

WAVE Analysis Toolbox
—
a tutorial
(Version 1.1)

for use with Matlab 4.x

Igor Rychlik Georg Lindgren

Department of Mathematical Statistics
University of Lund
Box 118, S-221 00 Lund, SWEDEN



First printing, May, 1995

Second printing, December 5, 1995

University of Lund and Lund Institute of Technology 1995:1
Department of Mathematical Statistics ISSN 0281-1944
ISRN: LUTFD2/TFMS - - 7001 - - SE

TABLE OF CONTENTS

| | |
|--|------------|
| Foreword | iii |
| 1 Random waves | 1 |
| 1.1 Introduction | 1 |
| 1.2 Definition of waves and wave characteristics | 1 |
| 1.3 Installation and technical details | 5 |
| 2 Finding the empirical wave characteristics | 7 |
| 2.1 Loading data and turning points | 7 |
| 2.2 Empirical cycle counts | 9 |
| 2.3 Crossing spectrum | 10 |
| 2.4 Significant waveheight | 11 |
| 2.5 Summary of empirical wave characteristics | 13 |
| 3 Descriptive statistics of wave characteristics | 15 |
| 3.1 Counting distribution | 15 |
| 3.2 Oscillation count | 16 |
| 3.3 Amplitude distribution | 17 |
| 3.4 Significant wave characteristics | 20 |
| 3.5 Discretization schemes | 20 |
| 3.6 Summary of descriptive wave characteristics | 23 |
| 4 Theoretical analysis of Gaussian transform models | 25 |
| 4.1 Stochastic wave models | 25 |
| 4.2 Simple approximations of wave distributions | 28 |
| 4.3 More accurate approximations | 34 |
| 4.4 Summary of Gaussian crest and trough functions | 44 |
| 5 Estimation of the Gaussian transform model | 45 |
| 5.1 The transformation g | 45 |
| 5.2 Estimation of the spectral density S | 46 |
| 5.3 The covariance function | 47 |
| 5.4 Validation of the model | 49 |
| 5.5 Summary of estimation procedures | 52 |

| | | |
|----------|---|-----------|
| 6 | A Markov Chain model for the trough-crest sequence | 53 |
| 6.1 | Rainflow filtered sea wave data | 53 |
| 6.2 | Oscillation count and rainflow matrix | 54 |
| 6.3 | Markov chain of turning points, Markov matrix | 54 |
| 6.4 | Rainflow filtered transformed Gaussian model | 55 |
| 6.5 | Summary of estimation procedures | 59 |
| 7 | Extreme value analysis | 61 |
| 7.1 | Weibull and Gumbel papers | 61 |
| 7.2 | Generalized Pareto and extreme value distribution | 62 |
| 7.3 | POT-analysis | 65 |
| 7.4 | Summary of extreme value procedures | 69 |
| APPENDIX | | |
| A | Some simulation programs for random processes | 71 |
| | References | 73 |
| | Index | 75 |

FOREWORD

This is a manual and tutorial for the `WAVE ANALYSIS TOOLBOX` for use together with `MATLAB`. It consists of a number of `MATLAB` m-files together with executable routines (from `FORTRAN` source), and it requires only a standard `MATLAB` setup, with no additional toolboxes. The routines are based on algorithms for extreme value and crossing analysis, developed over many years by the authors. The routines collected in the toolbox are specially designed for analysis of wave characteristics and they are described in a series of examples on wave data from sea surface measurements.

Many people have contributed to our understanding of the problems dealt with in this text, first of all Professor Ross Leadbetter at the Center for Stochastic Processes, University of North Carolina at Chapel Hill. Much of the text in this version was written while Igor Rychlik was visiting the Department of Mathematics, University of Queensland, Brisbane, Australia, during the spring semester 1995. Both authors are grateful to Dr Richard Wilson, Brisbane, and also to Dr Michel Olagnon, at Institut Francais de Recherches pour l'Exploitation de la Mer, Brest, for many fruitful discussions. Mats Frendahl, Pär Johansson, Finn Lindgren, and Jesper Rydén have contributed with many suggestions on the `MATLAB`-routines and on the presentation.

In this second printing of the tutorial, the call to some of the routines has been changed and a new Chapter 6 has been added.

The routines are available from the authors, under address

Igor Rychlik (or Georg Lindgren)
Department of mathematical statistics
Lund University
Box 118
S-220 00 Lund, Sweden

e-mail: igor@maths.lth.se or georg@maths.lth.se

The work has been supported by the Swedish Research Council for Engineering Sciences (Contract No. 222-93-846), the US Office of Naval Research (Grant N00014-93-1-0841) and by The Swedish Institute for Applied Mathematics (ITM).

CHAPTER 1

RANDOM WAVES

1.1 Introduction

In this manual we present a WAVE ANALYSIS TOOLBOX (WAT) in MATLAB for the analysis of wave characteristics of marine and other random wave data. The toolbox can handle different definitions of wave amplitude and wave length, and it can be used to estimate crossing spectra as well as energy spectra and significant wave height and other wave characteristics from wave data. Furthermore, it can handle non-Gaussian waves by estimating a nonparametric transformation that makes data more Gaussian, and also simulate Gaussian and non-Gaussian wave forms with prescribed crossing and energy spectra. The WAVE ANALYSIS TOOLBOX contains several algorithms for calculation of the theoretical distribution of wave amplitudes and wave length which do not require that the waves have a narrow energy spectrum. Programs for extreme value analysis are also included in the toolbox. A number of examples with ocean wave data are used to present the programs. Theoretical aspects of the algorithms are discussed in papers [10, 20, 21].

This first chapter contains the basic definitions of wave characteristics such as wavelength, amplitude, cycle counts, and crossing spectrum in a data set. Programs which can be used to extract these variables from data are described in Chapter 2. The empirical statistical properties of the wave characteristics can be found by means of programs described in Chapter 3, and in Chapter 4 we describe a number of programs used to calculate the theoretical distributions of variables such as wavelength and amplitude in Gaussian and transformed Gaussian wave models. Chapter 5 contains programs for estimation of wave transformation functions, spectral density and correlations, and in Chapter 6 we consider non-linear filtering of wave data. Chapter 7 finally, describes a number of programs for extreme value analysis.

1.2 Definition of waves and wave characteristics

Trough-crest and crest front

Let $x(t)$ be the height of the sea level at a fixed point as a function of time t . In oceanographic applications $x(t)$ is often seen as a sequence of waves where each wave can be described by means of its highest and lowest values (**crest** and **trough**), or by means of its **amplitude** (= crest - trough) and **wavelength**, describing the range and duration of a single wave. There is no general agreement about the formal definition of a wave. Often one uses the so called

mean downcrossing wave, where the wave is considered as that part of a wave profile that falls between consecutive downcrossings of the mean sea level, as in the following definition.

Definition 1.1 Let $x(t)$, $0 \leq t \leq T$, be a smooth function, in the sense of having a finite number of local extremes, and let u^{tc} be a fixed reference level, usually the mean of $x(t)$. Denote by t_i , $0 < t_1 < t_2 < \dots < t_n < T$, the times of downcrossings of u^{tc} by $x(t)$. The **crest** and **trough**, M_i^{tc} and m_i^{tc} , of the i^{th} wave are the global maximum and the global minimum of $x(t)$, for $t_i < t < t_{i+1}$, respectively. The sequence of (m_i^{tc}, M_i^{tc}) forms a **trough2crest** cycle count. Of special interest is the **crest front amplitude**, H_i^{tc} , defined as the height difference between the crest and the trough, i.e.

$$H_i^{tc} = M_i^{tc} - m_i^{tc} = \max_{t_i < t < t_{i+1}} x(t) - \min_{t_i < t < t_{i+1}} x(t).$$

Further, the **crest front wavelength** T_i^{tc} , of the i^{th} wave is the time distance between the minimum m_i^{tc} and the maximum M_i^{tc} . The terms *trough2crest amplitude* and *wavelength* are used as synonyms of *crest front amplitude* and *wavelength*. We shall also use the **half-wavelength** T_i^{utc} , defined as the distance between a downcrossing and the following upcrossing of u^{tc} .

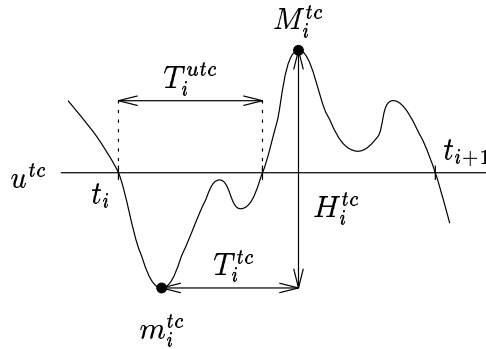


Figure 1.1: Definition of trough2crest waves.

Minimum-maximum waves

An important application of wave analysis is the prediction of big waves, and then one will often neglect small oscillations superimposed on major waves. The use of mean level crossings in the definition of a wave is particularly well suited for such studies. However, such waves are very complicated to analyse theoretically and for this reason we also need a simpler form of waves, consisting of pairs of a minimum and the following maximum.

Definition 1.2 As before, let $x(t)$, $0 \leq t \leq T$, be a smooth function. The times and values of the local extremes in $x(t)$ will be called a sequence of **turning points**. The sequence of heights of local minima and the following maxima in $x(t)$, are denoted by m_i, M_i , $i = 1, 2, \dots, n_{mM}$, respectively, and the i^{th} **min2Max** cycle is the pair (m_i, M_i) . (The number of local maxima is denoted by n_{mM} .) Further, the **min2Max wavelength** and **amplitude** of the i^{th} cycle is the pair $(T_i, H_i = M_i - m_i)$, where T_i is the time distance between the minimum m_i and the following maximum M_i . A **min2Max** cycle (m_i, M_i) with $m_i < u^{tc} < M_i$ will be called a **mean separated min2Max** cycle; here u^{tc} is a reference level, usually the mean level of the process. Similarly, **Max2min** cycles are defined from the pair (M_i, m_i) .

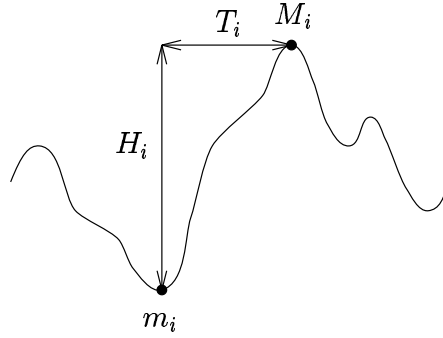


Figure 1.2: *Definition of min2Max waves.*

Rainflow waves

The rainflow method was introduced by Endo, the first paper in English being [2]. Endo's definition was a complicated recursive algorithm identifying closed hysteresis loops in the strain-stress plane. The first nonrecursive definition of the method was given by Rychlik in [16].

Definition 1.3 *To define the rainflow cycles, each local maximum in the load sequence has to be paired with one particular local minimum, found as follows: from a local maximum M_i with height u , say, one shall try to reach above u (in the forward or backward direction) with an as small downward excursion as possible. The minimum $m_i^+ = m_i^{\text{RFC}}$, which represents the smallest deviation from the maximum M_i , is defined to be the corresponding rainflow minimum; see Figure 1.3. Thus the rainflow cycle starting at M_i is (m_i^{RFC}, M_i) .*

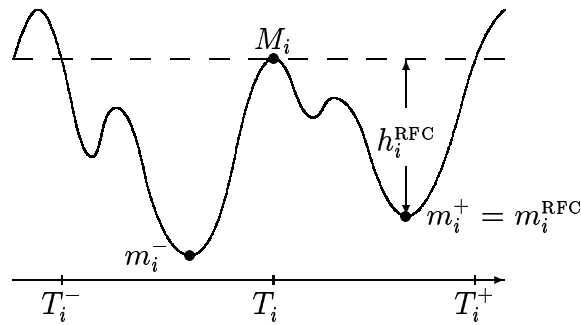


Figure 1.3: *Definition of rainflow cycle.*

Cycle counts and crossing spectrum

Many functions in the toolbox work both for min2Max and for Max2min cycles as well as for trough2crest and rainflow cycles. For simplicity of presentation, we shall define a *cycle count* as a sequence of pairs (x_i, y_i) , say, such that x_i occurs in time before y_i . The generic term cycle count, denoted by cc , will be used when we do not want to distinguish between min2Max, trough2crest, and rainflow cycles.

A somewhat different wave characteristic, the (empirical) *crossing spectrum*, defined as the number of times $x(t)$ upcrosses the level u as a function of u , $N(u)$, say. The crossing spectrum is a very important characteristic of $x(t)$, especially useful in the analysis of the height of big waves. The following alternative definition of $N(u)$ was given in [17],

$$N(u) = \sum_i 1_{(m_i, M_i)}(u), \quad (1.1)$$

where $1_{(a,b)}(u) = 1$, if $a < u < b$, and zero otherwise and (m_i, M_i) is the sequence of min2Max cycles. Obviously $N(u)$ is a stepwise constant discontinuous function. If we make the function left (or right) continuous, then it is enough to save the points of jumps and the corresponding values of $N(u)$ in order to completely reconstruct the entire spectrum. In the WAVE ANALYSIS TOOLBOX we choose such a modified definition of upcrossings given by

$$N(u) = \sum_i 1_{(m_i, M_i]}(u), \quad (1.2)$$

which makes $N(u)$ left continuous. Using formulas (1.1) or (1.2) we obtain that

$$\sum_i (M_i - m_i) = \int N(u) du, \quad (1.3)$$

which relates the observed average min2Max-amplitude $\sum_i (M_i - m_i)/n_{mM}$ to the integral of the crossing spectrum.

The crossing spectrum can be obtained from (1.1) using other cycle counts than min2Max cycles. Obviously, for any cycle count (x_i, y_i) , we can define

$$N^{cc}(u) = \sum_i 1_{(x_i, y_i]}(u);$$

note that for $x_i \geq y_i$ one has $1_{(x_i, y_i]} = 0$. For rainflow cycles we have that $N^{\text{RFC}}(u) = N(u)$, but for trough2crest cycles only the inequality $N^{tc}(u) = \sum_i 1_{(m_i^{tc}, M_i^{tc}]}(u) \leq N(u)$ holds. Since $N^{cc}(u)$ does not uniquely characterize a cycle count we introduce a more complicated "crossing spectrum", which we shall call the *counting distribution*. The counting distribution of a cycle count \mathbf{cc} is defined as follows,

$$N^{cc}(u, v) = \sum_i 1_{(x_i, +\infty)}(u) \cdot 1_{(-\infty, y_i)}(v) = \#\{(x_i, y_i) : x_i < u \text{ and } v < y_i\} \quad (1.4)$$

where $\#\{\cdot\}$ denotes the number of elements in the set $\{\cdot\}$. (Obviously $N^{cc}(u, u) = N^{cc}(u)$.)

As in the case of the crossing spectrum, we have some freedom in the definition of the counting distribution $N^{cc}(u, v)$, and we will also use the alternative formulation

$$N^{cc}(u, v) = \sum_i 1_{[x_i, +\infty)}(u) \cdot 1_{(-\infty, y_i]}(v) = \#\{(x_i, y_i) : x_i \leq u \text{ and } v \leq y_i\}. \quad (1.5)$$

We want to stress here that if the counting distribution $N^{cc}(u, v)$ defined by (1.4) or (1.5) is given then we can retrieve the cycle count \mathbf{cc} from it.

Now, we introduce a generalization of the crossing spectrum, the *oscillation count*, defined as

$$N^{\text{osc}}(u, v) = \text{number of passages from } u \text{ to } v \text{ by } x(t), \quad (1.6)$$

see Figure 1.4.

Obviously, the crossing spectrum can be computed from the oscillation distribution, by $N(u) = N^{\text{osc}}(u, u)$. Furthermore, it can be shown that $N^{\text{osc}}(u, v)$ is a counting distribution for a rainflow count; see [17].

