EMS wave logger data processing

Introduction

Waves can be measured in several ways. One way of measuring waves is by measuring the wave pressure at a certain depth using a pressure sensor and calculate the wave information from the pressure record. The EMS wave logger[1] uses a Honeywell MLH 050 PGP 06A pressure sensor. The information is stored by the logger on a SD card. The software in the logger controls the sample durations (from 1 to 30 minutes) and the sample intervals (from 15 min to 3 hours). The sampling rate is fixed to 4 Hz.

Wave pressure

In a regular, small amplitude wave the instantaneous water level is given by:

$$\eta = a \sin\theta \qquad (1)$$

Under a regular, small amplitude wave the pressure is given by:

$$p = -\rho g z + \rho g a \frac{\cosh k(h+z)}{\cosh kh} \sin\theta \qquad (2)$$

in which:

η	instantaneous water level (m)
p	pressure at the requested location (Pa)
ρ	density of the water (kg/m^3)
g	acceleration of gravity (m/s^2)
z	depth of the pressure (note: under water z is negative) (m)
a	amplitude of the wave (m)
k	wave number = $2\pi/L$
θ	phase angle of the wave (rad)
L	local wave length (m)

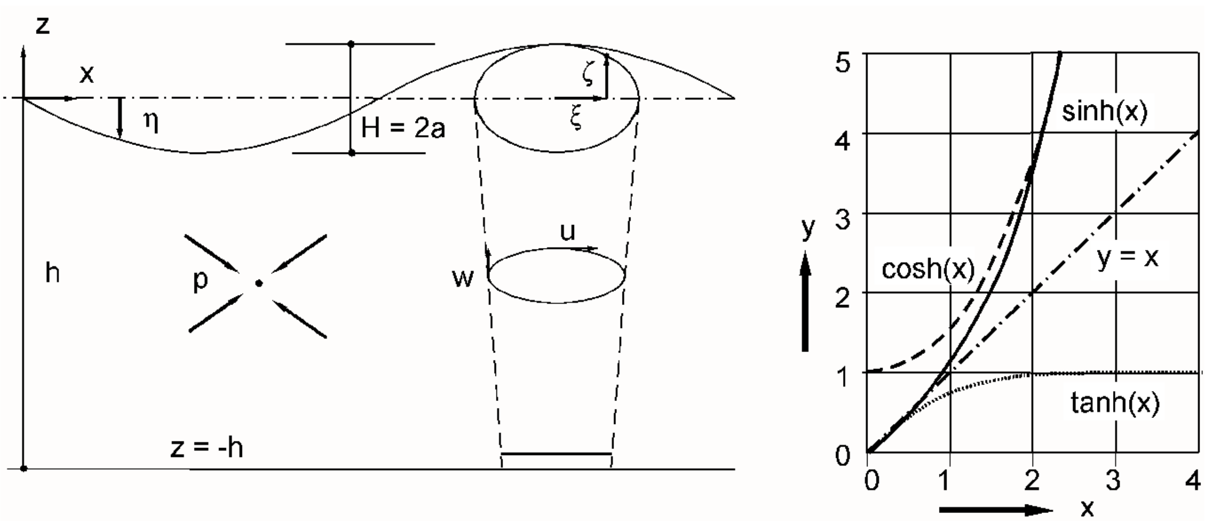


Figure 1: definitions in the wave pressure equation.

[1] The wave logger is produced by Environmental Mapping & Surveying, Durban, South Africa; see also Appendix 1.

When inserting eq. (1) into eq. (2) leads to:

$$p = -\rho g z + \rho g \eta \frac{\cosh k(h+z)}{\cosh kh} \tag{3}$$

In order to do this correct, the density of water needs to be known. This parameter (*Rho*) has to be set in the script. For fresh water ρ=1000, for sea water one should raise ρ up to 1030 kg/m^3.

Subtracting the hydrostatic pressure leads to:

$$\begin{aligned} \Delta p &= \rho g \eta \frac{\cosh k(h+z)}{\cosh kh} \\ \eta &= \frac{\Delta p}{\rho g} \frac{\cosh kh}{\cosh k(h+z)} \end{aligned} \tag{4}$$

In the Matlab script formula (4) is used to calculate the surface elevation. However, the parameters used are *Depth* and *Bottom*. *Depth* is the calculated depth of the sensor below the average surface and *Bottom* is the height of the sensor above the bed.
So:

h = *Depth* + *Bottom*
(*h*+z) = *Bottom*

And in case the pressure is measured at the bottom, this reduces to:

$$\eta = \frac{\Delta p}{\rho g} \cosh kh \tag{5}$$

Eq. (5) can be used to calculate the surface elevation from the pressure record. However, one should realise that this derivation is only valid for regular, small amplitude waves.

In shallow water usually waves cannot be considered small amplitude waves, and second order theory should be applied to describe the water surface elevation. The main difference between first order and higher order waves is the exact shape of the surface. Wave height and wave period are not very different. Because of this, first order wave theory is acceptable for this purpose.

In important drawback of determining wave heights from pressure records is that small, short waves cannot be measured at larger water depths.

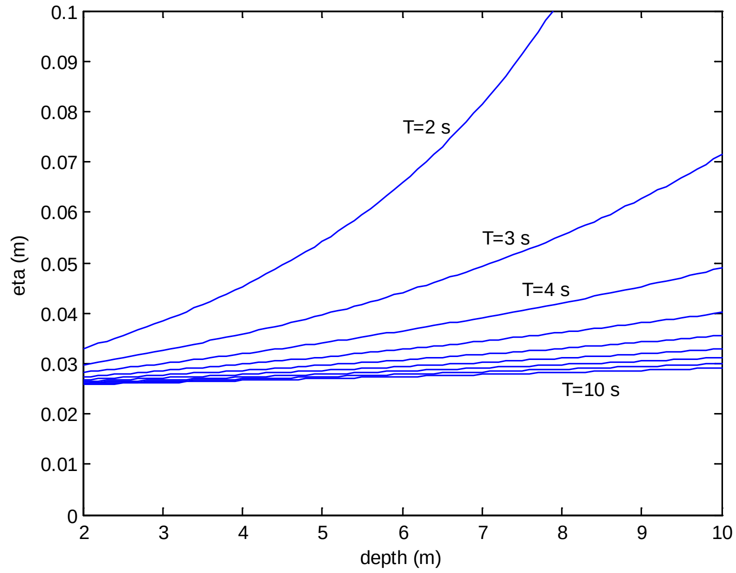


Figure 2: accuracy of the calculated water elevation as a function of water depth and wave period

The accuracy of the wave logger is in the order of 250 Pa. In fig. 2. the accuracy of the calculated water elevation is plotted as a function of the water depth and the wave period.

It is clear from the figure that in larger water depths the accuracy decreases significantly for the shorter wave periods. In general one may conclude that up to a water depth of 10 waves with a period of 5 seconds and larger are quite accurate.

Irregular waves (time domain)

In case of irregular waves the above theory is still valid, but cannot be applied without restrictions.

A real surface record may look like figure 3.

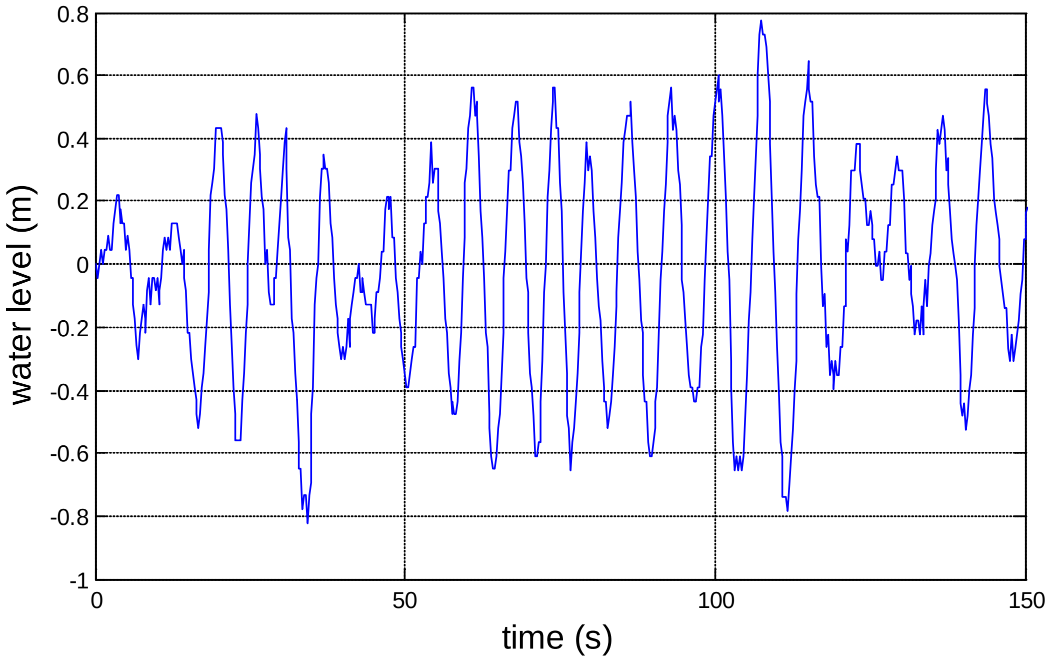


Figure 3: example of a surface water level record

As can be seen all individual waves have different periods and heights. To calculate back from a pressure record to a water surface level the individual wave periods should be known. In theory one could split a pressure record into individual waves and then calculate the resulting surface elevation for each individual wave. However this usually leads to large errors with shorter waves. Therefore one can better first calculate the average wave period in the whole record (i.e. the total observation period divided by the number of waves) and use that wave period for the calculation.
When doing so, the short waves in the wave field are not really accounted for. But these waves are also the lower waves, and therefore usually not the most relevant part of the wave height distribution.

