at each iteration, until the sequence of solution vectors has converged. This was the method adopted by

Rienecker and Fenton (1981), and was determined to converge quickly to a solution.

It is possible to obtain the Jacobian matrix by differentiating each of the equations with respect to each of the variables, as done by Rienecker and Fenton. The coding is simpler, however, if the derivatives are obtained numerically. That is, if variable z_j is changed by an amount δ_j , then on numerical evaluation of equation i before and after the increment the result is obtained

$$\frac{\partial f_i}{\partial z_j} \approx \frac{f_i(z_1, \dots, z_j + \Delta_j, \dots, z_{2N+10}) - f_i(z_1, \dots, z_j, \dots, z_{2N+10})}{\Delta_j}$$

and the whole Jacobian matrix can be determined by evaluation of each of the $2N + 10$ equations a total of $2N + 10$ times. The coding for this is simple. In the program, $\Delta_j = z_j/100$ if $z_j > 10^{-4}$, otherwise $\Delta_j = 10^{-5}$.

INITIAL LINEAR SOLUTION

To commence the iteration solution an initial estimate $\mathbf{z}^{(0)}$ must be known. This can be obtained from linear wave theory, for example, taking the first-order terms of the Stokes solution presented by Fenton (1983, 1985). The solution is in terms of kd and kH , and as a first step kd must be determined. If the wavelength is specified, then kd is simply $2\pi d/\lambda$. In many practical problems however, the wave period rather than the wavelength is known initially, and the first step is to solve the transcendental equation from linear theory:

$$\frac{2\pi}{\tau(gk)^{1/2}} - c_E(k/g)^{1/2} = (\tanh kd)^{1/2}.$$

As the value of c_E usually is much less than the value of wave speed, it is reasonable to leave that term out, giving the familiar linear dispersion relation. There have been several papers written on the numerical solution of this equation, which is a relatively straightforward operation. However, there is a simple explicit approximation given by Eckart (1952) which seems to be little known. The approximation is, if $\alpha = 4\pi^2 d/g\tau^2$, then

$$kd \approx \alpha(\coth \alpha)^{1/2}.$$

This approximation is accurate to within 5% for all values of kd , and is exact in the limits of deep and shallow water. A simple refinement of this approximation can be had by using one step of a Newton iteration with Eckart's solution as first estimate, to give

$$kd \approx \frac{\alpha + \beta^2 \operatorname{sech}^2 \beta}{\tanh \beta + \beta \operatorname{sech}^2 \beta},$$

where $\beta = \alpha(\coth \alpha)^{1/2}$. This result also is exact in the limit of both deep and shallow water. The greatest error over all wavelengths is 0.05%. It would seem to be a convenient approximation deserving of wide-

Table 2. Examples of data provided

Entry 6 from Table A0 of Cokelet (1977)	Figure 9 of Le Méhauté, Divoky, and Lin (1968)
'deep' 0. 'wavelength' 0.09762055 'euler' 0. 10 1	'finite' 0.548 'period' 7.369d-4 'stokes' 0. 10 4

spread use in applications of linear theory. Once an estimate of kd is obtained, the value of kH follows, and initial values for all the variables can be calculated using linear theory as set out in Table 1.

In developing the program it was determined that for high waves and long waves, when linear theory does not provide a good initial solution, the program did not converge to a solution. This was overcome easily by solving the problem in steps of wave height. If M steps are used, the initial solution is for a wave of zero wave height, using the solution given in Table 1. For a wave height H/M , the linear solution is used as an initial solution for an iterative solution. From this accurate solution for a wave of height H/M , and the linear solution for a wave of height 0, linear extrapolation can be used to give an initial estimate for a wave of height $2H/M$, then the iterative procedure can be followed. For subsequent heights the initial solution for each step then is the linear extrapolation from the two previous steps. For waves in deep water it was determined that this procedure was unnecessary, while for the most demanding situation examined at the program development stage, a high and long wave of height 55% of the depth and a length 31 times the depth, 4 steps were necessary. For each step in wave height, five iterations were enough for six-figure accuracy. Convergence almost everywhere was rapid, but for the highest waves became rather slower, reflecting the fact that the solution is close to a domain where no solutions exist.