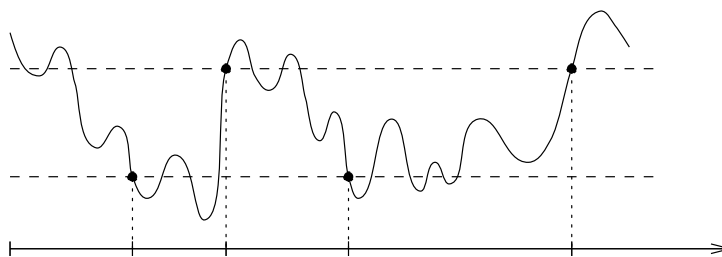


Figure 1.4: *Oscillation count between levels v and u ; here $N^{osc}(u, v) = 2$.*

1.3 Installation and technical details

The toolbox has been run successfully together with MATLAB (version 4.2) on PC with MS-Windows and on SUN and DEC workstations with UNIX. It should be possible to run it on a Macintosh with FORTRAN compiler. It is self contained and uses only routines from the standard MATLAB package. To increase computational speed, a number of routines are given as executable code, e.g. `progrname.exe` (from FORTRAN), which communicates with the MATLAB-routines by writing and reading external ASCII files.

Installation and search path

It is advisable to install the toolbox in a directory of its own, e.g. `C:\MATLAB\TOOLBOX\WAVE`. The WAVE ANALYSIS TOOLBOX m-file `wavepath.m` must be edited by the user and should contain the search path to the directory where the executable files are installed. The path to the WAVE ANALYSIS TOOLBOX m-files should also be added to the MATLAB path.

When the WAVE ANALYSIS TOOLBOX is run on a PC, a DOS-window is opened every time a `progrname.exe` routine is activated. Use the PIF editor in MS-Windows to edit `bang_ml.pif` in `C:\MATLAB\BIN` to close the DOS-window upon program completion if you don't want to do it by hand. Note that this will affect all `*.exe` calls in your MATLABsession.

Fortran codes

If you want to compile the FORTRAN source codes you can use any F77 or F90 compiler. For highest flexibility on a PC one should use a F90 compiler. The main routines are

```
getrfc  gettp  minmax  rfc_filt  simspec  wave_t  wave_th
```

Two packages of subroutines, `twog` and `rind`, have to be compiled and linked to the main routines `minmax`, `wave_t`, and `wave_th`. It is then important that `rind` is compiled and linked as the last package, to guarantee correct allocation of common blocks.

Help

There is a brief description of the routines at the end of each chapter in this tutorial. These descriptions state the simplest form of commands and argument lists. Most of the routines can take many more arguments, and do much more complicated things than described in this tutorial. The reader should consult the appropriate `help` text in MATLAB and the command `help wave`.

We also submit a data file `sea.dat` of wave measurements, used in the examples. The file `wattutor.m` contains all the MATLAB commands used to produce the figures in this tutorial. Open it with your favourite editor and insert the commands into your MATLAB window, or run it by issuing the command `\wattutor`. Beware that some of the accurate calculations in Chapters 4 and 6 can take some time.

CHAPTER 2

FINDING THE EMPIRICAL WAVE CHARACTERISTICS

2.1 Loading data and turning points

Assume that we have measured a sea level at a fixed point during a time interval $[0, T]$. The measurements will be called $x(t)$, $0 \leq t \leq T$. The $x(t)$ function is sampled with a fixed sampling frequency and a given resolution, i.e. the values of $x(t)$ are also discretized. (The effects of sampling can not always be neglected.)

Let the function $x(t)$ be saved as a two column ASCII file named `sea.dat` with time argument in the first column and function values in the second column. The data used in the examples are measurements at shallow water location, sampled with a sampling frequency of 4 Hz, and the units of measurement are seconds and meters. The file `sea.dat` is loaded into MATLAB and after the mean value has been subtracted the data are saved in the two column matrix `xx`,

```
>> load sea.dat; xx = sea;
>> xx(:,2) = sea(:,2) - mean(sea(:,2));
>> clear sea
>> plot(xx(1:400,1),xx(1:400,2)); hold
```

A part of the sea data is presented in Figure 2.1.

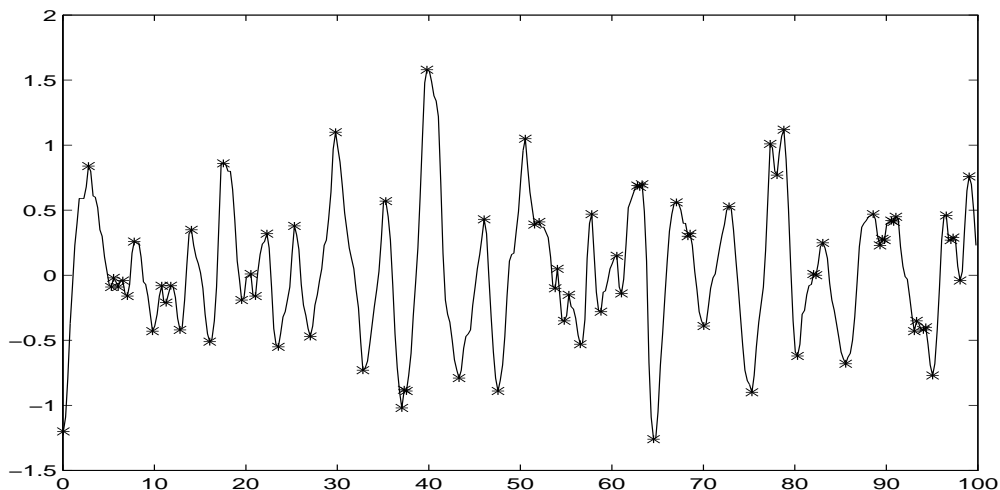


Figure 2.1: *A part of the sea data with turning points marked as stars.*

The first reduction of data is obtained by extracting the sequence of turning points TP from the load. This can be done using the function `dat2tp`,

```
>> TP = dat2tp(xx); plot(TP(1:80,1),TP(1:80,2),'*')
```

In noisy wave data the sequence of turning points will contain a large number of small cycles. These may be of little relevance for the further analysis, and should then be removed. This can be achieved by a second argument in the function `dat2tp(xx,h)`, which performs a *rainflow filtering* of the sequence. This is a nonlinear filter defined so that those local extremes in `xx` which are paired into rainflow cycles with amplitude less than `h` are removed from the sequence. More precisely, if `h < 0` then `dat2tp(xx,h)` returns `xx`, if `h = 0` (the default value) then it returns the sequence of turning points, and if `h > 0` cycles with amplitude less than `h` are removed. In Figure 2.2 we show the result of the following filtering commands.

```
>> TP_025 = dat2tp(xx, 0.25);
>> plot(TP_025(1:55,1), TP_025(1:55,2)); hold
>> plot(TP(:,1),TP(:,2),'-'); axis([0 30 -1.5 1.5])
>> TP_1 = dat2tp(xx, 1);
>> plot(TP_1(1:55,1), TP_1(1:55,2)); hold
>> plot(TP(:,1),TP(:,2),'-'); axis([0 30 -1.5 1.5])
```

It is easy to see that rainflow filtering is a similar kind of operation as gives the mean upcrossing waves. However, the rainflow filtering is a more flexible way to extract large waves. We return to the analysis of rainflow filtered data in Chapter 6.

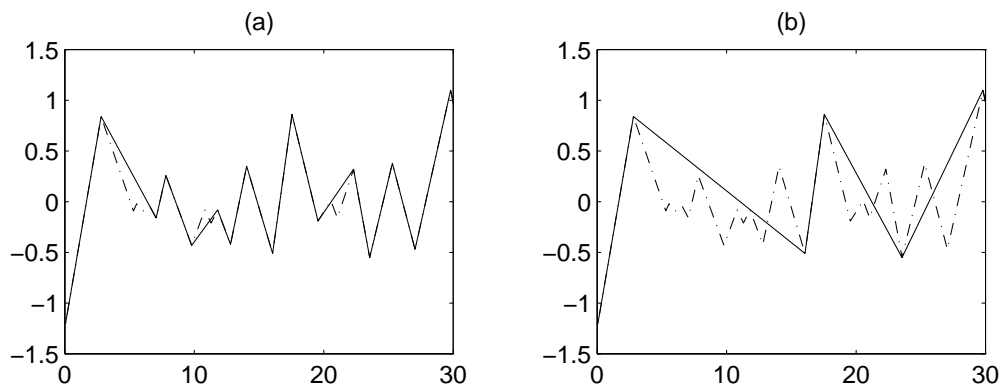


Figure 2.2: Turning points (dashdotted) and rainflow filtered sequences of turning points (solid) ; (a) $h = 0.25$, (b) $h = 1$.

By transforming `xx` into TP we have lost some information on how $x(t)$ develops between the local extremes, see Figure 2.1. However, we have considerably reduced the amount of input data. The sequences of min2Max cycles (m_i, M_i) , trough2crest cycles (m_i^{tc}, M_i^{tc}) , the rainflow cycles (m_i^{RFC}, M_i) , the min2Max and crest front wavelengths T and T^{tc} , and the crossing spectrum can all be calculated from the sequence of turning points alone. It is only for the half wavelength and for estimation of a power spectrum density that we need the

original sequence of measurements \mathbf{xx} . (The half wavelength T_hw can be extracted from data by means of the command $T_hw=dat2hwa(\mathbf{xx}, utc)$ with utc as the reference level.)

2.2 Empirical cycle counts

Denote the sequence of min2Max cycles by mM , the trough2crest cycles by TC , the rainflow cycles by RFC , the min2Max wavelengths by T_mM , and crest front wavelengths by T_tc . These sequences can be extracted from the sequence of turning points TP using the following WAVE ANALYSIS TOOLBOX functions (here with a reference level $u^{tc} = 0$),

```
>> mM = tp2mm(TP);
>> [TC T_tc] = tp2tc(TP,0);
>> RFC = tp2rfc(TP);
>> T_mM = tp2wa(TP);
```

The min2Max and crest front amplitudes denoted by H , H_tc , respectively, can be obtained by

```
>> H = mM(:,2) - mM(:,1);
>> H_tc = TC(:,2) - TC(:,1);
```

We now turn to the visualization of min2Max and trough2crest cycles. Consider a sequence of cycles (x_i, y_i) . The cycle (x_i, y_i) can be identified with a point in the plane with coordinates (x_i, y_i) . Consequently, a sequence of cycles becomes a cloud of dots in R^2 , and the shape of this cloud is very characteristic for different types of cycle distributions. Cycle counts can be plotted in the WAVE ANALYSIS TOOLBOX by means of a function `ccplot`. With two arguments it produces two plots for comparison of two sets of cycle counts. (A third argument defines the marker size.) The cycle counts in Figure 2.3, are produced by

```
>> ccplot(mM,TC,6)
```

The most severe waves in the sea data are represented by the dots in the NW corner of the plots in Figure 2.3. The dots along the SW-NE diagonal represent waves with very small amplitudes. The proportion of waves with small amplitudes is measured by the so called *irregularity factor*. More precisely, the irregularity factor of \mathbf{xx} is the ratio of the number of mean level upcrossings and the number of local maxima in \mathbf{xx} . Thus, since $u^{tc} = 0$, the irregularity factor `irr` can be computed as follows,

```
>> irr = length(TC)/length(mM)
```

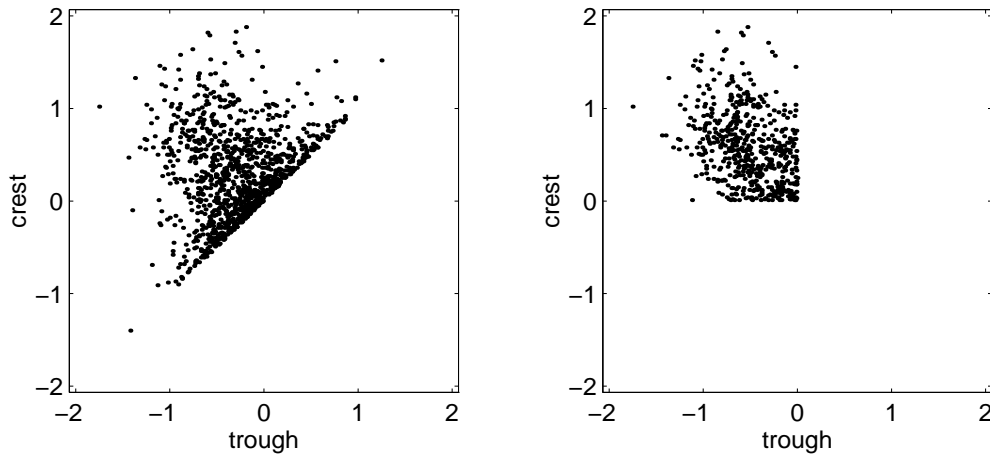


Figure 2.3: A *min2Max* cycle count, left plot, and *trough2crest* cycle count, right plot.

The number of *trough2crest* cycles (i.e. `length(TC)`) is equal to the number of *min2Max* cycles (m_i, M_i) such that $m_i < 0 < M_i$, and hence, equal to the number of dots in the set $\{(x, y) : x < 0 < y\}$ in the plane. Consequently, the irregularity factor of `xx` is the ratio of the numbers of dots in the quadrant $\{(x, y) : x < 0 < y\}$ and the total number of all dots; cf. Figure 2.3.

If only the extreme wave properties are of interest one should use the *trough2crest* cycles since they eliminate all *min2Max* cycles with small amplitudes. In general, the extreme properties of $x(t)$ are well described by the *trough2crest* cycles. However, the probabilistic properties of the *trough2crest* cycles are much more difficult to analyse than the properties of the *min2Max* cycles, and one may ask if it is possible to use the *min2Max* cycles (m_i, M_i) with $m_i < 0 < M_i$ instead of the *trough2crest* cycles. These cycles were called *mean separated min2Max* cycles in Definition 1.2, and they can be extracted from the sequence of *min2Max* cycles `mM` by the following instructions in the `WAVE ANALYSIS TOOLBOX` – see Figure 2.4 where we compare `TC` with `mM_np`,

```
>> index=find(mM(:,1)<0 & mM(:,2)>0);
>> mM_np = mM(index,:);
>> ccplot(TC,mM_np,6)
```

The `mM_np` cycle count is a first order approximation of the `TC` count in the sense that it has the same number of cycles as `TC` and that the counting distribution of `mM_np`, $N^{np}(u, v)$, say, satisfies $N^{np}(u, v) \leq N^{tc}(u, v)$, for all u, v . Furthermore, for many sea level measurements, the highest cycles in both cycle counts are identical, which gives the possibility to study the extreme properties of crests by means of extreme waves in `mM_np`; see also Figure 3.3 and Figure 4.13(a).

2.3 Crossing spectrum

Now, we shall discuss computation of the empirical crossing spectrum defined by (1.2), which we will call `cross`. In the `WAVE ANALYSIS TOOLBOX` the program `mm2cross` can be used to

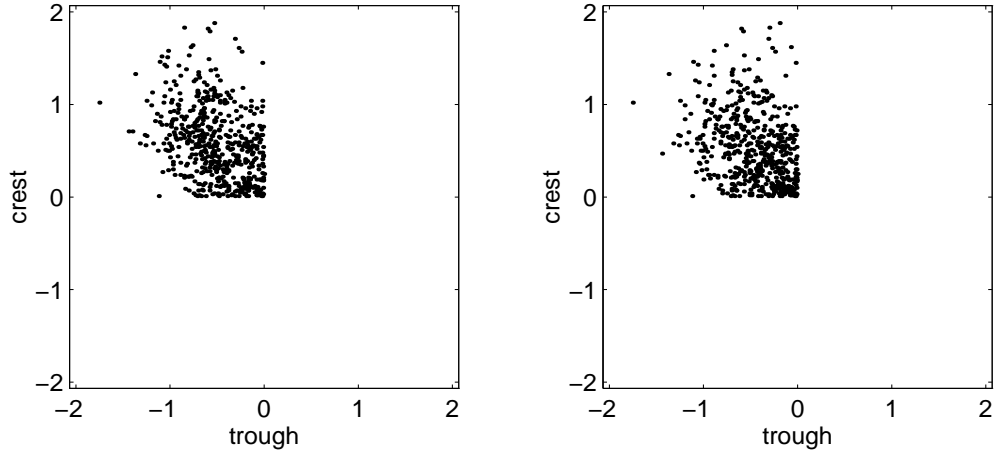


Figure 2.4: A trough2crest cycle count, left plot, and a plot of the mean separated min2Max cycle, right plot.

find the u -levels at which $N(u)$ jumps, and the values of $N(u)$ from the sequence of min2Max cycles. The following command puts these into a two column matrix `cross`, with the u -levels in the first column and the values of $N(u)$ in the second,

```
>> cross = mm2cross(mM);
```

The function also plots the crossing spectrum, shown in Figure 2.5(a). Note that, by the left continuity of $N(u)$, the matrix `cross` fully defines the crossing spectrum. Observe that if all local extremes have different values, the length of `cross` is equal to the total number of local extremes in $x(t)$, i.e. it is twice the length of `mM`. Consequently `cross` is a larger matrix than `mM`. (However, when the cycle values are discretized the length of `cross` will not exceed the number of discretizing levels.)

