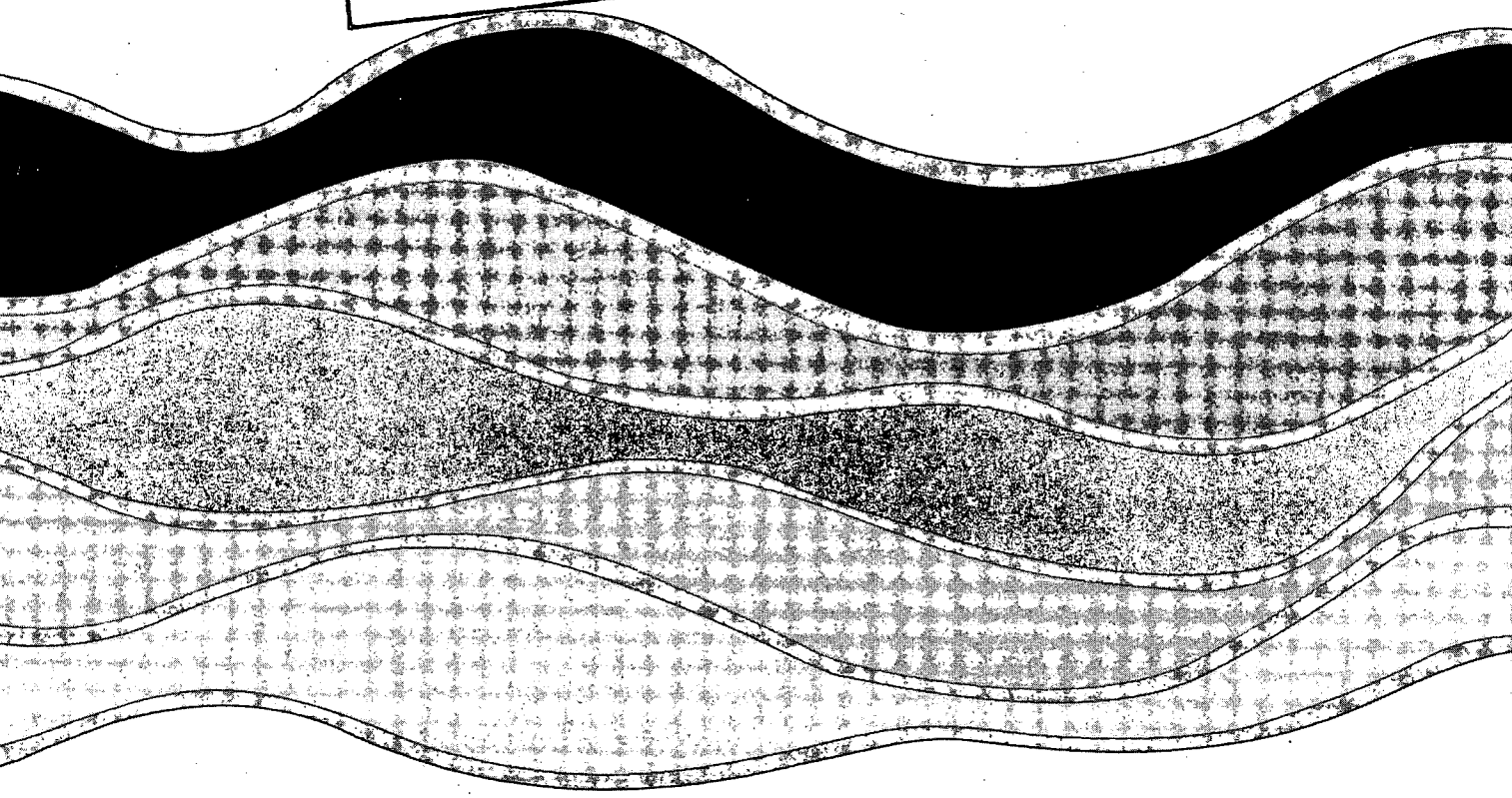
NATIONAL WATER RESEARCH INSTITUTE

INSTITUT NATIONAL de RECHERCHE sur les EAUX



ORBITAL.M - A COMPUTER PROGRAM TO CALCULATE THE VELOCITY FIELD BENEATH IRREGULAR WAVES

François Anctil

NWRI Contribution No. 92-02

ORBITAL.M - A COMPUTER PROGRAM TO CALCULATE THE VELOCITY FIELD BENEATH IRREGULAR WAVES

François Anctil

National Water Research Institute
Burlington, Ontario, L7R 4A6

December, 1991

MANAGEMENT PERSPECTIVE

The accurate prediction of wave-induced velocities is required for the safe engineering design of structures subjected to wave action. This report presents a new theoretically consistent method of predicting the velocity field beneath an irregular wavetrain. A laboratory data set, which provides unique detailed measurements of the velocities near the surface in wave crests, is used to demonstrate the relative accuracy of this new method of prediction. The results show that the new method provides more accurate predictions than previously employed prediction methods. This information could help improve designs, thereby reducing the risk of structural failure that might pose an environmental danger.

This is a companion report to NWRI 91-111 ("A simple method for calculating the velocity field beneath irregular waves" by M. Donelan, F. Anctil and J. Doering). It is a description of the computer code that follows from the theory outlined in the earlier report.

SOMMAIRE À L'INTENTION DE LA DIRECTION

La prévision exacte des vitesses induites par les vagues est nécessaire pour la conception technique de structures pouvant être soumises sans danger à l'action des vagues. On trouvera dans le présent rapport une nouvelle méthode de prévision du champ de vitesse sous un train de vagues irrégulier qui est conforme aux données théoriques. Un ensemble de données expérimentales, qui fournissent des mesures détaillées spéciales des vitesses près de la surface dans les crêtes de vague, est utilisé pour montrer la précision relative de cette nouvelle méthode de prévision. Les résultats montrent que cette méthode permet d'obtenir des prévisions plus précises que les méthodes de prévision utilisées antérieurement. Ces informations permettraient d'améliorer les conceptions, réduisant ainsi le risque de défaillance des structures qui peuvent éventuellement constituer un danger environnemental.

Il s'agit d'un rapport connexe à la Contribution de l'INRE n° 91-111 (*A simple method for calculating the velocity field beneath irregular waves* de M. Donelan, F. Anctil et J. Doering). On y décrit le code machine qui découle de la théorie énoncée dans le document mentionné ci-dessus.

ABSTRACT

A new, theoretically consistent method for estimating water velocities beneath surface waves has been developed at the National Water Research Institute (NWRI Contribution No. 91-111 "A simple method for calculating the velocity field beneath irregular waves" by M. Donelan, F. Anctil and J. Doering - also published in *Coastal Engineering,* Vol.16, 1992). This report describes the computer code required to implement the method. A machine-readable copy of the code may be purchased from the National Water Research Institute for a nominal fee.

RÉSUMÉ

L'Institut national de recherche sur les eaux a élaboré une nouvelle méthode de prévision du champ de vitesse sous les vagues de surface, conforme aux données théoriques (Contribution de l'INRE n° 91-111 *A simple method for calculating the velocity field beneath irregular waves* par M. Donelan, F. Anctil et J. Doering, publiée également dans ***Coastal Engineering***, Vol. 16, 1992). Ce rapport décrit le code machine nécessaire pour appliquer la méthode. Pour un coût minime, une copie compréhensible par machine du code peut être obtenue auprès de l'Institut national de recherche sur les eaux.

1. Introduction

It is generally much simpler to measure the surface displacement than the associated velocity field. It is however possible to use the surface displacement measurements to predict the velocity associated with the propagation of waves. The literature is replete with methods to achieve this, based on the linear Gaussian model. They are reviewed and compared by Donelan et al. (1992), who in addition proposed a new theoretically consistent method called superposition. This manual is intended for the description of the MatLab function Orbital.m that takes advantage of the new method to compute the velocity field beneath known irregular waves.

The computational procedure of the superposition method is explained in the next section. Readers interested in the theoretical justification of the method are referred to Donelan et al. (1992). Essential MatLab notions are presented in section 3 to help readers, non-initiated with MatLab, to understand the program coding. Then specific notions about the function are given in section 4, especially to describe the input and output arguments. Finally, references are listed in section 5, and the code is given in section 6.

2. The superposition procedure

Before a velocity prediction can be made, the physical parameters describing the surface displacement measurements must be estimated. This is done by decomposing the surface elevation signal into a series of sinusoidal components, via fast Fourier transformation. In short, the surface displacement is Fourier transformed, and then the amplitude and the phase of each complex Fourier coefficient is retained. The original signal can then be reconstructed by summing all the components.

The velocity field is constructed in a similar way, i.e. by summing the velocity contribution of each component. However, various methods have been proposed to achieve this. The superposition procedure is founded on the assumption that each wave component

rides over longer ones. So starting with the low frequency end of the spectrum, the influence of each component is considered one after the other, taking the previous surface displacement total as mean water level for the present component. This is particular to the superposition procedure: surface displacement and velocity field being constructed simultaneously. It is then possible that the desired vertical position for the velocity prediction is for a while located above the surface, especially if it is above mean water level. In such cases the contribution of the component is limited to its own surface velocity. Finally, at the end, all positions that are still located above the water surface are set to zero.