DESCRIPTION AND OPERATION OF PROGRAM

The program was written in FORTRAN, and developed on a VAX 11/750 computer in the School of Mathematics at the University of New South Wales. Calculations on that machine were accurate to some six decimal places in single precision. The program was developed in double precision, and is presented in the Appendix. To convert to single precision it is necessary only to change the "implicit double precision . . ." at the top of each subroutine to "implicit real . . .", and to change the names of the LINPACK routines used from "dgefa" and "dgesl" to "sgefa" and "sgesl" respectively.

Each dimensioned variable in the program is shown as having a length of 41, in excess of that

required for some variables, this number being selected as being prime and uncommon, so that a global edit command could change simply to a different value. In fact, the length required is $2N + 10$, so that as it is printed, a value of $N = 15$ could be used, certainly enough for most environmental and engineering problems. In operation on the VAX computer, total storage of about 55K was required. To solve a typical problem took 5.2 sec in single and 8.9 sec in double precision.

Data input

The data are to be provided in free format, and examples of data are given in Table 2. In the first line the character variable 'deep' or 'finite' is given, specifying the depth, followed by the numerical value of H/d , which is meaningless for the former. Line two contains either 'wavelength' and the numerical value of H/λ , or 'period' followed by the value of $H/g\tau^2$. Next, 'euler' or 'stokes' is given, followed respectively by the value of $c_E/(gH)^{1/2}$ or $c_S/(gH)^{1/2}$. In the last line two integers are given, the first is the value of N , the number of Fourier coefficients (and the number of intervals into which one-half the wave is divided for computations), the second is M , the number of steps in height. Table 2 contains two examples of data. The first is for a wave in deep water, with a height to wavelength ratio of 0.09762055, the 6th entry in Table A0 of Cokelet (1977), and having a height 69% of the maximum in deep water. Results obtained from this data will be presented subsequently. The second set of data is for the wave in figure 9 of Le Méhauté, Divoky, and Lin (1968), a high and long wave with $H/D = 0.548$, $H/g\tau^2 = 0.0007369$, and as the tank was closed $c_S = 0$. The solution showed that the wave was 31 times as long as the depth. In this situation, four height steps were determined necessary for convergence. As a rough guide, for waves in deeper water it was determined that 5 Fourier coefficients and 1 height step usually were sufficient for an accurate solution. For longer waves 10 and more coefficients were needed, and up to 4 steps in height. In any application it is simple to examine the adequacy of the value of N from the magnitudes of the B_j printed out, the last coefficient B_N having to be sufficiently small that truncation there is sufficiently accurate. In the program, variable "crit" is given a value of 0.001. If

at any iteration, the sum of the magnitudes of the corrections $\sum_j |z_j^{(n+1)} - z_j^{(n)}|$ is less than this number, the iteration process stops. If the height step is the last (or only) one, a value of 10^{-6} is used instead. After nine iterations, if the sum of the magnitudes of the corrections is larger than this number, the program is halted with a message. Selecting a larger value of M , the number of steps in height, should make the process converge more quickly.

Overall, experience gained with the program gave the impression of it being robust. If the solution process did not converge at any stage, simply using a larger number of steps in height was enough to ensure convergence, except in the vicinity of the highest waves, when the solution oscillated without converging completely, although it never diverged there, and was always at least of engineering accuracy.

For waves in deeper water (kd large) it was determined that evaluation of the hyperbolic functions $\cosh jkd$ led to computer overflow errors. For $N = 10$ this occurred for waves shorter than $\frac{2}{3}$ of the depth. As in this situation the velocity on the bottom is some 10^{-4} times that at the surface, it is simple and reasonable to use the 'deep' option for calculations, when the problem does not arise.