Similarly to the command just described, `mm2cross(TC)` will give the crest2trough cycles $N^{tc}(u)$.

2.4 Significant waveheight

A simple measure of severity of waves is the so called *significant wave characteristic*. Particularly important are the significant wave amplitude and the significant crest height, defined as follows.

Let $x(t)$, $0 \leq t \leq T$, be a smooth function. Then the significant amplitude $H_{1/3}^{tc}$, is the average of the highest one-third of the amplitudes H_i^{tc} , i.e.

$$H_{1/3}^{tc} = \frac{3}{n} \sum_{i=[2n/3]}^n H_{(i)}^{tc}, \quad (2.1)$$

where $H_{(1)}^{tc} \leq \dots \leq H_{(n)}^{tc}$ are ordered crest front amplitudes H_i^{tc} . Similarly, the significant crest $M_{1/3}^{tc}$, and trough $m_{1/3}^{tc}$, are defined by

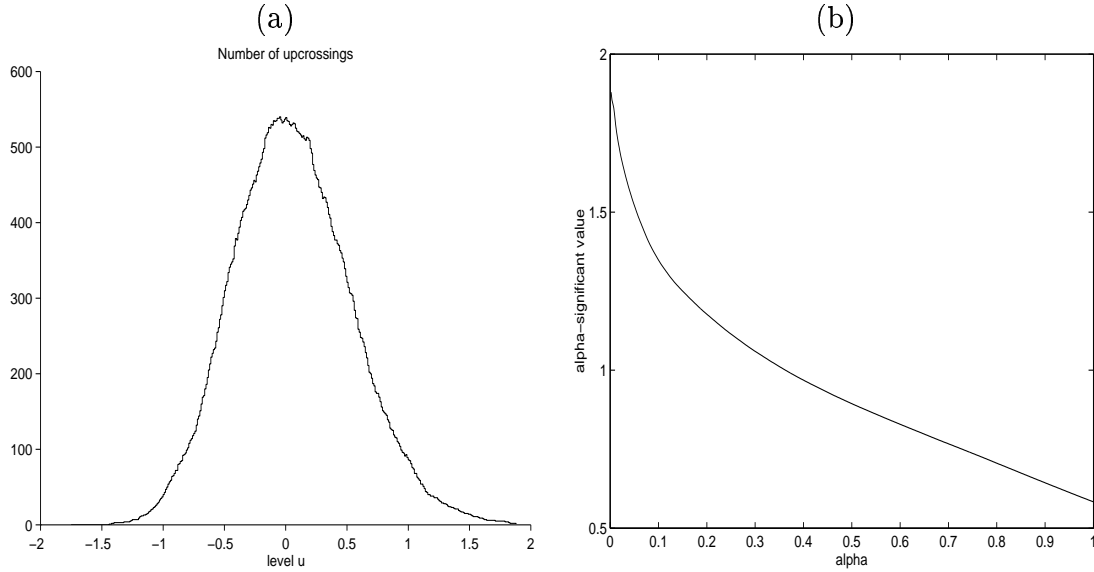


Figure 2.5: (a) Crossing spectrum; (b) α -significant crest height as a function of α .

$$M_{1/3}^{tc} = \frac{3}{n} \sum_{i=[2n/3]}^n M_{(i)}^{tc}, \quad m_{1/3}^{tc} = \frac{3}{n} \sum_{i=1}^{[n/3]} m_{(i)}^{tc}, \quad (2.2)$$

where $M_{(1)}^{tc} \leq \dots \leq M_{(n)}^{tc}$, and $m_{(1)}^{tc} \leq \dots \leq m_{(n)}^{tc}$, are the ordered crest and trough heights M_i^{tc} and m_i^{tc} , respectively.

When studying extreme properties of waves the so called α -significant crest height, defined as follows

$$M_{\alpha}^{tc} = \frac{1}{\alpha n} \sum_{i=[(1-\alpha)n]}^n M_{(i)}^{tc}, \quad 0 < \alpha \leq 1, \quad (2.3)$$

is also useful, see [18] for more discussion on α -significant wave characteristics.

Given a vector of wave characteristics, the α -significant value of the vector is computed in the `WAVE ANALYSIS TOOLBOX` by the function `vc2sign`. For example, the significant crest and trough with $\alpha = 1/3$ are computed as follows:

```
>> Cs = vc2sign(TC(:,2),1/3)
>> Ts = -vc2sign(-TC(:,1),1/3)
```

Note that the first element is 1/3 in `Cs` and $-1/3$ in `Ts`; the negative sign indicates that we have calculated the lower significant value, i.e. the average of the third lowest values.

If only one input argument is used in `vc2sign` then the function computes a three column matrix with decreasing values of α , equally spaced between 0 and 1, the α -significant values, and the α -quantiles of the crossings distribution, in the three columns,

```
>> Cs = vc2sign(TC(:,2)); plot(Cs(:,1),Cs(:,2))
```

see Figure 2.5(b).

2.5 Summary of empirical wave characteristics

| | |
|-----------------------|---|
| <code>ccplot</code> | <code>ccplot(cc)</code> |
| | Visualizes a cycle count <code>cc</code> as points in the plane. |
| <code>dat2hwa</code> | <code>T_hw = dat2hwa(xx,utc)</code> |
| | Extracts half wavelengths <code>T_hw</code> from data <code>xx</code> ; <code>utc</code> is the reference level, usually set equal to 0. |
| <code>dat2tp</code> | <code>TP = dat2tp(xx,h)</code> |
| | Extracts the sequence of turning points from data <code>xx</code> if <code>h = 0</code> end removes small oscillations if <code>h > 0</code> . |
| <code>mm2cross</code> | <code>cross = mm2cross(mM)</code> |
| | Extracts the crossing spectrum from <code>min2Max</code> cycles. |
| <code>tp2mm</code> | <code>mM = tp2mm(TP)</code> |
| | Extracts the <code>min2Max</code> -cycles from the sequence of turning points. |
| <code>tp2rfc</code> | <code>RFC = tp2rfc(TP)</code> |
| | Extracts rainflow cycles <code>RFC</code> from turning points. |
| <code>tp2tc</code> | <code>[TC T_tc] = tp2tc(TP,u)</code> |
| | Extracts crests and troughs <code>TC</code> in a sequence of turning points, where <code>u</code> is the reference level, and finds the corresponding crest front wavelengths <code>T_tc</code> . |
| <code>tp2wa</code> | <code>T = tp2wa(TP)</code> |
| | Extracts the <code>min2Max</code> wavelengths <code>T</code> from turning points. |
| <code>vc2sign</code> | <code>v_s = vc2sign(TC(:,2),ratio)</code> |
| | Computes the <code>ratio</code> -significant crest height. |

CHAPTER 3

DESCRIPTIVE STATISTICS OF WAVE CHARACTERISTICS

In this chapter we illustrate some functions in the `WAVE ANALYSIS TOOLBOX` which can be used to quantitatively describe the wave characteristics. In many of these functions the observed values are replaced by discretized values with a finite number of levels, chosen by the user. We begin with a discussion of data reduction and discretization functions.

3.1 Counting distribution

In the toolbox we shall use level in an equidistant grid $u_1 > \dots > u_n$, say, defined by specifying a **parameter vector** $[u_n \ u_1 \ n]$, where $u_n < u_1$ are the lowest and highest grid level, respectively, and n is the number of levels. The parameter vector will be denoted by **param**. A vector $\mathbf{u} = [u_1, u_2, \dots, u_n]$ with 5 levels between 2 and -2 is computed as follows,

```

>> param = [-2  2  5];  u = fliplr(levels(param))

```

When **param** is used as an argument to any descriptive `WAVE ANALYSIS TOOLBOX` function all cycle values are discretized to the levels calculated by the `levels` function. (The levels are assumed to be in decreasing order, which explains the `fliplr` command in `u = fliplr(levels(param));`.)

After discretization, a min2Max cycle cc_i is represented by a pair (u_k, u_l) which can either be illustrated by a point in R^2 or as an $n \times n$ -matrix F_i of zeros except with a single element equal to 1, $f_{l,n-k+1} = 1$. A discretized cycle count $cc_i, i = 1, \dots, N$ is uniquely represented in a form of a **histogram matrix** $F = \sum_{i=1}^N F_i$, where N is the total number of cycles. The name histogram matrix is motivated by the following obvious property. If the sequence of discretized cycles is (x_i^d, y_i^d) , then the histogram matrix $F = (f_{kl}), k, l = 1, \dots, n$, is

$$f_{kl} = \#\{i; x_i^d = u_{n-l+1}, y_i^d = u_k\}.$$

Both the discretization and the computation of the matrix F is performed by the `WAVE ANALYSIS TOOLBOX` function `cc2fr`. For example, given a min2Max cycle count `mM`, we can define the **param** vector by

```
>> param = [min(mM(:,1)) max(mM(:,2)) 51];
```

which will give an equidistant grid of 51 levels starting at the global maximum of $x(t)$ and ending at the global minimum of $x(t)$. The histogram matrix is then computed by

```
>> F = cc2fr(param,mM);
```

The histogram matrix, normalized by the total number of elements, will be called a **frequency matrix**, explaining our notation `fr`.

The counting distribution, with the default definition for $N(u, v)$ given by (1.4), is now computed from `F` by a function `fr2nt`. It has an inverse `nt2fr`, and we can check the accuracy of the inversion as follows, The result of the last operation should be zero (to machine accuracy).

```
>> NT = fr2nt(F);
>> F1 = nt2fr(NT); sum(sum(abs(F-F1)))
```

By (1.4), the counting distribution $N(u, v)$ is the number of cycles (x_i, y_i) such that $v < y_i$ and $x_i < u$. Observe, however, that the definition of $N(u, v)$ is general, and applies also to cycle counts such that $x_i > y_i$, which is the case for Max2min cycles.

The counting distribution $N(u, v)$ can be visualized by means of the WAVE ANALYSIS TOOLBOX function `cocc`, which plots a cycle count `cc` together with isolines of $N(u, v)$, specified by an argument `clevels`. For example, the following commands give Figure 3.1(a), when applied to the sea data,

```
>> clevels = [0.5 1 2 5 10 20 50 75];
>> cocc(param,mM,filt(NT,2),clevels,6);
```

3.2 Oscillation count

The oscillation count N^{osc} was defined by (1.6). Since N^{osc} is also a counting distribution for the rainflow cycle count it can be computed as follows,

```
>> RFC = tp2rfc(TP);
>> f_rfc = cc2fr(param, RFC);
>> N_osc = fr2nt(f_rfc);
```

Usually the histogram matrix `f_rfc` is very irregular and it needs to be smoothed before it can be plotted and used for further calculations. This can be done using a general WAVE ANALYSIS TOOLBOX filtering function `filt`, which uses a kernel type smoother,

```
>> cocc(param,RFC,filt(f_rfc,3),clevels,6);
```

The result is shown in Figure 3.1(b).

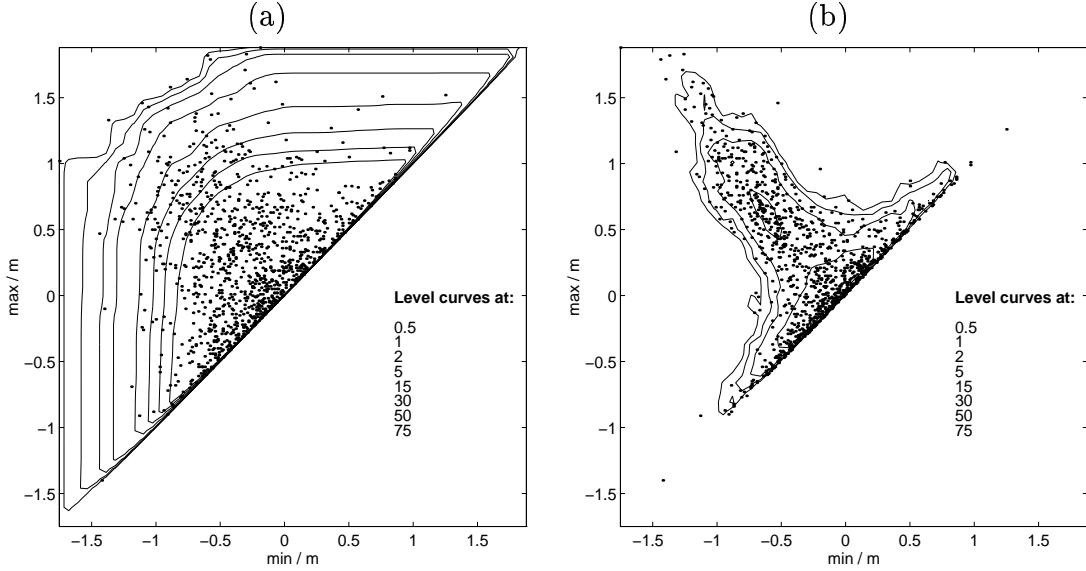


Figure 3.1: (a) Isolines of the counting distribution $N(u, v)$ of the min2Max count and the cycles given by dots; (b) Rainflow cycles and smoothed histogram.

Finally, we defined in Section 2.1 the rainflow filtered sequence of turning points, and used this nonlinear filter to smooth the observed data; see Figure 2.2. Note that `tp2rfc(TP_rfc)` gives the same rainflow cycles as `tp2rfc(TP)` for amplitudes exceeding h .

3.3 Amplitude distribution

Given a cycle count (x_i, y_i) , the sequence of observed amplitudes is defined by $h_i = y_i - x_i$. In the previous chapter we have computed the min2Max and trough2crest amplitudes, H and H_{tc} , respectively, for an observed wave sequence. For the discretized sequence (x_i^d, y_i^d) a histogram of amplitudes can be obtained from the histogram matrix F using a function `fr2amp`,

```
>> f_H = fr2amp(param,F); stairs(f_H(:,1),f_H(:,2))
```

giving Figure 3.2(a). The matrix f_H is a two column matrix with a vector of amplitudes $amp = [amp_i = u_{n-i+1} - u_n], i = 1, \dots, n$ in the first column, and a histogram vector with the i^{th} element equal to $\#\{(x_k^d, y_k^d) : y_k^d - x_k^d = amp_i\}$ in the second column.