3. Essential MatLab notions

MatLab (The MathWorks, 1988) is a software system that provides an interactive environment for matrix computations. Data can either be loaded directly from a disk file or manually via the keyboard. Then, operations to the data can be typed in directly, while creating new variables as lines are interpreted by the software. A series of commands can also be grouped in a disk file called function, to be executed as one command. There are two advantages in using a package such as MatLab instead of standard computing langages like C or FORTRAN: the environment is interactive allowing the user to freely analyse data, and the commands are vectorized allowing fast and concise treatment of huge matrices.

A brief presentation of the MatLab software is given herein. This is not a MatLab user's manual, but a concise description of the main trends of MatLab, to allow an understanding of the code of the function Orbital.m by people not familiar with the software.

Variable type. There is only one variable type in MatLab, viz.: double precision floating point matrix, for which some or all elements may be complex numbers. There is no need to declare in advance the size of the variable, because it can be expanded at will. It is however a good memory management practice to use the maximum size of a variable at its first occurence, if it is to be expanded often. All variables are matrices that can be operated

as a whole. For example:

```
B = A
```

would create a new variable B of the same size and having the same elements as variable A. There is no need to use loops, as for most programming langages, to perform this operation element by element. Note that if B already exists, its old value is overwritten whatever its previous size.

Mathematical operators. The main MatLab operators are +, -, *, /, and ^, which respectively stand for addition, substraction, multiplication, division and power. The order of interpretation of a series of operators is similar to other programming language use. There is however one main difference: the operation can be performed on the whole matrix or element by element. This distinction is crucial to the understanding of MatLab. For example the symbol * performs a matrix multiplication, following the principles of standard algebra; however, the combination .* performs an element by element multiplication. In the latter case, both variables should have the same size to allow the operation to take place on elements that occupy the same location inside the matrices. So, whenever a dot is placed before a mathematical operator, it implies that it is performed element by element.

Variable declaration. A variable can be declared in various ways. If elements of a variable are to be enumerated one by one, they are placed between square brackets. Inside the brackets, a space or a comma is equivalent to a concatenation (the value is put alongside) and a semi-colon is equivalent to a carriage return (the value is put underneath). For example, a 2 by 3 matrix may be declared as the following:

```
A = [ 1 2 3 ; 4 5 6 ]
```

An efficient way to declare a series of monotically increasing or decreasing numbers is to use colons to separate the starting value, the increment (default value is 1), and the ending value. For example:

```
A = 10 : -1 : 0
```

yields a row of 11 elements decreasing in steps of 1, from 10 to 0. Finally, the apostrophe operator is used to transpose a matrix, an efficient way to convert a column into a row and vice-versa.

Sub-matrix declaration. To access a particular element or a group of elements of a matrix, its position has to be specified after the matrix name, between round brackets, with a comma separating row and column numbers. For example, A(2, 3:5) yields the three elements of row 2, located from column 3 to 5. Note that in this case the output is a 1 by 3 matrix. A complete row or column can be accessed by using a colon in place of specific numbers.

Predefined variables. MatLab automatically defines a small number of variables at startup. The two most useful ones are pi and i or j (the square root of -1). Those values can be overidden by the user.

Looping. To repeat a group of commands a number of times is called looping. The syntax is as follows:

```
for counter = 1 : N
   statement(s)
end
```

The command for, along with a counter, allows the specification of the number of repetitions wanted (here N), while the command end limits the extent of the group of statements to be repeated. Since MatLab is vectorized, looping should be avoided whenever possible for efficiency reasons.

Relational operators. The relational operators in MatLab are: ==,~=, <, >, <=, and >=, which respectively stand for equal to, not equal to, smaller than, greater than, smaller or equal to, greater or equal to. In addition, & and | respectively mean and, and or.

Making decisions. It is possible to take different actions depending on the results of a test. The syntax is as follows:

```
if   test1
   statement(s)
elseif  test2
   statement(s)
else
   statement(s)
end
```

The command "if" will cause the execution of the following group of statements, if the test is all zeros. The extent of the group is limited by the command end or by the commands else or else if. The former command indicates the beginning of an alternate group of statements to be executed if the test is not all zeros. The latter command is similar, but adds a new test prior the execution of the alternate group.

Command execution. A command is executed when the return key is activated, or at the end of each line inside a function. If the command line is terminated by a semi-colon, the answer is not echoed on the screen. Three consecutive dots at the end of a line informs MatLab that the command expands over an additional line.

Functions. Users may define functions to perform series of specific commands. Functions are thus the key to efficient MatLab use. The software already comes with a large number of functions, that are disk files identified with an .M extension (named M-files). Following is a list of the functions called within Orbital.m. Note that the character % is used to document a function; it indicates to the interpreter that the rest of that line should not be executed.

ABS(X)	Returns the absolute value of the elements of X. When X is complex, ABS(X) returns the complex modulus (magnitude) of X.
ANGLE(X)	Returns the phase angles of a matrix with complex elements.

MEAN(X)	Returns the mean value of vector X.
ERROR('MSG')	Displays the text MSG and causes an error exit from an M-file.
EXP(X)	Returns the exponential of the elements of X, e to the X.
FFT(X)	Returns the discrete Fourier transform of each column of X.
FIND(X)	Finds indices of the non-zero elements in a vector. For example, FIND(X > 100) returns the indices of X, where it is greater than 100.
LENGTH(X)	Returns the length of vector X.
NARGIN	Inside the body of a user-defined function, the permanent variable NARGIN indicates the number of input arguments that were used to call the function.
SIZE(X)	If X is an M-by-N matrix, then SIZE(X) returns [M, N].
WAVEK(F, H)	Returns a column vector of wavenumbers for given frequencies F and water depth H, using the linear wave dispersion relation. WAVEK is a user defined function.
ZEROS(M, N)	ZEROS(M, N) returns an M-by-N matrix of zeros.

4. Notions specific to Orbital.m

The proper use of Orbital.m demands a complete understanding of the input and output arguments. There are 4 input arguments: eta, rate, h, and z. eta is the discretized signal of the surface displacement, an n by 1 matrix, for which n is the number of values. Since eta is Fourier transformed, the most efficient FFT algorithm requires that n be a power of 2; however, MatLab also offers a second, but slower, algorithm that allows n to be any factor of 2. rate is the sampling rate of the surface displacement measurements (in Hz), and it allows the program to determine the frequency of each component. h is the water depth, estimated from the mean water level (MWL)[1] to the bottom. It is a positive value that has

---

MWL: MWL signifies the mean water level for the entire set of eta values; i.e. it is the average position of the surface.

mwl: In this program each successively shorter wave rides on the surface given by the sum of all longer waves. This

a direct impact on the velocity contribution of each component. Finally, z, an n by 1 matrix, is the desired vertical location for computing the velocity at each time step. Its value is zero at the MWL, and negative under the MWL. If z is the same for all time steps, it is equivalent to measurements from a fixed instrument. It is also possible to simulate measurements from a following device that guarantees measurements at a fixed distance from the surface by using z = eta - x, in which x is the constant distance from the surface.

There are two output arguments: u and w, n by 1 matrices that respectively stand for the horizontal and vertical velocity field. In its present form, the program assumes that the waves are two-dimensional. If it is desired to incorporate some known directional spreading, this may be accomplished in a statistical sense by selecting the horizontal direction of u at any instant from a population of angles that reflects the directional spread.

5. Orbital.m

```
function [ u, w ] = orbital( eta, rate, h, z )
%       [ u, w ] = orbital( eta, rate, h, z )
%
% This function evaluates the orbital velocities at specified vertical
% locations around the mean water level (MWL), from a known
% surface displacement time series.
%
%  eta = surface displacement time series,       [m] - matrix [n,1].
% rate = sampling rate of surface displacement,  [Hz] - scalar [1,1].
%    h = water depth (from MWL to the bottom),   [m] - scalar [1,1].
%    z = vertical locations of calculations,     [m] - matrix [n,1].
%        (z = 0 at MWL; z < 0 under MWL)
%
%    u = horizontal orbital velocities,          [m/s] - matrix [n,1].
%    w = vertical orbital velocities,            [m/s] - matrix [n,1].
```