For the time domain analysis therefore a surface elevation is generated using eq. (5) and the mean period of the observation period.
With a counting algorithm then the individual waves are distinguished in this generated surface elevation record. To separate the waves the downward crossing method is used. From each wave the height is determined. Subsequently the waves are sorted on individual height. In the Matlab script the array with sorted wave heights is called "Hsorted2". The number of waves found is called "counter". In this report the following notation is used:

N	Number of waves
H_i	individual wave
H_{mean}	average wave height
H_{rms}	root mean square wave height
$H_{1/3}$	Significant wave height

In the next step the time domain variables are determined:

$$H_{mean} = \frac{1}{N} \sum H_i \tag{6}$$

$$H_{rms} = \frac{1}{N}\sqrt{\sum H_i^2} \tag{7}$$

$$H_s = H_{1/3} = \frac{1}{N/3}\sum_{\frac{2}{3}N}^{N} H_i \tag{8}$$

Also are determined the wave height exceeded by 2%, 1% and 0.1% of the waves. These wave heights are determined in three ways:

1. Directly from the observations;
2. Using a Rayleigh distribution;
3. Using the Composite Weibull Distribution according to Battjes and Groenendijk [2000].

With method 2 a Rayleigh distribution of the waves is assumed. This assumption is usually correct in deep water. For the Rayleigh distribution is valid:

$$\begin{aligned} H_{2\%} &= 1.40 H_s \\ H_{1\%} &= 1.50 H_s \\ H_{0.1\%} &= 1.85 H_s \end{aligned} \tag{9}$$

The equation for the Rayleigh distribution is:

$$P\{\underline{H} > H\} = \exp\left[-1\left(\frac{H}{H_s}\right)^2\right] \tag{10}$$

In shallow water the highest waves in the record are already broken. Therefore the Rayleigh distribution is not valid any more. Battjes and Groenendijk [2000] suggested a approximation of this distribution using:

$$\Pr(\underline{H} \le H) = \begin{cases} F_1(H) = 1 - \exp\left[-\left(\dfrac{H}{H_1}\right)^2\right] & H \le H_{tr} \\ F_2(H) = 1 - \exp\left[-\left(\dfrac{H}{H_2}\right)^{3.6}\right] & H > H_{tr} \end{cases} \tag{11}$$

Below a transitional value of the wave height (H_{tr}), the Rayleigh distribution remains valid. Above this value the exponent in the distribution has a different value (3.6). The values H_1 and H_2 were fitted from measurements and tabulated in the original paper. These values depend on the water depth and the bed slope. Figure 4 below shows the ratio $H_{x\%}/H_{rms}$ for various values of H_{tr}/H_{rms}.

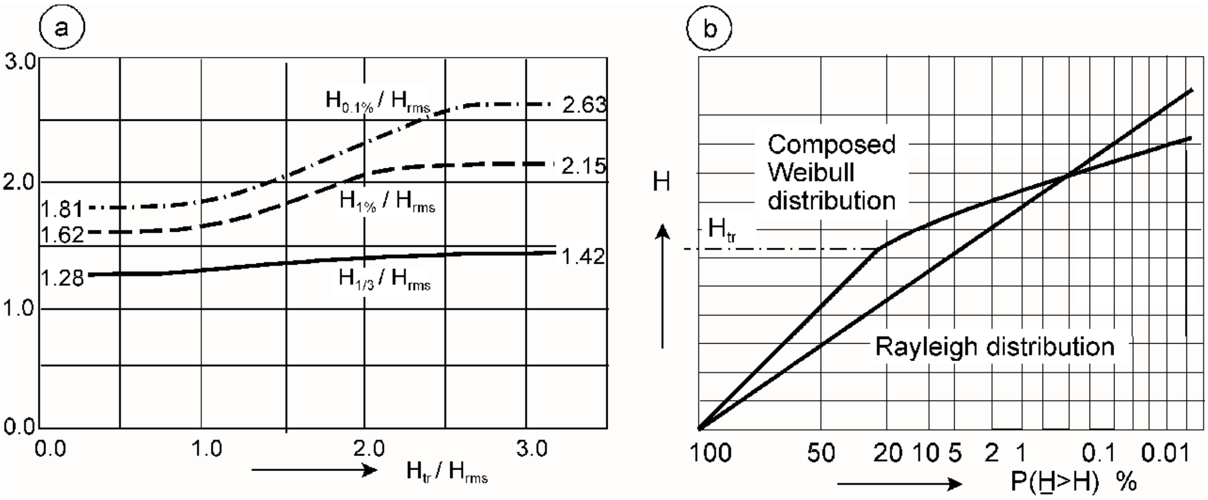


Figure 4: Wave height distribution in shallow water (Battjes/Groenendijk, 2000)

The parameters from the Composite Weibull Distribution are loaded into the script from the matlab script BG.

Note that in case of a limited number of waves in the record (<1000), no observed value of $H_{0.1\%}$ can be determined. The script gives then as output NaN (Not a Number).

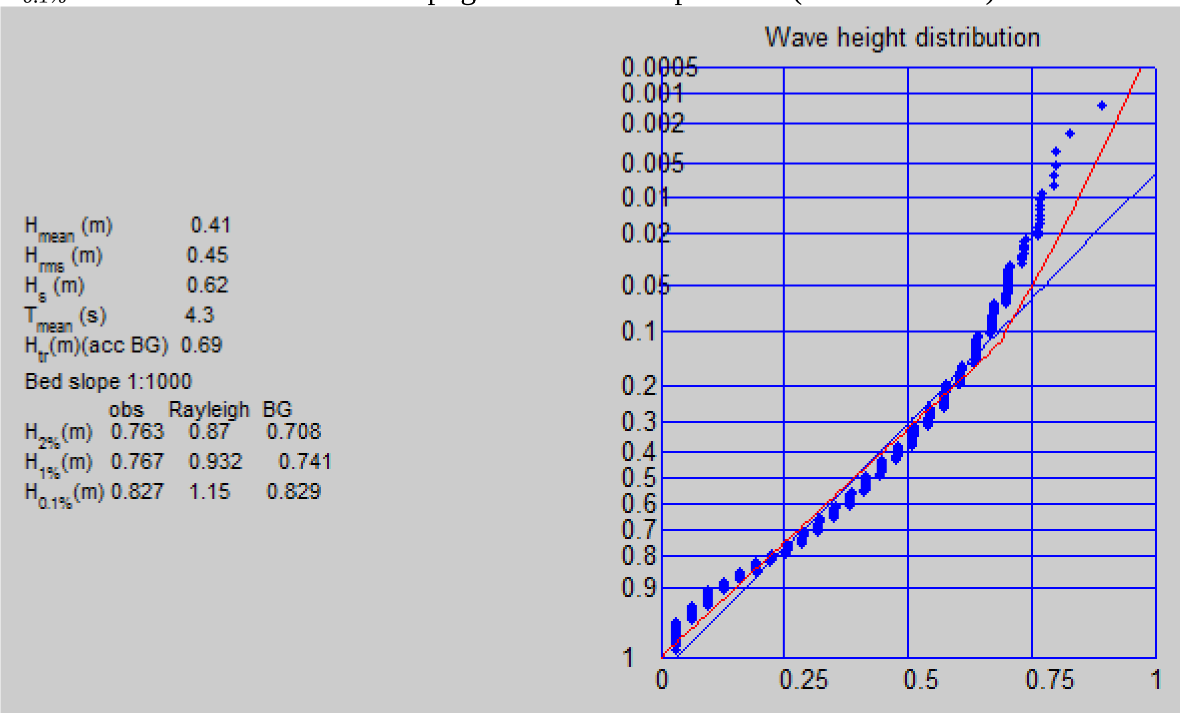


Figure 5: example of the output of a wave height distribution

In the output of the script the blue line is the Rayleigh fit, the red line the Battjes-Groenendijk fit and the blue crosses the observed data. This dataset was observed on a natural beach in very shallow water (1.9 m).

The extreme wave heights in the record are relevant for the design of coastal structures. The stability of armour units depends on the highest values in the record. Assuming a Rayleigh distribution usually gives a significant overestimation in relatively shallow water, leading to a too heavy structure. Note that with a frequency domain analysis (see below) one cannot determine these highest waves in the record.

Irregular waves (frequency domain)

The time domain analysis is not very accurate in determining the periods of the wave. Also it does not give information on the shape of the wave spectrum. Therefore a frequency domain analysis is needed. In principle two methods are possible:

1. Determine the surface elevation of the water and apply spectral analysis on this surface elevation;
2. Determine the spectrum of the pressure and calculate from the pressure spectrum the surface elevation spectrum (wave energy spectrum).

