DAFTAR PUSTAKA

Bambang, Triatmodjo. 1999. *Teknik Pantai.* Yogyakarta.

Houlthuijsen, Leo H. 2007. *Waves in Oceanic and Coastal Waters*. New York: Cambridge University Press.

Janssen, Peter. 2016. "*The Impact of Ocean Waves*." Grantham Institude

Mayasari, Dini. 2001. "*Analisis Spektrum Gelombang Laut*." (Universitas Diponegoro).

Sifnioti, D. E, T. H. Soukissian, and S. E. Poulus. 2015. "*Short-Term Spectral and Statistical Analysis of Sea Surface Elevation Data From buoys Located in the Greek Sea*." Marine Research.

WAFO, Group. 2017. *a Matlab Toolbox for Analysis of Random Waves and Loads.* Sweden: Lund University.

Pierson, J. Willard, and Lionel Moskowitz. 1964. "*A Proposed Spectral Form for Fully Developed Wind Seas Based on the Similarity Theory of S. A. Kitaigrodskii*." Journal of Geophysical Research Vol. 69.

https://kkp.go.id/artikel/maritim-indonesia (Diakses pada tanggal 16 Juli 2018).

https://www.ngdc.noaa.gov/mgg/global/etopo1_ocean_volumes.html (Diakses pada tanggal 25 April 2020).

LAMPIRAN

Lampiran 1. Scrift file data dilaut utara Atlantik yang diakses melalui http://www.maths.lth.se/matstat/wafo/documentation/wafodoc/wafo/data/northsea.html

```
NORTHSEA  coastline map of The Nortsea

   CALL:  map = load('northsea.dat');

  Size              :     60646 X 2
  Sampling Rate     :
  Device            :
  Source            :
http://crusty.er.usgs.gov/coast/getcoast.html
  Format            :    ascii, c1: longitude c2: latitude
  Description       :
    NORTHSEA.DAT contains data for plotting a map of The Northsea.
    The data is obtained from USGS coastline extractor.

   Example: the map is seen by

    plot(map(:,1),map(:,2)), hold on
    text( 1,62     ,'Statfjord A'), line([1.8, 1.8], [62  ,61.2 ])
    plot(1.8,61.2,'x')
    text(1,60.5,'Gullfaks C'),      line([1.8, 2.3 ], [60.5 ,61.20
])
    plot(2.30,61.20,'x')
    text(1,59.1,'Frigg'),           line([1.8, 2.0 ], [59.1 ,59.9 ])
    plot(2.0,59.90,'x'),
    text(1,57.6,'Sleipner'),        line([1.8, 1.9 ], [57.60 ,58.4
])
    plot(1.90,58.40,'x')
    text(1,56.9,'Draupner'),        line([1.8, 2.6 ], [56.90 ,57.7
])
    plot(2.6,57.7,'x'),
    text(10.40,60.10,'Oslo'),      plot(10.80,59.85,'h')
    text(10.00,63.05,'Trondheim'), plot(10.80,63.40,'h')
    text(4.00,58.80,'Stavanger'),  plot(5.52,58.90,'h')
    text(3.50,60.30,'Bergen'),     plot(5.20,60.30,'h') , hold off
```

