Appendix B. Summary of the Matlab scripts

B.1. Introduction

In this appendix the most important used Matlab programmes are commented. They are either scripts or functions that analyse the wave records obtained with SWASH or compute wave overtopping.

All scripts that deal with a wave record need to define first two vectors with the output water elevation time series of SWASH. A vector `t` is defined with the output times and a vector `W` with the water level measures.

Either in scripts and functions, the time step is obtained from dividing the time-record length between the number of intervals. This gives more flexibility to the programmes.

B.2. Wave per wave analysis

Only one script is used to analyse the wave-record wave per wave. This script could be easily converted into a Matlab function.

Analysis.m

This script analyses the wave record wave per wave and gives the mean water level, the significant wave height, the mean wave height, the root mean square wave height and the mean wave period. It also computes a vector with all the wave heights and wave periods though they are not given as a direct output. However, the maximum and minimum values of each parameter can easily be known by asking the maximum and minimum values of the vector `H`.

It also plots the water level record, but the title must be customized each run according to the input data. Furthermore, the variance of the water elevation is computed and it is used to give $H_{m0} = 4 \cdot \sigma$, $H_s = 3.8 \cdot \sigma$ and $H_{rms} = \sqrt{8} \cdot \sigma$ as a contrasting values.

The waves are identified and split according to the zero down-crossing criterion. If the mean water level is quite below or above zero, it will not work properly. The reason is that some waves will not be considered because the zero line is not crossed. This leads to larger wave periods, and usually indicates reflection. At the graph a very long standing wave may be observed.

The code is:

```
%Analysis.m
n=length(W);
s=0;
%At: time interval between the results written on the table!
%At=t(2)-t(1);
At=(max(t)-min(t))/(length(t)-1);
%At=0.05;

P=[];
for i=(1+1):(n-1)
```

```
    if W(i-1)>0 & W(i+1)<0;
        %if W(i+1)<0; %Zero-down crossing criterion
        %Be careful! The majority of the points are selected twice!
        s=s+1;
        P(s)=i; %It indicates the changing place
        %The difference between the points lets us know Tz
        %end
    end
end
m=length(P); %m changing points, m-1 waves
%The double points are deleted
s=0;
for i=1:(m-1)
    if (P(i+1)-P(i))==1;
        P(i)=0;
        s=s+1;%Number of points to delete
    end
end
P=sort(P);
P=P((s+1):m);
T=[];
m=length(P);
for i=1:(m-1)
    T(i)=At*(P(i+1)-P(i)); %Wave periods T
end

%Obtaining Hcrest, Tcrest
%Hcrest
H=[];

for j=1:(m-1) %For each wave (m-1)
    a=[];
    %Wave heigh of each wave
    a=W(P(j):P(j+1));
    %Look at the maximum and minimum
    mx=max(a);
    %[max,M]=max(a);
    mn=min(a);
    %[min,l]=min(a);
    %Hcrest
    H(j)=mx-mn;
    %Absolute position of the crests k=k+P(j-1)
    %Vector with the distances between crests (maximums)
    %B(j)=k;
end

%Mean Values
Wm=mean(W)
disp('From wave per wave analysis')
Hc=mean(H)
Tm=mean(T)
h=sort(H);
%Remember that we have m-1 waves
n=round((m-1)/3);
h=h(2*(n+1):(m-1));
HSIG=mean(h)

%Root Mean Square Wave Height
HRMS=sqrt(mean(H.^2))
```

```
%Plot
figure(1)
plot(t,W,'b')
xlabel('time(s)')
ylabel('water level (m)')
title('Water level at')
xlim([round(min(t)) round(max(t))])

%Variance
disp('From the variance')
m0=var(W);
H_m0=4*sqrt(m0)
H_rms=sqrt(8*m0)
H_s=3.8*sqrt(m0)
```

B.3. Spectral wave analysis

spectral_analysis.m

This is a Matlab function that from 3 inputs: the time series input (for instance, `t`), the water elevation record (the previous `W`) and the number of periodograms; builds the wave spectrum; computes the spectral parameters m_0, m_1, m_2 and m_{-1}; and gives the following output values: H_{m0}, H_{rms}, T_p, T_{m01} and $T_{m-1,0}$. Two output spectrum plots are given (one with the wave frequency and the other with the wave period).