Method 1 is standard in case the surface elevation was measured directly (e.g. with wave buoys or step gauges). However, when applying this method for pressure records, inaccuracies are included because of the fact that the calculated surface elevation is smoothed and does not contain all period information any more.
Therefore for pressure data method 2 is to be preferred.

In the script the pressure record is processed by the script crosgk. This script, developed by Klopman of Deltares, determines the power cross spectrum using a Fast Fourier Method. This results in a pressure spectrum:

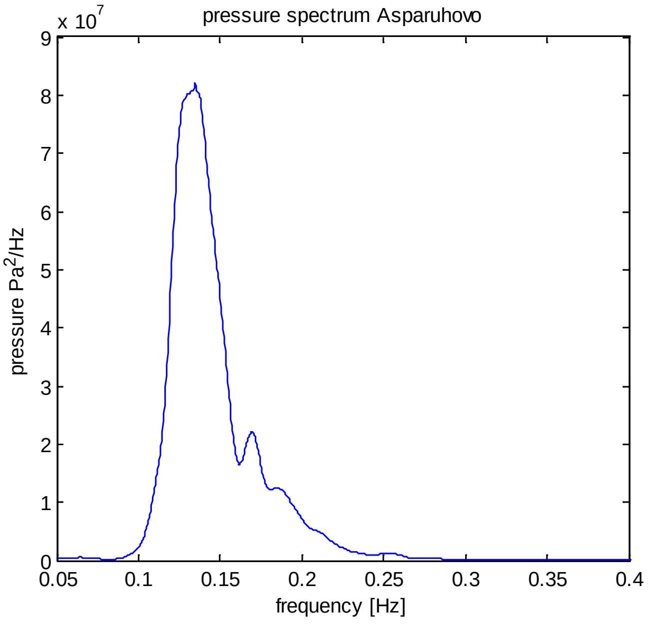


Figure 6: A pressure spectrum derived from the pressure data

A certain degree of smoothing of the spectrum is recommended. This value can be set by changing the parameter M in the script. M=0 is no smoothing. Practical values for M are between 20 and 100.
For each frequency bin one can calculate from the pressure value the elevation value (on the vertical axis is the pressure in Pa^2/Hz, in the wave energy spectrum we need m^2/Hz). This means that the same formula can be used as was applied in the time domain analysis (eq. (5)).

Now the exact value for the period is known, because that is given by the number of the frequency bin.
This process results in the energy spectrum:

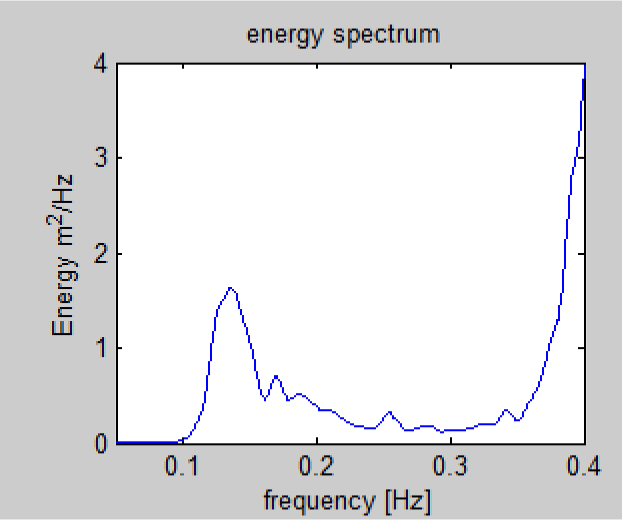


Figure 7: The energy spectrum calculated from the pressure spectrum of figure 6.

It is clear that the peak at the right side of the spectrum is an artefact. It is caused by the fact that a small value of the pressure is multiplied with an extremely high multiplier. This has no physical meaning. The above data were measured on a water depth of 8.5 m. Small variations in the pressure (less than the accuracy of the meter) are interpreted by eq. (5) as being caused by huge waves of very short periods. While in reality these variations are only random noise.

Therefore the spectrum should be cut off in this case at a value of approximately 0.28 Hz (see also figure 6). With the parameters “cutoff” and “high” this is realized. The variable “cutoff” limits the analysis, while “high” only changes the axis of the plot. See figure 8 for the result.

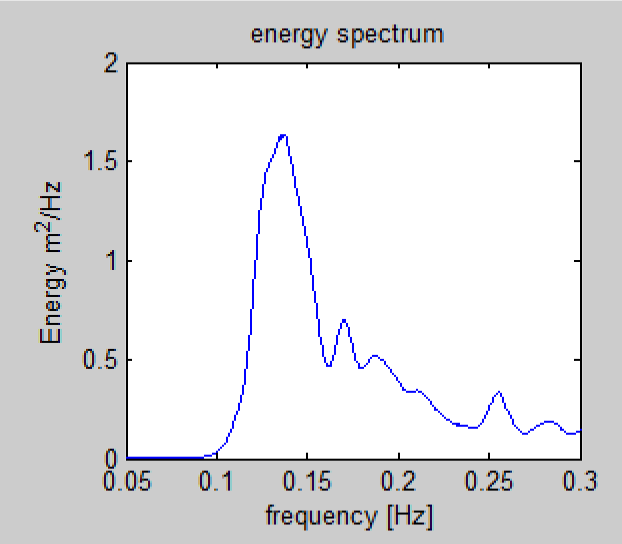


Figure 8: corrected wave energy spectrum using the parameters “cutoff” and “high”

One can determine the values for “cutoff” and “high” by using the plot of the pressure spectrum. This plot is not printed on default, only when the parameter “extraplot” is set to 1.

There is also a value “low”. This value should be used for deleting long periodic waves which have nothing to do with the waves to be analysed (e.g. seiches).

From the spectrum various moments are computed. A moment of a spectrum is the product of arm and area. For higher order moments, one raises the arm to a certain power. The zero-th order moment is the area multiplied with the arm to the power zero, which is in fact only the surface area. The zero-th order moment gives the total energy in the spectrum. The general equation of the moment is:

$$m_n = \int_0^\infty f^n S(f)df \tag{12}$$

The zero-th and the 1st order moment are:

$$\begin{aligned} m_0 &= \int_0^\infty S(f)df \approx \frac{1}{2}\sum_{i=1}^{N} e_i^2 \\ m_1 &= \int_0^\infty f \cdot S(f)df \approx \frac{1}{2}\sum_{i=1}^{N} f \cdot e_i^2 \end{aligned} \tag{13}$$

The script calculates m_{-1}, m_0, m_1 and m_2. From these moments the following values are calculated:

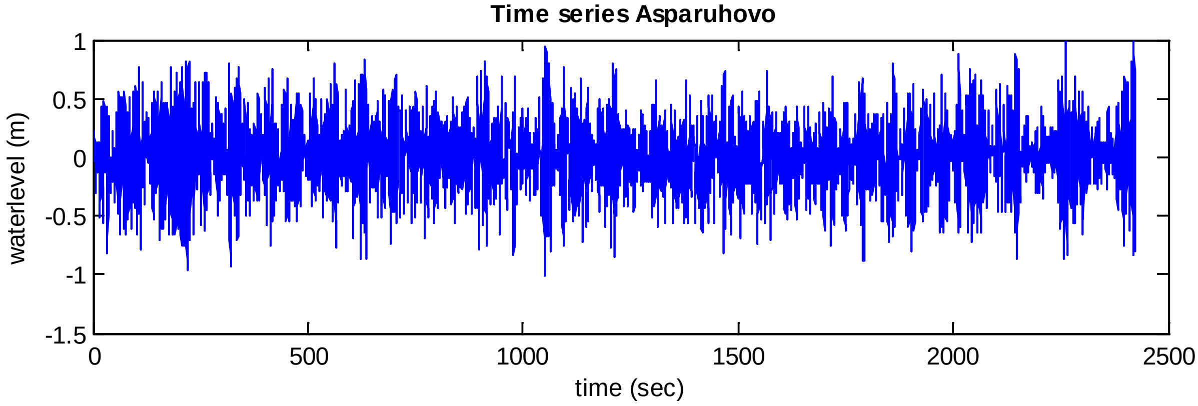
$$\begin{aligned} H_{rms} &= \sqrt{8m_0} \\ H_{m0} &= 4\sqrt{m_0} \end{aligned} \tag{14}$$

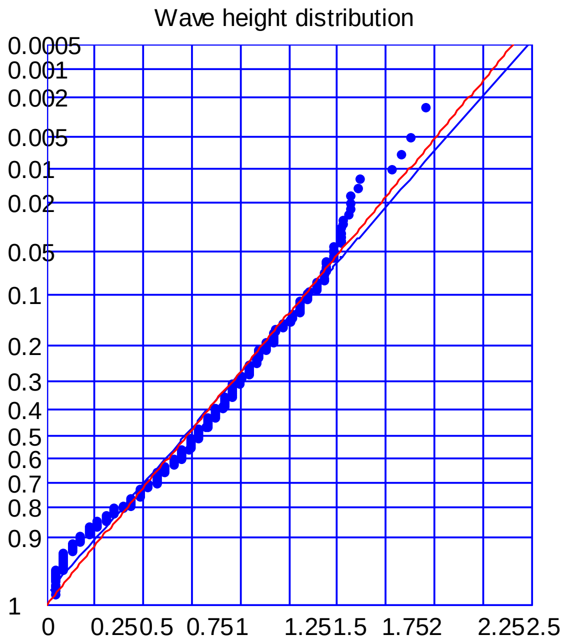
and

$$\begin{aligned} T_m &= \sqrt{\frac{m_0}{m_2}} \\ T_{0,1} &= \frac{m_0}{m_1} \\ T_{-1,0} &= \frac{m_{-1}}{m_0} \end{aligned} \tag{15}$$

One should expect that H_{m0} and $H_{1/3}$ are equal. If this is not the case, one should verify if the settings of the cutoff parameters.