---

time varying surface is treated as the mean water level for the new component. In order to distinguish between this and the overall mean water level, the latter is written with uppercase letters (MWL), while the former is written with lowercase letters (mwl).

```
%

% The computational method used is called Superposition.  It is based
% on the linear Gaussian wave model.  Ref: Donelan, M.A., Anctil, F.
% and Doering, J.C. (1992) "A simple method for calculating the velocity
% field beneath irregular waves." Coastal Engineering, v16.
%
% coding by F. Anctil, 1991.
%

%       VERIFICATION OF THE NUMBER OF INPUT ARGUMENTS.
%
if nargin ~= 4
  error( '4 input arguments are needed' ) ;
end

%       VERIFICATION OF THE LENGTH AND SIZE OF INPUT VARIABLES.
%
eta = eta(:) ; leta = length( eta ) ;
[ lz, wz ] = size( z ) ;

if leta ~= lz
  error( 'the input arguments eta and z should be of same length' ) ;
end

%       SURFACE DISPLACEMENT PHYSICAL PARAMETERS.
%
ETA = fft(eta - mean ( eta )) ;              % Fourier components
leta2 = 1 + ( leta / 2 ) ;
ETA = 2 .* [ ETA(1:leta/2); ETA(leta2) ] ; % keep half of the comp.

A = abs( ETA ) ./ leta ;                  % amplitude, [leta2,1]
ph = -angle( ETA ) ;                      % phase
```

```
f = ( 0 : leta2-1 )' ./ leta .* rate ;      % frequency
om = 2 .* pi .* f ;                         % angular frequency
t = ( 0 : lz-1 )' ./ rate ;                 % time

k = wavek( f, h ) ;                         % wavenumber
relh = k * h ;                              % relative depth

%          BASE GROUND OF THE SUPERPOSITION METHOD
%
% Each wave component rides over the longer ones.  So, starting with
% the low frequency end of the spectrum, the influence of each Fourier
% component is considered one after the other, taking the previous
% total surface height as mean water level (mwl) for the new component.

%          INITIALIZATION OF THE MAIN VARIABLES
%
etatot = zeros( lz, 1 ) ;
surf   = zeros( lz, 2 ) ;
u      = zeros( lz, 1 ) ;
w      = zeros( lz, 1 ) ;

%          STEP BY STEP CONSTRUCTION OF THE ORBITAL TIME SERIES,
%          ONE FOURIER COMPONENT AT A TIME.
%
for iom = 1 : leta2 ;
 phase = ph(iom) - om(iom) .* t ;
                                            % surface displacement
 eta2 = A(iom) .* cos( phase ) ;
                                            % SURFACE ORB. VEL.
 Au  = A(iom) * om(iom) .* cos(phase) ;
 Aw  = A(iom) * om(iom) .* sin(phase) ;
                                            % deep water (relh>2pi)
 if relh(iom) > 2*pi
   part = exp( k(iom) .* eta2 ) ;
```

```
  surf(:,1) = surf(:,1) + part .* Au;
  surf(:,2) = surf(:,2) + part .* Aw;
                                        % intermediate water
elseif relh(iom) ~= 0
  surf(:,1) = surf(:,1) + cosh( k(iom) .* ( eta2 + h ) ) ...
            ./ sinh( k(iom) .* h ) .* Au ;
  surf(:,2) = surf(:,2) + sinh( k(iom) .* ( eta2 + h ) ) ...
            ./ sinh( k(iom) .* h ) .* Aw ;
end
                                        % ORBITAL VELOCITIES
                                        % for z < etatot
ind = find( z < etatot ) ;
if length(ind) > 0
  %
  % For the new component, mwl is the sum of all
  % previous components.  Correct z in accordance.
  %
  zz = z(ind) - etatot(ind) ;
                                        % deep water
  if relh(iom) > 2*pi
    part = exp( k(iom) .* zz ) ;
    u(ind) = u(ind) + part .* Au(ind) ;
    w(ind) = w(ind) + part .* Aw(ind) ;
                                        % intermediate water
  elseif relh(iom) ~= 0
    u(ind) = u(ind) + cosh( k(iom) .* ( zz + h ) ) ...
           ./ sinh( k(iom) .* h ) .* Au(ind) ;
    w(ind) = w(ind) + sinh( k(iom) .* ( zz + h ) ) ...
           ./ sinh( k(iom) .* h ) .* Aw(ind) ;
  end
end
                                        % ORBITAL VELOCITIES
                                        % z >= etatot
ind =find( z >= etatot ) ;
```

```
  lind = length( ind ) ;
  if lind > 0
    u(ind) = surf(ind,1) ;
    w(ind) = surf(ind,2) ;
  end
                                        % add surface displ.
                                        % to previous sum
  etatot = etatot + eta2 ;
end

%          ELIMINATE VALUES FOR POSITIONS ABOVE THE WATER SURFACE
%
ind = find( z > etatot ) ;
lind = length( ind ) ;
if lind > 1
  u(ind) = zeros( lind, 1 );
  w(ind) = zeros( lind, 1 );
elseif lind > 0
  u(ind) = 0;
  w(ind) = 0;
end
```

6. Wavek.m

```
function k = wavek( f, H, tol )
%        k = wavek( f, H, tol )
%
% Returns a column vector of wavenumbers k (1/m) at frequencies f (Hz)
% using the linear wave dispersion relation for water depth H (m).  "tol"
% is an optional error bound on the predicted f's (the default is 1e-4).
% Note: f must increase monotonically from small to large frequencies.
%

% coding by Eugene Terray, 1990.
%
```

```
                % some initial housekeeping

if nargin<3, tol = 1e-4; end          % default tolerance
g = 9.80171;   c = 2*pi*sqrt(H/g);     % g at 40N (m/s^2)
f = c * f(:); Nf = length(f);          % non-dimensionalize f
k = zeros(Nf,1);
                % use approximations for large & small f
                   % define f LARGE if error > 110% * tol
f_r = 1.1*sqrt( abs(log(tol/2))/2 );
nr = find( f >= f_r ) ;       % if so, use the limiting deep-water
if isempty(nr)                % dispersion relation  k = f^2
   nr = Nf+1 ;
else
   k(nr) = ( f(nr).^2 ) ;
   nr = min( nr ) ;
end
                         % f is SMALL if error > 70% * tol
f_l = 0.7*( tol * 604800/281 )^(1/8);
nl = find( f <= f_l ) ;       % if so, use shallow water expansion
if isempty(nl)                % k = f*(1+f^2/6+11f^4/360+17f^6/5040)
   nl = 0 ;
else
   k(nl) = f(nl) .* polyval([17/5040,11/360,1/6,1], f(nl).^2) ;
   nl = max( nl ) ;
end

%  fprintf('\n Non-linear Range: %g < f < %g (Hz)', f_l/c, f_r/c)
%  fprintf('\n                          %g < n < %g  \n', nl, nr)

                % calculate the nonlinear k(f) regime by solving
n = (nl+1):1:(nr-1) ;    % f^2 - k*tanh(k) == 0 using Newton-Raphson
if ~isempty( n )
   dk = 2*tol ;
   k(n) = f(n).^2 ;          % use deep water limit as initial guess
```

```
    while max( abs(dk) ) > tol      % Newton-Raphson iteration loop
       t = tanh( k(n) ) ;
       dk = -(f(n).^2 - k(n).*t) ./ ( t + k(n).*( 1 - t.^2 ) ) ;
       k(n) = k(n) - dk ;
    end
    k(n) = abs( k(n) ) ;      % f(k) = f(-k), so k>0 == k<0 roots
end

N = min( find( k>50 ) ) ;     % diagnostics: inform user if err > tol
if isempty(N),  N = Nf;  end
err = abs( f(2:N) - sqrt( k(2:N).*tanh(k(2:N)) ) )./f(2:N) ;
if max( err ) > tol,  fprintf('\n WAVEK: error exceeds %g \n', tol),  end
%  fprintf('\n max error = %g \n', max( err )),  plot(err,'w')

k = k / H ;                   % finally, give k dimensions of 1/meter
```

7. Acknowledgment

The author wishes to acknowledge Eugene Terray for providing the Wavek.m computer program.