Lampiran 2. *Source code* spektrum JONSWAP yang diakses melalui

http://www.maths.lth.se/matstat/wafo/documentation/wafodoc/wafo/spec/jonswap.html

```
001 function S1 = jonswap(w1,sdata,plotflag)
002 %JONSWAP Calculates (and plots) a JONSWAP spectral density
003 %
004 % CALL:  S = jonswap(w,sdata,plotflag);
005 %        S = jonswap(wc,sdata,plotflag);
006 %
007 %   S      = a struct containing the spectral density. See
                datastructures
008 %   w      = angular frequency (default linspace(0,wc,257))
009 %   wc     = angular cutoff frequency (default 33/Tp)
010 %   sdata  = [Hm0 Tp gamma sa sb A], where
011 %          Hm0  = significant wave height (default 7 (m))
012 %          Tp   = peak period (default 11 (sec))
013 %          gamma= peakedness factor determines the concentraton
014 %            of the spectrum on the peak frequency,  1 <= gamma
                <= 7.
015 %   (default depending on Hm0, Tp, see below)
016 %          sa,sb = spectral width parameters (default 0.07 0.09)
017 %          A     = alpha, normalization factor, (default -1)
018 %                  A<0 : A calculated by integration so that int
              S dw =Hm0^2/16
019 %                 A==0 : A = 5.061*Hm0^2/Tp^4*(1
                         0.287*log(gamma))
020 %                 A>0  : A = A
021 %       plotflag = 0, do not plot the spectrum (default).
022 %                  1, plot the spectrum.
023 %
024 %  For zero values, NaN's or parameters not specified in DATA
the
025 %  default values are used.
026 %
027 %
028 %         S(w) = A*g^2/w^5*exp(-5/4(wp/w)^4)*j^exp(-.5*((w/wp-
                   1)/s)^2)
029 %    where
030 %          s    = sa w<=wp
031 %                 sb w>wp (wp = angular peak frequency)
032 %          j    = gamma,  (j=1, => Bretschneider spectrum)
033 %
034 %  This spectrum is assumed to be especially suitable for the
```

```
      North Sea,
035 %  and does not represent a fully developed sea. It is a
      reasonable model for
036 %  wind generated sea when 3.6*sqrt(Hm0) < Tp < 5*sqrt(Hm0)
037 %  A standard value for gamma is 3.3. However, a more correct
      approach is
038 %  to relate gamma to Hm0:
039 %        D = 0.036-0.0056*Tp/sqrt(Hm0);
040 %        gamma = exp(3.484*(1-0.1975*D*Tp^4/(Hm0^2)));
041 %  This parameterization is based on qualitative considerations
      of deep water
042 %  wave data from the North Sea, see Torsethaugen et. al. (1984)
043 %  Here gamma is limited to 1..7.
044 %
045 %  The relation between the peak period and mean zero-upcrossing
      period
046 %  may be approximated by
047 %         Tz = Tp/(1.30301-0.01698*gamma+0.12102/gamma)
048 %
049 % Example:  % Bretschneider spectrum Hm0=7, Tp=11
050 %      S = jonswap([],[0 0 1])
051 %
052 % See also  pmspec, torsethaugen, simpson
053
054 % References:
055 % Torsethaugen et al. (1984)
056 % Characteristica for extreme Sea States on the Norwegian
      continental shelf.
057 % Report No. STF60 A84123. Norwegian Hydrodyn. Lab., Trondheim
058 %
059 % Hasselman et al. (1973)
060 % Measurements of Wind-Wave Growth and Swell Decay during the
      Joint
061 % North Sea Project (JONSWAP).
062 % Ergansungsheft, Reihe A(8), Nr. 12, Deutschen Hydrografischen
      Zeitschrift.
063
064
065 % Tested on: matlab 6.0, 5.3
066 % History:
067 % revised pab June 2005
068 % -fixed a bug in help header: the jonswap range is now correct
069 % revised pab 11jan2004
070 % - replaced code with call to getjonswappeakedness
071 % revised jr 22.08.2001
072 % - correction in formula for S(w) in help: j^exp(-.5*((w-
      wp)/s*wp)^2)
073 %   (the first minus sign added)
074 % revised pab 01.04.2001
075 % - added wc to input
076 % revised jr 30.10 2000
077 %   - changed 'data' to 'sdata' in the function call
```

```
078 % revised pab 20.09.2000
079 %   - changed default w: made it dependent on Tp
080 % revised es 25.05.00
081 %   - revision of help text
082 % revised pab 16.02.2000
083 %  - fixed a bug for sa,sb and the automatic calculation of
      gamma.
084 %  - added sa, sb, A=alpha to data input
085 %  - restricted values of gamma to 1..7
086 % revised by pab 01.12.99
087 %  added gamma to data input
088 % revised by pab 11.08.99
089 % changed so that parameters are only dependent on the
090 % seastate parameters Hm0 and Tp.
091 % also checks if Hm0 and Tp are reasonable.
092
093
094 %NOTE: In order to calculate the short term statistics of the
      response,
095 %      it is extremely important that the resolution of the
     transfer
096 %      function is sufficiently good. In addition, the transfer
    function
097 %      must cover a sufficietn range of wave periods, especially
     in the
098 %      range where the wave spectrum contains most of its
099 %      energy. VIOLATION OF THIS MAY LEAD TO MEANINGLESS RESULTS
      FROM THE
100 %      CALCULATIONS OF SHORT TERM STATISTICS. The highest wave
period
101 %      should therefore be at least 2.5 to 3 times the highest
peak
102 %      period in the transfer function. The lowest period should
be selected
103 %      so that the transfer function value is low. This low
range is
104 %      especially important when studying velocities and
accelerations.
105
106 monitor=0;
107
108 if nargin<3|isempty(plotflag),  plotflag=0;end
109
110 Hm0=7;Tp=11; gam=0; sa=0.07; sb=0.09; A=-1;% default values
111 data2=[Hm0 Tp gam sa sb A];
112 nd2=length(data2);
113 if (nargin>1) & ~isempty(sdata),
114   nd=length(sdata);
115   ind=find(~isnan(sdata(1:min(nd,nd2))));
116   if any(ind) % replace default values with those from input
data
117     data2(ind)=sdata(ind);
```

```
118   end
119 end
120 if (nd2>0) & (data2(1)>0),
121   Hm0 = data2(1);
122 end
123 if (nd2>1) & (data2(2)>0),
124   Tp = data2(2);
125 end
126 if (nd2>2) & (data2(3)>=1) & (data2(3)<=7),
127   gam = data2(3);
128 end
129 if (nd2>3) & (data2(4)>0),
130   sa = data2(4);
131 end
132 if (nd2>4) & (data2(5)>0),
133   sb = data2(5);
134 end
135 if (nd2>5) ,
136   A = data2(6);
137 end
138
139 w = [];
140 if nargin<1|isempty(w1),
141   wc = 33/Tp;
142 elseif length(w1)==1,
143   wc = w1;
144 else
145   w = w1 ;
146 end
147 nw = 257;
148 if isempty(w),
149   w = linspace(0,wc,nw).';
150 end
151
152
153 n=length(w);
154 S1=createspec;
155 S1.S=zeros(n,1);
156 S1.w=w(:);
157 S1.norm=0; % The spectrum is not normalized
158 S1.note=['JONSWAP, Hm0 = ' num2str(Hm0)  ', Tp = ' num2str(Tp)];
159
160 M=4;
161 N=5;
162 wp=2*pi/Tp;
163
164
165 if gam<1
166   gam = getjonswappeakedness(Hm0,Tp);
167 end
168 S1.note=[S1.note ', gamma = ' num2str(gam)];
169 %end
```

```
170
171
172
173 if Tp>5*sqrt(Hm0) | Tp<3.6*sqrt(Hm0)
174   disp('Warning: Hm0,Tp is outside the JONSWAP range')
175   disp('The validity of the spectral density is questionable')
176 end
177 if gam>7|gam<1
178   disp('Warning: gamma is outside the valid range')
179   disp('The validity of the spectral density is questionable')
180 end
181
182
183 % for w>wp
184 k=(w>wp);
185 S1.S(k)=1./(w(k).^N).*(gam.^(exp(-(w(k)/wp-1).^2 ...
186                 /(2*sb^2)))).*exp(-N/M*(wp./w(k)).^M);
187 % for 0<w<=wp
188 k=~k;
189 k(1)=k(1)*(w(1)>0); % avoid division by zero
190 S1.S(k)=1./(w(k).^N).*(gam.^(exp(-(w(k)/wp-1).^2 ...
191     /(2*sa^2)))).*exp(-N/M*(wp./w(k)).^M);
192
193
194 g=gravity; % acceleration of gravity
195
196 if A<0, % normalizing by integration
197   A=(Hm0/g)^2/16/simpson(w,S1.S);% make sure m0=Hm0^2/16=int
S(w)dw
198 elseif A==0,% original normalization
199   % NOTE: that  Hm0^2/16 generally is not equal to intS(w)dw
200   %       with this definition of A if sa or sb are changed from
the
201   %       default values
202   A=5.061*Hm0^2/Tp^4*(1-0.287*log(gam)); % approx D
203 end
204
205 S1.S=S1.S*A*g^2; %normalization
206
207 if monitor
208   D=max(0,0.036-0.0056*Tp/sqrt(Hm0)); % approx
5.061*Hm0^2/Tp^4*(1-0.287*log(gam));
209   disp(['sa, sb       = ' num2str([sa sb])])
210   disp(['alpha, gamma = ' num2str([A gam])])
211   disp(['Hm0, Tp      = ' num2str([Hm0 Tp])])
212   disp(['D            = ' num2str(D)])
213 end
214
215 if plotflag
216   wspecplot(S1,plotflag)
217 end
218
```