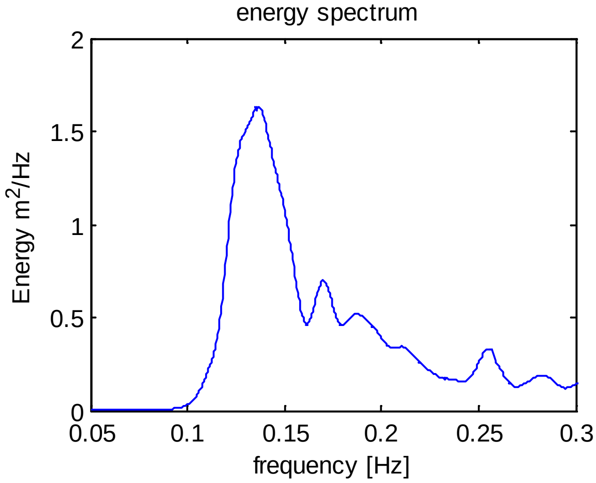
An overview of all standard output of the script is given on the next page.





H_{mean} (m) 0.77
H_{rms} (m) 0.87
H_s (m) 1.2
T_{mean} (s) 6.1
H_{tr}(m)(acc BG) 3
Bed slope 1:1000

	obs	Rayleigh	BG
$H_{2\%}$(m)	1.56	1.72	1.73
$H_{1\%}$(m)	1.61	1.84	1.88
$H_{0.1\%}$(m)	NaN	2.27	2.3



H_{rms} (m) 1.2
H_{m0} (m) 1.7
T_m (s) 3.5
$T_{m0,1}$ (s) 3.8
$T_{m-1,0}$(s) 4.8
T_{peak}(s) 2.5

Average depht (m) 8.5
water density (kgm^3) 1018

Calibration

The pressure sensor of the wave logger has been calibrated in a deep water pit. From this calibration was the following relation found:

$$p = 254V - 43250 \quad \text{(Pa)} \tag{16}$$

In this formula a correction is included for the atmospheric pressure. Variations in the atmospheric pressure have influence on the results, but this effect is very small.
V is the reading from the wavelogger.

The wavelogger has been applied also during one of the Bardex experiments at the large wave flume of Deltares, Netherlands (the Deltaflume). In this flume prototype size waves can be generated. The flume is equipped with standard high performance wave gauges with a very high sampling interval (20 Hz). Appendix 2 shows a comparison of the data from the EMS wavelogger, processed with the TU Delft script and the results from Deltares equipment in the flume.

User Manual

The data are processed by the Matlab script Wavelogger.m. This script processes files with one observation only. The Wavelogger produces one single datafile with all observations in sequence. The first step in processing is splitting the data from the logger in separate files. This can be done by any program. A nice tool is the program TextSplitter.exe.
Load the output file from the Wavelogger into this program, enter the number of observations and let te program slit the datafile into separate chunck of file. When the file is large, this may take several minutes; be patient. The number of observations can be found at the end of the file. The observation number is the number after $D,.
It is recommended to use filenames without an underscore ('_'), because this symbol makes that letters are printed as subscript in the Matlab script.
All observations from one ensemble should have the same name, and a different number, e.g.: Logger2-012Part1.txt; Logger2-012Part2.txt; Logger2-012Part3.txt;…..etc

Individual observations can be processed by the Matlab scripts wavelogger.m.
In line 12 and 13 one should NofFiles to 1.
Then in lines 58 to 72 the other input parameters can be set.

In case of processing multiple observations set NofFiles to the number of observations. If one intends to skip the first files (because they do not contain information), set the StartFile to a higher number. This might be useful when the Wavelogger already started to observe before it was placed on its final location.
In case of multiple observations no plots of individual observations are made (like spectra and exceedance lines). In the command window the number of processed file is shown, so one can follow the progress of the calculation.

After completion of all processing the data are written to an outputfile. This file has the same name as the inputfiles of the observations (but without the sequence number) and the extension .mat (e.g. Logger2-012Part.mat)

One can plot the results with the Matlab script PlotSeries.m. In this script one can set the first point to be plotted (startpunt) and the last point (eindpunt). This means that if one has a record of 600 points and on intends to plot the series from 10 to 595, startpunt=10 and eindpunt=5.

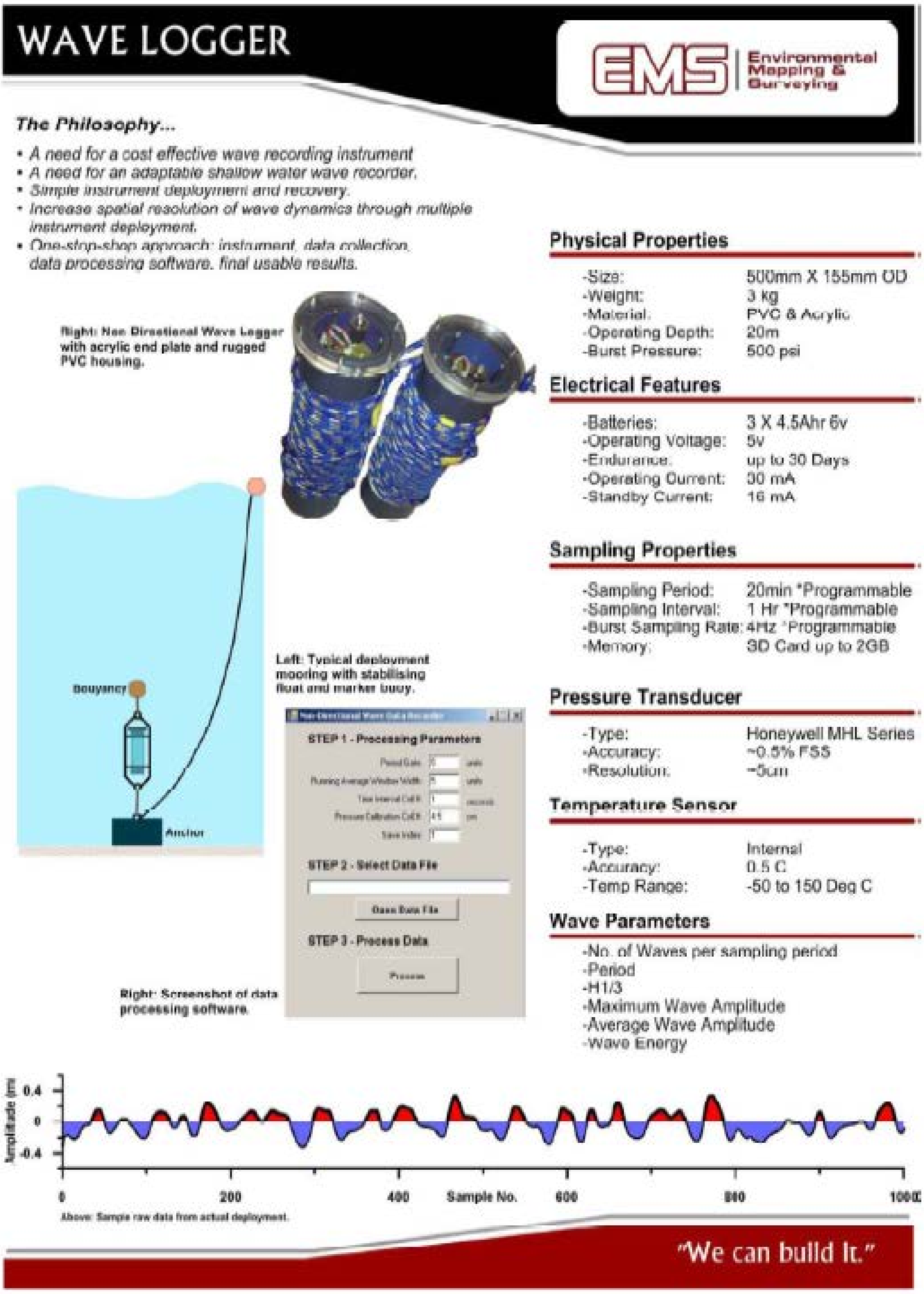
Errors

In some cases the time domain data are plotted, but the frequency domain data are not, or are printed incorrectly. In that case one should wonder if the length of the record is sufficient. One should have preferably at least 100 waves in a record.