Histograms of the crest and trough values of the cycles, f_y and f_x , respectively, can be easily extracted from the histogram matrix F . The following MATLAB-commands result in Figure 3.2,

```
>> f_x = [levels(param)' sum(F)'];
>> f_y = [fliplr(levels(param))' sum(F')'];
>> subplot(211) stairs(f_x(:,1),f_x(:,2));
>> subplot(212) stairs(f_y(:,1),f_y(:,2));
```

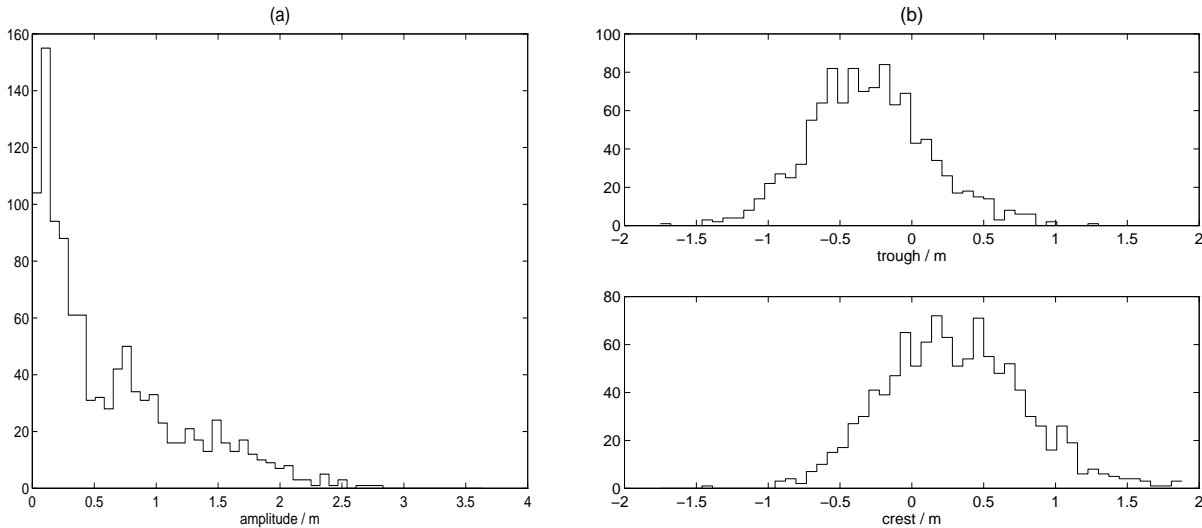


Figure 3.2: (a) Histogram of min2Max amplitudes; (b) Histograms of crest and trough values in `sea.dat`.

In Chapter 2 we introduced the mean separated min2Max cycle counts with positive maxima and negative minima, stored in a matrix `mM_np`, and indicated that they can be used to approximate the wave crests. In order to motivate this claim we shall compare the histogram of the crests heights with the histogram of maxima in `mM_np`. Since the discretized cycles are represented by a histogram matrix `F` we can compute the histogram of discretized maxima in `mM_np` from `F`. This can be done using a function `fr2pmax`,

```
>> f_pmax = fr2pmax(param,F,0);
>> stairs(f_pmax(:,1),f_pmax(:,2))
```

The argument 0 in the `fr2pmax` denotes the reference level, i.e. in this case the zero level.

As mentioned before, the number of mean separated min2Max cycles is equal to the number of crest2trough cycles and hence we can compute the irregularity factor as

```
>> irr = sum(f_pmax(:,2))/sum(sum(F))
```

Observe that we obtained a different value than on page 9, where we used

```
>> irr = length(TC)/length(mM)
```

In order to avoid such inconsistencies we need to choose the discretization levels u_i so that they include the reference level $u^{tc} = 0$. (This will solve our problem since this discretization procedure preserves a crossing spectrum at the levels u_i and if zero is one of discretization levels then $\text{sum}(f_pmax(:,2))$ is equal to the number of zero-upcrossings by x .) Consequently we propose a different `param` vector and compute a new histogram matrix `F`,

```
>> param_5 = [-2 2 51]; F=cc2fr(param_5,mM);
>> f_pmax = fr2pmax(param_5,F,0); irr = sum(f_pmax(:,2))/sum(sum(F))
```

To again illustrate the approximation of `trough2crest` cycles by the mean separated `min2Max` cycles we compare `f_pmax` with a histogram of discretized crests heights,

```
>> F_tc = cc2fr(param_5,TC); f_tcmax = fr2pmax(param_5,F_tc,0);
>> stairs(f_tcmax(:,1),f_tcmax(:,2)); hold
>> plot(f_pmax(:,1),f_pmax(:,2),'o');
```

The resulting plot is given in Figure 3.3(a) and we can see that the histogram of true crest heights `f_tcmax` differs from that of mean separated crest heights `f_pmax` only for small crest heights.

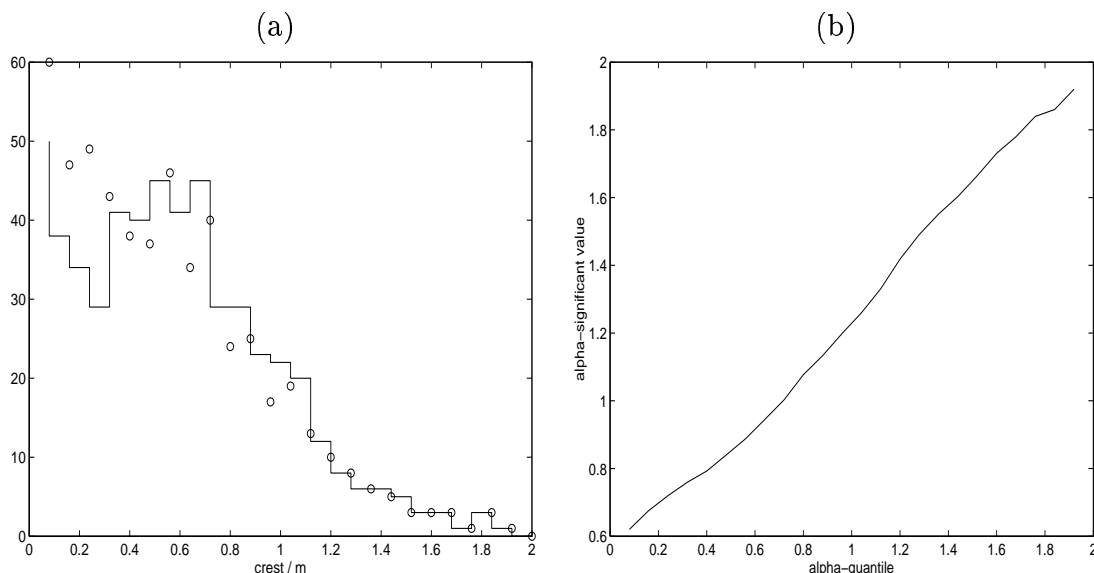


Figure 3.3: (a) A histogram of crest heights (solid line) compared with the crest heights in the mean separated `min2Max` cycles (circles); (b) A plot of α -significant values of crests heights against α -quantiles.

3.4 Significant wave characteristics

We now turn to the computation of significant wave characteristics given the histograms `f_H`, `f_pmax`, `f_tcmax`, `f_y`, or `f_x`. This can be done using a function `hs2sign`. For example, a significant crest height can be computed as follows,

```
>> Cs = hs2sign(f_tcmax);   Cs3 = hs2sign(f_tcmax,1/3)
```

Next, we can plot the α -quantile against α -significant crest height, see Figure 3.3(b),

```
>> plot(Cs(:,3),Cs(:,2))
```

Such plots are used in extreme value theory to check the fit of extreme data to a Generalized Pareto distribution; see [3] and 7.2. For data which follow a Generalized Pareto distribution the α -significant value is a linear function of the α -quantile. The WAVE ANALYSIS TOOLBOX contains algorithms for extreme value analysis, i.a. for estimation of parameters in the Generalized Pareto distribution (GPD) and in the Generalized Extreme Value distribution (GEV); see Chapter 7.

3.5 Discretization schemes

We finish with some less important remarks, which can be omitted at first reading. In the previous chapter, we have mentioned that the counting distribution of the min2Max cycles contains the upcrossing intensity along the diagonal. Using the function `pickdiag` one can extract the diagonal from the discretized crossing distribution `NT`. Figure 3.4 is produced by the following statements,

```
>> d = pickdiag(NT);
>> stairs(levels(param),d), hold
>> stairs(cross(:,1),cross(:,2))
```

The discretization of the continuous data can be done in many different ways. Since the counting distribution $N^{cc}(u, v)$ defines a cycle count, we propose to use such a discretization algorithm that the counting distributions computed for the original and discretized cycles agree at the levels u_i . In Section 1.2 we introduced two definitions (1.4) and (1.5) for $N^{cc}(u, v)$, here denoted $N^a(u, v)$ and $N^b(u, v)$, respectively, viz.

$$N^a(u, v) = \#\{(x_i, y_i) : x_i < u \text{ and } v < y_i\},$$

$$N^b(u, v) = \#\{(x_i, y_i) : x_i \leq u \text{ and } v \leq y_i\},$$

where $\#\{\cdot\}$ is the number of elements in the set $\{\cdot\}$.

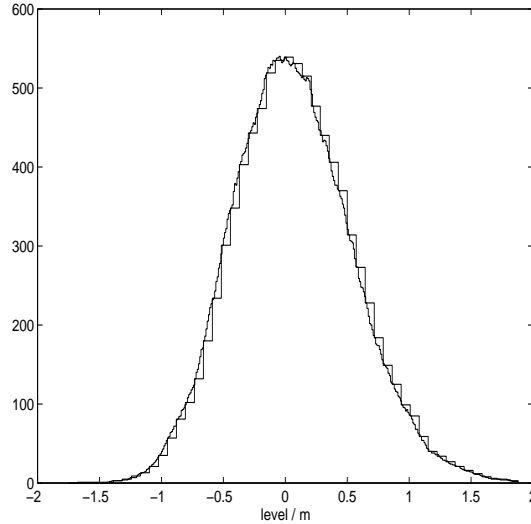


Figure 3.4: Comparison between the crossing spectrum of \mathbf{xx} and the crossing spectrum for the discretized cycle count.

If the counting distribution $N^a(u, v)$ is used to characterize the cycle count, the following scheme is obtained: if x_i is in the interval $[u_i, u_{i-1})$ then the discretized value is u_i , and if y_i is in $(u_i, u_{i-1}]$ then the discretized value is u_{i-1} . (Note that $u_i < u_{i-1}$ but that we do not assume $x_i < y_i$.) Consequently, we are systematically overestimating cycle amplitudes $y_i - x_i$. This is the default algorithm in the `WAVE ANALYSIS TOOLBOX`.

An optional scheme is based on the counting distribution $N^b(u, v)$, which leads to the following scheme: if x_i is in the interval $(u_i, u_{i-1}]$ then the discretized value is u_{i-1} , and if y_i is in $[u_i, u_{i-1})$ then the discretized value is u_i . Here we are systematically underestimating cycles amplitudes $y_i - x_i$. (This scheme is an optional choice.)

The difference between these two discretization methods can be visualized as follows: if we have a number of points, representing cycles, in a small square in the R^2 plane, then the first discretization algorithm puts all points in the north-west corner of the square, while the second algorithm puts them in the south-east corner. Consequently, the results are just a simple translation of each other. However, there are some theoretical differences between the two methods and for very broadband functions the optional discretization is more suitable to use.

Since the histogram matrix contains all information about discretized cycles, we practically never need to compute them explicitly. However, for completeness we show how `cc2fr` is operating. First one transforms cycles to indices of discretized levels by using a function `mkdisc`,

```
>> cl = mkdisc(param_5, mM);
```

then a function `cl2fr` transforms `cl` into a histogram matrix. Finally, we may obtain the discretized min2Max cycle count (m_i^d, M_i^d) , say, by

```
>> u = fliplr(levels(param_5));
>> mM_d = [u(c1(:,2))' u(c1(:,1))'];
```

By randomizing the cycles in the corresponding square one can get a better picture of the actual cycle distribution. The WAVE ANALYSIS TOOLBOX-function `dc2cc` does this. Figure 3.5 shows the discretized `min2Max` cycles and the continuous cycles obtained by randomization. The WAVE ANALYSIS TOOLBOX-command which produces the cycle plots is

```
>> ccplot(dc2cc(param_5, mM_d),mM_d,6);
```

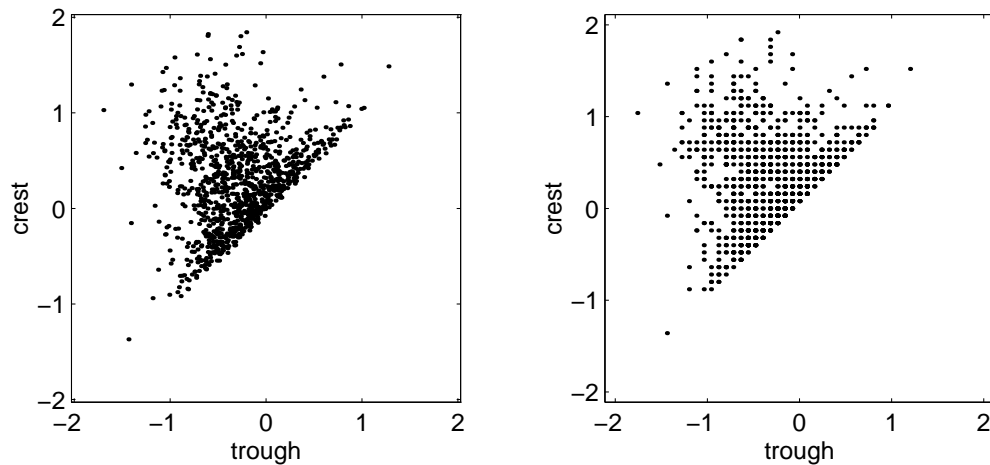


Figure 3.5: *Right plot: Discretized `min2Max` cycles, showing also where the histogram matrix `F` has nonzero elements. Left plot: A continuous `min2Max` cycle count obtained by randomization from the discrete cycle count.*

3.6 Summary of descriptive wave characteristics

| | |
|-----------------------|---|
| <code>cc2fr</code> | <code>F = cc2fr(param,cc)</code> |
| | Discretizes a cycle count <code>cc</code> using a grid defined by <code>param</code> and computes a histogram matrix. |
| <code>cl2fr</code> | <code>F = cl2fr(cl)</code> |
| | Transforms indices of discretized cycles <code>cl</code> into a histogram matrix. |
| <code>cocc</code> | <code>cocc(param,mM,NT)</code> |
| | Plots cycle count <code>cc</code> as points in the plane together with isolines of a matrix, here the counting distribution $N(u, v)$. |
| <code>dc2cc</code> | <code>dc2cc(param,mM_d)</code> |
| | Randomizes a discrete cycle count <code>mM_d</code> . |
| <code>filt</code> | <code>filt(F,k)</code> |
| | Smooths a histogram matrix <code>k</code> times. |
| <code>fr2amp</code> | <code>f_amp = fr2amp(param,F)</code> |
| | Extracts the histogram of amplitudes $h_i = y_i - x_i$ where <code>F</code> is a histogram matrix for cycle count of cycle count (x_i, y_i) . |
| <code>fr2nt</code> | <code>NT = fr2nt(F)</code> |
| | Computes the counting distribution $N(u, v)$ from the histogram matrix. |
| <code>fr2pmax</code> | <code>f_pmax = fr2pmax(param,F,u)</code> |
| | Extracts the histogram of cycles with crests higher than <code>u</code> and troughs lower than <code>u</code> ; by default <code>u=0</code> . |
| <code>hs2sign</code> | <code>v_s = hs2sign(f_H,ratio)</code> |
| | Computes the ratio-significant value given a histogram, here of the min2Max amplitudes <code>f_H</code> . |
| <code>levels</code> | <code>u = fliplr(levels(param))</code> gives a vector of discretization levels $u = [u_1, u_2, \dots, u_n]$ in decreasing order. |
| <code>mkdisc</code> | <code>cl = mkdisc(param,cc)</code> |
| | Discretizes a cycle count <code>cc</code> , using a grid defined by <code>param</code> . |
| <code>nt2fr</code> | <code>F = nt2fr(NT)</code> |
| | The inverse to the function <code>fr2nt</code> . |
| <code>param</code> | <code>param = [a b n]</code> |
| | Parameter vector which specifies that n discretization levels $b = u_1 > u_2 > \dots > u_n = a$ should be used. |
| <code>pickdiag</code> | <code>d = pickdiag(NT)</code> |
| | Returns the south-west/north-east diagonal of a square matrix, here <code>NT</code> . |

CHAPTER 4

THEORETICAL ANALYSIS OF GAUSSIAN TRANSFORM MODELS

4.1 Stochastic wave models

4.1.1 Gaussian and non-Gaussian waves

Here we shall analyse the wave data in $x(t)$ in the framework of stochastic processes, where $x(t)$ can be seen as an outcome or a sample function of a stochastic process $X(t)$. An important problem is then to compute the theoretical distributions of wave characteristics from the properties of the stochastic process $X(t)$.