Lampiran 3. Scrift spektrum JONSWAP yang digunakan.

```
Hm0 = 7; Tp = 12; plotflag = 1; clf
ST = jonswap([],[Hm0 Tp],plotflag);
plotspec(ST,plotflag);
ylim(['auto']);
grid on
figure
dt = 0.1; N = 2000;
xs = spec2sdat(ST,N,dt); clf
waveplot(xs,'-')
ylim(['auto']);
grid on

%% stat
S = jonswap;
[ch, R, txt] = spec2char(S,[1 2 3 4 5 6]);
ch0 = cell2struct(num2cell(ch),txt,2);
```

Lampiran 4. *Source code* **Spec2sdat** yang diakses melalui

http://www.maths.lth.se/matstat/wafo/documentation/wafodoc/wafo/wsim/spec2sdat.html

```
function [x,xder]=spec2sdat(S,np,dt,iseed,method)
%SPEC2SDAT Simulates a Gaussian process and its derivative from
spectrum
%
%   CALL: [xs, xsder] = spec2sdat(S,[np cases],dt,iseed,method);
%
%         xs    = a cases+1 column matrix  ( t,X1(t) X2(t) ...).
%         xsder = a cases+1 column matrix  ( t,X1'(t) X2'(t) ...).
%         S     = a spectral density structure
%         np    = giving np load points.  (default length(S)-1=n-1).
%                 If np>n-1 it is assummed that R(k)=0 for all k>n-1
%         cases = number of cases, i.e. number of replicates
(default=1)
%            dt = step in grid (default dt is defined by the Nyquist
freq)
%         iseed = starting seed number for the random number
generator
%                 (default none is set)
%        method = 'exact'  : simulation using cov2sdat
%                 'random' : random phase and amplitude simulation
```

```
(default)
%
%  SPEC2SDAT performs a fast and exact simulation of stationary zero
mean
%  Gaussian process through circulant embedding of the covariance
matrix
%  or by summation of sinus functions with random amplitudes and
random
%  phase angle.
%
%  If the spectrum has a non-empty field .tr, then the
transformation is
%  applied to the simulated data, the result is a simulation of a
transformed
%  Gaussian process.
%
%  NB! The method 'exact' simulation may give high frequency ripple
when
%  used with a small dt. In this case the method 'random' works
better.
%
% Example:
%  np =100; dt = .2;
% [x1 x2] = spec2sdat(jonswap,np,dt);
% waveplot(x1,'r',x2,'g',1,1)
%
%  %More extensive test
%  Sj = jonswap
%  [x2, x1] = spec2sdat(Sj,[20000,20]);
%  [sk,ku]= spec2skew(Sj);
%  truth1 = [0, sqrt(spec2mom(Sj,1)), sk, ku-3];
%  funs = {@mean,  @std, @skew, @kurt}
%  for i = 1:4,
%      trueval = truth1(i);
%      fun = funs{i};
%      res = fun(x2(:,2:end), 1);
%      m = mean(res);
%      sa = std(res);
%      [abs(m-trueval)<sa, trueval, m, sa]
%  end
%
% See also  cov2sdat, gaus2dat

% Reference
% C.R Dietrich and G. N. Newsam (1997)
% "Fast and exact simulation of stationary
%  Gaussian process through circulant embedding
%  of the Covariance matrix"
%  SIAM J. SCI. COMPT. Vol 18, No 4, pp. 1088-1107
%
% Hudspeth, R.T. and Borgman, L.E. (1979)
% "Efficient FFT simulation of Digital Time sequences"
```

```
% Journal of the Engineering Mechanics Division, ASCE, Vol. 105, No.
EM2,
%

% Tested on: Matlab 5.3
% History:
% Revised pab Mar2005
%
% Revised pab Feb2004
% - changed seed to state
% revised pab 21.07.2002
% -
% revised pab 12.10.2001
% - added example
% - the derivative, xder, is now calculated
%   correctly using fft for method = 'random'.
%   derivate.m is no longer needed!
% revised pab 11.10.2001
% - added a trick to avoid adding high frequency noise to spectrum
when
%   method = 'exact'
% revised pab 21.01.2001
% - random is now available for 1D wavenumber spectra as well
% revised ir 11.06.00 - introducing dt
% revised es 24.05.00 - fixed default value for np, and small
changes to help
% revised pab 13.03.2000
% - fixed default value for np
% revised pab 24.01.2000
% - added method random from L. Borgman fwavsim (slightly faster and
needs
%   less memory)
% - added iseed
% revised es 12.01.2000:  enable dir. spectrum input (change of
spec2cov call)
% revised pab 12.10.1999
%  simplified call by calling cov2sdat
%    last modified by Per A. Brodtkorb 19.08.98

if nargin<5||isempty(method)
  method='random';% 'exact' is slightly slower than method=='random'
end
if nargin<4||isempty(iseed)
 iseed=[];
else
  try
    randn('state',iseed)
  catch
    randn('seed',iseed); % set the the seed
  end
end
```

```
if nargin<3||isempty(dt)
 S1=S;
else
  S1=specinterp(S,dt); % interpolate spectrum
end

ftype = freqtype(S);
freq  = S.(ftype);

Nt = length(freq);

S = S1;

if nargin<2||isempty(np)
  np=Nt-1;
end
if strcmpi(method,'exact')
  R = spec2cov(S,0,[],1,0,0);

  if strcmpi(ftype,'f')
    T = Nt/(2*freq(end));
  else
    T = Nt*pi/freq(end);
  end
  ltype = lagtype(R);

  ix = find(R.(ltype)>T,'first');

  % Trick to avoid adding high frequency noise to the spectrum
  if ~isempty(ix),
    R.R(ix:end)=0;
  end

  if nargout>1
    [x, xder]=cov2sdat(R,np,iseed);
  else
    x=cov2sdat(R,np,iseed);
  end
  return
end

switch  length(np)
  case 1, cases=1;
  case 2, cases=np(2); np=np(1);
  otherwise, error('Wrong input. Too many arguments')
end
if mod(np,2),
  np=np+1;
end % make sure it is even
```

```
if 0
  % Do the simulation with normalized spectrum
  % Normalize the spectrum
  [Sn,mn4,m0,m2] = specnorm(S);
  % Normalization constants
  tnorm = 2*pi*sqrt(m0/m2);
  xnorm = sqrt( tnorm*m0/(2*pi));
  fs    = Sn.(ftype);
  Si    = Sn.S(2:end-1);
else
  tnorm = 1;
  xnorm = 1;

  fs    = S.(ftype);
  Si    = S.S(2:end-1);

  switch ftype
    case 'f'
    case {'w','k'}
      Si=Si*2*pi;
      fs=fs/2/pi;
    otherwise
      error('Not implemented for wavenumber spectra')
  end
end

x=zeros(np,cases+1);
if nargout==2
  xder=x;
end


dT = 1/(2*fs(end)); % dT

df = 1/(np*dT);


% interpolate for freq.  [1:(N/2)-1]*df and create 2-sided,
uncentered spectra
% -------------------------------------------------------------------
----------
f = (1:(np/2)-1).'*df;

fs(1)   = [];
fs(end) = [];
Fs      = [0; fs(:); (np/2)*df];
Su      = [0; abs(Si(:))/2; 0];
```

```
Si = interp1(Fs,Su,f,'linear');
% Si = interp1q(Fs,Su,f);
Su=[0; Si; 0; Si((np/2)-1:-1:1)];

clear Si Fs

% Generate standard normal random numbers for the simulations
% ------------------------------------------------------------
Zr = randn((np/2)+1,cases);
Zi = [zeros(1,cases); randn((np/2)-1,cases); zeros(1,cases)];

A                = zeros(np,cases);
A(1:(np/2+1),:)  = Zr - sqrt(-1)*Zi; clear Zr Zi
A((np/2+2):np,:) = conj(A(np/2:-1:2,:));
A(1,:)           = A(1,:)*sqrt(2);
A((np/2)+1,:)    = A((np/2)+1,:)*sqrt(2);


% Make simulated time series
% --------------------------

T    = (np-1)*dT;
Ssqr = sqrt(Su*df/2);

% stochastic amplitude
A    = A.*Ssqr(:,ones(1,cases));


% Deterministic amplitude
%A     =  sqrt(2)*Ssqr(:,ones(1,cases)).*exp(sqrt(-
1)*atan2(imag(A),real(A)));
clear Su Ssqr


x(:,2:end) = xnorm*real(fft(A));

x(:,1)     = linspace(0,T*tnorm,np)'; %(0:dT:(np-1)*dT).';




if nargout==2,
  %xder=derivate(x(:,1),x(:,2:end));

  % new call pab 12.10.2001
  w = 2*pi*[0; f;0;-f(end:-1:1)] ;
  A = -i*A.*w(:,ones(1,cases));
```

```
  xder(:,2:(cases+1)) = real(fft(A))*xnorm/tnorm;
  xder(:,1)           = x(:,1);
end


if isfield(S,'tr') && ~isempty(S.tr)
  disp('   Transforming data.')
  g=S.tr;
  G=fliplr(g); % the invers of g
  if nargout==2, % gaus2dat
    for ix=1:cases
      tmp=tranproc([x(:,ix+1) xder(:,ix+1)],G);
      x(:,ix+1)=tmp(:,1);
      xder(:,ix+1)=tmp(:,2);
    end
  else
    for ix=1:cases % gaus2dat
      x(:,ix+1)=tranproc(x(:,ix+1),G);
    end
  end
end



return

% Old call kept just in case



% df=1/(np*dT);
%
%
% % Interpolate for freq.  [1:(N/2)-1]*df and create 2-sided,
uncentered spectra
% % ------------------------------------------------------------------
------------
% f=(1:(np/2)-1)'*df;
%
% fs(1)=[];fs(end)=[];
% Fs=[0; fs(:); (np/2)*df];
% Su=[0; Si(:); 0];
% Si = interp1(Fs,Su,f,'linear')/2;
%
% Su=[0; Si; 0; Si((np/2)-1:-1:1)];
% clear Si
%
% % Generate standard normal random numbers for the simulations
% % ---------------------------------------------------------
% %Zr=randn((np/2)+1,cases);
```

```
% %Zi=[zeros(1,cases); randn((np/2)-1,cases); zeros(1,cases)];
% %AA=Zr-sqrt(-1)*Zi;
%
% AA = randn((np/2)+1,cases)-sqrt(-1)*[zeros(1,cases); randn((np/2)-
1,cases); zeros(1,cases)];
% A=[AA;  conj(AA(np/2:-1:2,:))];
% clear AA
% A(1,:)=A(1,:)*sqrt(2);
% A((np/2)+1,:)=A((np/2)+1,:)*sqrt(2);
%
% % Make simulated time series
% % --------------------------
% T    = np*dT;
% Ssqr = sqrt(Su*df/2);
% A    = A.*Ssqr(:,ones(1,cases));
% clear Su Ssqr
%
% x(:,2:end)=real(fft(A));
%
% x(:,1)=(0:dT:(np-1)*dT).';
```