External subroutines used

At each step of the iteration it is necessary to solve the matrix equation. The LINPACK subroutine library (Dongarra and others, 1979) was determined to be satisfactory, and at the time of writing was available widely. From that library of linear algebra subroutines, "dgefa" and "dgesl" are used, which do refer to other external references from the library. In the event of this library being unavailable, any routine for the solution of a system of linear equations should be suitable. The square matrix "a" has a rank of $2N + 10$, but a specified dimension in the program of 41. The vector of right sides, $-f$ is stored in "b". After returning, "b" contains the solution vector of corrections.

Output of results

The program prints out the solution vector z at each iteration, as a guide so that the process can be monitored. After convergence at the height required, the program calls a subroutine "output" which calculates some extra quantities which may be useful in practice, and prints out the results. The first $2N + 10$ numbers output, which are titled, form the solution vector z . That is, the various dimensions of the wave-train, velocities, volume flux, Bernoulli constant, the point values of the surface elevation, and the Fourier coefficients B_j . The program at this stage also calculates the discrete Fourier transform of the surface elevations, denoted here by Y_j , and which are stored in dimensioned variable "y":

$$Y_j = \frac{2}{N} \sum_{m=0}^N k\eta_m \cos \frac{mj\pi}{N},$$

where the Σ' means that in the summation, the contribution at 0 and N is multiplied by $\frac{1}{2}$.

Next, the values of some of the integral quantities of the wave train are printed. These have been calculated by the formulae given by Cokelet (1977), and in terms of quantities defined in this paper are:

Impulse

$$\frac{I}{\rho(g/k^3)^{1/2}} = q(k^3/g)^{1/2} + kdc_E(k/g)^{1/2},$$

Kinetic energy

$$\frac{T}{\rho g/k^2} = \frac{1}{2} \left[c(k/g)^{1/2} \frac{I}{\rho(g/k^3)^{1/2}} + c_E(k/g)^{1/2} (q(k^3/g)^{1/2} - \bar{u}(k/g)^{1/2} kd) \right],$$

Potential energy

$$\frac{V}{\rho g/k^2} = \frac{1}{4N} \left[(k\eta_0)^2 + (k\eta_N)^2 + 2 \sum_{m=1}^{N-1} (k\eta_m)^2 \right],$$

Mean square of bed velocity

$$\frac{\bar{u}_b^2 k}{g} = 2 \frac{rk}{g} - \frac{c^2 k}{g},$$

Radiation stress

$$\frac{S_{xx}}{\rho g/k^2} = 4 \frac{T}{\rho g k^2} - 3 \frac{V}{\rho g k^2} + \frac{\bar{u}_b^2 k}{g} kd + 2c_E(k/g)^{1/2} (\bar{u}(k/g)^{1/2} kd - q(k^3/g)^{1/2}),$$

Wave power

$$\frac{F}{\rho g^{3/2} k^{-5/2}} = c(k/g)^{1/2} \left(3 \frac{T}{\rho g k^2} - 2 \frac{V}{\rho g k^2} \right) + \frac{1}{2} \frac{\bar{u}_b^2 k}{g} \left(\frac{I}{\rho(g/k^3)^{1/2}} + kdc(k/g)^{1/2} \right) + c(k/g)^{1/2} c_E(k/g)^{1/2} (kd\bar{u}(k/g)^{1/2} - q(k^3/g)^{1/2}).$$

Finally, for finite depth only, the values of the discharge Q per unit span, the Bernoulli constant R , and the momentum flux S per unit span are printed. The first two have been defined previously, the last is calculated:

Momentum flux

$$\frac{S}{\rho g/k^2} = \frac{S_{xx}}{\rho g/k^2} - 2c(k/g)^{1/2} \frac{I}{\rho(g/k^3)^{1/2}} + kd \left(\frac{c^2 k}{g} + \frac{1}{2} kd \right).$$