The sample is divided in the specified number of periodograms and a wave spectrum is computed for each "stretch". Afterwards, they are averaged to compute the "final" wave spectrum.

The more periodograms are used, the smoother the wave spectrum will be. This is a way to improve the obtained spectrum and delete the noise. However, a longer record is needed, because around 8-10 minutes of wave record are required to get a reliable spectrum, otherwise no long waves would be included in the spectrum (narrow spectrum range). It is also advised in the consulted literature (Steward, 2008) to consider between 10 and 30 periodograms. This is the reason why I enlarged the "working" SWASH simulations up to 90 minutes, so that I had an enough long wave record to average 10 periodograms. However, hardly any differences are found between a spectrum computed with 5 or 10 periodograms (45 or 90 minutes), but a smoother shape.

Each wave spectrum is computed through the Matlab function `periodogram`, which also gives the frequency vector. This function computes the FFT from the input water elevation vector. Because the FFT algorithm works efficiently with a vector length that is a power of 2, the function `periodogram` pads the input vector with zeros until a number of elements that is a power of 2 is obtained. This number has been previously computed from the length of each stretch.

Then two plots with the Power Spectral Density against wave frequency and wave period are drawn.

The next step is computing the characteristic wave parameters from the wave spectrum.

To obtain them we need to calculate first the spectral moments. The zero spectral moment m_0 is sufficient to compute the significant wave height. But if we want to compute the mean wave period T_{m01}, we need to compute m_1 too. Besides, m_2 is required for the mean zero-crossing period T_0 and m_{-1}, for T_{m-10}.

For each of these spectral moments an integral must be numerically computed. Since the analytical function is unknown and we only have a set of equidistant points, a Newton-Cotes quadrature must be used. As the spectrum graph has been built using linear interpolation, a composite trapezoidal rule is used for the numerical integration. This method converges with increasing number of subintervals. We will need one subinterval less than the number of points of the spectrum.

The integral of m_{-1} presents some problems, because the function to be integrated is divided by a zero value of the frequency at the first point. It can be solved by removing this point of the computation. The value of the spectrum at that point should go to zero, so the indeterminacy should not be a problem and the error is likely to be low.

The peak wave period is computed using the Matlab `max` function, which indicates the maximum value of the given vector (in this case the spectral density vector) and its position. Then, the associated wave frequency or period can be easily obtained.