References

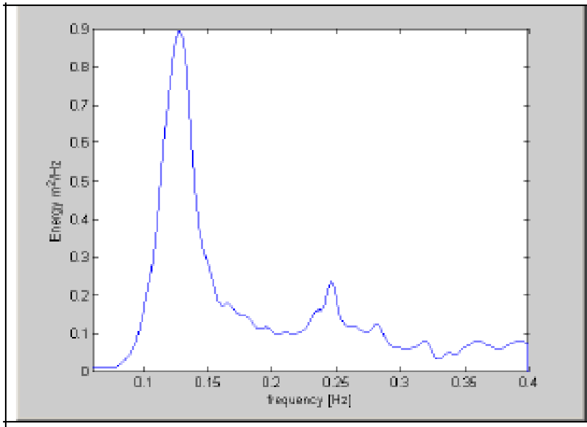
BATTJES, J.A., GROENENDIJK, H.W. [2000] Wave height distribution on shallow foreshores, *Coastal Eng. Vol 40, pp 161-182*.

Appendix 1 Description of the wave logger

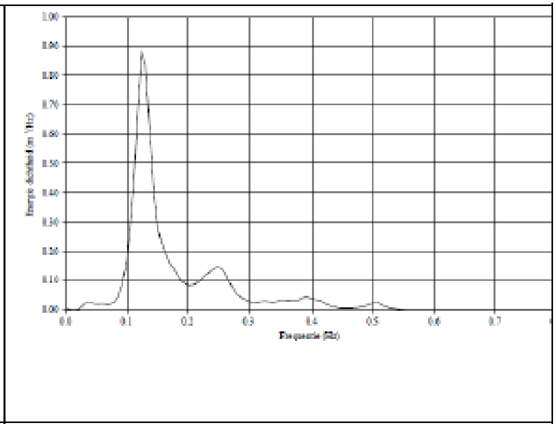


Appendix 2: Wave logger calibration

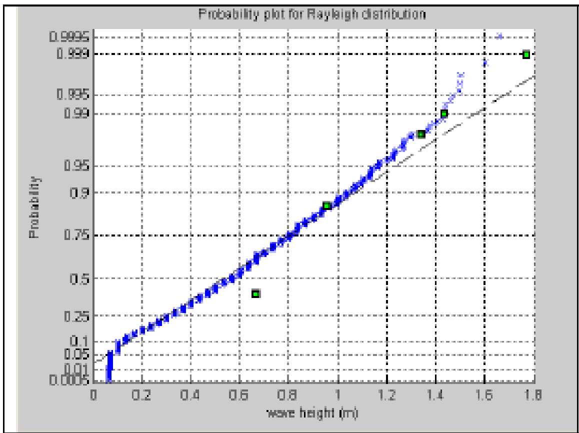
Calibration test EMS wave logger - 7 June 2012 - Deltaflume, Deltares



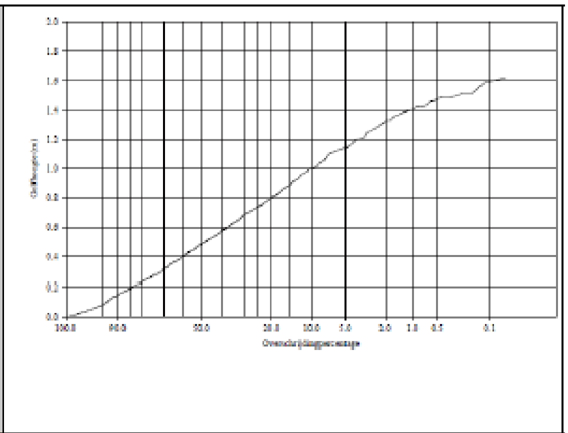
Measured spectrum with EMS wave logger



Measured spectrum according to Deltares equipment



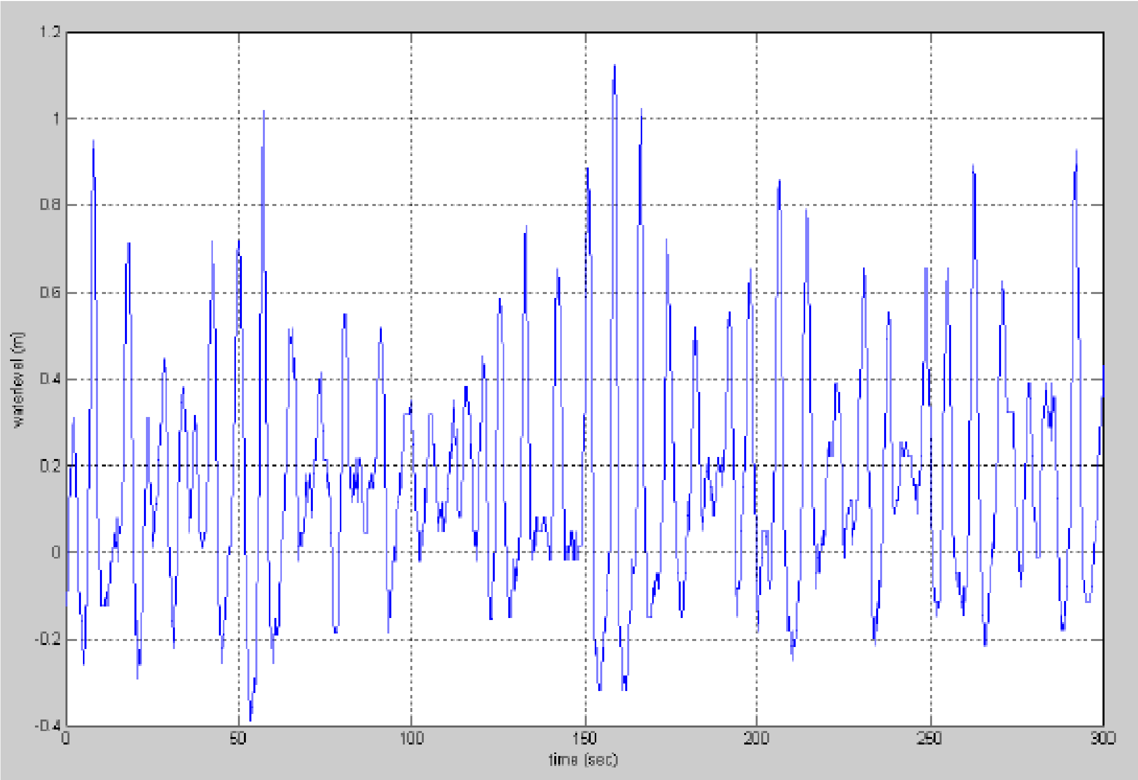
Measured exceedance line with EMS wave logger



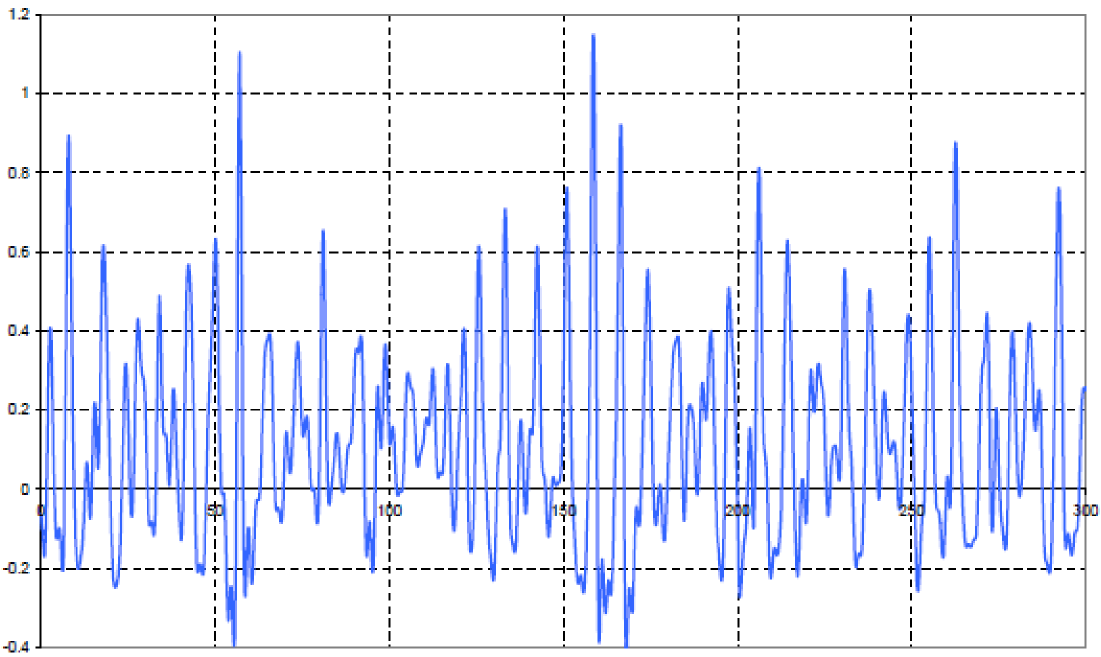
Measured spectrum according to Deltares equipment

measured unit	EMS wave logger	Deltares Equipment
Hm0	0.93	0.93
H1/3	0.96	0.94
H2%	1.34	1.38
Hmax	1.66	1.79
T01	5.47	5.53
T02		4.88
Tm-1,0	6.50	7.90
Tpeak	7.87	7.94
Sampling frequency	4 Hz	20 Hz

Calibration test EMS wave logger - 7 June 2012 - Deltaflume, Deltares



Wave surface elevation measured with EMS wave logger



Wave surface elevation measured with Deltares equipment
The measurements by EMS wave logger and Deltares Equipment were not at exactly the same location.