Table 3 contains an example of the final output of the program, that resulting from the data given in Table 2(a). Comparison with the results of Cokelet (1977) show that these agree with his results to the figures shown. The small negative value for the mean square of the bed velocity is a measure of the errors in the

Table 3. Example of final output from program

Solution, non-dimensionalized by wavenumber				
Wave height	.613368			
Wave period	5.994646			
Wave speed	1.048133			
Mean Eulerian fluid velocity	.000000			
Mean mass transport velocity	.000000			
Mean fluid speed relative to wave	1.048133			
Volume flux due to waves	.043984			
Bernoulli constant	.549291			
Surface elevations - crest to trough				
.3611 .3262 .2421 .1410 .0422 -.0459 -.1196 -.1775 -.2190 -.2440 -.2523				
Fourier coefficients				
1 .287008	2 .004529	3 .000375	4 .000038	5 .000004
6 .000001	7 .000000	8 .000000	9 .000000	10 .000000
Integral quantities				
Impulse (I)	.439839e-01			
Kinetic energy (T)	.230505e-01			
Potential energy (V)	.219555e-01			
Mean square of bed velocity	-.609826e-08			
Radiation stress (Sxx)	.263353e-01			
Wave power (F)	.264552e-01			

solution—at the bottom of the deep ocean this should of course be zero, and certainly should not be negative.

Calculations of fluid velocity and pressure

In many applications the integral quantities mentioned are not required. Instead, local values of velocity and pressure are needed. These can be obtained from a call to subroutine "point", included at the end of the program, but not called from the main program. The velocities and pressures calculated by that program are those in the physical frame through which the waves travel at velocity *c*. It is shown easily for a coordinate system (*X*, *Y*) with origin at the same level as that of (*x*, *y*), with *y* = *Y* and *x* = *X* - *ct*, where *t* is time such that *t* = 0 when the wave crest is above the origin of the (*X*, *Y*) frame, then the velocities in this frame (*U*, *V*) are such that *U* = *u* + *c* and *V* = *v*, which gives

$$U(k/g)^{1/2} = c(k/g)^{1/2} - \bar{u}(k/g)^{1/2} + \sum_{j=1}^n jB_j \frac{\cosh jk(d+y)}{\cosh jkd} \cos jk(X-ct),$$

and

$$V(k/g)^{1/2} = \sum_{j=1}^N jB_j \frac{\sinh jk(d+y)}{\cosh jkd} \sin jk(X-ct).$$

The pressure *p*(*X*, *Y*, *t*) then is calculated easily from Bernoulli's theorem:

$$\frac{p}{\rho g/k^2} = \frac{rk}{g} - kY - \frac{1}{2}[(U(k/g)^{1/2} - c(k/g)^{1/2})^2 + (V(k/g)^{1/2})^2].$$

The surface elevation is calculated from the interpolating Fourier series which passes through the point values *kη_m*:

$$k\eta = \sum_{j=1}^N Y_j \cos jk(X-ct).$$

For *k*(*X* - *ct*) stored in "kx", and *kY* stored in "ky", the statement

"call point(kx,ky,u,v,press,elevn)"

returns the value of *U*(*k/g*)^{1/2} in "u", *V*(*k/g*)^{1/2} in "v", *p*/ρ*gk*⁻² in "press" and the surface elevation above that point, *kη*(*X* - *ct*) in "elevn".

In some applications it also is necessary to calculate derivatives of the fluid velocities, such as in applications of Morison's equation. Values of ∂*U*/∂*t*, ∂*V*/∂*t*, ∂*U*/∂*X*, ∂*V*/∂*X*, ∂*U*/∂*Y*, and ∂*V*/∂*Y* may be obtained by differentiating the above expressions for *U* and *V*, and adding some simple lines of code to subroutine "point", which, it is hoped, is sufficiently transparent to enable this to be done easily.