Lampiran 5. Spec2sdat scrift

```
clc
clear
close all

%% construct spectrum and plot
Hm0 = 7;    % m
Tp = 12;     % s
plotflag = 1;
ST = jonswap([],[Hm0 Tp],plotflag);
plotspec(ST,plotflag);
xlim([0 3]);
%% stat
[ch, R, txt] = spec2char(ST,[1 2 3 4 5 6]);
ch0 = cell2struct(num2cell(ch),txt,2)

%% construct data from spectrum and plot
dt = 0.1; N = 20000;
xs = spec2sdat(ST,N,dt);

%% plot wave
figure
waveplot(xs(1:3600,:),'-')

%% extract period
[T, ind] = dat2wa(xs,0,'c2c'); % Returns crest2crest waveperiods
figure;
```

```
bins = 20;
histgrm(T,bins);
xlabel('T(s)');
ylabel('Occurence(count)');

Tsort = sort(T,'descend');
Tm = mean(T);
Tp = T-Tm;
n = size(Tsort,1);
n13 = ceil(1/3*n);  T13 = mean(Tsort(1:n13));
n10 = ceil(1/10*n); T10 = mean(Tsort(1:n10));

disp(['Tmean = ',num2str(mean(Tsort))]);
disp(['T1/3  = ',num2str(T13)]);
disp(['T1/10 = ',num2str(T10)]);
disp(['Tmax  = ',num2str(Tsort(1))]);

%% extract steep and wave height
rate=8; method=1;
[S,H] = dat2steep(xs,rate,method);
figure
bins = 20;
histgrm(H);
xlabel('H(m)');
ylabel('Occurence(count)');
xlim([0 inf]);

%% calculate Hm0, H1/3, H1/10
Hsort = sort(H,'descend');
hm = mean(H);
hp = H-hm;
n = size(Hsort,1);
n13 = ceil(1/3*n);  H13 = mean(Hsort(1:n13));
n10 = ceil(1/10*n); H10 = mean(Hsort(1:n10));

disp(['Hmean = ',num2str(mean(Hsort))]);
disp(['H1/3  = ',num2str(H13)]);
disp(['H1/10 = ',num2str(H10)]);
disp(['Hmax  = ',num2str(Hsort(1))]);

%% scatter wave height vs period
figure
scatter(H(2:end),T,'o');
xlabel('H(m)');
ylabel('T(s)');

%% scatter steep vs wave height
scatter(S,H,'o');
xlabel('Steepness');
ylabel('H(m)');
```