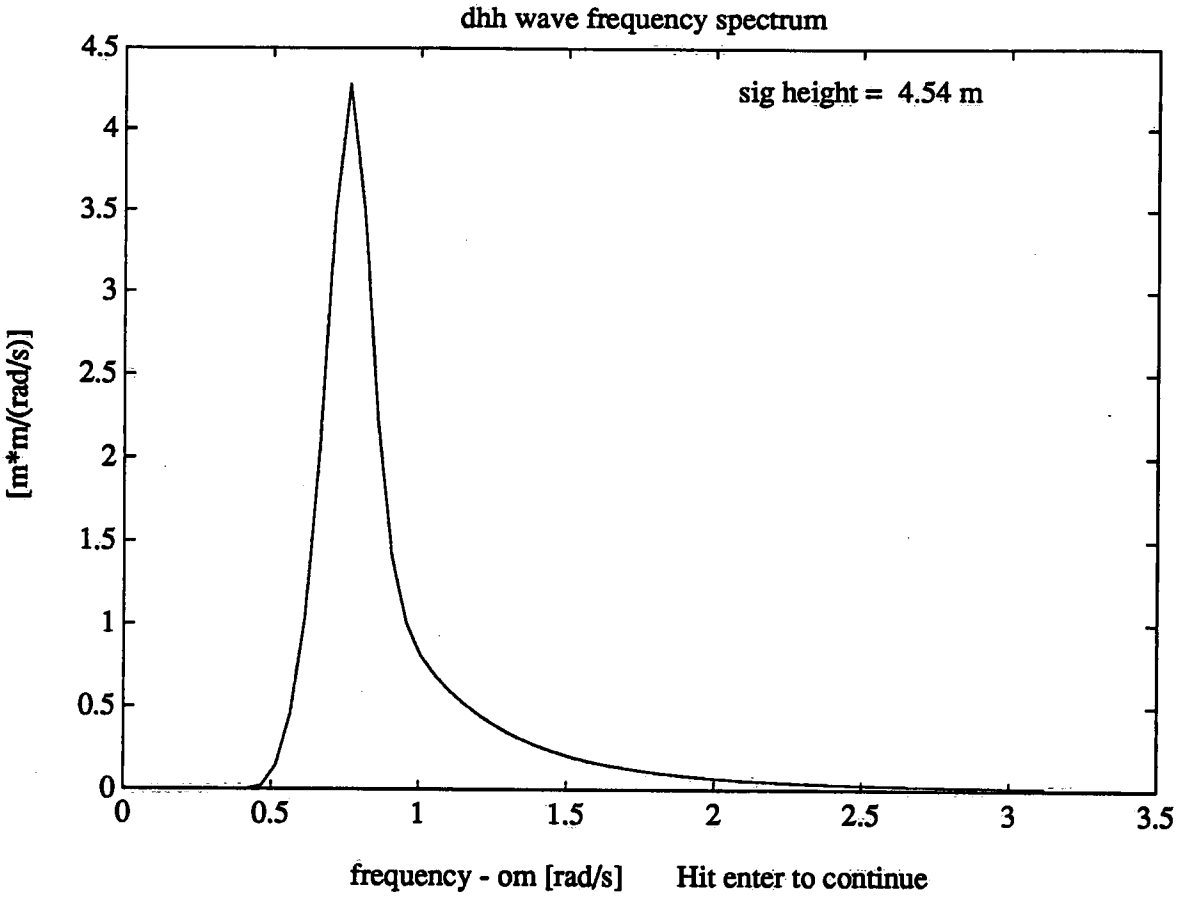
8. References

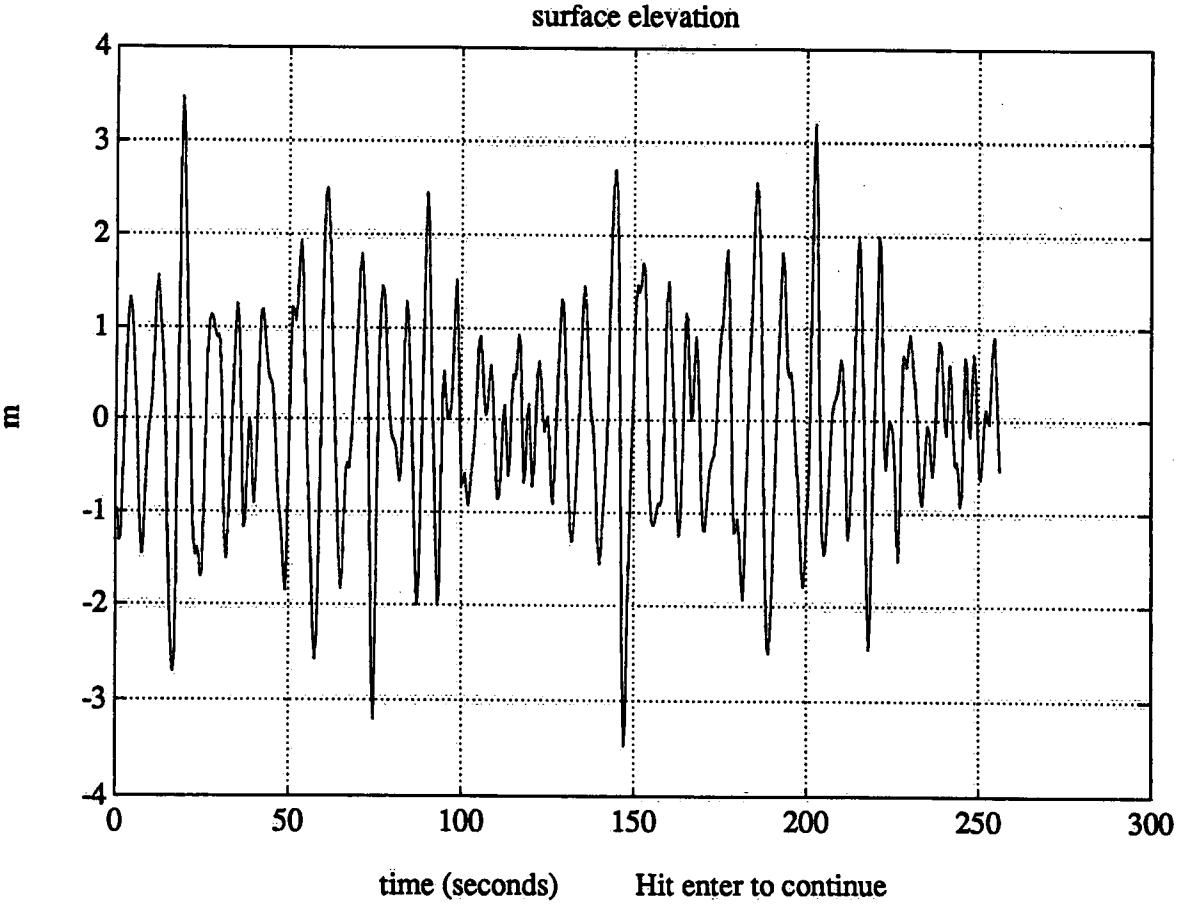
Donelan, M.A., Anctil, F. and Doering, J.C. 1992. A simple method for calculating the velocity field beneath irregular waves. Coastal Engineering, 16, (in press).

The MathWorks. 1988. Signal Processing Toolbox User's guide. South Natick, Massachusetts, 167 p.