The standard assumptions for a sea state under stationary conditions are that the model $X(t)$ is a stationary and ergodic stochastic process with mean $E[X(t)]$ assumed to be zero and a spectral density $S(f)$. The correlation structure of the process is defined by its covariance function

$$r(t) = \text{Cov}[X(s), X(s+t)] = \int_0^\infty \cos(\lambda) S(\lambda) d\lambda.$$

The first two (even) spectral moments, $\lambda_0 = V[X(t)] = \int_0^\infty S(\lambda) d\lambda$ and $\lambda_2 = V[X'(t)] = \int_0^\infty \lambda^2 S(\lambda) d\lambda$, are important quantities for the smoothness of the process.

The knowledge of which kind of spectral density $S(\lambda)$ are suitable to describe sea state data is well established from experimental studies. An important problem is how to compute the wave characteristic distributions when $S(\lambda)$ is given. This is not possible in general, but if, in addition, we assume that $X(t)$ is a Gaussian process, then $S(\lambda)$ fully defines the statistical properties of $X(t)$, and the wave characteristic distributions as well. However, even for Gaussian processes, there are no known explicit exact solutions to the problem, except in some degenerate cases, but very accurate numerical approximations do exist, see [10, 20].

Real data $x(t)$ seldom perfectly support the Gaussian assumption for the process $X(t)$. But since the Gaussian case is well understood and there are approximative methods to obtain wave characteristics from the spectral density $S(\lambda)$ for Gaussian processes, one often looks for a model of the sea state in the class of Gaussian processes. Furthermore, in previous work [21] we have found that for many sea wave data, even such that are clearly non-Gaussian, the wavelength and amplitude densities can be very accurately approximated using the Gaussian process model.

However, the Gaussian model can lead to less satisfactory results when it comes to the distribution of crest heights or joint densities of troughs and crests. In that case we found in [21] that a simple transformed Gaussian process used to model $x(t)$ gave good approximations for those densities.

Consequently, in the WAVE ANALYSIS TOOLBOX we shall model $x(t)$ by $X(t)$ which is a function of a single Gaussian process $\tilde{X}(t)$, i.e.

$$X(t) = G(\tilde{X}(t)), \quad (4.1)$$

where $G(\cdot)$ is a continuously differentiable function with positive derivative. Note that, once the distributions of crests, troughs, amplitudes or wavelengths in a Gaussian process $\tilde{X}(t)$ are computed then the corresponding wave distributions in $X(t)$ are obtained by simple variable transformations involving only the inverse of G which we shall denote by g . Actually we shall use the function g to define the transformation instead of G , and use the relation $\tilde{x}(t) = g(x(t))$ between the real sea data $x(t)$ and the transformed data $\tilde{x}(t)$.

If the model (4.1) is correct, then $\tilde{x}(t)$ should be a sample function of a process with Gaussian marginal distributions. Obviously, a Gaussian model is obtained by using a linear function $g(y) = ay + b$, where a, b are constants.

4.1.2 Spectral density S and transformation g

The knowledge of which kind of spectral density $S(f)$ is suitable to describe sea state data is well established from experimental studies. One often uses some parametric form of spectral density functions, e.g. a JONSWAP-spectrum given by the following formula:

$$S(\lambda) = \frac{9.81^2 \cdot \alpha}{\lambda^5} e^{-\beta(\lambda_m/\lambda)^4} \gamma \psi(\lambda), \quad \psi(\lambda) = e^{-(\lambda - \lambda_m)^2 / (2\sigma^2 \lambda_m^2)},$$

$$\sigma = \begin{cases} \sigma_a & \text{for } |\lambda| \leq \lambda_m \\ \sigma_b & \text{otherwise.} \end{cases}$$

This formula is programmed in a WAVE ANALYSIS TOOLBOX function `jonswap`, which evaluates and plots the spectral density $S(\lambda)$. In the example here we have chosen a JONSWAP spectrum with $\gamma = 1$, which reduces the spectrum to the Pierson-Moscowitz spectrum. We also have chosen $\beta = 0.09$, $\alpha = 0.0029$ and $\lambda_m = 0.075$. (These parameters are default values in `jonswap`.) Consequently the spectrum is obtained by the command

```
>> spec=jonswap;
```

The resulting spectrum is shown in Figure 4.1(a). Observe that the diagram always shows a normalized spectrum, such that the two first even spectral moments are one, i.e. the process and its derivative both have standard deviation equal to 1.

For the transformation g there are some parametric functions proposed in the literature, e.g. the one proposed by Ochi and Ahn [11]:

$$g(y) = \frac{1}{\sigma\gamma}(1 - e^{-\gamma y}), \quad \gamma = \begin{cases} \gamma_a & \text{if } y \geq 0, \\ \gamma_b & \text{otherwise,} \end{cases}$$

and $\sigma > 0$. Parameters used here are $\sigma = 1$, $\gamma_a = 0.1$, and $\gamma_b = 0.15$. (These parameters are the default values in the WAVE ANALYSIS TOOLBOX function `ochi`.) The simplest way to define this standard transformation is by the command

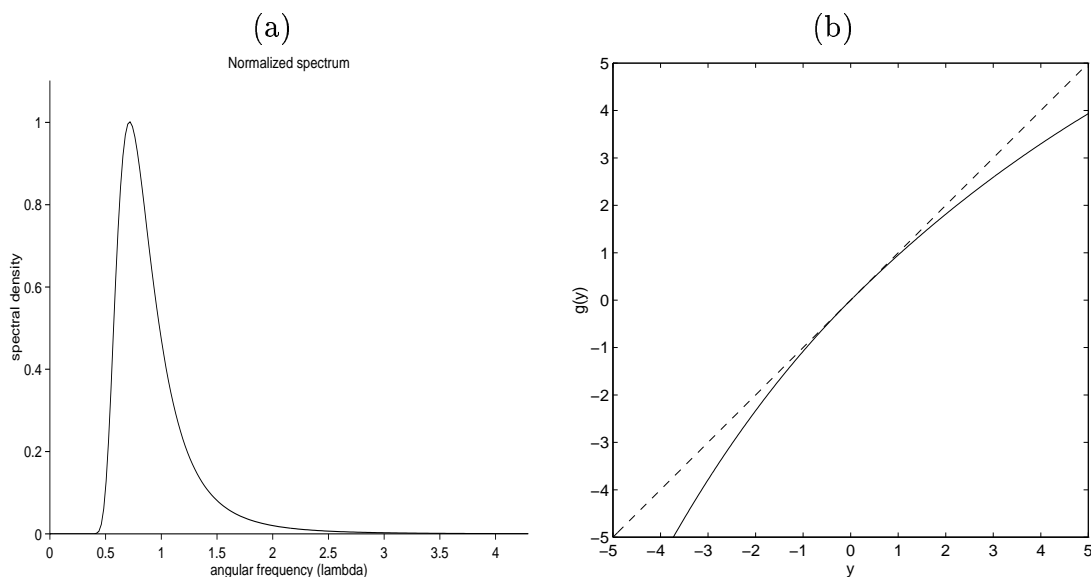


Figure 4.1: (a) Standard JONSWAP spectrum, (b) The Ochi transformation g compared with the Gaussian model $g(y) = y$ (dashed line).

```
>> g = ochi;
```

This command will result in a two column matrix \mathbf{g} with equally spaced y -values in the first column and the values of $g(y)$ in the second column. This function is shown in Figure 4.1(b), where it is compared to the Gaussian model ($g(y) = y$) given by a dashed line. We can see from the diagram that the waves in a transformed Gaussian process $X(t) = G(\tilde{X}(t))$, will have an excess of high crests and shallow troughs compared with waves in the Gaussian process \tilde{X} . For example, since $g(3.5) \approx 3$ and $g(-2.5) \approx -3$, a crest value of 3 in a Gaussian process will correspond to a crest value of approximately 3.5 in the transformed process, and a Gaussian trough value of -3 will correspond to a trough of approximately height -2.5 in the transformed process. In order to illustrate this fact we have simulated a sample of $X(t)$, using the WAVE ANALYSIS TOOLBOX function `simspec`,

```
>> L = simspec([15 3 0.1],spec,g);
```

Here, the argument `[15 3 0.1]` specifies that $2^{15} = 32768$ sample points should be simulated, with a time step of 0.1. The parameter 3 specifies the number of alias intervals used in the spectrum; see the help function for details.

The mean and standard deviation of the simulated \mathbf{L} is obtained by

```
>> mL = mean(L(:,2));  sL = std(L(:,2));
```

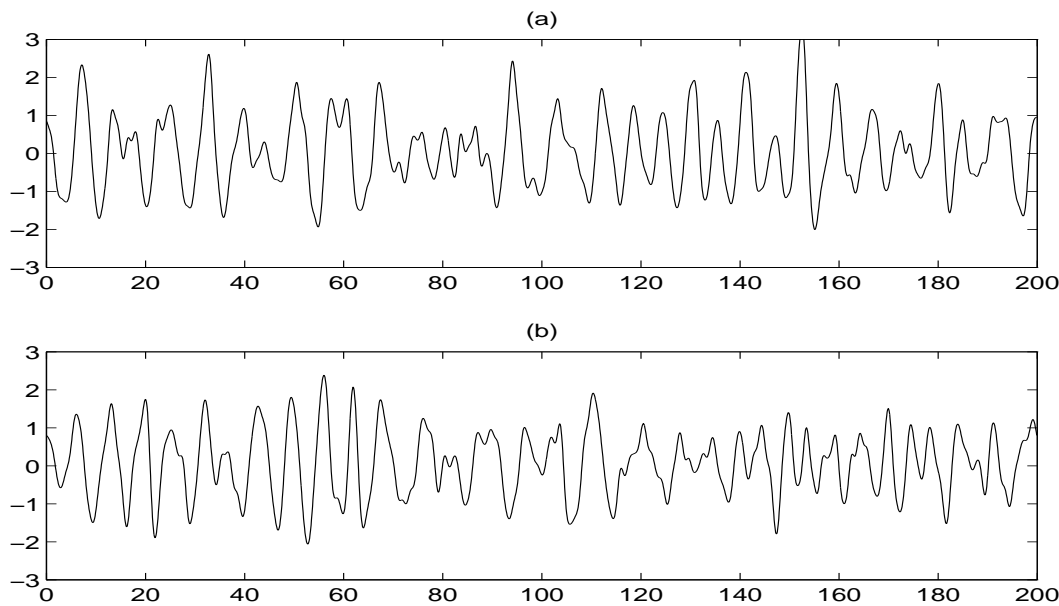


Figure 4.2: Part of simulated sample paths of (a) $X(t) = G(\tilde{X}(t))$, (b) $\tilde{X}(t)$.

A part of the simulated L is shown in Figure 4.2(a) and compared with the Gaussian sample path (Figure 4.2(b)) with the same spectral density, and with the same mean and standard deviation as L , obtained by

```
>> LG = simspec([15 3 0.1],spec); LG(:, 2:3) = mL + sL*LG(:,2:3);
```

4.2 Simple approximations of wave distributions

4.2.1 Density of minima and maxima

One of the most important wave-characteristics is the joint density of wave crests and troughs. The `WAVE ANALYSIS TOOLBOX` contains algorithms for very accurate approximations of this density for a Gaussian processes with a general spectral density, and also for a transformed Gaussian process. However, in order to compute the distribution of trough and crest heights in a trough2crest cycle, we first need to compute the joint density of the height of a minimum and the following maximum in a min2Max cycle, using the `WAVE ANALYSIS TOOLBOX` function `minmax`.

The function `minmax` works by first calculating the probability density of minimum and maximum heights in a Gaussian process $\tilde{X}(t) = g(X(t))$. This density is then transformed into the distribution of min2Max heights in the original process $X(t)$. The transformation `g` is used as input parameter together with the spectral density `spec`.

We start with the simplest and fastest approximation, using the default parameters and then we show how to compute more accurate approximations. The following command calculates and plots the bivariate density of a minimum and the following maximum,

```
>> [f_mM f_M f_m param] = minmax(spec,g);
```

The matrix `f_mM` contains the joint density of minimum and the following maximum computed at a grid defined by the vector `param`, chosen automatically by the program `minmax`; see Section 3.5 for discussion of the discretization procedure. The two column matrices `f_M` and `f_m` contain the marginal densities of maxima and minima, respectively, with arguments t and density values $f(t)$ in the two columns. Isolines of the matrix `f_mM` are shown in Figure 4.3; (these are plotted automatically by the function `minmax`).

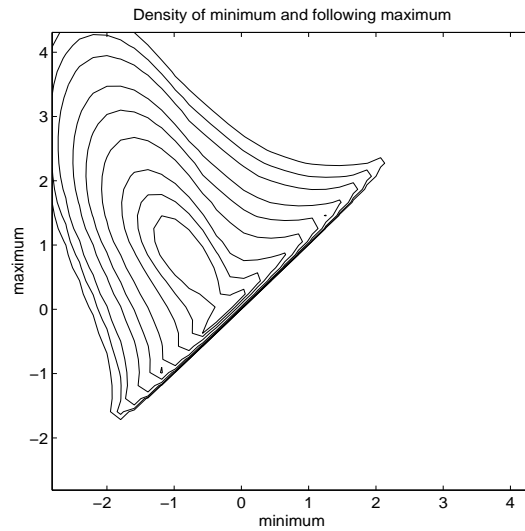


Figure 4.3: *Isolines of the density of minimum and the following maximum.*

From Figure 4.3 we can see that the grid defined by `param` does not cover the whole region of variation of the density and therefore we choose our own grid and recalculate the density,

```
>> param = [-3 5 81]; f_mM = minmax(spec,g,0,param);
```

Here the grid is explicitly specified by the last input, and third input 0 indicates that the simplest approximation will be computed.

The definition of the probability density of the minimum and the following maximum is discussed in detail in [10]. Here we shall only give a brief description.

In the `WAVE ANALYSIS TOOLBOX` a `min2Max` cycle count is represented as a point process (cloud of dots) in R^2 . For each new simulated sample L (or LG) we obtain a different configuration of points. The average density of points (per area unit) is called the *intensity* of cycles. Obviously, the intensity is not constant over R^2 since there are a few big waves and plenty of small ones. The integral of the intensity over a given region is the expected number of points in that region.

Now, the joint probability density of minimum and the following maximum, which for the transformed Gaussian process is the same as the joint probability density of minimum and the previous maximum, is defined as the ratio of the intensity of cycles over the total expected number of cycles. We prefer to use the intensities instead of the joint probability densities

because they can be easily visually compared with the point process of cycles and, what is even more important, the approximations of the intensity are usually very accurate for big waves, e.g. in the important region, while the intensity in the region where the waves are small is very hard to approximate and requires a lot of computational time. The probability density is obtained by normalizing the intensity, and the inaccurate approximation of the intensity for small cycles is then corrupting the probability density in the important region of big waves.