Appendix 3: Matlab scripts

Wavelogger.m

```
% Script for processing pressure data from the EMS wavelogger
% Script developed by H.J. Verhagen, Delft University of Technology
% January 2013
clear;
clc;
%define basic variables
Rho=1018;
g=9.81;
Avolt=254;     %Calibration value A of the pressure meter
Bvolt=-43250;  %Calibration value B of the pressure meter

TanAlfa = 0.001;   %bedslope

load Asparuhovo_121002_1.txt;         %fill in filename to be processed
rawdata=Asparuhovo_121002_1;          %fill in filename without .txt
heading='Asparuhovo';   %fill in any text as heading of the figure

% additional input data for spectrum
%===================================
M = 50;        % higher values of M give more smoothing of the spectrum
               % low value for M is 20, high value for M is 100
high = 0.3;    % highest frequency for plots  default high=0.4
low = 0.05;    % lowest frequency for plots   default low =0.05
cutoff=0.2;    % parameter to regulate end of spectrum, to be calibrated
               % from the pressure spectrum default cutoff =0.2
upgrade=1;     %calabration coeffectien for transformation pressure to
height

extraplot = 1; % extraplot = 1 gives additional plots
interval=.25;  %interval is the sample frequency interval of the sensor
               % do not change this value unless another wavelogger is used

%transformation from Volts to pressure using calibration constants Avolt
%and Bvolt
%Avolt has dimension Pa/V, Bvolt has dimension Pa
P=Avolt*rawdata+Bvolt;

n=numel(rawdata);      %count number of samples
tottime=n*interval;    %calculate total duration of observation in sec
ttime=(interval:interval:tottime); %create array with time
time=ttime.';           %change orientation of matrix
% calculate regression coefficients to compensate for change in waterlevel
% during the observations
%plot(time,rawdata);
%plot(time,P);
BB=polyfit(time,P,1); %regression analysis to determine real waterdepth at
any moment
                      %and correct for hydrostatic pressure
Intercept=BB(2);
Slope=BB(1);
Pwave=P-Intercept-time*Slope;

Pstatic=P-Pwave;      %Hydrostatic pressure
```

```
Depth=Pstatic/Rho/g;
AverageDepth=mean(Depth);

%Optional figure to plot waterdepth as function of time
if (extraplot==1)
   figure
   plot(time,Depth);
   ylabel('depth (m)');
   xlabel('time (sec)');
   title(['average waterdepth ',heading]);


%Optional figuer to plot wave pressure as function of time
   figure
   plot(time,Pwave);
   xlabel('time (sec)');
   ylabel('pressure (Pa)');
   title(['presure variations ',heading]);
end;
%==========================================================================
==
%Time domain analysis
counter=0;
Pmin=0;
Pmax=0;
period=0;
for k=1:n-1
    if(Pwave(k+1)*Pwave(k)<0 && Pwave(k+1)<0)  %new wave starts at k+1
        counter=counter+1;
        T(counter)=period;
        L0=g/2/pi*period*period;
        if (Depth(k)/L0 < 0.36)
            L=sqrt(g*Depth(k))*(1-Depth(k)/L0)*period;
        else
            L=L0;
        end;
        H(counter)=(Pmax-Pmin)/Rho/g*cosh(2*pi/L/Depth(k));
        Pmax=0;
        Pmin=0;
        period=0;
    end;
    if(Pwave(k)>Pmax)  Pmax=Pwave(k);
    end;
    if (Pwave(k)<Pmin) Pmin=Pwave(k);
    end;
    period=period+interval;
end;
%counter = number of waves found
Hsorted=sort(H);
kk=0;
for k=1:counter
    if  Hsorted(k)>0.00
        kk=kk+1;
        Hsorted2(kk)=Hsorted(k);
    end;
end;
counter=kk;

Tsorted=sort(T);
```

```
Tmean = mean (Tsorted);
%loop to estimate water level variations
for k=1:n
        period=Tmean;
        L0=g/2/pi*period*period;
        if (Depth(k)/L0 < 0.36)
            L=sqrt(g*Depth(k))*(1-Depth(k)/L0)*period;
        else
            L=L0;
        end;
        eta(k)=(Pwave(k))/Rho/g*cosh(2*pi/L*Depth(k));
end;

figure
subplot(4,1,1);
plot(time,eta);
xlabel('time (sec)');
ylabel('waterlevel (m)');
title(['{\bf Time series ',heading,'}']);
%end waterlevel loop

%start loop to determine individual waves from waterlevel
etamin=0;
etamax=0;
counter=0;
for k=1:n-1
    if(eta(k+1)*eta(k)<0 && eta(k+1)<0)  %new wave starts at k+1
        counter=counter+1;
        Hzz(counter)=etamax-etamin;
        etamin=0;
        etamax=0;
    end;
    if(eta(k)>etamax)  etamax=eta(k);
    end;
    if (eta(k)<etamin) etamin=eta(k);
    end;
end;
%end loop to determine individual waves
Hsorted3=sort(Hzz);
kk=0;
for k=1:counter
    if  Hsorted3(k)>0.00
        kk=kk+1;
        Hsorted2(kk)=Hsorted3(k);
    end;
end;
counter=kk;

for k=1 : counter
prob(k)= (counter-k+1)/counter;
end;

Hmean=mean(Hsorted2);
Hrms = sqrt(mean(Hsorted2.^2)); % rms wave height

from=round(2*counter/3);
to=round(counter);
Hs=mean(Hsorted2(from:to)); %H 1/3, or significant wave height
```

```
%calculation of the H2%
Twopercent=round(counter-counter/50);
if (Twopercent==counter)
H2percenteMeasured=NaN;
else
H2percentMeasured=Hsorted2(Twopercent);
end;
H2percentMeasured;
H2percRayleigh=1.4*Hs;
%caculation of the H1%
Onepercent=round(counter-counter/100);
if (Onepercent==counter)
H1percenteMeasured=NaN;
else
H1percentMeasured=Hsorted2(Onepercent);
end;
H1percentMeasured;
H1percRayleigh=1.5*Hs;
%calculation of the H0.1%
Onepermille=round(counter-counter/1000);
if (Onepermille==counter)
H1permilleMeasured=NaN;
else
H1permilleMeasured=Hsorted2(Onepermille);
end;
H1permilleMeasured;
H1permilleRayleigh=1.85*Hs;

% Compositie weibull ditribution for shallow water
% source: Wave height distributions on shallow foreshores
%          Coastal Engineering, Volume 40, Issue 3, June 2000, Pages 161-182
%          Jurjen A Battjes, Heiko W Groenendijk

Htr=(0.35+5.8*TanAlfa)*AverageDepth; % Transistion wave height acc. to BG
(eq. 8)

H3=Hrms*BG(Htr,3);
H10=Hrms*BG(Htr,10);
H2=Hrms*BG(Htr,0.02);
H1=Hrms*BG(Htr,0.01);
H01=Hrms*BG(Htr,0.001);

%plot Rayleigh graph
h = axes('Position',[0 0 1 1],'Visible','off');
axes('Position',[.55 .40 .35 .30])

for i=1:counter
   probsorted2(i)=1-i/(counter+1);
end;
xmark=[0:0.25:1.3*Hsorted2(counter) ];
coef=Rayleigh(Hsorted2,probsorted2,xmark,Hrms,Htr);

ylabel('Probability');
xlabel('wave height (m)');
title('Wave height distribution ');
string=num2str(Hmean,2);
str(1)= {['H_{mean} (m)             ',string]};
```

```
string=num2str(Hrms,2);
str(2)= {['H_{rms} (m)              ',string]};
string=num2str(Hs,2);
str(3)= {['H_{s} (m)                 ',string]};
string=num2str(Tmean,2);
str(4)= {['T_{mean} (s)             ',string]};
string=num2str(Htr,2);
str(5)= {['H_{tr}(m)(acc BG)  '   ,string]};
string=num2str(1/TanAlfa);
str(6)= {['Bed slope 1:',string]};
str(7)={'              obs    Rayleigh  BG'};
string=num2str(H2percentMeasured,3);
stringB=num2str(H2percRayleigh,3);stringC=num2str(H2,3);
str(8)= {['H_{2%}(m)   ',string,'    ',stringB,'      ',stringC]};
string=num2str(H1percentMeasured,3);
stringB=num2str(H1percRayleigh,3);;stringC=num2str(H1,3);
str(9)= {['H_{1%}(m)   ',string,'    ',stringB,'      ',stringC]};
string=num2str(H1permilleMeasured,3);
stringB=num2str(H1permilleRayleigh,3);;stringC=num2str(H01,3);
str(10)= {['H_{0.1%}(m) ',string,'    ',stringB,'      ',stringC]};
set(gcf,'CurrentAxes',h)
text(.1,.55,str,'FontSize',8)



%==================================================================
% simple script utilizing crosgk (by G. Klopman) to obtain
% spectral estimate
%==================================================================
% data  contains the data
% N  is the number of samples per data segment (power of 2)
% M  is the number of frequency bins over which is smoothed (optional),
%    no smoothing for M=1 (default)
% DT is the time step (optional), default DT=1
% DW is the data window type (optional): DW = 1 for Hann window (default)
%                                        DW = 2 for rectangular window
% stats : display resolution, degrees of freedom (optimal, YES=1, NO=0)
%
% Output:
% P contains the (cross-)spectral estimates: column 1 = Pxx, 2 = Pyy, 3 =
Pxy
% F contains the frequencies at which P is given load time series

DT = interval;
    data = Pwave;
    [P,F,dof]=crosgk(data,data,length(data),M,DT,1,0);

 %recalcultate pressure spectrum to energy spectrum
  eta=1:length(F);  % length (F) is number of frequency bins
  for i=1:length(F)
      eta(i)=0;
  end;
  m0=0;   %zero-th moment
  m1=0;   %first moment
  m2=0;   %second moment
  m01=0;  %first negative moment {m(-1,0)}
  deltaF= F(31)-F(30);
```

```
emax=0;
% claculation loop to transform pressure spectrum to energy spectrum and
% to calculate the moments of the spectrum
% low and high frequencies are deleted, range from 200 to 0.2*max
% frequency bin, 200 means f= 200*deltaF, which is approx. 30 seconds
% 0.2*length(F)*deltaF = 0.4, so Tmin - 2.5 seconds
for i=50:cutoff*length(F)
    T=1/F(i);
    pr=sqrt(P(i,1));   %pr=pressure value in pressure spectrum
    L0=1.56*T*T;
    if (AverageDepth/L0<0.36)
        L=sqrt(g*AverageDepth)*(1-AverageDepth/L0)*T;
    else
        L=L0;
    end;
    e=upgrade*(pr/(Rho*g)*cosh(2*pi/L*AverageDepth))^2; %
e=energiedichtheid in Hz/m2
    eta(i)=e;    % replacing pressure value to energy value in spectrum
    if e>emax
        emax=e;
        Tpeak=T;
    end;
    m0 =m0 +e*deltaF;
    m1 =m1 +e*deltaF*F(i);
    m2 =m2 +e*deltaF*F(i)^2;
    if (F(i)>0)
        m01=m01+e*deltaF/F(i);
    end;
end;
Hm0=4*sqrt(m0);     %one may assume Hs = Hm0
Hrmss=sqrt(8*m0);    %root mean square height from spectrum
Tm=sqrt(m0/m2);     %spectral approximation of mean period
T01=m0/m1;          %Period based on first moment
T10=m01/m0;         %Period based on first negative moment


% plot spectrum and print output
h = axes('Position',[0 0 1 1],'Visible','off');
axes('Position',[.55 .05 .35 .20])
plot(F,eta)
axis([low high 0 10000])
axis 'auto y'
xlabel('frequency [Hz]');
ylabel ('Energy m^2/Hz');
title('energy spectrum ');
string=num2str(Hrmss,2);
str(1)= {['H_{rms} (m) ',string]};
string=num2str(Hm0,2);
str(2)= {['H_{m0} (m)  ',string]};
string=num2str(Tm,2);
str(3)= {['T_{m} (s)    ',string]};
string=num2str(T01,2);
str(4)= {['T_{m0,1} (s) ',string]};
string=num2str(T10,2);
str(5)= {['T_{m-1,0}(s) ',string]};
string=num2str(Tpeak,2);
str(6)= {['T_{peak}(s)  ',string]};
str(7)={' '};
str(8)={' '};
```

```
  string=num2str(AverageDepth,2);
  str(9)={['Average depht (m) ',string]};
  string=num2str(Rho);
  str(10)={['water density (kgm^{3})    ',string]};
  set(gcf,'CurrentAxes',h)
  text(.1,.15,str,'FontSize',8)


 %Optional plot pressure spectrum
if (extraplot==1)
    DT = interval;
    data = Pwave;
    [P,F,dof]=crosgk(data,data,length(data),M,DT,1,0);

    figure
    plot(F,P(:,1))     % F is pressure^2/Hz
    axis ([low high 0 1500])
    axis 'auto y'
    xlabel('frequency [Hz]');
    ylabel ('pressure Pa^2/Hz');
    title(['pressure spectrum ',heading]);
end;
```