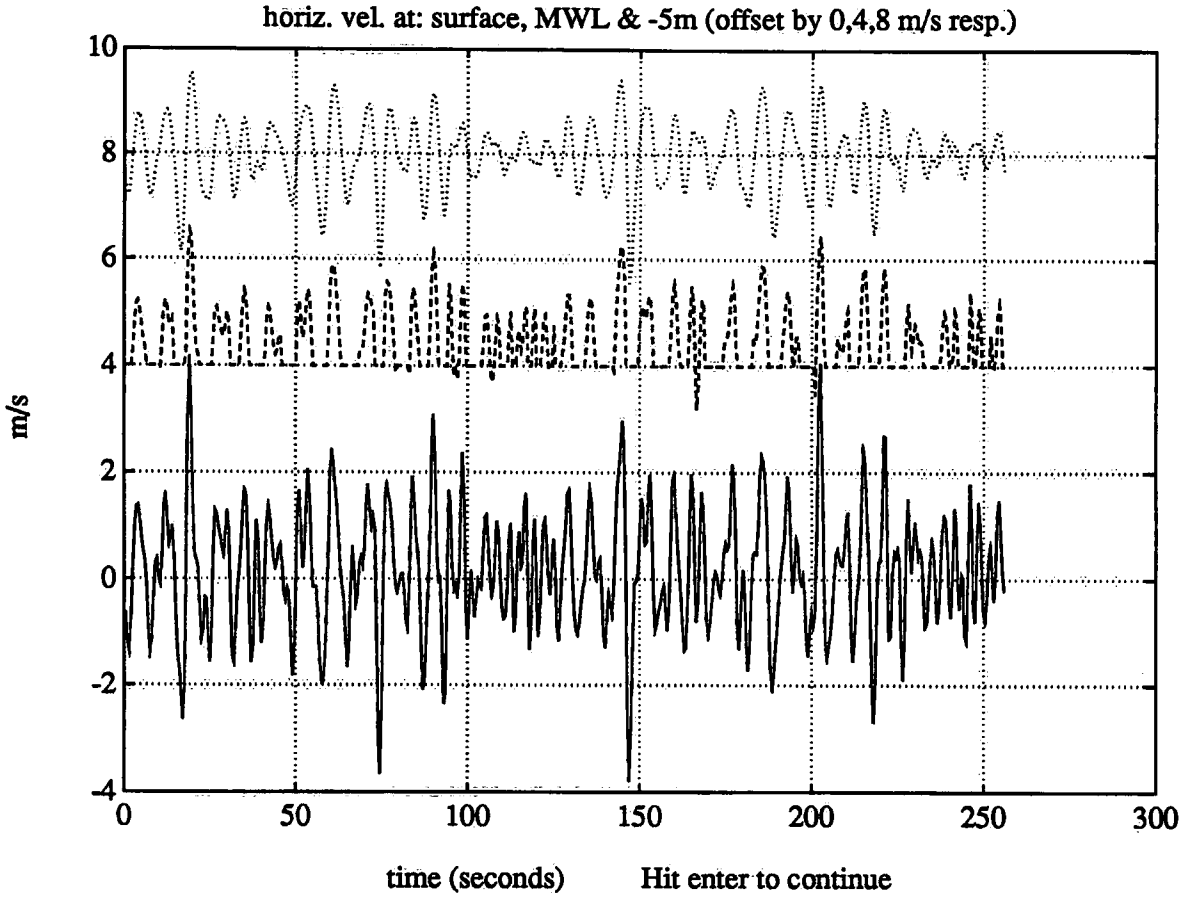
APPENDIX

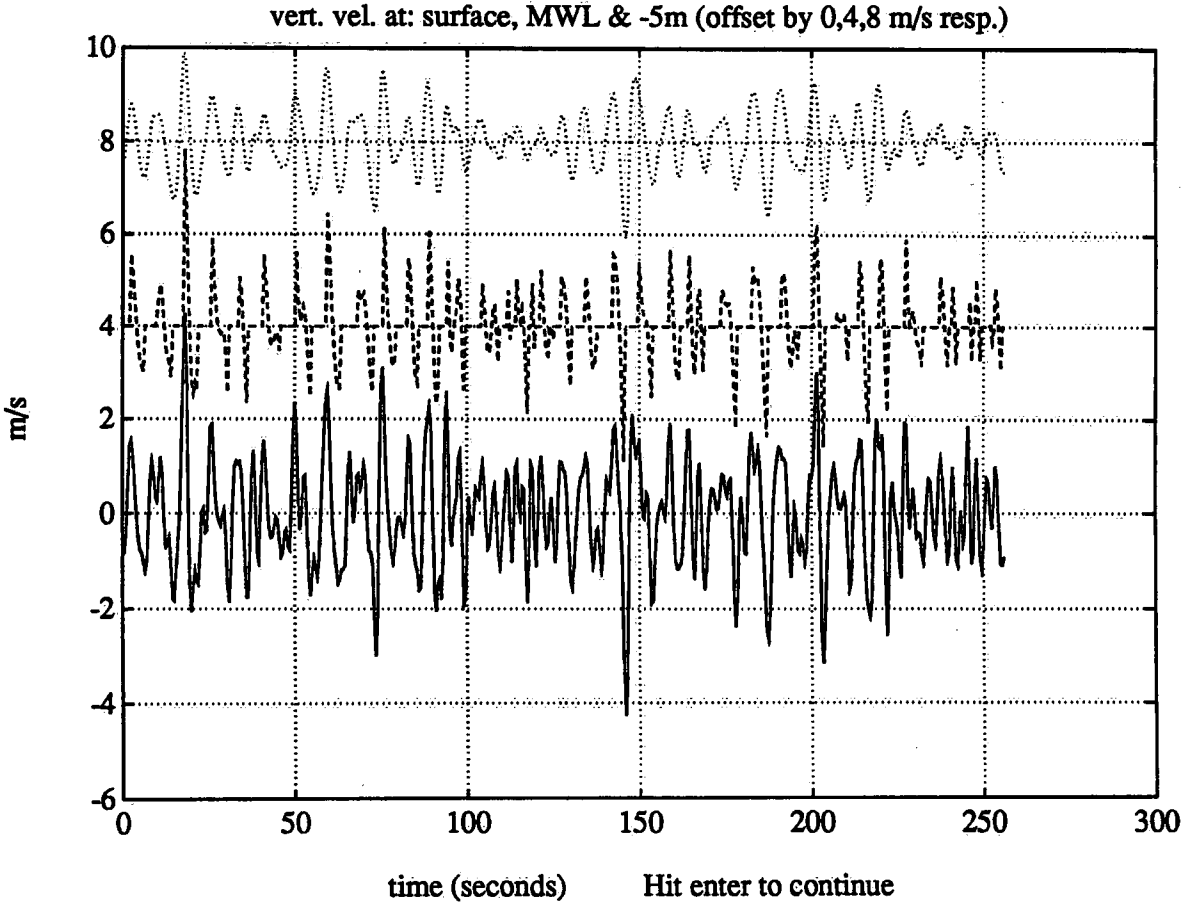
(Sample Output and Listing of Computer Programs on Disk)



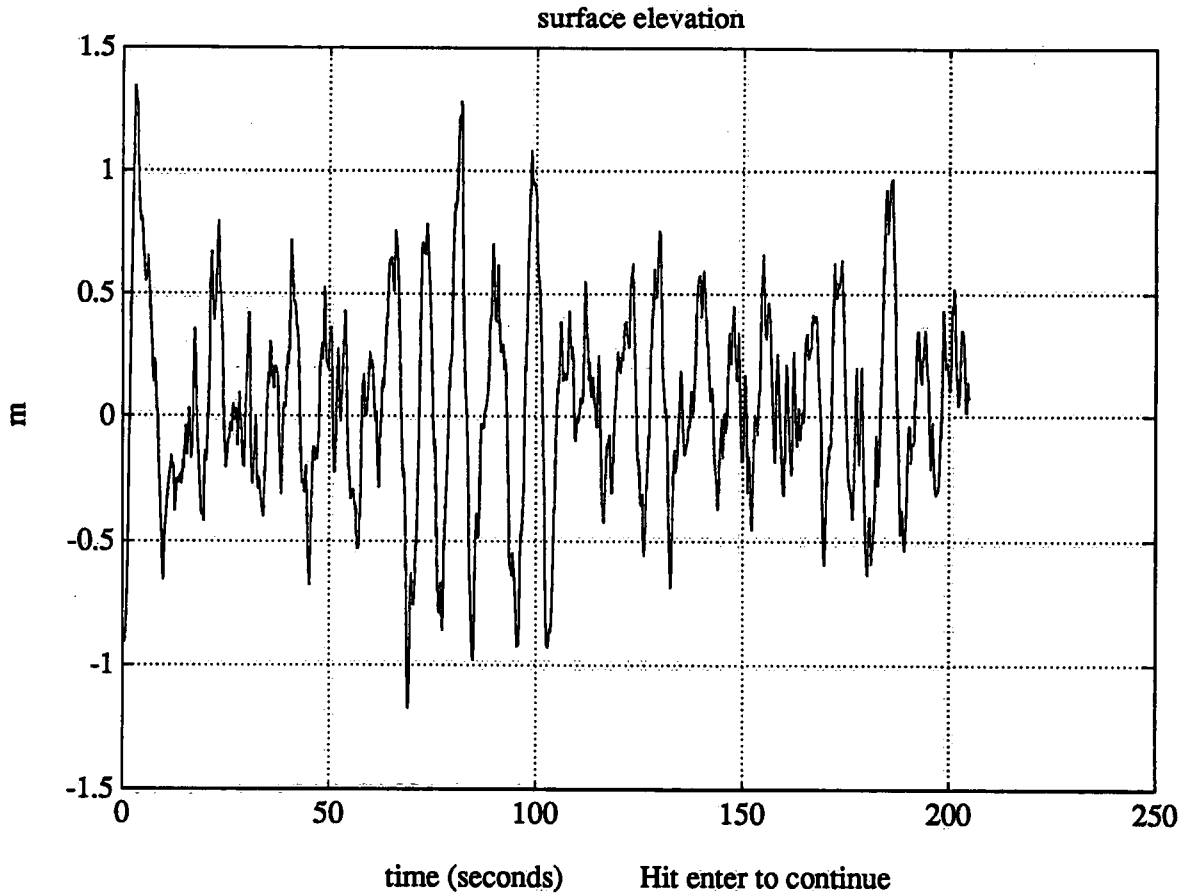


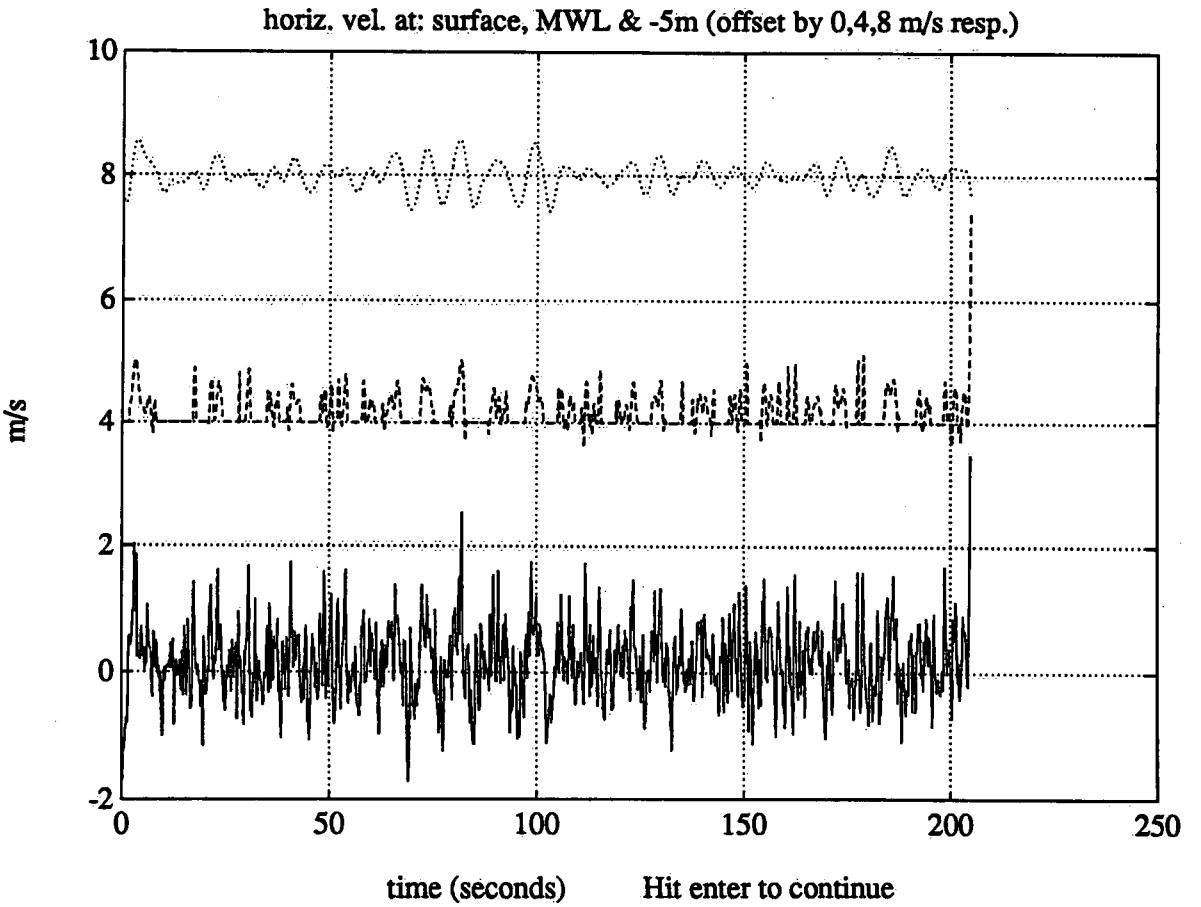
Output of READTHIS for simulated time series