Given the density f_{mM} , the intensity of min2Max cycles is obtained by multiplying f_{mM} by the expected number of cycles N , say. For the transformed normalized Gaussian process with variance one, N can be computed using Rice's formula

$$N = T \cdot \frac{1}{2\pi} \sqrt{\lambda_4},$$

where λ_4 is the fourth spectral moment in the normalized spectral density S , and T is the observation length. The moment λ_4 of the normalized spectral density can be computed as follows,

```
>> [nspec x14] = normspec(spec);
```

Here, `nspec` contains the normalized spectrum and `x14` = λ_4 is the fourth spectral moment of `nspec`. For our simulations, the expected number of cycles N , will be computed as follows,

```
>> NmM = 2^15*0.1*sqrt(x14)/2/pi
```

which gives $N = NmM = 763.15$. Note that the irregularity factor is $1/\sqrt{\lambda_4}$. i.e.

```
>> irr = 1/sqrt(x14)
```

The min2Max pairs (m_i, M_i) in the simulated L can be found by

```
>> TP = dat2tp(L(:, [1 2])); mM = tp2mm(TP);
```

(Observe that we use only the first two columns in L ; the third column contains the derivative of the simulated process). To visualize the intensity of min2Max cycles computed by `minmax` we use the WAVE ANALYSIS TOOLBOX function `cocc`,

```
>> cocc(param, mM, NmM*f_mM)
```

The algorithm plots contour lines on automatically chosen levels. These may not be satisfactory, and one can input better suggestions into `cocc` together with the size of dots. The plot in Figure 4.4(a) is obtained by the following command,

```
>> clevels = [1 2.5 5 10 25 50 100 250];
>> cocc(param,mM,NmM*f_mM,clevels,6)
```

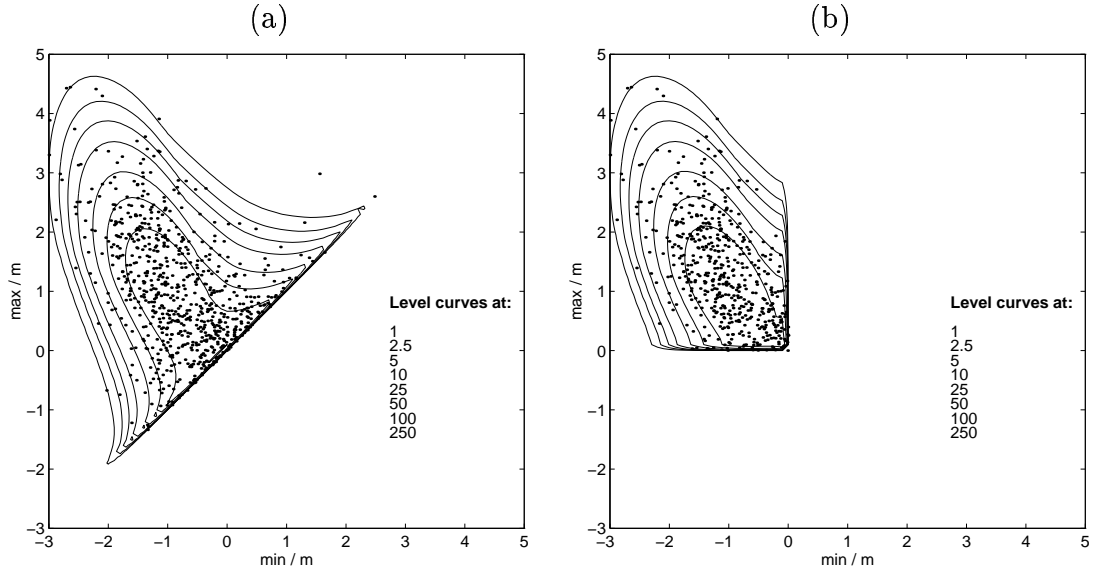


Figure 4.4: *Isolines of the theoretical intensity of min2Max cycles (m, M) (in (a)) and trough2crest cycles (m^{tc}, M^{tc}) (in (b)) for a transformed Gaussian process $X(t)$, together with the observed (m_i, M_i) and (m_i^{tc}, M_i^{tc}) in the simulation L (dots).*

4.2.2 Density of trough and crest height

We now turn to the computation of the density of trough and crest heights in a trough2crest pair. The algorithm is described in [21] and implemented in the WAVE ANALYSIS TOOLBOX function `mctp2tc`. (the name `mctp2tc` is short for Markov Chain of Turning Points to Trough-Crest cycles.) The following sequence of commands gives the plot in Figure 4.4(b),

```
>> f_tc = mctp2tc(f_mM,0,param);
>> [TC T_tc] = tp2tc(TP,0);
>> cocc(param,TC,NmM*f_tc,clevels,6)
```

The fact that in `cocc` we use the matrix `NmM*f_tc` and not `length(TC)*f_tc` needs some explanation. The matrix `f_mM` contains the intensity of min2Max cycles per local maximum. Therefore `f_tc`, obtained from `f_mM` by means of `mctp2tc`, has the same units, i.e. it is the intensity of trough2crest cycles per local maximum. The function `mctp2tc` works in such a way that the output has the same units as the input. For example, we can directly compute the intensity of trough2crest cycles and also check the agreement of the two methods, by the commands

```
>> f_tc1 = mctp2tc(NmM*f_mM,0,param); sum(sum(abs(f_tc1-NmM*f_tc)))
```

We now want to check the accuracy of the approximations `f_mM` and `f_tc`. This can be done by means of the function `empdistr`. We transform the theoretical `f_mM` matrix of `min2Max` intensities into an expected histogram matrix and also compute histograms of the observed `min2Max` amplitudes and `trough2crest` crests heights. The following commands will give the distributions in the form of two column matrices `f_H` and `f_crest` with amplitudes and crest heights in the first column and theoretical probabilities in the second,

```
>> delta = (param(2)-param(1))/(param(3)-1);
>> F_mM = NmM*f_mM*delta^2;
>> F_tc = NmM*f_tc*delta^2;
>> f_H = fr2amp(param,F_mM);
>> f_crest = fr2pmax(param,F_tc);
```

(The function `fr2pmax` can take the reference `elvel utc` as third argument; the default value is `utc = 0`.)

Next, we compare the empirical distributions of crest front amplitudes and crest heights in the simulation with the theoretical distributions `f_H` and `f_crest`. The following commands produce the plots in Figure 4.5,

```
>> empdistr(mM(:,2)-mM(:,1),0.1,f_H);
>> empdistr(TC(:,2),0.1,f_crest);
```

Here, the second input `0.1` cuts off the small amplitudes (crests) with height smaller than `0.1`. As can be seen from Figure 4.5, even these simple approximations are reasonably accurate. We shall describe more accurate approximations in the next section.

Rayleigh approximation

We end this section with some additional comments about the simplest approximation. Common practice in ocean engineering is to use the Rayleigh distribution to approximate the crest height distribution. This is a very good approximation for narrow banded Gaussian processes, but as we shall see, this will not be the case for our process. In order to compare `f_crest` with the Rayleigh distribution and with the observed crest distribution we shall use the discretization defined by `param` instead of the MATLAB function `hist`. After the commands

```
>> F_TC = cc2fr(param,TC);
>> f_crestTC = fr2pmax(param,F_TC);
```

`f_crestTC` will contain the histogram values of the observed discretized crest heights. Next, the Rayleigh histogram is computed as

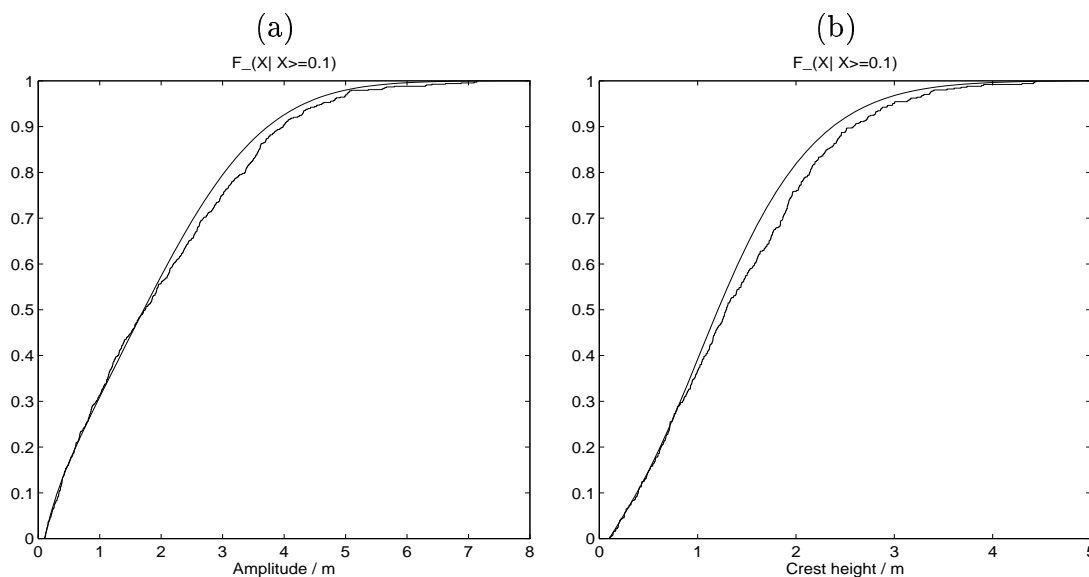


Figure 4.5: Empirical distributions of min2Max amplitudes (in (a)) and crest heights (in (b)), together with theoretical distributions calculated by the simplest approximation.

```
>> f_r = f_crest;
>> f_r(:,2) = f_r(:,1).*exp(-.5*((f_r(:,1))/sL).^2);
```

and in order to get the same number of crests in the Rayleigh histogram as in `f_crest` we need the following renormalization

```
>> f_r(:,2) = sum(f_crest(:,2))*f_r(:,2)/sum(f_r(:,2));
```

The empirical histogram can now be compared with the two theoretical distributions. The following commands result in the plot in Figure 4.6(a):

```
>> stairs(f_crestTC(:,1),f_crestTC(:,2)); hold
>> plot(f_crest(:,1),f_crest(:,2)); plot(f_r(:,1),f_r(:,2),'--')
```

Another way to compare the obtained distributions is to use the significant values. Figure 4.6(b) shows the empirical significant values together with the theoretical significant values in the two distributions, calculated by the following set of commands,

```
>> sr = hs2sign(f_r); SC = vc2sign(TC(:,2));
>> sc = hs2sign(f_crest); sTC = hs2sign(f_crestTC);
>> clf; plot(SC(:,1),SC(:,2),'.'); hold
>> stairs(sTC(:,1),sTC(:,2));
>> plot(sc(:,1),sc(:,2)); plot(sr(:,1),sr(:,2),'--')
```

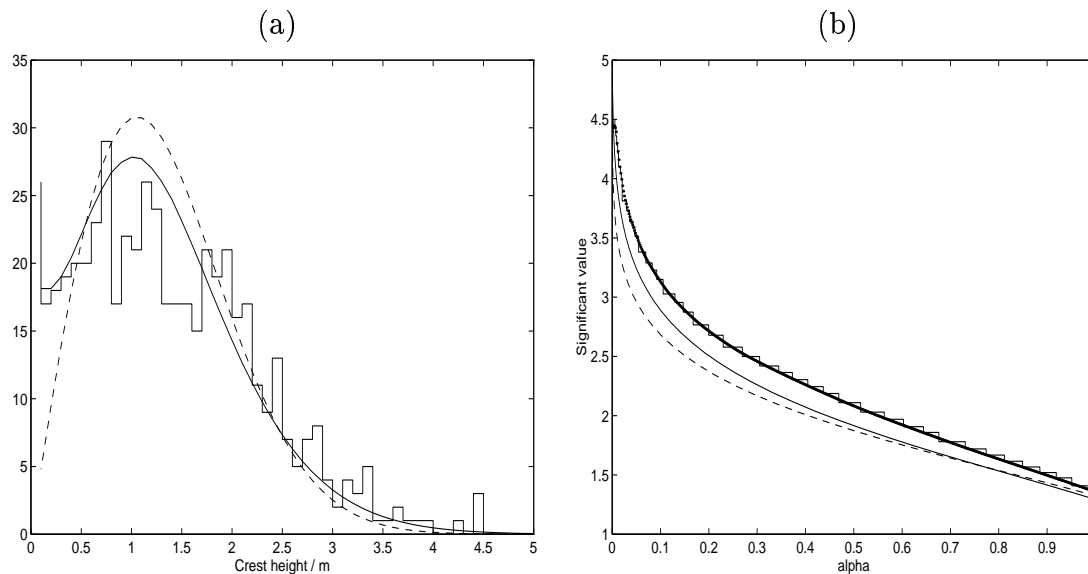


Figure 4.6: (a) Histogram of crest heights together with a theoretical crest distribution (solid line) and the Rayleigh approximation (dashed line). (b) Observed significant α -values (broken curve) together with calculated values from different approximations: solid curve = theoretical distribution, dashed curve = Rayleigh distribution, dotted curve = positive crest histogram.

As we can see, the discretization does not affect the distribution of crest heights. We can also see that the Rayleigh approximation is not very accurate.

4.3 More accurate approximations

4.3.1 The regression approximation

The more advanced approximations of the trough and crest heights in the WAVE ANALYSIS TOOLBOX are based on regression approximations of higher order of the joint density of minimum and the following maximum in the transformed Gaussian model. The increased accuracy is obtained by introducing certain *regressors*, which are used to describe the variability of the sample paths between the local extremes. The number of regressors is controlled by an input parameter called NIT ("number of integrations"). It is well known that, for high waves, the sample paths of a Gaussian process are almost deterministic between the extremes – this explains the good accuracy of the simplest approximation (NIT = 0) for large waves, as well as that of the Rayleigh distribution to describe the crest front amplitude for narrow band Gaussian processes. The small and moderate waves for broad band processes are more difficult to approximate and the sample variation has to be described stochastically using more regressors, i.e. NIT > 0; see [20].

4.3.2 The min2Max wavelength T

In the simple approximations described in the previous sections we used the default value $NIT = 0$. We shall now describe how to calculate good approximations of the trough2crest and min2Max cycles by choosing higher values of NIT . We start with the distribution of the distance between local extremes, i.e. the wavelength T . This can be done using the function `wave_t`,

```
>> f_t = wave_t(spec,0,3);
```

This call needs some comments. First, the output matrix `f_t` contains the density of wavelength T and, as mentioned before, it is the same for the transformed Gaussian process (simulation given in `L`) and a Gaussian process (simulation given in `LG`). Therefore we do not need to input the transformation `g`. The second input `0` indicates that the density of the full wavelength is computed. If this input parameter is equal to `1` the density of the half wavelength T^{utc} will be computed instead. The third input parameter is NIT , and in this case up to 3 regressors can be used in the approximation. The grid for the density calculation is chosen automatically by the function, which also as default gives a plot of the density; see Figure 4.7(a).

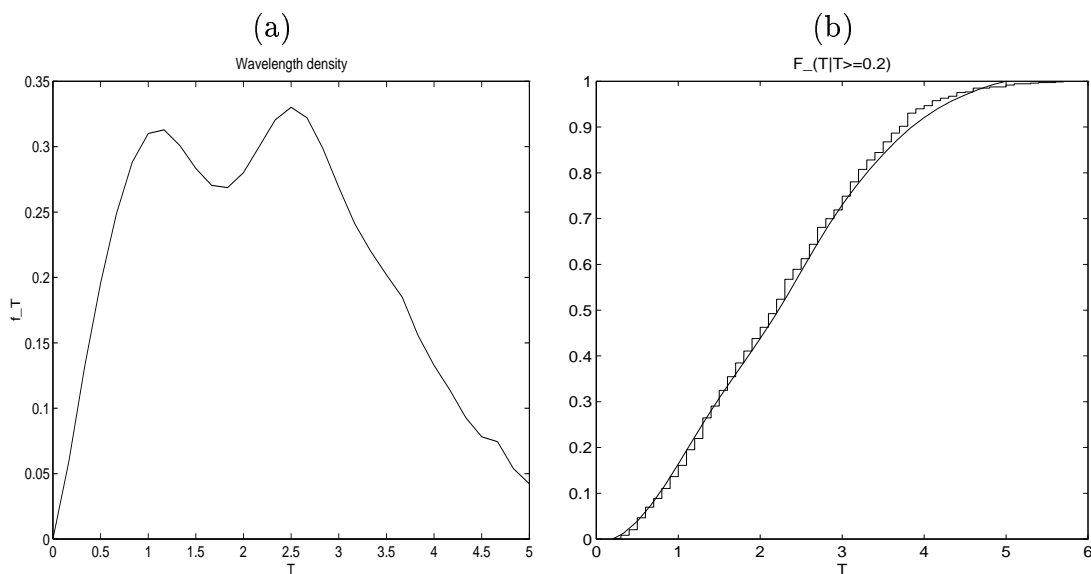


Figure 4.7: (a) Density of min2Max wavelength T calculated with $NIT = 3$ regressors; (b) Empirical and theoretical wavelength distribution functions.