TU Delft
Technische Universiteit Delft

Rayleigh.m

```
function coef = rayleigh(xi,yi,xmark,Hrms,Htr)
% ----------------------------------------------------------------
%      function coef = rayleigh(xi,yi,xmark)
%
%   Rayleigh Probability Paper: The paper is marked with probability
%   of exceedance [0.002, 0.005, 0.01, 0.05, 0.1:0.1:0.9]  (horizontal
%   lines associated with these values are drawn).  In addition, it draws
%   the linear curve fitting for input data points.
%
%   xi = horizontal input (physical quantity)
%   yi = vertical input (in probability of exceedance)
%   xmark = vertical grid lines to be plotted
%   coef = slope and intercept of the linear regression line.
x = xmark;
poe = [1:-0.1:0.1, 0.05,0.02,0.01,0.005,0.002,0.001,0.0005]; % prob.of
exceedance
y = sqrt(-log(poe));
n = length(y);
m = length(x);
axis([x(1) x(m) y(1) y(n)]);
dx = abs(x(m)-x(1))/25;
xh = [x(1) x(m)];
% working on horizontal line
for k=1:n,
    yh=[y(k) y(k)];
    line(xh,yh);
    text(x(1)-2*dx,y(k),num2str(poe(k)));  % prob. of exceedance
end
% set(gca,'linestyle',':');
% working on vertical line (horizontal scale)
dy=abs(y(n)-y(1))/25;
yv=[y(1),y(n)];
for k=1:m,
    xv=[x(k) x(k)];
    line(xv,yv);
    text(x(k)-dx/4,y(1)-dy,num2str(xmark(k)));
end
axis('off');
% x is wave height
% y is probability (correct numbers)
% z is scaled probability for Rayleigh paper
% plotting data points, convert probabilties for plot
%
zi = sqrt(-log(yi));
hold on
plot(xi,zi,'.')
%
% to draw regression line
%
coef=polyfit(xi,zi,1);
xa = 0.5 * min(xi);
xb = 1.5 * max(xi);
xx = [xa xi xb];
yy = polyval(coef,xx);
plot(xx,yy,'-');
%
H1=BG(Htr/Hrms,1);
```

```
H2=BG(Htr/Hrms,2);
xxx=[xmark(1):.01:xmark(m)];
lxxx=length(xxx);
%calculate plot with CWD of Battjes Groenendijk
for j=1:lxxx
  if (xxx(j)<Htr)
    yyy(j)=1-exp(-power((xxx(j)/(H1*Hrms)),2.0));  %BG 2
    else
    yyy(j)=1-exp(-power((xxx(j)/(H2*Hrms)),3.6));  %BG 3.6
    end;
end;
%convert probabilties to scale for plot
zzz=sqrt(-log(1-yyy));
hold on

plot(xxx,zzz,'r');
title('wave height (m)')
```