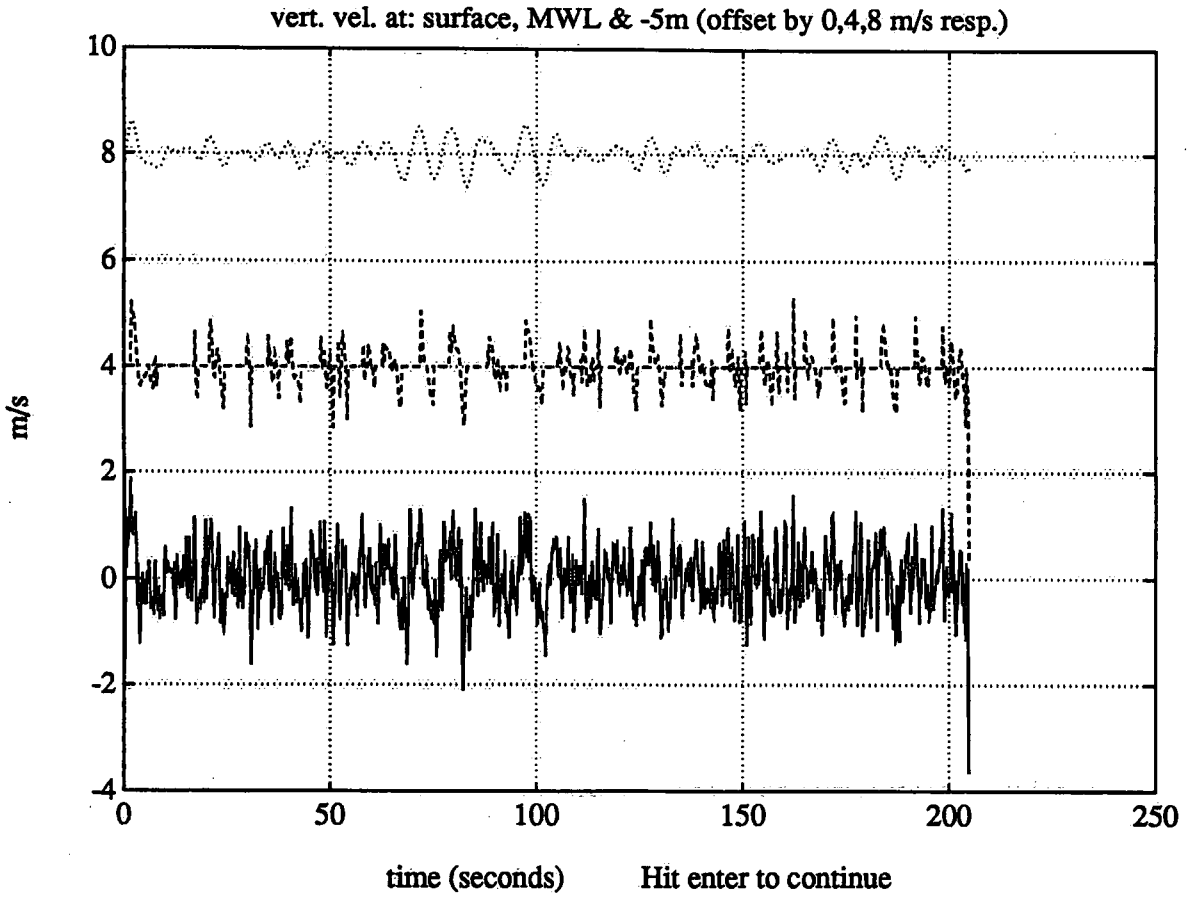


Output of READTHIS for simulated time series





Output of READTHIS for real time series



Output of READTHIS for real time series

```
%
%  ************************************************************
%
% READTHIS is an executable file.  After reading the instuctions
% below, simply type "readthis" (in MATLAB environment) to see
% the example work.
%
% The program halts to display each plot.  Hit return to continue.
% The plots are routed to example.met.
%
!erase example.met
%
%  ************************************************************
%
% ***************************** NOTES ****************************
%
% NOTE 1:
%
% ORBITAL may be used directly as described in the manual, i.e. you
% supply the surface elevation as measured and it computes velocity
% components [u,w] at the desired locations.  Type "help orbital"
% for details.
%
% If no measured data are available a simulated surface elevation
% record (time history) may be created using the functions DHHFRQSP
% and SURFACE1 supplied herewith.
%
% DHHFRQSP creates a frequency spectrum to your specifications.
% Type "help dhhfrqsp" for details.
%
% SURFACE1 uses the spectrum created by DHHFRQSP to construct a
% time series of surface elevation by combining the Fourier components
% in random phase.  The record repeats itself with the periodicity of
% the first non-zero Fourier component.  In the example below this is
% 256 seconds.  Type "help surface1" for details.
%
% The example below computes the velocity components at the surface,
% at the mean water level and at 5m depth.  You may choose either a
% simulated series or a segment of real data.
%
% ------------------------------------------------------------
%
% NOTE 2:
%
% Note that the surface values are recovered by asking for a depth
% marginally below the surface.  Without this small tolerance
% (depthtol) some values would be taken to be above the surface
% (due to numerical imprecision of the computer) and set to zero.
% When a short time series is being analysed there will be some
% window leakage and the above problem will be exacerbated.
% There are two ways to fix the problem: (1) increase depthtol
% (here set to 1 mm); (2) increase the length of the time series.
%
% ------------------------------------------------------------
%
% NOTE 3:
%
% The values at the mean water level are, of course, zero in the
% troughs.
```

```
%
% ---------------------------------------------------------------
%
% NOTE 4:
%
% Whenever desired vertical locations (last argument of ORBITAL) are
% referenced to the surface elevation (measured or simulated), you
% must first remove the mean of the surface elevation.
%
% ---------------------------------------------------------------
%
% NOTE 5:
%
% Although MATLAB will do prime number FFTs, set run length to be
% a power of 2 to maximize the speed of execution.
%
%  ***************************************************************
%
%
%
 choice = input('Input "s" for simulated data or "r" for real data   ','s');
 depth = 50;  % water is 50m deep.
 depthtol = 0.001;  % depthtol is the closest to the surface that a consiste
                    % result can be expected (see above).
% runlth is the length of the run in seconds.
% sampling frequency  = 1/delt in Hertz.
 delt = .2;
 runlth = 1024 * delt;
 if choice == 's'
 delt = .5;
 runlth = 512 * delt;
 lom = 1/runlth;  % set lowest frequency to be 1/(length of record).
 om=(lom:2*lom:.5)*2*pi;  % set radian frequencies of spectrum.
 S=dhhfrqsp(om,31*lom*2*pi,2);  % compute spectrum with peak at (31*lom = )
                                % 0.121Hz, and forcing (ratio of wind speed
                                % to phase speed at spectral peak) of 2.
meta example
 eta=surface1(S,om,0,(delt:delt:256),depth);  % compute surface elevation
                                              % at x=0 and every half second
                                              % for 256 seconds.
 tt = (1:512); t = tt * delt;  % compute time in seconds for plots.
 end
 if choice == 'r'
 tt = (1:1024); t = tt * delt;  % compute time in seconds for plots.
 load etaa
 eta = eta(tt);
 end
 plot(t,eta(tt))
 grid
 title('surface elevation')
 xlabel('time (seconds)          Hit enter to continue')
 ylabel('m')
meta example
pause
if choice == 's'
fprintf('ORBITAL is calculating.  It will take about 1.5 minutes on a 486 PC
else
fprintf('ORBITAL is calculating.  It will take about 5.5 minutes on a 486 PC
end
```

```
fprintf('\n')
eta = eta -mean(eta);  % remove mean of surface elevation
[us,ws]=orbital(eta,1/delt,depth,eta-depthtol);
[u0,w0]=orbital(eta,1/delt,depth,zeros(length(eta),1));
[u5,w5]=orbital(eta,1/delt,depth,-5*ones(length(eta),1));
 plot(t,[us(tt),u0(tt)+4,u5(tt)+8])
 grid
 title('horiz. vel. at: surface, MWL & -5m (offset by 0,4,8 m/s resp.)')
 xlabel('time (seconds)          Hit enter to continue')
 ylabel('m/s')
meta example
pause
 plot(t,[ws(tt),w0(tt)+4,w5(tt)+8])
 grid
 title('vert. vel. at: surface, MWL & -5m (offset by 0,4,8 m/s resp.)')
 xlabel('time (seconds)          Hit enter to continue')
 ylabel('m/s')
meta example
```

```
function [ u, w ] = orbital( eta, rate, h, z )
%        [ u, w ] = orbital( eta, rate, h, z )
%
% This function evaluates the orbital velocities at specified vertical
% locations around the mean water level (MWL), from a known surface
% displacement time series.
%
%  eta = surface displacement time series,          [m] - matrix [n,1].
% rate = sampling rate of surface displacement,   [Hz] - scalar [1,1].
%    h = water depth (from MWL to the bottom),     [m] - scalar [1,1].
%    z = vertical locations of calculations,       [m] - matrix [n,1].
%        (z = 0 at MWL; z < 0 under MWL)
%
%    u = horizontal orbital velocities,          [m/s] - matrix [n,1].
%    w = vertical orbital velocities,            [m/s] - matrix [n,1].
%

% The computational method used is called Superposition.  It is based
% on the linear Gaussian wave model.  Ref: Donelan, M.A., Anctil, F.