This function was written from a previous programme called `Spectral.m`. The code of this function is:

```
function [Hm0,Tp,Tm01,Tm_10,Hrms]=spectral_analysis(t,W,p)
%Function that builds the wave spectrum from a wave record (water
elevation)
%and give some output parameters. The output parameters are the
significant
%wave height and the mean wave period.
%It is based on Spectral.m Script (and Spectral2.m) reconverted into a
%function
%
% Input parameters
% t = time series input
% W = Water elevation series input
% p = number of periodograms
%
% Output parameters
% Hm0 = Spectral wave height/significant wave height = 4*sqrt(m0)
% HRMS = Root mean square wave height = sqrt(8*m0)
% Tp = peak wave period
% Tm01 = mean wave period = m0/m1
% Tm_10 = m_1/m0

%At: time interval between the results written on the table!
%At=t(2)-t(1);
At=(max(t)-min(t))/(length(t)-1);
%Sampling frequency
fr=1/At;
```

```
%Length of the water elevation record
n=length(W);
%number of periodograms
%p=10;
%p=5;
%W is divided into 20 samples/periodograms 18000/20=900
samples/periodogram
%W is divided into 10 periodograms 18000/10=1800 samples/peridogram
%W is divided into 5 periodograms 18000/5=3600 samples/periodogram
%Very long records needed!!! 90 min for 10 periodograms or 45 for 5.

%Periodogram samples
x=[];
for i=1:p
   x(:,i)=W((1+(i-1)*n/p):((n/p)*i));
end

%Next power of 2 greater or equal to length of the sample vector to
%calculate fft. FFT needs a length power of 2 to be efficient. For a
%shorter record, zeros will be added to reach this value.
nfft=2^(nextpow2(length(x(:,1))));

Pxx=[];
for i=1:p
%PSD
[Pxx(:,i),f]=periodogram(x(:,i),[],nfft,fr);
%[Pxx(:,i),w]=periodogram(x(:,i),[],nfft,fr);
end

%Average periodograms
mW=sum(Pxx,2)/p;

%f = Frequency vector
%f=w

%Wave period vector for the plot
ft=1./f;
%ft=1./w;

%Wave frequency spectral density plot
figure(2)
plot(f,mW)
xlabel('Frequency (Hz)')
ylabel('Power Spectral Density (m^2/Hz)')
title('Power wave Spectrum at x= of flume')

%Wave Spectral density plot with wave period
figure(3)
plot(ft,mW);
xlabel('Wave period (s)')
ylabel('Power Spectral Density')
title('Power wave Spectrum at x= of flume')

%Computation of some wave parameters.
%To do that 2 integrals must be numerically computed. Since the
%analytical function is unknown and the available data is a set of
points
%(theorically equidistant), a Newton-Cotes quadrature will be used.
```

```
%As the spectrum graph is built using linear interpolation, a
composite
%trapezoidal rule will be used for numerical integration.

%Spectrum function E(f)=mW
%total number of points
m=length(mW);
%number of subintervals
m=m-1;
%length of the subinterval
Ax=(f(m+1)-f(1))/m;

%Integral I1 m0=int{E(f)df} I1
s=0;
for i=2:m
    s=s+mW(i); %sum(2,m) f(xi)
end
m0=0.5*Ax*(mW(1)+2*s +mW(m+1));

%Integral I2 m1=int{f*E(f)df} I2
s=0;
%f=f*E(f)=f*mW
for i=2:m
    s=s+f(i)*mW(i);
end
m1=0.5*Ax*(f(1)*mW(1)+2*s+f(m+1)*mW(m+1));

%Integral I3 m2=int{f^2*E(f)df} I3
s=0;
for i=2:m
    s=s+(f(i))^2*mW(i);
end
m2=0.5*Ax*((f(1))^2*mW(1)+2*s+(f(m+1))^2*mW(m+1));

%Integral I4 m_1=int{f^(-1)*E(f)df} I4
s=0;
for i=3:m
    s=s+mW(i)/f(i);
end
m_1=0.5*Ax*(1/f(2)*mW(2)+2*s+1/f(m+1)*mW(m+1));
%Note: f(1)=0 and so it is expluded. The integral is not fully
computed.
%There is a mistake there!

%Spectrum analysis output
%Significant wave height
%Hm0=4*sqrt(int{E(f)df})
Hm0=4*sqrt(m0);
%Mean spectral wave period
%Tm01=(int{E(f)df}/(f*int{E(f)df}))
Tm01=m0/m1;
%Peak wave period
[M,i]=max(mW);
fm=f(i);
Tp=1/fm;
%Mean zero-crossing period
T0=sqrt(m0/m2);
%Mean spectral wave period Tm-1,0 with a higher weight of the longer
waves
```

```
Tm_10=m_1/m0;
%Remember that one point has been ommited to compute Tm_10. I am not
sure
%if it does not affect the result!

%Root Mean Square Wave height
Hrms=sqrt(2)/2*Hm0;
```

B.4. Dike overtopping

Two functions were created to compute wave overtopping for dikes according to the Eurotop formulas with the purpose of speeding up all the calculations. All stretches of the formulas are included and compared. Both of them have as an input H_{m0} and T_{m01} at the toe of the dike (without reflection), the crest freeboard and the slope tangent; and give as an output the run-up, the discharge, the dimensionless discharge, the wave steepness $s_{m-1,0}$ and $\xi_{m-1,0}$.

dike_overtopping.m

This function computes the mean wave overtopping taking into account the probabilistic formulations. The deterministic expressions are also in and could be easily used by changing the code.