REFERENCES

Chaplin, J. R., 1980, Developments of stream-function wave theory: Coastal Engng, v. 3, no. 3, p. 179-205.
 Chappellear, J. E., 1961, Direct numerical calculation of wave properties: Jour. Geophys. Res., v. 66, no. 2, p. 501-508.
 Cokelet, E. D., 1977, Steep gravity waves in water of arbitrary uniform depth: Phil. Trans. Royal Soc. London Ser. A, v. 286, no. 1335, p. 183-230.
 Dean, R. G., 1965, Stream function representation of non-linear ocean waves: Jour. Geophys. Res., v. 70, no. 18, p. 4561-4572.
 Dongarra, J. J., Moler, C. B., Bunch, J. R., and Stewart, G. W., 1979, LINPACK User's Guide, S.I.A.M. Philadelphia.
 Eckart, C., 1952, The propagation of gravity waves from deep to shallow water: National Bureau of Standards Circular 521, Washington, D.C., p. 165-173.
 Fenton, J. D., 1979, A high-order cnoidal wave theory: Jour. Fluid Mech., v. 94, no. 1, p. 129-161.
 Fenton, J. D., 1983, On the application of steady wave theories: Proc. 6th Australian Conf. on Coastal and Ocean Engineering (Gold Coast, July 1983), Inst. Engineers, Australia, p. 64-68.
 Fenton, J. D., 1985, A fifth-order Stokes theory for steady waves: Jour. Waterway Port Coastal and Ocean Engng ASCE, v. 111, no. 2, p. 216-234.
 Le Méhauté, B., Divoky, D., and Lin, A., 1968, Shallow water waves: a comparison of theories and experiments: Proc. 11th Conf. Coastal Engng, ASCE, v. 1, p. 86-107.
 Rienecker, M. M., and Fenton, J. D., 1981, A Fourier approximation method for steady water waves: Jour. Fluid Mech., v. 104, p. 119-137.

APPENDIX

Program Steady

```

PROGRAM STEADY
C CALCULATION OF STEADY WAVES.
  implicit double precision(a-h,k-l,o-z)
  character*10 depth,case,currnt
  common /one/n,num,pi,hoverd,height,value,depth,case,currnt
  common /two/z(41),cosa(0:41),sina(0:41),coeff(41),sol(41,2),y(41)
  dimension rhs1(41),rhs2(41),a(41,41),b(41),ipvt(41)
C
C INPUT DATA
C
C   "depth" IS EITHER 'deep' OR 'finite'.
C   "hoverd" IS WAVE HEIGHT/DEPTH.
  read(5,*)depth,hoverd
C   "case" IS EITHER 'period' OR 'wavelength'.
C   "height" IS HEIGHT/LENGTH IF "case" IS 'wavelength'.
C   "height" IS HEIGHT/(g*T**2) IF "case" IS 'period'.
  read(5,*)case,height
C   "currnt" IS EITHER 'euler' OR 'stokes'.
C   "value" is the magnitude of the mean Eulerian or Stokes velocities,
C   non-dimensionalized with respect to wave height.
  read(5,*)currnt,value
C   "n" is the number of terms in the Fourier series and the number
C   of intervals in half a wavelength.
C   "nstep" is the number of steps in wave height.
  read(5,*)n,nstep
C   "number" is the number of iterations for each wave height step.
  number=9
C   "crit" is the criterion for convergence. If the sum of magnitudes
C   of corrections is smaller than crit, the iteration stops.
  crit=1.d-3
  write(6,20)depth,hoverd
  write(6,21)height,case
  write(6,22)currnt,value
  num=2*n+10
  pi=4.d0*atan(1.d0)
  dhe=height/nstep
  dho=hoverd/nstep
C
C COMMENCE STEPPING THROUGH STEPS IN WAVE HEIGHT.
C
  do 1 ns=1,nstep
  write(6,23)ns,nstep
  height=ns*dhe
  hoverd=ns*dho
C
C CALCULATE INITIAL LINEAR SOLUTION.
C
  if(ns.le.1)then
    call init
  else
C
C OR, EXTRAPOLATE FOR NEXT WAVE HEIGHT, IF NECESSARY.
  do 3 i=1,num
  3  z(i)=2.*sol(i,2)-sol(i,1)
  endif
C
C COMMENCE ITERATIVE SOLUTION
C
  do 4 iter=1,number
  write(6,24)iter
C
C CALCULATE RIGHT SIDES OF EQUATIONS AND DIFFERENTIATE NUMERICALLY
C TO OBTAIN JACOBIAN MATRIX.
C
  call eqns(rhs1)
  do 5 i=1,num