In Figure 4.7(b) we compare the empirical distribution function of the wavelength with the theoretical distribution obtained from the density `f_t`,

```
>> T_mM = tp2wa(TP); empdistr(T_mM,0.2,f_t)
```

From the figure we can see that the accuracy of the approximation is not satisfactory. This can be checked also by computing the integral of the density (it should be 1),

```
>> diff(f_t(:,1))'*f_t(2:length(f_t),2)
```

To increase accuracy we choose our own grid and a higher $NIT = 5$:

```
>> t = [0:0.15:6]; f_t5 = wave_t(spec,0,5,0,t,15);
>> empdistr(T_mM,0.2,f_t5);
```

Now, we have also specified the reference level $u_{tc} = 0$, which is not needed here but in the calculation of the half wavelength distribution. Note that the grid specified by t has to start at zero but it needs not be equidistant. The last input parameter 15 specifies the number of points ($32768 = 2^{15}$) used by the FFT transform to compute the covariance structure from the spectral density. In Figure 4.8 we give the plots corresponding to Figure 4.7 but for this new approximation. As seen, the accuracy is now very good.

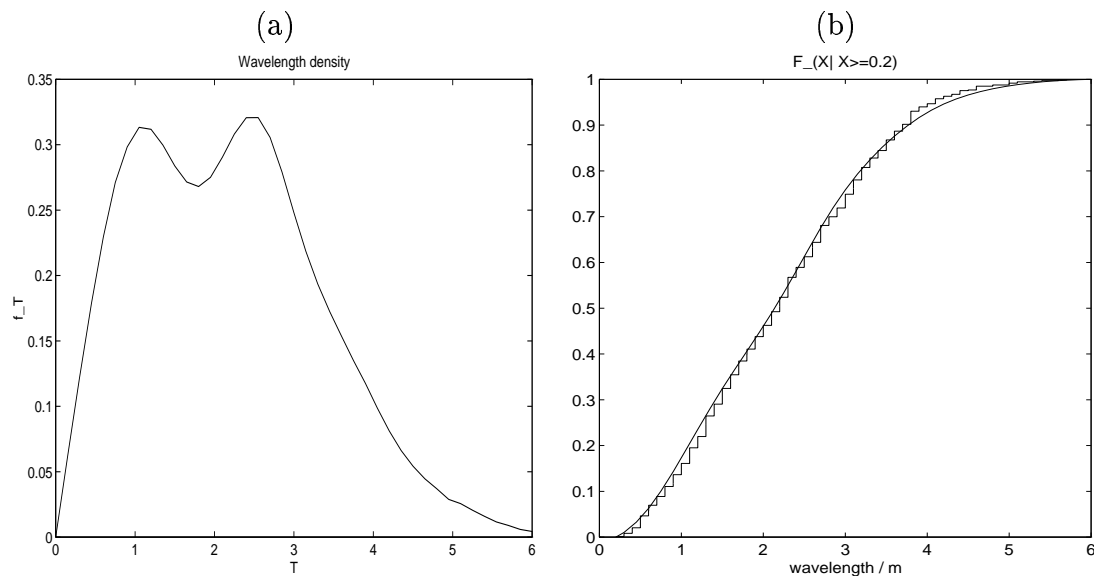


Figure 4.8: (a) Density of min2Max wavelength T calculated with $NIT = 5$ regressors; (b) Empirical and theoretical wavelength distribution functions.

4.3.3 Joint density of min2Max and trough2crest cycles

We turn now to the density of the height of minima and maxima in a min2Max pair or a trough2crest pair. Since the height of a minimum and a maximum can be considered as two random variables we will choose the NIT parameter to be 3,

```
>> f_mM3 = minmax(spec,g,3,param,t,15);
>> f_tc3 = mctp2tc(f_mM3,0,param);
```

The obtained densities, i.e. `f_mM3` for the `min2Max` pair (m_i, M_i) and `f_tc3` for the `trough2crest` pair (m_i^{tc}, M_i^{tc}) , are compared with the simulated `min2Max` and `trough2crest` cycles by means of the following commands,

```
>> cocc(param,mM,NmM*f_mM3,clevels,6)
>> cocc(param,TC,NmM*f_tc3,clevels,6)
```

The resulting two plots are shown in Figure 4.9 which should be compared to Figure 4.4.

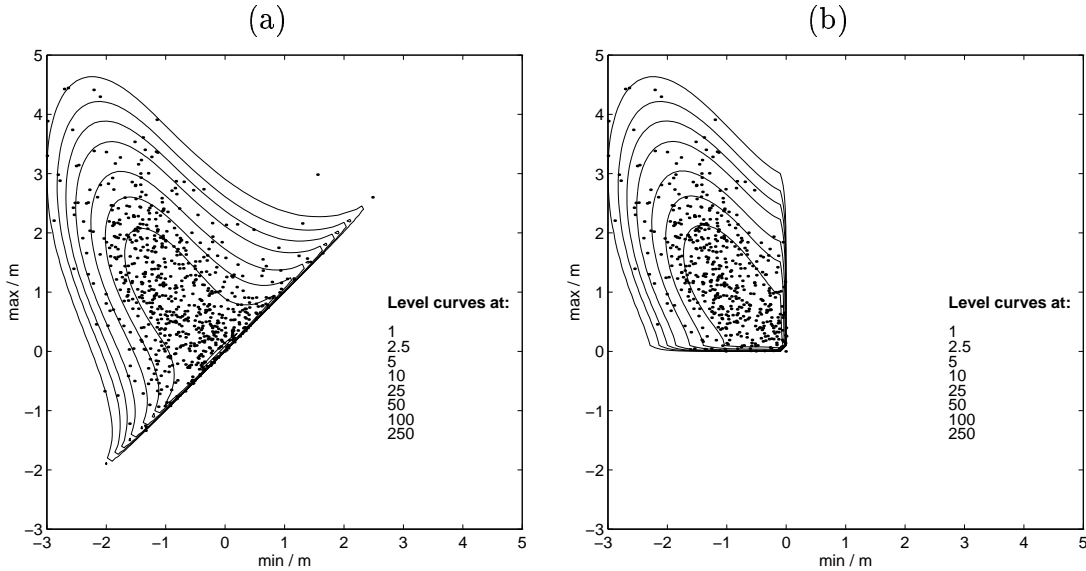


Figure 4.9: *Isolines of intensities of min2Max cycles (m, M) (in (a)) and trough2crest cycles (m^{tc}, M^{tc}) (in (b)) for a transformed Gaussian process $X(t)$, together with the observed (m_i, M_i) and (m_i^{tc}, M_i^{tc}) in the simulation L (dots).*

As before, we check the accuracy of the approximations `f_mM3` and `f_tc3` by comparing the amplitude and crest distributions with observations,

```
>> delta = (param(2)-param(1))/(param(3)-1);
>> F_c3 = NmM*f_mM3*delta^2;
>> f_H3 = fr2amp(param,F_c3);
>> H_mM = mM(:,2)-mM(:,1);
>> empdistr(H_mM,0.1,f_H3);
```

A plot of the empirical distribution of the `min2Max` amplitude is shown in Figure 4.10(a), and in Figure 4.10(b) we show the corresponding `trough2crest` amplitude distribution by means of

```
>> F_tc3 = NmM*f_tc3*delta^2;
>> f_crest3 = fr2pmax(param,F_tc3);
>> empdistr(TC(:,2),0.1,f_crest3);
```

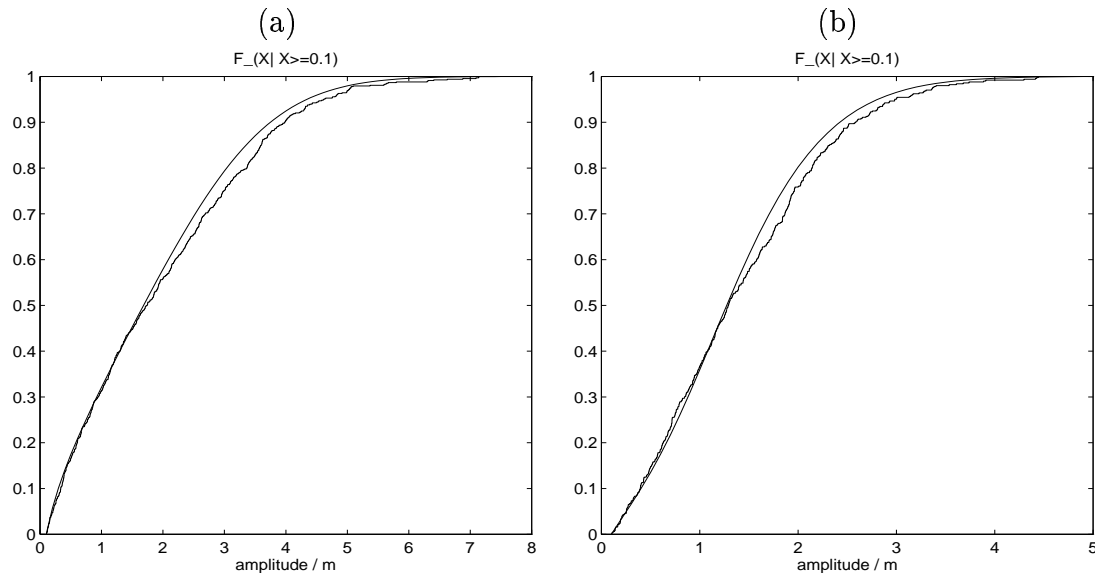


Figure 4.10: Empirical distribution functions of *min2Max* (in (a)) and *trough2crest* (in (b)) amplitudes together with accurate theoretical approximations, $NIT = 3$.

The accuracy of the approximations is now excellent.

4.3.4 Joint density of wavelength and amplitude

Another important wave characteristic is the joint density of wavelength and amplitude. We start with the *min2Max* wavelength and amplitude.

min2Max amplitude and wavelength

The wavelength and amplitude of *min2Max* can be computed using the `WAVE ANALYSIS TOOLBOX` program `wave_th`. The simplest approximation, computed at a regular dense grid, is obtained using the call

```
>> wt = [0:0.1:6]; wh = [0:0.1:7];
>> [f_th0 f_th0_t f_th0_h] = wave_th(spec,0,g,0,wt,wh);
```

As in the `wave_t` program the second input '0' indicates that *min2Max* wavelength and amplitude will be computed, while the fourth input is the NIT parameter, which in this simple example is set to zero.

Isolines of the density are compared with the observations in Figure 4.11(a). Since the function `wave_th` allows for a non-equidistant grid we cannot use the function `cocc` to visualize the density `f_th0`. This is instead done using a special function `cowa`,

```
>> cowa([T_mM H_mM],wt,wh,NmM*f_th0, clevels,6)
```

In Figure 4.11(a) we can see that the density has a very peaked maximum at low amplitudes (because the spectrum is broad banded) making it difficult to integrate out the marginal densities of wavelength and amplitude. (That is also why we allow here for a non-equidistant step grid.) Consequently, the densities of the min2Max wavelength computed by means of `wave_t` and of the amplitude obtained from `minmax` and `fr2amp` are more reliable than those obtained from `wave_th`.

We now turn to higher order approximations of the wavelength and amplitude density. In the previous section we needed `NIT = 3` to get an accurate approximation of the joint density of minimum and the following maximum and `NIT = 5` for the marginal density of wavelength. Since amplitude can be seen as one regressor we can compute a good approximation for the density using `NIT = 4`. Isolines of the density `f_th4` are shown in Figure 4.11(b),

```
>> [f_th4 f_th4_t f_th4_h] = wave_th(spec,0,g,4,wt,wh);
>> cowa([T_mM H_mM],wt,wh,NmM*f_th4, clevels,6)
```

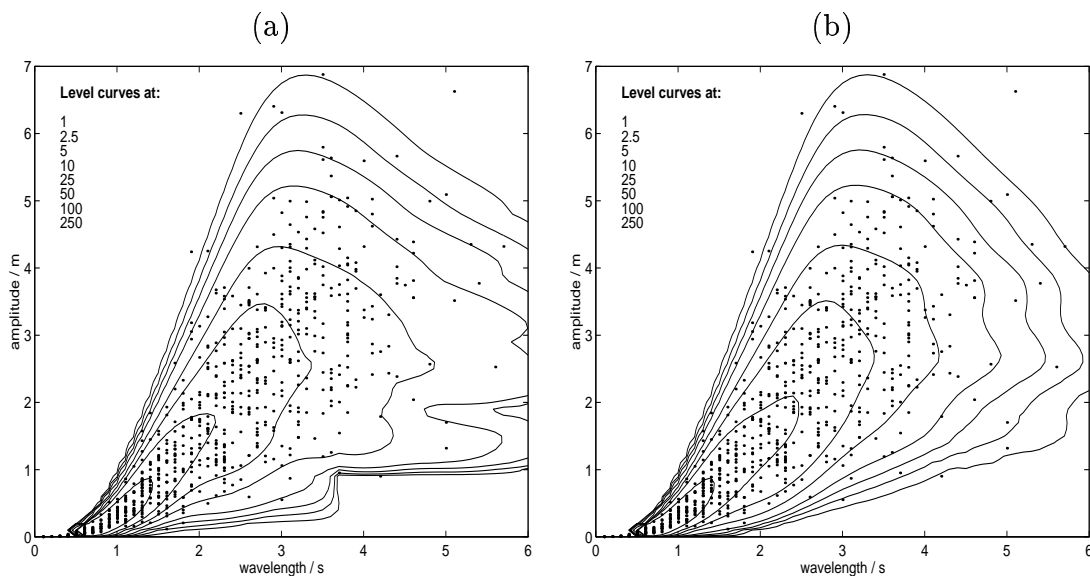


Figure 4.11: Isolines of the intensities of min2Max cycles (m, M) for a transformed Gaussian process $X(t)$ together with the observed (m_i, M_i) in the simulation L (dots); `NIT = 0` in (a) and `NIT = 4` in (b).