The function code is:

```
function [Ru2,q,q_ad,Xim_10,sm_10]=
dike_overtopping(Hm0,Tm_10,Rc,slope)
%Dike_q.m
%This script computes wave overtopping and run-up on dikes applying
the
%formulas from the Eurotop Manual.
%
%INPUT
%Hm0 Significant wave height
%Tm_10 mean wave period
%Rc Crest freeboard (Over sea level!!)
%slope tan alpha
%
%OUTPUT
%Ru2 Ru2% Run-up
%q Mean wave overtopping
%q_ad dimensionless mean wave overtopping
%Xim_10 Iribarren number
%sm_10 Wave steepness

%Water depth
h=10;
%Dike freeboard
%Rc=2.5; %Over sea level!!
%Slope tan alpha
%slope=0.5;
gammaf=1.0; %smooth slope
gammab=1.0; %no berm
gammaB=1.0; %normal waves
gammav=1.0; %no vertical wall at the top
```

```
%Wave parameters
%Hm0=3.5;
%Tm_10=10;

Lm_10=9.81*Tm_10^2/(2*pi);
sm_10=sqrt(Hm0/Lm_10);
Xim_10=slope/sqrt(Hm0/Lm_10);

%Ru2%/Hm0
Ru2=1.65*gammaf*gammab*gammaB*Xim_10;
m=1.0*gammab*gammaf*gammaB*(4-1.5/sqrt(Xim_10));
if Ru2>m;
    Ru2=m;
end
%Wave run-up
Ru2=Ru2*Hm0;

if Xim_10<5;
    % Probabilistic
    q=0.067/sqrt(slope)*gammab*Xim_10*exp(-
4.75*Rc/(Xim_10*Hm0*gammab*gammaf*gammaB*gammav));
    m=0.2*exp(-2.6*Rc/(Hm0*gammaf*gammaB));
    % Deterministic
    % q=0.067/sqrt(slope)*gammab*Xim_10*exp(-
4.3*Rc/(Xim_10*Hm0*gammab*gammaf*gammaB*gammav));
    % m=0.2*exp(-2.3*Rc/(Hm0*gammaf*gammaB));
    if q>m;
    q=m;
    end
elseif Xim_10>7;
    % Probabilistic
    q=10^(-0.92)*exp(-Rc/(gammaf*gammaB*Hm0*(0.33+0.022*Xim_10)));
    % Deterministic
    % q=0.21*exp(-Rc/(gammaf*gammaB*Hm0*(0.33+0.022*Xim_10)));
else
    %disp('error')
    disp('interpolation needed')
    a=0.067/sqrt(slope)*gammab*5*exp(-
4.75*Rc/(5*Hm0*gammab*gammaf*gammaB*gammav));
    b=0.2*exp(-2.6*Rc/(Hm0*gammaf*gammaB));
    if a>b;
        a=b;
    end
    c=10^(-0.92)*exp(-Rc/(gammaf*gammaB*Hm0*(0.33+0.022*7)));
    q=a+(c-a)/(7-5)*(Xim_10-5);
end
%Wave overtopping
q_ad=q;
q=q*sqrt(9.81*(Hm0^3));
```

dike_overtopping_q.m

This function has a different purpose. It computes the wave run-up and wave overtopping that are exceeded with a probability `1-p`. Therefore, the function has an extra input parameter `p`, which can be 0.025, 0.05, 0.5, 0.95 or 0.975.

The code is:

```
function
[Ru2,q,q_ad,Xim_10,sm_10]=dike_overtopping_p(Hm0,Tm_10,Rc,slope,p)
%Dike_q.m
%This script computes wave overtopping and run-up on dikes applying
the
%formulas from the Eurotop Manual.
%
%INPUT
%Hm0 Significant wave height
%Tm_10 mean wave period
%Rc Crest freeboard (Over sea level!!)
%slope tan alpha
%p percentile (2.5%, 5%, mean (50%), 95% and 97.5%). Unitary input!!
%
%OUTPUT
%Ru2 Ru2% Run-up
%q Mean wave overtopping
%q_ad dimensionless mean wave overtopping
%Xim_10 Iribarren number
%sm_10 Wave steepness
%The output discharges are exceeded with a probability of 1-p.

%Water depth
h=10;
%Dike freeboard
%Rc=2.5; %Over sea level!!
%Slope tan alpha
%slope=0.5;
gammaf=1.0; %smooth slope
gammab=1.0; %no berm
gammaB=1.0; %normal waves
gammav=1.0; %no vertical wall at the top

if p==0.5;
    x=0;
elseif p==0.05;
    x=-1.64;
elseif p==0.95;
    x=1.64;
elseif p==0.025;
    x=-1.96;
elseif p==0.975;
    x=1.96
else
    disp('Error: this value can not be an input')
end

%Wave parameters
%Hm0=3.5;
%Tm_10=10;

Lm_10=9.81*Tm_10^2/(2*pi);
sm_10=sqrt(Hm0/Lm_10);
Xim_10=slope/sqrt(Hm0/Lm_10);

%Ru2%/Hm0
Ru2=(1.65+x*0.1105)*gammaf*gammab*gammaB*Xim_10;
m=1.0*gammab*gammaf*gammaB*((4+x*0.28)-(1.5+x*0.105)/sqrt(Xim_10));
if Ru2>m;
```

```
    Ru2=m;
end
%Wave run-up
Ru2=Ru2*Hm0;

if Xim_10<5;
    % Probabilistic
    q=0.067/sqrt(slope)*gammab*Xim_10*exp((-
4.75+x*0.5)*Rc/(Xim_10*Hm0*gammab*gammaf*gammaB*gammav));
    m=0.2*exp((-2.6+x*0.35)*Rc/(Hm0*gammaf*gammaB));
    if q>m;
    q=m;
    end
elseif Xim_10>7;
    % Probabilistic
    q=10^(-0.92+x*0.24)*exp(-
Rc/(gammaf*gammaB*Hm0*(0.33+0.022*Xim_10)));
else
    %disp('error')
    disp('interpolation needed')
    a=0.067/sqrt(slope)*gammab*Xim_10*exp((-
4.75+x*0.5)*Rc/(Xim_10*Hm0*gammab*gammaf*gammaB*gammav));
    b=0.2*exp((-2.6+x*0.35)*Rc/(Hm0*gammaf*gammaB));
    if a>b;
        a=b;
    end
    c=10^(-0.92+x*0.24)*exp(-
Rc/(gammaf*gammaB*Hm0*(0.33+0.022*Xim_10)));
    q=a+(c-a)/(7-5)*(Xim_10-5);
end
%Wave overtopping
q_ad=q;
q=q*sqrt(9.81*(Hm0^3));
```