```

```

h=0.01*z(i)
if(abs(z(i)).lt.1.d-4)h=1.d-5
z(i)=z(i)+h
call eqns(rhs2)
z(i)=z(i)-h
b(i)=-rhs1(i)
  do 6 j=1,num
6   a(j,i)=(rhs2(j)-rhs1(j))/h
5   continue
C
C SOLVE MATRIX EQUATION AND CORRECT VARIABLES, USING "LINPACK" ROUTINES.
C
C The matrix equation [a(i,j)][correction vector]=[b(i)] is to be solved.
C
  call dgefa(a,41,num,ipvt,info)
  if(info.ne.0)then
    write(6,27)
    stop
  endif
  call dgesl(a,41,num,ipvt,b,0)
C
C The b(i) are now the corrections to each variable.
C
  sum=0.d0
  do 7 i=1,num
    sum=sum+abs(b(i))
7   z(i)=z(i)+b(i)
  write(6,25)(z(i),i=1,num)
  crit=crit
  if(ns.eq.nstep)crit=0.01*crit
  if(sum.lt.crit)goto 8
4  continue
  write(6,26)number
  stop
8  if(ns.eq.1)then
  do 9 i=1,num
9   sol(i,2)=z(i)
  else
  do 10 i=1,num
  sol(i,1)=sol(i,2)
10  sol(i,2)=z(i)
  endif
1  continue
C
C OUTPUT OF RESULTS
C
  call output
20 format(//,'Depth: ',a6,', Height/Depth',f7.4)
21 format(/,'Wave height',f9.6,', dimensionless with respect to ',
1a10)
22 format(//,'Current criterion ',a6,', Magnitude ',f5.2)
23 format(//,'Height step ',i2,' of ',i2)
24 format(//,'Iteration ',i3)
25 format(/,'Solution vector',10(/,6e13.6))
26 format(//,'Did not converge sufficiently after ',i3,' iterations.')
27 format(/,'Matrix singular')
  stop
  end
C
C SUBROUTINE TO CALCULATE INITIAL SOLUTION FROM LINEAR WAVE THEORY.
C
  subroutine init
  implicit double precision(a-h,k-l,o-z)
  character*10 depth,case,currnt
  common /one/n,num,pi,hoverd,height,value,depth,case,currnt
  common /two/z(41),cosa(0:41),sina(0:41),coeff(41),sol(41,2),y(41)
  if(depth.eq.'finite')then
    if(case.eq.'period')then
      a=4.*pi*pi*height/hoverd
      b=a/sqrt(tanh(a))
      t=tanh(b)
      z(1)=b+(a-b*t)/(t+b*(1.-t*t))

```

```

else
  z(1)=2.*pi*height/hoverd
endif
z(2)=z(1)*hoverd
z(4)=sqrt(tanh(z(1)))
else
  z(1)=-1.d0
  z(4)=1.d0
  if(case.eq.'period')then
    z(2)=4.*pi*pi*height
  else
    z(2)=2.*pi*height
  endif
endif
z(3)=2.*pi/z(4)
if(currnt.eq.'euler')then
  z(5)=value*sqrt(z(2))
  z(6)=0.d0
else
  z(6)=value*sqrt(z(2))
  z(5)=0.d0
endif
z(7)=z(4)
z(8)=0.d0
z(9)=0.5*z(7)**2
cosa(0)=1.d0
sina(0)=0.d0
z(10)=0.5*z(2)
do 1 i=1,n
  cosa(i)=cos(i*pi/n)
  cosa(i+n)=cos((i+n)*pi/n)
  sina(i)=sin(i*pi/n)
  sina(i+n)=sin((i+n)*pi/n)
  z(n+i+10)=0.d0
1  z(i+10)=0.5*z(2)*cosa(i)
  z(n+11)=0.5*z(2)/z(7)

write(6,2)(z(i),i=1,num)
2 format(/,'Initial linear solution',10(/,6e13.6))
do 3 i=1,9
3  sol(i,1)=z(i)
  sol(i,2)=0.d0
do 4 i=10,num
4  sol(i,1)=0.d0
return
end