% and Doering, J.C. (1992) "A simple method for calculating the velocity
% field beneath irregular waves." Coastal Engineering, v16.
%
% coding by F. Anctil, 1991.
%

%          VERIFICATION OF THE NUMBER OF INPUT ARGUMENTS.
%
if nargin ~= 4
  error( '4 input arguments are needed' ) ;
end

%          VERIFICATION OF THE LENGTH AND SIZE OF INPUT VARIABLES.
%
eta = eta(:) ; leta = length( eta ) ;
[ lz, wz ] = size( z ) ;

if leta ~= lz
  error( 'the input arguments eta and z should be of same length' ) ;
end

%          SURFACE DISPLACEMENT PHYSICAL PARAMETERS.
%
ETA = fft((eta-mean(eta)) ) ;                % Fourier components
leta2 = 1 + ( leta / 2 ) ;
ETA = 2 .* [ ETA(1:leta/2); ETA(leta2) ] ; % keep half of the comp.

A = abs( ETA ) ./ leta ;                     % amplitude, [leta2,1]
ph = -angle( ETA ) ;                         % phase

f = ( 0 : leta2-1 )' ./ leta .* rate ;       % frequency
om = 2 .* pi .* f ;                          % angular frequency
t = ( 0 : lz-1 )' ./ rate ;                  % time

k = wavek( f, h ) ;                          % wavenumber
relh = k * h ;                               % relative depth

%          BASE GROUND OF THE SUPERPOSITION METHOD
%
% Each wave component rides over the longer ones.  So, starting with
```

```
% the low frequency end of the spectrum, the influence of each Fourier
% component is considered one after the other, taking the previous
% total surface height as mean water level (mwl) for the new component.

%          INITIALIZATION OF THE MAIN VARIABLES
%
etatot = zeros( lz, 1 ) ;
surf   = zeros( lz, 2 ) ;
u      = zeros( lz, 1 ) ;
w      = zeros( lz, 1 ) ;

%          STEP BY STEP CONSTRUCTION OF THE ORBITAL TIME SERIES,
%          ONE FOURIER COMPONENT AT THE TIME.
%
for iom = 1 : leta2 ;
  phase = ph(iom) - om(iom) .* t ;
                                                  % surface displacement
  eta2 = A(iom) .* cos( phase ) ;
                                                  % SURFACE ORB. VEL.
  Au  = A(iom) * om(iom) .* cos(phase) ;
  Aw  = A(iom) * om(iom) .* sin(phase) ;
                                                  % deep water (relh>2*pi)
  if relh(iom) > 2*pi
    part = exp( k(iom) .* eta2 ) ;
    surf(:,1) = surf(:,1) + part .* Au;
    surf(:,2) = surf(:,2) + part .* Aw;
                                                  % intermediate water
  elseif relh(iom) ~= 0
    surf(:,1) = surf(:,1) + cosh( k(iom) .* ( eta2 + h ) ) ...
                ./ sinh( k(iom) .* h ) .* Au ;
    surf(:,2) = surf(:,2) + sinh( k(iom) .* ( eta2 + h ) ) ...
                ./ sinh( k(iom) .* h ) .* Aw ;
  end
                                                  % ORBITAL VELOCITIES
                                                  % for z < etatot
  ind = find( z < etatot ) ;
  if length(ind) > 0
    %
    % For the new component, mwl is the sum of all
    % previous components.  Correct z in accordance.
    %
    zz = z(ind) - etatot(ind) ;
                                                  % deep water
    if relh(iom) > 2*pi
      part = exp( k(iom) .* zz ) ;
      u(ind) = u(ind) + part .* Au(ind) ;
      w(ind) = w(ind) + part .* Aw(ind) ;
                                                  % intermediate water
    elseif relh(iom) ~= 0
      u(ind) = u(ind) + cosh( k(iom) .* ( zz + h ) ) ...
               ./ sinh( k(iom) .* h ) .* Au(ind) ;
      w(ind) = w(ind) + sinh( k(iom) .* ( zz + h ) ) ...
               ./ sinh( k(iom) .* h ) .* Aw(ind) ;
    end
  end
                                                  % ORBITAL VELOCITIES
                                                  % z >= etatot
  ind =find( z >= etatot ) ;
  lind = length( ind ) ;
```

```
  if lind > 0
    u(ind) = surf(ind,1) ;
    w(ind) = surf(ind,2) ;
  end

                                                    % add surface displ.
                                                    % to previous sum
  etatot = etatot + eta2 ;
end

%          ELIMINATE VALUES FOR POSITIONS ABOVE THE WATER SURFACE
%
ind = find( z > etatot ) ;
lind = length( ind ) ;
if lind > 1
  u(ind) = zeros( lind, 1 );
  w(ind) = zeros( lind, 1 );
elseif lind > 0
  u(ind) = 0;
  w(ind) = 0;
end
```

```
function F=dhhfrqsp(om,omp,Uccp)
%        F=dhhfrqsp(om,omp,Uccp)
%
%  This function CONSTRUCT A FREQUENCY SPCTRUM according to
%     the work of Donelan, Hamilton & Hui (1985), i.e. for
%     deep-water wind-generated waves.
%
%  F    - frequency spectrum, in [m*m/(rad/s)]    - vector
%  om   - omega, angular frequency, in [rad/s]    - vector
%  omp  - peak angular frequency of S             - scalar
%  Uccp - wind forcing parameter (Uc / cp)        - scalar
%         where Uc - component of the 20 m wind
%                    at omp, in [m/s];
%                cp - corresponding phase speed,
%                    in [m/s].

%  Donelan, M.A,, Hamilton, J. & Hui, W.H. 1985. Directional
%     spectra of wind-generated waves. Philosophical Transaction
%     of the Royal Society of London, A315, 509-562.
%
if Uccp < 0.83  |  Uccp > 5
  disp('This theory applies only to 0.83 < Uccp < 5')
end
om= om(:)'; lom= length(om); g= 9.81;
%
% construction of the FREQUENCY SPECTRUM (F)
%
% gamma - peak enhancement factor.
% sigma - peak width factor.
% alpha - equilibrium range
% gam_exponent
%
if Uccp < 1
  gamma= 1.7;
else
  gamma= 1.7 +6.0 *log10(Uccp);
end
sigma =0.08 *(1 +4 /Uccp^3);
alpha =0.006 *Uccp^0.55;

%
% note: there is a bug in MATLAB for handling exponential of values
%       -709 to -745.  Knowing that exponential results tend to zero
%       for large negative values, we set here zero has results for
%       any computations involving values less than -50.
%       exp(-50) =1.92874e-22
%
gam_exp_part =-(om -omp).^2 /(2 *sigma*sigma *omp*omp);
gam_exponent =zeros(1,lom);
ind =find( gam_exp_part > -50 );
lind =length(ind);
if lind >= 1
    gam_exponent(ind) =exp(gam_exp_part(ind));
end
%
F= alpha * g*g *om.^(-5) .*(om/omp) .*exp(-(omp*ones(1,lom)./om).^4) ..
   .*(gamma*ones(1,lom)).^gam_exponent;

plot(om,F);
```