B.5. Rubble mound overtopping

mound_overtopping.m

This function is analogous to `dike_overtopping.m` and computes wave overtopping for rubble mound breakwaters according to the Eurotop Manual.

The wave overtopping at the beginning and at the back of the crest are computed. The function has two extra input values, the roughness influence factor and the crest width, which is needed to compute the crest width reduction.

The code is:

```
function [Ru2,q_t,q_b,q_ad]=
mound_overtopping(Hm0,Tm_10,Rc,slope,gammaf,Gc)

%This function computes wave overtopping and run-up on rubble mounds
applying the
%formulas from the Eurotop Manual.
%
%INPUT
%Hm0 Significant wave height
```

```
%Tm_10 mean wave period
%Rc Crest freeboard (over sea level!!!)
%slope tan alpha
%gammaf Roughness factor
%Gc Crest width
%
%OUTPUT
%Ru2 Ru2% Run-up
%q_ad Mean wave overtopping (dimensionless without crest reduction)
%q_t Mean wave overtopping at the beginning of the crest
%q_b Mean wave overtopping at the back of the crest
%
%SI Units

%Water depth
h=10;
%Crest width
%Gc=5;
%Slope tan alpha
%slope=0.5;
%gammaf=0.6; %1 rock layer, impermeable core
gammab=1.0; %no berm
gammaB=1.0; %normal waves
gammav=1.0; %no vertical wall at the top

%Wave parameters
%Hm0=3.5;
%Tm_10=10;

Lm_10=9.81*Tm_10^2/(2*pi);
Xim_10=slope/sqrt(Hm0/Lm_10)

if Xim_10>10;
    gammaf_surging=1.8;
elseif Xim_10<=10 && Xim_10>=1.8;
    gammaf_surging=gammaf+(Xim_10-1.8)*(1-gammaf)/8.2;
else
end

%Ru2%/Hm0
Ru2=1.65*gammaf*gammab*gammaB*Xim_10;
m=1.0*gammab*gammaf_surging*gammaB*(4-1.5/sqrt(Xim_10));
if Ru2>m;
    Ru2=m;
end
%Wave run-up
Ru2=Ru2*Hm0

%Crest effect
Cr=3.06*exp(-1.5*Gc/Hm0);
if Cr>1;
    Cr=1;
end

q=0.067/sqrt(slope)*gammab*Xim_10*exp(-
4.75*Rc/(Xim_10*Hm0*gammab*gammaf*gammaB*gammav));
m=0.2*exp(-2.6*Rc/(Hm0*gammaf*gammaB));
    if q>m;
    q=m;
```

```
    end

%Wave overtopping
disp('Wave overtopping at the beginning of the crest')
q_t=q*sqrt(9.81*(Hm0^3))
disp('Wave overtopping at the back of the crest')
q_b=q_t*Cr
q_ad=q;
```