Lampiran 6. *Source code* **Dat2spec**

http://www.maths.lth.se/matstat/wafo/documentation/wafodoc/wafo/onedim/dat2spec.html

```
function [S,fcut] = dat2spec(xn,varargin)
%DAT2SPEC Estimate one-sided spectral density from data.
%
% CALL:  S = dat2spec(x,L,g,plotflag,p,method,dflag,ftype)
%
%         S = A structure containing:
%             S    = spectral density
%             w    = angular frequency
%             tr   = transformation g
%             h    = water depth (default inf)
%             type = 'freq'
%             note = Memorandum string
%             date = Date and time of creation
%             L    = maximum lag size of the window function.
%             CI   = lower and upper confidence constant
%             p    = confidence level. (Default 0.95).
%             Bw   = Bandwidth of the smoothing window which is used
%                    in the estimated spectrum. (rad/sec or Hz)
%
%         x =  m column data matrix with sampled times in the first
column
%              and values the next columns.
%
%         L = maximum lag size of the window function.
%             If no value is given the lag size is set to
%             be the lag where the auto correlation is less than
%             2 standard deviations. (maximum 300)
%
%         g = the transformation assuming that x is a sample of a
%             transformed Gaussian process. If g is empty then
%             x  is a sample of a Gaussian process (Default)
%
%  plotflag = 1 plots the spectrum, S,
%             2 plot 10log10(S) and
%             3 plots both the above plots
%
%  Method   = 'cov'   Frequency smoothing using a parzen window
function
%                     on the estimated autocovariance function.
(default)
%             'psd'   Welch's averaged periodogram method with no
overlapping
%                     batches
%             'psdo'  Welch's averaged periodogram method with
overlapping
%                     batches
```

```
%              'pmem'  Maximum Entropy Method (psd using the Yule-
Walker
%                      AR method)
%              'pburg' Burg's method.
%
%  dflag    = specifies a detrending performed on the signal before
estimation.
%              'mean','linear' or 'ma' (= moving average)  (default
'mean')
%  ftype    = frequency type, 'w' or 'f'  (default 'w')
%
% Method == 'cov','psd':
%  As L decreases the estimate becomes smoother and Bw increases. If
we
%  want to resolve peaks in S which is Bf (Hz or rad/sec) apart then
Bw < Bf.
%
% Method == 'pmem','pburg':
%  L denotes the order of the AR (AutoRegressive) model.
%
%  NOTE: The strings method,dflag and ftype may be given anywhere
after x
%        and in any order.
%
% Example
%  x = load('sea.dat');
%  S = dat2spec(x);
%  plotspec(S);
%
%  close all;
%
% See also   dat2tr, dat2cov

% Secret option: if chopOffHighFreq==1
%  Chop off high frequencies in order to
%  get the same irregularity factor in the spectrum as
%  in x.



% References:
% Georg Lindgren, Holger Rootzen, Maria Sandsten (2013), Chapman &
Hall.
% "Stationary stochastic processes for scientists and engineers",
Ch 9.
%
% Gareth Janacek and Louise Swift (1993)
% "TIME SERIES forecasting, simulation, applications",
% pp 75--76 and 261--268
%
% Emanuel Parzen (1962),
% "Stochastic Processes", HOLDEN-DAY,
```

```
% pp 66--103

%

% Tested on: Matlab 5.3
% history:
% revised pab 03.11.2000
% - changed call from chi2inv to wchi2inv
% revised pab 10.07.00
% - fixed a bug for m>2 and g is given: replaced dat2gaus call with
tranproc
% revised pab 12.06.2000
% - added method,dflag,ftype to input arguments
% - removed dT wdef from input arguments
% revised pab 14.02.2000
%  - added rate for interpolation in frequency domain
% revised pab 20.01.2000
%  - added detrending option linear, ma mean
%  - added tapering of data before estimation
% Modified by Per A. Brodtkorb 14.08.98,25.05.98
%  - add a nugget effect to ensure that round off errors
%    do not result in negative spectral estimates
% Modified by svi 29.09.99
%  - The program is not estimating transformation g any more.
% modified pab 22.10.1999
%  - fixed so that x in fact can be a m column matrix
%  - updated info put into the spectral structure.
%  - updated help header
% modified pab 03.11.1999
%  - fixed a bug: line 152 wrong array dim. when m>2


% Initialize constants
%~~~~~~~~~~~~~~~~~~~~
nugget   = 0; %10^-12;
rate     = 2; % interpolationrate for frequency
tapery   = 0; % taper the data before the analysis
wdef     = 1; % 1=parzen window 2=hanning window, 3= bartlett window

% Default values:
%~~~~~~~~~~~~~~~~
L        = [];
g        = [];
plotflag = 0;
p        = 0.95;
chopOffHighFreq=0;   % chop off high frequencies in order to get the
same
                     % irregularity factor in the spectrum as in the
data
                     % may not be a good idea => default is 0
```

```
method   = 'cov';  % cov. other options from signal toolbox: psd =
welch's method, pyulear,pmem =
                  % maximum entropy method
dflag    = 'mean'; %'ma','linear' 'mean','none' % detrending option
ftype    = 'w'  ;  %options are 'f' and 'w'


P  = varargin;
Np = length(P);
if Np>0
  strix = zeros(1,Np);
  for ix=1:Np, % finding symbol strings
    strix(ix)=ischar(P{ix});
  end
  k = find(strix);
  Nk = length(k);
  if Nk>0
    if Nk>3,
      warning('WAFO:DAT2SPEC','More than 3 strings are not
allowed'),
    end
    for ix = k
      switch lower(P{ix})
        case {'f','w'},                                    ftype  =
P{ix};
        case {'cov','pmem','mem','psd','psdo','pburg'}, method =
P{ix};
        case {'mean','ma','linear','none'},                dflag =
P{ix};
        otherwise,
          warning('WAFO:DAT2SPEC',['Unknown option:' P{ix}])
      end
    end
    Np = Np-Nk;
    P  = {P{find(~strix)}}; % remove strings from input
  end

  if Np>0 && ~isempty(P{1}), L        = P{1};end
  if Np>1 && ~isempty(P{2}), g        = P{2};end
  if Np>2 && ~isempty(P{3}), plotflag = P{3};end
  if Np>3 && ~isempty(P{4}), p        = P{4};end
  if Np>4 && ~isempty(P{5}), chopOffHighFreq = P{5};end
end


%[L,g,plotflag,p,method,dflag,ftype,chopOffHighFreq] =
d2schk(varargin);


if (nargout == 0) && (plotflag==0)
```

```
  plotflag = 1;
end

xx     = xn;
[n m]  = size(xx);

if min(m,n)==1,
  xx = [ (1:n)' xx(:)];
  n  = max(m,n);
  m  = 2;
  disp('Warning: The sampling frequency is undetermined and set to 1
Hz.')
end
dT = xx(2,1)-xx(1,1);
L = min(L,n);

if  (isempty(g)),
  yy = xx;
else
  yy = xx;
  for ix=2:m
    yy(:,ix)= tranproc(xx(:,ix),g);
  end
  %yy = dat2gaus(xx,g);
end

%display('****1');
%break;
switch lower(dflag)
  case 'mean',
    ma        = mean(yy(:,2:m));
    yy(:,2:m) = (yy(:,2:m)-ma(ones(n,1),:) );
  case 'linear',
    yy(:,2:m) = detrend(yy(:,2:m),1); % signal toolbox detrend
  case 'ma',
    dL        = ceil(1200/2/dT); % approximately 20 min. moving
average
    yy(:,2:m) = detrendma(yy(:,2:m),dL);
    dflag     ='mean';