TU Delft
Technische Universiteit Delft

crosgk.m

```
function [P,F,dof]=crosgk(X,Y,N,M,DT,DW,stats);

% CROSGK   Power cross-spectrum computation, with smoothing in the
%          frequency domain
%
% Usage: [P,F]=CROSGK(X,Y,N,M,DT,DW,stats)
%
% Input:
% X  contains the data of series 1
% Y  contains the data of series 2
% N  is the number of samples per data segment (power of 2)
% M  is the number of frequency bins over which is smoothed (optional),
%    no smoothing for M=1 (default)
% DT is the time step (optional), default DT=1
% DW is the data window type (optional): DW = 1 for Hann window (default)
%                                        DW = 2 for rectangular window
% stats : display resolution, degrees of freedom (optimal, YES=1, NO=0)
%
% Output:
% P contains the (cross-)spectral estimates: column 1 = Pxx, 2 = Pyy, 3 =
Pxy
% F contains the frequencies at which P is given
%

%
% Gert Klopman, Delft Hydraulics, 1995
%

if nargin < 4,
  M = 1;
end;

if nargin < 5,
    DT = 1;
end;

if nargin < 6,
  DW = 1;
end;

if nargin < 7,
  stats = 1;
end;

df = 1 / (N * DT) ;

% data window

w = [];
if DW == 1,
  % Hann
  w  = hanning(N);
  dj = N/2;
else,
  % rectangle
```

```
  w  = ones(N,1);
  dj = N;
end;
varw = sum (w.^2) / N ;


% summation over segments

nx     = max(size(X));
ny     = max(size(Y));
avgx   = sum(X) / nx;
avgy   = sum(Y) / ny;
px     = zeros(size(w));
py     = zeros(size(w));
Pxx    = zeros(size(w));
Pxy    = zeros(size(w));
Pyy    = zeros(size(w));
ns     = 0;

for j=[1:dj:nx-N+1],

  ns = ns + 1;

  %   compute FFT of signals

  px = X([j:j+N-1]') - avgx;
  px = w .* px ;
  px = fft(px) ;

  py = Y([j:j+N-1]') - avgy;
  py = w .* py ;
  py = fft(py) ;

  % compute periodogram

  Pxx = Pxx + px .* conj(px) ;
  Pyy = Pyy + py .* conj(py) ;
  Pxy = Pxy + py .* conj(px) ;

end;

Pxx = (2 / (ns * (N^2) * varw * df)) * Pxx ;
Pyy = (2 / (ns * (N^2) * varw * df)) * Pyy ;
Pxy = (2 / (ns * (N^2) * varw * df)) * Pxy ;

% smoothing

if M>1,
  w = [];
  w = hamming(M);
  w = w / sum(w);
  w = [w(ceil((M+1)/2):M); zeros(N-M,1); w(1:ceil((M+1)/2)-1)];
  w = fft(w);
  Pxx = fft(Pxx);
  Pyy = fft(Pyy);
  Pxy = fft(Pxy);
  Pxx = ifft(w .* Pxx);
```

```
  Pyy = ifft(w .* Pyy);
  Pxy = ifft(w .* Pxy);
end;

Pxx = Pxx(1:N/2);
Pyy = Pyy(1:N/2);
Pxy = Pxy(1:N/2);

% frequencies

F = [];
F = ([1:1:N/2]' - 1) * df;

% signal variance

if DW == 1,
  nn = (ns + 1) * N / 2;
else,
  nn = ns * N;
end;
avgx  = sum (X(1:nn)) / nn;
varx  = sum ((X(1:nn) - avgx).^2) / (nn - 1);
avgy  = sum (Y(1:nn)) / nn;
vary  = sum ((Y(1:nn) - avgy).^2) / (nn - 1);
covxy = sum ((X(1:nn) - avgx) .* (Y(1:nn) - avgy)) / (nn - 1);

m0xx    = (0.5 * Pxx(1) + sum(Pxx(2:N/2-1)) + 0.5 * Pxx(N/2)) * df;
m0yy    = (0.5 * Pyy(1) + sum(Pyy(2:N/2-1)) + 0.5 * Pyy(N/2)) * df;
m0xy    = (0.5 * Pxy(1) + sum(Pxy(2:N/2-1)) + 0.5 * Pxy(N/2)) * df;

%disp(['m0x / varx = ' num2str(m0xx./varx) '  ;  m0y / vary = '
num2str(m0yy./vary) '  ; m0xy / varxy = ' num2str(real(m0xy)./covxy) '  '])


Pxx = Pxx * (varx  / m0xx);
Pyy = Pyy * (vary  / m0yy);
Pxy = Pxy * (covxy / real(m0xy));

P = [Pxx, Pyy, Pxy];

% output spectrum characteristics
dof = floor(2*ns*(M+1)/2/(3-DW));
if stats == 1
fprintf('number of samples used : %8.0f\n', nn);
fprintf('degrees of freedom     : %8.0f\n', floor(2*ns*(M+1)/2/(3-DW)));
fprintf('resolution             : %13.5f\n', (3-DW)*df*(M+1)/2);
end
%
```

BG.m

```
function [Hnorm]=BG(Htr,Proba);
% Function developed by H.J. Verhagen, Delft University of Technology
% January 2013

% Input:
% Htr - H transition acc Battjes-Groenendijk (eq 8 in paper)

% Proba code for required exceendandce:
%     Proba = 3     ==> H1/3
%     Proba = 10    ==> H1/10
%     Proba = 0.02  ==> H2%
%     Proba = 0.01  ==> H1%
%     Proba = 0.001 ==> H0.1%

% Output Hnorm

% source: Wave height distributions on shallow foreshores
%         Coastal Engineering, Volume 40, Issue 3, June 2000, Pages 161-182
%         Jurjen A Battjes, Heiko W Groenendijk

%table from Battjes groenendijk
% Htr     H1      H2      H1/3    H1/10   H2%     H1%     H0.1%
BGtable=[
0.05    12.193  1.060   1.279   1.466   1.548   1.620   1.813
0.1     7.003   1.060   1.279   1.466   1.548   1.620   1.813
0.15    5.063   1.060   1.279   1.466   1.548   1.620   1.813
0.2     4.022   1.060   1.279   1.466   1.548   1.620   1.813
0.25    3.365   1.060   1.279   1.466   1.548   1.620   1.813
0.3     2.908   1.060   1.279   1.466   1.548   1.620   1.813
0.35    2.571   1.060   1.279   1.466   1.548   1.620   1.813
0.4     2.311   1.060   1.279   1.466   1.548   1.620   1.813
0.45    2.104   1.060   1.279   1.466   1.549   1.620   1.813
0.5     1.936   1.061   1.280   1.467   1.549   1.621   1.814
0.55    1.796   1.061   1.281   1.468   1.550   1.622   1.815
0.6     1.678   1.062   1.282   1.469   1.552   1.624   1.817
0.65    1.578   1.064   1.284   1.471   1.554   1.626   1.820
0.7     1.492   1.066   1.286   1.474   1.557   1.629   1.823
0.75    1.419   1.069   1.290   1.478   1.561   1.634   1.828
0.8     1.356   1.073   1.294   1.483   1.567   1.639   1.835
0.85    1.302   1.077   1.300   1.490   1.573   1.646   1.843
0.9     1.256   1.083   1.307   1.498   1.582   1.655   1.852
0.95    1.216   1.090   1.315   1.507   1.591   1.665   1.864
1       1.182   1.097   1.324   1.518   1.603   1.677   1.877
1.05    1.153   1.106   1.335   1.530   1.616   1.690   1.892
1.1     1.128   1.116   1.346   1.543   1.630   1.705   1.909
1.15    1.108   1.126   1.359   1.558   1.645   1.721   1.927
1.2     1.090   1.138   1.371   1.573   1.662   1.739   1.946
1.25    1.075   1.150   1.381   1.590   1.679   1.757   1.967
1.3     1.063   1.162   1.389   1.607   1.698   1.776   1.988
1.35    1.052   1.175   1.395   1.626   1.717   1.796   2.011
1.4     1.043   1.189   1.399   1.644   1.737   1.817   2.034
1.45    1.036   1.203   1.403   1.664   1.757   1.838   2.058
1.5     1.030   1.217   1.406   1.683   1.778   1.860   2.082
1.55    1.024   1.231   1.408   1.703   1.799   1.882   2.106
1.6     1.020   1.246   1.410   1.721   1.820   1.904   2.131
1.65    1.016   1.261   1.411   1.736   1.841   1.927   2.156
```

```
1.7     1.013   1.275   1.412   1.749   1.863   1.949   2.182
1.75    1.011   1.290   1.413   1.759   1.884   1.972   2.207
1.8     1.009   1.305   1.413   1.767   1.906   1.994   2.232
1.85    1.007   1.320   1.414   1.773   1.927   2.017   2.257
1.9     1.006   1.334   1.414   1.779   1.949   2.039   2.282
1.95    1.004   1.349   1.415   1.783   1.970   2.062   2.307
2       1.004   1.363   1.415   1.786   1.985   2.084   2.332
2.05    1.003   1.378   1.415   1.789   1.983   2.106   2.357
2.1     1.002   1.392   1.415   1.791   1.982   2.128   2.382
2.15    1.002   1.407   1.415   1.793   1.981   2.150   2.406
2.2     1.001   1.421   1.415   1.795   1.981   2.149   2.430
2.25    1.001   1.435   1.415   1.796   1.980   2.148   2.454
2.3     1.001   1.449   1.415   1.797   1.979   2.148   2.478
2.35    1.001   1.462   1.415   1.797   1.979   2.147   2.502
2.4     1.000   1.476   1.416   1.798   1.979   2.147   2.525
2.45    1.000   1.490   1.416   1.798   1.979   2.147   2.548
2.5     1.000   1.503   1.416   1.799   1.978   2.147   2.571
2.55    1.000   1.516   1.416   1.799   1.978   2.146   2.593
2.6     1.000   1.529   1.416   1.799   1.978   2.146   2.616
2.65    1.000   1.542   1.416   1.799   1.978   2.146   2.629
2.7     1.000   1.555   1.416   1.799   1.978   2.146   2.629
2.75    1.000   1.568   1.416   1.800   1.978   2.146   2.628
2.8     1.000   1.580   1.416   1.800   1.978   2.146   2.628
2.85    1.000   1.593   1.416   1.800   1.978   2.146   2.628
2.9     1.000   1.605   1.416   1.800   1.978   2.146   2.628
2.95    1.000   1.617   1.416   1.800   1.978   2.146   2.628
3       1.000   1.630   1.416   1.800   1.978   2.146   2.628];

j=0;
 if (Proba==1)     j=2; end;
 if (Proba==2)     j=3; end;
 if (Proba==3)     j=4; end;
 if (Proba==10)    j=5; end;
 if (Proba==0.02)  j=6; end;
 if (Proba==0.01)  j=7; end;
 if (Proba==0.001) j=8; end;
  i=round(Htr/0.05);
 if (Htr>3)    i=60; end;
 if (Htr<.05) i=1;  end;

 if(j==0)
   Hnorm=NaN;
   else
   Hnorm=BGtable(i,j);
   end;
end
```