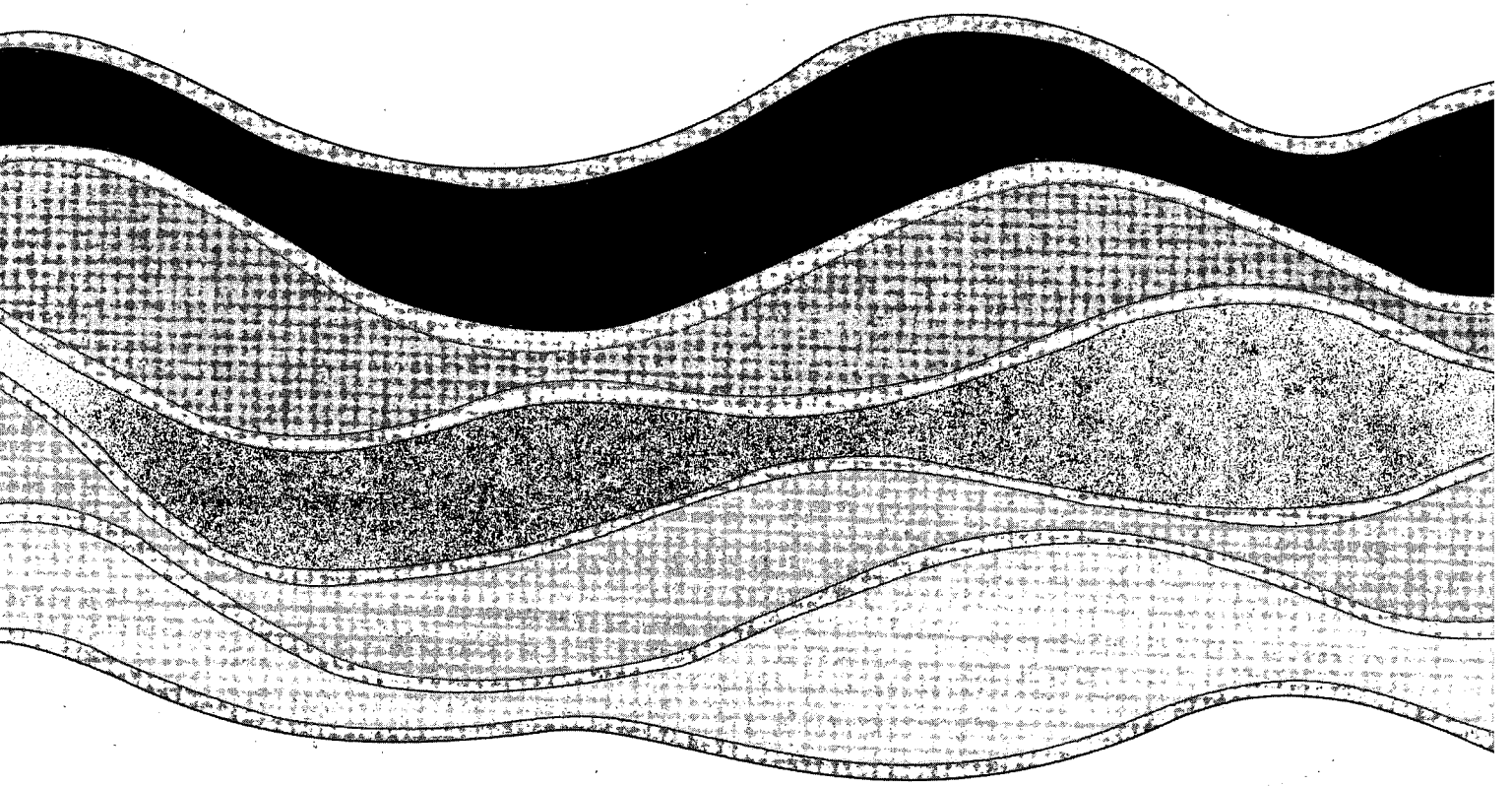
```
title('dhh wave frequency spectrum');
xlabel('frequency - om [rad/s]        Hit enter to continue');
ylabel('[m*m/(rad/s)]');

dom=om(2)-om(1);
sight=4*(sum(F*dom)).^0.5;
s=sprintf(' sig height = %5.2f m',sight);
text(0.6,0.87,s,'sc');
pause;
```

```
function eta=SURFACE1(S,om,x,t,h)
S = S';
%
%        eta=SURFACE1(S,om,x,t,h)
%
%        WAVE SURFACE, NO APPROXIMATION
% The function calculates surface displacement from a one-dimensional
% (frequency) wave spectrum S using random phases. S is a matrix.
% The dimension of the spectrum is m*m/(rad/s)
%
% om=  vector of angular frequencies. length(om)=length(S)
%
% Surface displacement is calculated at points defined by vector x
% at time t. If lenght(t) is 1 then the vector x defines the domain.
% If t is a vector, then the surface is calculated at points (x(i),t);
%
% In case of unevenly spaced omega, SURFACE1 can be called also by
%
%          eta=SURFACE1(S,[om0;om],x,t,h)

% om0= the lower limit of the first frequency band.
%      To be used when omega is not evenly spaced. If omega0 is
%      omitted when omega is unevenly spaced a small error will
%      result.
%
%  WARNING !  It is assumed that omega is the unweigted mean of the
%             frequency band eg. om(1)=(om0 + upper limit)/2.
%             If om0 is used but it is not exactly correct, results
%             will be completely wrong!!!
%

rand('seed',0) % seed is determined here, change if necessary
g=9.81;
m = length(S);
om=om(:);
lom=length(om);

% create difference matrix
if lom==m
   if m==1
      dom=1;
   else
      dom=diff(om);
      dom=[dom(1);dom];
   end
elseif lom==m+1
   oml=om(1);
   om(1)=[];
   for i=1:m
     dom(i,1)=2*(om(i)-oml);
     oml=oml+dom(i);
   end;
else
   error('Spectrum and omega incompatible')
end;

d=2*dom;
A=sqrt(S.*d);
```

```
k= wavek(om/2/pi,h);
Nx=length(x); T=length(t);
if length(om) > 1
   omphase = 2*pi*rand(om);
else
   omphase = 2*pi*rand(1);
end;

   for j=1:Nx
      phase=k*x(j)+omphase; disp(j);
      for i=1:T
         eta(i,j)=sum(sum(A.*cos(phase-om*t(i))));
      end;
   end;
end;
```

```
function k = wavek( f, H, tol )
%        k = wavek( f, H, tol )
%
% Returns a column vector of wavenumbers k (1/m) at frequencies f (Hz)
% using the linear wave dispersion relation for water depth H (m).  "tol"
% is an optional error bound on the predicted f's (the default is 1e-4).
% Note: f must increase monotonically from small to large frequencies.
%

% coding by Eugene Terray, 1990.
%
                                % some initial housekeeping

if nargin<3,  tol = 1e-4;  end          % default tolerance
g = 9.80171;   c = 2*pi*sqrt(H/g);      % g at 40N (m/s^2)
f = c * f(:);  Nf = length(f);          % non-dimensionalize f
k = zeros(Nf,1);
                                % use approximations for large & small f

f_r = 1.1*sqrt( abs(log(tol/2))/2 );    % define f LARGE if error > 110% * tol
nr = find( f >= f_r ) ;                 % if so, use the limiting deep-water
if isempty(nr)                          % dispersion relation  k = f^2
    nr = Nf+1 ;
else
    k(nr) = ( f(nr).^2 ) ;
    nr = min( nr ) ;
end

f_l = 0.7*( tol * 604800/281 )^(1/8);   % f is SMALL if error > 70% * tol
nl = find( f <= f_l ) ;                 % if so, use shallow water expansion
if isempty(nl)                          % k = f*(1+f^2/6+11f^4/360+17f^6/5040)
    nl = 0 ;
else
    k(nl) = f(nl) .* polyval([17/5040,11/360,1/6,1], f(nl).^2) ;
    nl = max( nl ) ;
end

%  fprintf('\n Non-linear Range: %g < f < %g (Hz)', f_l/c, f_r/c)
%  fprintf('\n                   %g < n < %g  \n', nl, nr)

                                % calculate the nonlinear k(f) regime by solving
n = (nl+1):1:(nr-1) ;           % f^2 - k*tanh(k) == 0 using Newton-Raphson
if ~isempty( n )
    dk = 2*tol ;
    k(n) = f(n).^2 ;                    % use deep water limit as initial guess

    while max( abs(dk) ) > tol          % Newton-Raphson iteration loop
        t = tanh( k(n) ) ;
        dk = -(f(n).^2 - k(n).*t) ./ ( t + k(n).*( 1 - t.^2 ) ) ;
        k(n) = k(n) - dk ;
    end
    k(n) = abs( k(n) ) ;                % f(k) = f(-k), so k>0 == k<0 roots
end

N = min( find( k>50 ) ) ;               % diagnostics: inform user if err > tol
if isempty(N),  N = Nf;  end
err = abs( f(2:N) - sqrt( k(2:N).*tanh(k(2:N)) ) )./f(2:N) ;
if max( err ) > tol,  fprintf('\n WAVEK: error exceeds %g \n', tol),  end
%  fprintf('\n max error = %g \n', max( err )),  plot(err,'w')

k = k / H ;                             % finally, give k dimensions of 1/meter
```

```
function y=sech(x)
%        y=sech(x)
%
% This function evaluates the HYPERBOLIC SECANT (sech) from
%   exponential functions.

% Abramowitz M. & Stegun I.A. 1964. Handbook of Mathematical Functions.
%   National Bureau of standards, AMS55, Washington (section 4.5).
%
y= 2 ./(exp(x) +exp(-x));
```



NATIONAL WATER RESEARCH INSTITUTE
P.O. BOX 5050, BURLINGTON, ONTARIO L7R 4A6

Environment Canada Environnement Canada

Canada

INSTITUT NATIONAL DE RECHERCHE SUR LES EAUX
C.P. 5050, BURLINGTON (ONTARIO) L7R 4A6