end
% By using a tapered data window to smooth the data at each
% end of the record has the effect of sharpening
% the spectral window.
% NB! the resulting spectral estimate must be
% normalized in order to correct for the loss of
% amplitude (energy) caused by the data taper.
sa = std(yy(:,2:m));
if tapery
 taper     = bingham(n);
 yy(:,2:m) = taper(:,ones(1,m-1)).*yy(:,2:m);
end
```

```
max_L     = min(300,n); % maximum lag if L is undetermined
changed_L = 0;
if isempty(L)
  L = min(n-2,ceil(4/3*max_L));
  changed_L = 1;
end

if strcmp(method,'cov') || changed_L,
  r=0;
  stdev=0;
  for ix=2:m
    R = dat2cov(yy(:,[1 ix]));
    r = r+R.R(:);
    stdev = stdev+R.stdev(:);
  end
  r       = r/(m-1);
  R.stdev = mean(sa.^2)/r(1)*stdev/(m-1);
  r       = r*mean(sa.^2)/r(1);
  R.R     = r;
  %covplot(R,150)
  if   changed_L,
    %finding where ACF is less than 2 st. deviations.
    L = find(abs(r(1:max_L))>2*R.stdev(1:max_L))+1; % a better L
value
    L = min(L(end),max_L);

    if wdef==1   % modify L so that hanning and Parzen give appr.
the same result
      L = min(floor(4*L/3),n-2);
    end
    disp(['The default L is set to ' num2str(L) ])
  end
end

if wdef==1              % Parzen window
  v   = floor(3.71*n/L);   % degrees of freedom used in chi^2
distribution
  win = parzen(2*L-1);     % Does not give negative estimates of the
spectral density
  Be  = 2*pi*1.33/(L*dT);  % bandwidth (rad/sec)

elseif wdef==2          % Hanning window
  v   = floor(2.67*n/L);   % degrees of freedom used in chi^2
distribution
  win = hanning(2*L-1);    % May give negative estimates of the
spectral density
  Be  = 2*pi/(L*dT);       % bandwidth (rad/sec)

else wdef==3             % Bartlett window
  v   = floor(3*n/L);    % degrees of freedom used in chi^2
```

```
distribution
     win = bartlett(2*L-1);
  Be  = 2*pi*1.33/(L*dT);  % bandwidth (rad/sec)
end

nf   = rate*2^nextpow2(2*L-2);  %  Interpolate the spectrum with
rate
nfft = 2*nf;

S      = createspec('freq',ftype);
S.tr   = g;
S.note = ['dat2spec(',inputname(1),'), Method = ' method ];
S.norm = 0; % not normalized
S.L    = L;



S.S = zeros(nf+1,m-1);

switch lower(method)
  case {'psd','psdo'} % from signal toolbox
    noverlap   = 0;
    if method(end)=='o',
      noverlap   = floor(L/2);
    end
    S.noverlap = noverlap;
    if 1,
      vararg = cell(1,3*~isempty(p));
      [Rper
vararg{:}]=welch_psd(yy(:,2:m),'nfft',nfft,'window',win,'overlap',no
verlap,'p',p,'dflag',dflag);
      if m>2
        Rper = mean(Rper,2);
      end
      if ~isempty(p)
        Sc = [mean(vararg{3}(:,1:m),2),mean(vararg{3}(:,m+1:end),2)]
;
        [maxS ind] = max(Rper);
        S.CI = Sc(ind,:)/Rper(ind);
        S.p  = p;
      end
    else
      % from signal toolbox
      [Rper,Sc f] = psd(yy(:,ix),nfft,1/dT,win,noverlap,p,dflag);
      for ix=3:m
        Rper = Rper + psd(yy(:,ix),nfft,1/dT,win,noverlap,p,dflag);
      end
      Rper = Rper/(m-1);

     if (  ~isempty(p) ),       % Confidence interval constants
```

```
        [maxS ind] = max(Rper);
        S.CI = Sc(ind,:)/Rper(ind);
        S.p  = p;
      end
    end
  case {'pyulear','pmem','mem'} % from signal toolbox
    Rper = pyulear(yy(:,2),L,nfft,1/dT);
    for ix=3:m
      Rper = Rper + pyulear(yy(:,ix),L,nfft,1/dT);
    end
    Rper = Rper/(m-1);
  case 'pburg',  % from signal toolbox
    Rper = pburg(yy(:,2),L,nfft,1/dT);
    for ix=3:m
      Rper = Rper + pburg(yy(:,ix),L,nfft,1/dT);
    end
    Rper = Rper/(m-1);
  otherwise, % cov method
  % add a nugget effect to ensure that round off errors
  % do not result in negative spectral estimates
  r    = r+nugget;
  rwin = r(1:L).*win(L:(2*L-1));
  Rper = real(fft([rwin; zeros(nfft-(2*L-1),1); rwin(L:-
1:2)],nfft));
  %f    =  [0:(nf)]'/nf/(2*dT);
  if (  ~isempty(p) ),
    alpha = (1-p);
    % Confidence interval constants
    S.CI = [v/invchi2( 1-alpha/2 ,v) v/invchi2( alpha/2 ,v)];
    S.p  = p;
  end
end

ind = find(Rper<0);
if any(ind)
  Rper(ind) = 0; % set negative values to zero
  warning('WAFO:DAT2SPEC','negative spectral estimates')
end

if strcmp(ftype,'w')
  S.w  = (0:nf)'/nf*pi/dT;           % (rad/s)
  S.S  = real(Rper(1:(nf+1),1))*dT/pi; % (m^2*s/rad)one sided
spectrum
  S.Bw = Be;
else % ftype == f
  S.f  = (0:nf)'/nf/2/dT;            % frequency Hz if dT is in
seconds
  S.S  = 2*real(Rper(1:(nf+1),1))*dT;  % (m^2*s) one sided spectrum
  S.Bw = Be/(2*pi);                    % bandwidth in Hz
end
```

```
N = floor(nf/10);
% cutting off high frequencies
% in this way may not be a very good idea.

if ((N>3) && chopOffHighFreq),
  % The data must be Gaussian in order for this proc to be correct.
  [g0 test cmax irr]  = dat2tr(xx,'nonlinear');
  ind=(nf-4):(nf+1);

  [Sn,m4] = specnorm(S,0);

  while (sqrt(m4) > irr) && (ind(1)>1)
    S.S(ind)    = 0;
    [Sn,m4]     = specnorm(S,0);
    ind         = ind-5;
  end
  fcut = S.w(min(ind(end)+5,nf)); % cut off frequency
  if ind(1)<1,
    disp('DAT2SPEC: Error in cutting off high frequencies, try other
L-values')
  end
end



%-----------------------------------
%
%   Plotting the Spectral Density
%
%-----------------------------------

if plotflag>0
 plotspec(S,plotflag)
end

return

function [L,g,plotflag,p,method,dflag,ftype,chopOffHighFreq] =
d2schk(P)
% D2SCHK Helper function for dat2spec.
%
% CALL
[L,g,plotflag,p,method,dflag,ftype,chopOffHighFreq]=d2schk(P)
%
%   P = the cell array P of input arguments (between 0 and 7
elements)
%  xx = must be a two column vector.
%
```

```
% Default values:
%~~~~~~~~~~~~~~~~
L        = [];
g        = [];
plotflag = 0;
p        = 0.95;
chopOffHighFreq=0;   % chop off high frequencies in order to get the
same
                     % irregularity factor in the spectrum as in the
data
                     % may not be a good idea => default is 0

method   = 'cov';  % cov. other options from signal toolbox: psd =
welch's method, pyulear,pmem =
                   % maximum entropy method
dflag    = 'mean'; %'ma','linear' 'mean','none' % detrending option
ftype    = 'w'  ;  %options are 'f' and 'w'



Np=length(P);
strix=zeros(1,Np);
for ix=1:Np, % finding symbol strings
 strix(ix)=ischar(P{ix});
end
k = find(strix);
if any(k)
  Nk=length(k);
  if Nk>3
    warning('WAFO:DAT2SPEC','More than 3 strings are not allowed in
')
  end
  for ix = k
    switch lower(P{ix})
    case {'f','w'},                                  ftype  = P{ix};
    case {'cov','pmem','mem','psd','psdo','pburg'}, method = P{ix};
    case {'mean','ma','linear','none'},              dflag = P{ix};
    otherwise,
      warning('WAFO:DAT2SPEC',['Unknown option:' P{ix}])
    end
  end
  Np=Np-Nk;
  P={P{find(~strix)}}; % remove strings from input
end

if Np>0 && ~isempty(P{1}), L        = P{1};end
if Np>1 && ~isempty(P{2}), g        = P{2};end
if Np>2 && ~isempty(P{3}), plotflag = P{3};end
if Np>3 && ~isempty(P{4}), p        = P{4};end
if Np>4 && ~isempty(P{5}), chopOffHighFreq = P{5};end
```