```

C

C SUBROUTINE FOR EVALUATION OF EQUATIONS.

C

```

subroutine eqns(rhs)
implicit double precision(a-h,k-l,o-z)
character*10 depth,case,currnt
common /one/n,num,pi,hoverd,height,value,depth,case,currnt
common /two/z(41),cosa(0:41),sina(0:41),coeff(41),sol(41,2),y(41)
dimension rhs(41)
if(depth.eq.'finite')then
  rhs(1)=z(2)-z(1)*hoverd
else
  rhs(1)=z(1)+1.d0
endif
if(case.eq.'wavelength')then
  rhs(2)=z(2)-2.*pi*height
else
  rhs(2)=z(2)-height*z(3)**2
endif
rhs(3)=z(4)*z(3)-pi-pi
rhs(4)=z(5)+z(7)-z(4)
rhs(5)=z(6)+z(7)-z(4)
if(depth.eq.'finite')then
  rhs(5)=rhs(5)-z(8)/z(1)
do 2 i=1,n

```

```

2  coeff(i)=z(n+i+10)/cosh(i*z(1))
   endif
   it=6
   if(currnt.eq.'euler')it=5
   rhs(6)=z(it)-value*sqrt(z(2))
   rhs(7)=z(10)+z(n+10)
   do 1 i=1,n-1
1  rhs(7)=rhs(7)+z(10+i)+z(10+i)
   rhs(8)=z(10)-z(n+10)-z(2)
   do 3 m=0,n
   psi=0.d0
   u=0.d0
   v=0.d0
   if(depth.eq.'finite')then
   do 4 j=1,n
   nm=mod(m*j,n+n)
   e=exp(j*(z(1)+z(10+m)))
   s=0.5*(e-1./e)
   c=0.5*(e+1./e)
   psi=psi+coeff(j)*s*cosa(nm)
   u=u+j*coeff(j)*c*cosa(nm)
   v=v+j*coeff(j)*s*sina(nm)
4  continue
   else
   do 5 j=1,n
   nm=mod(m*j,n+n)
   e=exp(j*z(10+m))
   psi=psi+z(n+j+10)*e*cosa(nm)
   u=u+j*z(n+j+10)*e*cosa(nm)
5  v=v+j*z(n+j+10)*e*sina(nm)
   endif
   rhs(m+9)=psi-z(8)-z(7)*z(m+10)
   rhs(n+m+10)=0.5*((-z(7)+u)**2+v**2)+z(m+10)-z(9)
3  continue
   return
   end
C
C SUBROUTINE FOR OUTPUT OF RESULTS
C
   subroutine output
   implicit double precision(a-h,k-l,o-z)
   character*10 depth,case,currnt
   common /one/n,num,pi,hoverd,height,value,depth,case,currnt
   common /two/z(41),cosa(0:41),sina(0:41),coeff(41),sol(41,2),y(41)
C Calculate Fourier coefficients of surface elevation
   do 10 j=1,n
   sum=0.5d0*(z(10)+z(n+10)*(-1.d0)**j)
   do 11 m=1,n-1
11  sum=sum+z(10+m)*cosa(mod(m*j,n+n))
10  y(j)=2.*sum/n
   write(6,1)
1  format(//,'Solution, non-dimensionalized by wavenumber',/)
   if(depth.eq.'finite')write(6,2)z(1)
2  format('Water depth',f10.6,/)
   write(6,3)(z(i),i=2,9)
3  format('Wave height',f10.6,/,
1'Wave period',f10.6,/,
1'Wave speed',f10.6,/,
1'Mean Eulerian fluid velocity',f10.6,/,
1'Mean mass transport velocity',f10.6,/,
1'Mean fluid speed relative to wave',f10.6,/,
1'Volume flux due to waves',f10.6,/,
1'Bernoulli constant',f10.6,/)
   write(6,4)(z(i),i=10,n+10)
4  format(//,'Surface elevations - crest to trough',//,2(18f7.4,))
   write(6,5)(i,z(i+n+10),i=1,n)
5  format(//,'Fourier coefficients',/,
110(5(i3,f9.6,3x),/))
   pulse=z(8)+z(1)*z(5)
   ke=0.5*(z(4)*pulse+z(5)*(z(8)-z(7)*z(1)))
   pe=0.5*(z(10)**2+z(n+10)**2)
   do 7 i=1,n-1