Lampiran 7. Scrift karakteristik statistik

```
waveData = load('sea.dat');
 clf
 waveplot(waveData,1,1,'r-','r');
 wp1 = waveData(1:150,:);
 waveplot(wp1,'r-','bo');
 wp2 = waveData(50:100,:);
 waveplot(wp2,'r-','bo');
 waveMean = mean(waveData(:,2))
 waveStd = std(waveData(:,2))
 waveMax = max(waveData(:,2))
 waveMin = min(waveData(:,2))
waveData(:,2) = waveData(:,2) - waveMean;
lc = dat2lc(waveData);
plotflag = 2;
lcplot(lc,plotflag,0,waveStd)
T = max(waveData(:,1))-min(waveData(:,1))
f0 = interp1(lc(:,1),lc(:,2),0)/T

 rf = dat2cor(waveData,150,2)
%
%   assert(rf.R(1:3)', [ 1.0, 0.931593322326778, 0.764943054479577],
1e-10);
%   close all;

 waveData(:,2) = waveData(:,2) - waveMean;
 lc = dat2lc(waveData);
 plotflag = 2;
 lcplot(lc,plotflag,0,waveSTD)
tp = dat2tp(waveData);
fm = length(tp)/(2*T)             % frequency of maxima
alfa = f0/fm
waveplot(waveData,tp,'k-','+',1,1)
axis([0 2 -inf inf])
dt = diff(waveData(1:2,1));
dcrit = 5*dt;
ddcrit = 9.81/2*dt*dt;
zcrit = 0;
[inds indg] = findoutliers(waveData,zcrit,dcrit,ddcrit);
```