```

```

7  pe=pe+z(10+i)**2
   pe=pe/(2.*n)
   ub2=2.*z(9)-z(4)**2
   sxx=4.*ke-3.*pe+ub2*z(1)+2.*z(5)*(z(7)*z(1)-z(8))
   f=z(4)*(3.*ke-2.*pe)+0.5*ub2*(pulse+z(4)*z(1))
   1+z(4)*z(5)*(z(7)*z(1)-z(8))
   write(6,8)pulse,ke,pe,ub2,sxx,f
8  format(/,'Integral quantities',//,
   1'Impulse (I)           ',e13.6,//,
   1'Kinetic energy (T)    ',e13.6,//,
   1'Potential energy (V)  ',e13.6,//,
   1'Mean square of bed velocity ',e13.6,//,
   1'Radiation stress (Sxx) ',e13.6,//,
   1'Wave power (F)        ',e13.6)
   if(depth.eq.'finite')then
     q=z(7)*z(1)-z(8)
     r=z(9)+z(1)
     s=sxx-2.*z(4)*pulse+(z(4)**2+0.5*z(1))*z(1)
     write(6,9)q,r,s
9  format(/,'Invariants for finite depth',//,
   1'Volume flux (Q)       ',f9.6,//,
   1'Bernoulli constant (R) ',f9.6,//,
   1'Momentum flux (S)    ',f9.6)
   endif
   return
   end

```

C
C SUBROUTINE FOR CALCULATION OF FREE SURFACE ELEVATION elevn AT kx ,
C AND VELOCITY (u,v) AND PRESSURE press AT POINT (kx,ky).
C

```

subroutine point(kx,ky,u,v,press,elevn)
implicit double precision(a-h,k-l,o-z)
character*10 depth,case,currnt
common /one/n,num,pi,hoverd,height,value,depth,case,currnt
common /two/z(41),cosa(0:41),sina(0:41),coeff(41),sol(41,2),y(41)
elevn=0.5*y(n)*cos(n*kx)
do 1 j=1,n-1
1  elevn=elevn+y(j)*cos(j*kx)
   u=z(4)-z(7)
   v=0.d0
   if(depth.eq.'finite')then
     do 4 j=1,n
       e=exp(j*(z(1)+ky))
       s=0.5*(e-1./e)
       c=0.5*(e+1./e)
       b=z(n+j+10)/cosh(j*z(1))
       u=u+j*b*c*cos(j*kx)
4      v=v+j*b*s*sin(j*kx)
     else
       do 5 j=1,n
         e=exp(j*ky)
         u=u+j*z(n+j+10)*e*cos(j*kx)
5        v=v+j*z(n+j+10)*e*sin(j*kx)
     endif
   press=z(9)-ky-0.5*((u-z(4))**2+v*v)
   return
   end

```