JURUSAN FISIKA
FAKULTAS MIPA
UNIVERSITAS HASANUDDIN
KAMPUS TAMALANREA JL.PERINTIS KEMERDEKAAN.10 MAKASSAR 90245
TELP (0411)587634 fax (0411-587634)

KARTU KONTROL
BIMBINGAN TUGAS AKHIR

NAMA : RUSMIATI
NO.POKOK : H22114006
Program studi : GEOFISIKA
Nama pembimbing T.A : Prof. Dr. Eng, Dadang Ahmad Suriamihardja, M. Eng
- Dr. Muh. Aimuddin Hamzah, M. Eng

NO	HARI/ TANGGAL	KONSULTASI BIMBINGAN TUGAS AKHIR		PARAF/ PEMBIMBING
		MATERI KONSULTASI		
1	Kamis 17/05/2018	Bahas literatur		
2	Senin 17/09/2018	Bab I dan Bab II		
3	Senin 10/12/2018	Bab III		
4	Selasa 05/02/2019	Bab III		
5	Jumat 15/02/2019	Bab I, II, dan III		
6	Senin 25/03/2019	Simulasi Proposal Penelitian		
7	Selasa 18/02/2020	Asistensi hasil		
8	Rabu 19/02/2020	Asistensi hasil		
9	Jumat 28/03/2020	Asistensi hasil		
10	Selasa 03/03/2020	Asistensi hasil (bab IV)		
11	Kamis 12/10/2020	Asistensi pembahasan bab IV		
12	Kamis 29/10/2020	Asistensi pembahasan dan penulisan		
13	Senin 07/12/2020	Asistensi draft		
14				
15				
16				
17				
18				
19				
20				

Makassar.....20
Sekertaris Jurusan Fisika

CATATAN
DIPERBOLEHKAN MELAKSANAKAN SEMINAR I/II
JIKA MENGIKUTI SEMINAR MINIMAL 10 KALI

Syamsuddin,S.Si,MT
NP.197401152002121001