We shall now discuss the accuracy of the approximations. Comparing Figures 4.11(a) and (b), we can see that the main part of the density for wavelength and amplitude is approximated correctly by `f_th0` and only the more irregular long waves have not been modeled correctly, and hence the marginal density of wavelength is not very accurate; see Figure 4.12(a). However, even if `f_th0` is not very smooth, the marginal density `f_th0_h` is a reasonably good approximation of the density of amplitude, see Figure 4.12(b). (Figure 4.12 is

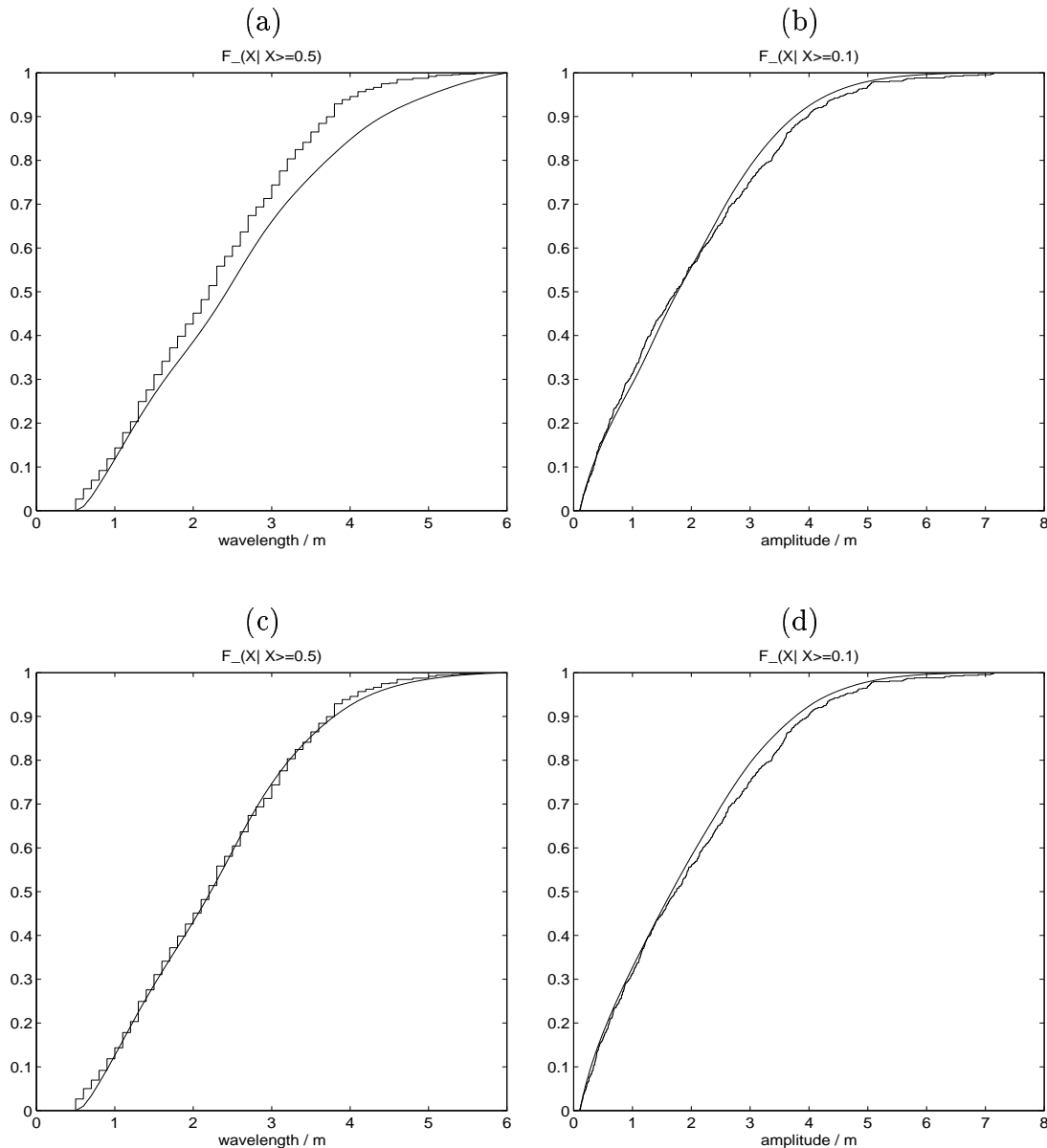


Figure 4.12: Empirical and theoretical distributions of *min2Max* wavelength (in (a) and (c)) and amplitude (in (b) and (d)); inaccurate and accurate approximations with $NIT = 0$ and $NIT = 4$.

obtained using `empdistr(T_mM,0.5,f_th0_t)`; and `empdistr(H_mM,0.1,f_th0_h)`;) We also check the accuracy of the approximation based on $NIT = 4$ by comparing the empirical distributions of amplitude and wavelength with the densities `f_th4_t` and `f_th4_h`. The plots are given in Figure 4.12(c) and (d), and the accuracy of the approximations is very good.

Trough2crest amplitude and wavelength

This section can be omitted at first reading. We shall here analyse the wavelength and amplitude distribution for *trough2crest* waves. In order to obtain enough high wave we simulate three long sequences of waves, extracted the *min2Max* cycles `mM` and *trough2crest*

cycles TC and combined them into one. We shall use $3*NmM$ as the expected number of extremes instead of NmM . The sequence of commands is

```
>> L = simspec([15 3 0.1],spec,g);
>> TP = dat2tp(L(:,[1 2])); mM=tp2mm(TP);
>> T_mM = tp2wa(TP); [TC T_tc] = tp2tc(TP,0);
>> for i=1:2
    L = simspec([15 3 0.1],spec,g);
    TP = dat2tp(L(:,[1 2]));
    mM=[mM ; tp2mm(TP)]; T_mM = [T_mM ; tp2wa(TP)];
    [TC0 T_tc0] = tp2tc(TP,0);
    TC=[TC; TC0]; T_tc=[T_tc ; T_tc0];
end
```

In the first section we introduced the mean separated min2Max waves, simply by considering those min2Max cycles which cross the reference level. Clearly those waves are as many as the trough2crest waves and we have shown that the large trough2crest waves are identical to the large mean separated min2Max waves. In order to illustrate this fact we first find the mean separated min2Max cycles with positive maxima and negative minima, which we shall denote mMp ,

```
>> index = find(mM(:,1)<0 & 0<mM(:,2));
>> mMp = mM(index,:);
>> T_mMp = T_mM(index,:);
>> H_mMp = mMp(:,2)-mMp(:,1);
```

We shall also need the trough2crest amplitudes,

```
>> H_tc = TC(:,2)-TC(:,1);
```

To find the large waves in mMp we consider only waves with amplitudes greater than 4,

```
>> index4 = find(H_mMp>4);
>> H_mMp4 = H_mMp(index4); T_mMp4 = T_mMp(index4);
>> Index4 = find(H_tc>4);
>> H_tc4 = H_tc(Index4); T_tc4 = T_tc(Index4);
>> plot(T_tc4,H_tc4,'o'); hold;
>> plot(T_mMp4,H_mMp4,'.'); axis([0 8 3 8]);
```

The plot is presented in Figure 4.13(a) where the circles represent the trough2crest wavelength and amplitude and the dots are the mean separated min2Max wavelength and amplitude for large waves. We can see that most of the circles contain dots inside, showing that we can use large min2Max waves to approximate the large trough2crest waves.

The program `wave_th` can also be used to compute an approximation of the mean separated min2Max wavelength and amplitude distribution, and we directly compute the most accurate approximation using $NIT = 4$,

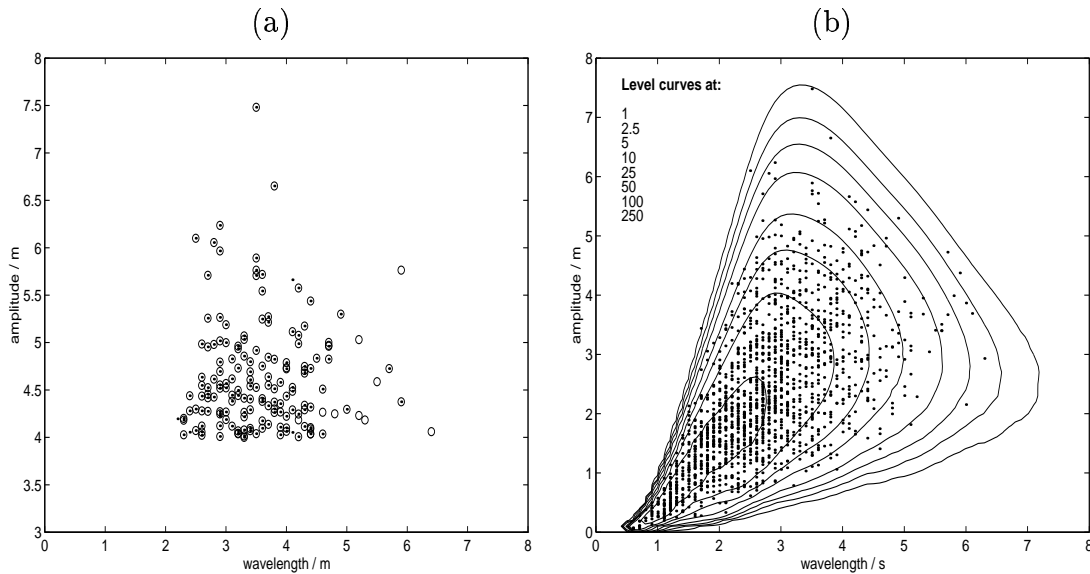


Figure 4.13: (a) Observed trough2crest (circles) and mean separated min2Max (dots) wavelength and amplitude. (b) Accurate theoretical density of min2Max wavelength and amplitude, $NIT = 4$.

```
>> mL = mean(L(:,2));
>> utc = mL;
>> wt = [0:0.1:8]; wh = [0:0.1:8];
>> [f_th4_p f_th4_t_p f_th4_h_p] = wave_th(spec,1,g,4,wt,wh,utc,15,20);
```

The inputs '1' and '4' define that the mean separated min2Max wavelength and amplitude density has to be computed with $NIT = 4$. The eighth input '15' defines the number of times points in the FFT algorithm used to compute the covariances from the spectrum and the last input '20' gives the number of nodes used for description of the height of local maxima. The default value for this argument is '16' which is too much for narrow band processes but too small for the broad banded, which we can see in Figure 4.11 where the density is somewhat irregular for the long wavelengths. However, this is only an aesthetic problem, since the accuracy of `f_th4` is otherwise very good, and it can be reduced by increasing the number of nodes describing the heights of local maxima. The price to pay is the computation time. (Observe that only `spec` is required for `wave_th`, all other inputs have default settings.) Figure 4.13(b) shows isolines of `f_th4_p` together with observed wavelengths and amplitudes,

```
>> cowa([T_mMp H_mMp],wt,wh,3*NmM*f_th4_p,clevels,6)
```

Now we turn to the accuracy of the marginal densities `f_th4_t_p` and `f_th4_h_p`,

```
>> empdistr(T_mMp,0,f_th4_t_p);
>> empdistr(H_mMp,0,f_th4_h_p);
```

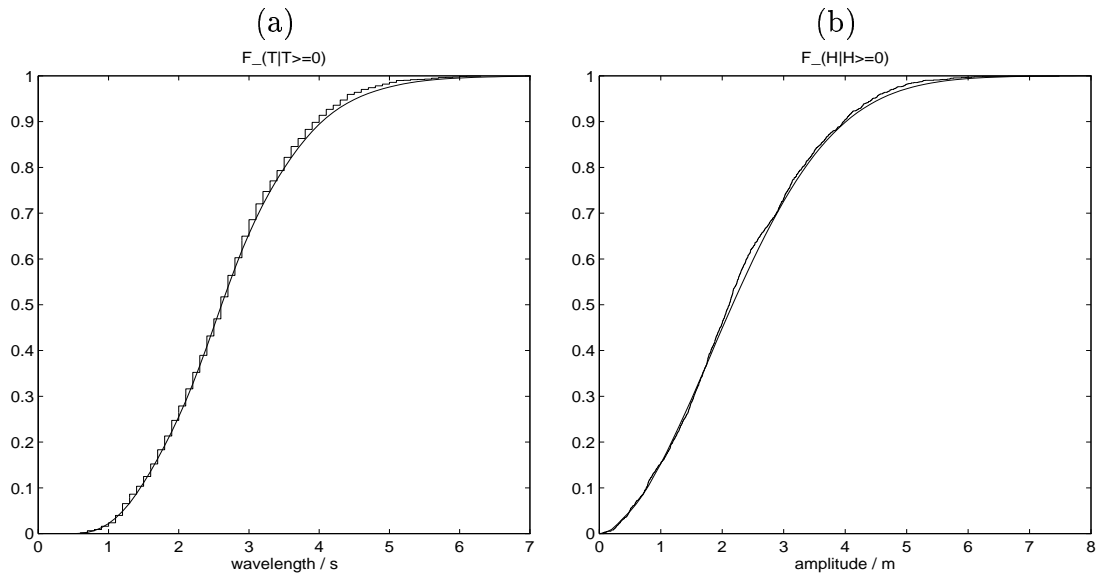


Figure 4.14: Empirical and theoretical densities of mean separated *min2Max* wavelength (in (a)) and amplitude (in (b)); $NIT = 4$.

The plots are given in Figure 4.14 and we can see that the accuracy is very good.

In Chapter 6 we shall return to some of the functions computed in this chapter, in particular to the densities `f_mM3` and `f_ct3`, and therefore we save the current variables in a MATLAB-file

```
>> save Chap_4
```

4.4 Summary of Gaussian crest and trough functions

| | |
|-----------------------|--|
| <code>cowa</code> | <code>cowa([T H],,wt,wh,f_th)</code> |
| | Plots wavelengths T and amplitudes H as points in the plane together with isolines of a matrix <code>f_th</code> with the wavelength/amplitude density. Vectors <code>wt</code> , <code>wh</code> contain the grid points on which the matrix is computed. |
| <code>empdistr</code> | <code>empdistr(obs,cut,dens)</code> |
| | Plots the empirical cumulative distribution function for the observations in the column vector <code>obs</code> and compares with the distribution given by the matrix <code>dens</code> . Here <code>dens</code> is a two column matrix with values in the first column and density values in the second column. Only values larger than <code>cut</code> are considered. |
| <code>jonswap</code> | <code>spec = jonswap(data)</code> |
| | Gives the jonswap spectral density with frequencies and parameters specified in <code>data</code> . |
| <code>mctp2tc</code> | <code>f_tc = mctp2tc(f_mM,def,param)</code> |
| | Calculates the joint intensity (or density) of troughs and crests from the intensity (density) <code>f_mM</code> of min2Max cycles. |
| <code>minmax</code> | <code>[fmM fM fm]=minmax(S,g,nit)</code> |
| | Calculates the joint density of minimum and the following maximum <code>fmM</code> and the marginal densities of maxima <code>fM</code> and minima <code>fm</code> . Accuracy specified by <code>nit</code> . |
| <code>normspec</code> | <code>[normS xl4] = normspec(spec)</code> |
| | Normalizes the spectral density <code>spec</code> and calculates the fourth spectral moment <code>xl4</code> . |
| <code>ochi</code> | <code>g = ochi</code> |
| | Gives the Ochi transformation $g(y)$ with default values. |
| <code>simspec</code> | <code>L = simspec([k na dt],S,g)</code> |
| | Simulates 2^k points in a Gaussian (or transformed Gaussian if <code>g</code> is specified) process with spectral density <code>S</code> ; time separation <code>dt</code> . |
| <code>wave_t</code> | <code>f_t = wave_t(S,def)</code> |
| | Calculates density of min2Max wavelength T , if <code>def=0</code> , or density of half wavelength T^{utc} , if <code>def=1</code> . |
| <code>wave_th</code> | <code>[f_th f_t f_h] = wave_th(S,def,g)</code> |
| | Calculates joint density <code>f_th</code> of min2Max wavelength and amplitude T, H , together with marginal densities <code>f_t</code> and <code>f_h</code> , if <code>def=0</code> , or the corresponding densities of mean separated wavelength and amplitude, if <code>def=1</code> . |

CHAPTER 5

ESTIMATION OF THE GAUSSIAN TRANSFORM MODEL

5.1 The transformation g

In the `WAVE ANALYSIS TOOLBOX` we are modelling the sea level using the transformed Gaussian process $X(t) = G(\tilde{X}(t))$ defined in Section 4.1.1. The process is specified by two matrices `spec` and `g`, containing the spectral density function $S(f)$ and the transformation $g = G^{-1}$, respectively. In the previous chapter we used theoretically derived spectral densities (the JONSWAP spectrum) and the transformation proposed by Ochi and Ahn [11]. In this section we shall present programs which estimate the spectral density and the transformation given the observations (saved in `sea.dat`). The theoretical background and detailed discussion of the method is given in [21]. In that paper we proposed to choose g so that the observed crossing spectra (`cross`) agrees with the expected crossing spectrum for the transformed Gaussian process. More precisely, the transformation is defined to make the expected crossing intensity of the sea level equal to the expected crossing intensity of the transformed Gaussian process. (Of course, the true crossing rate for the sea level is unknown and has to be estimated before we can define the transformation g .)

The transformation `g` is estimated from the crossing spectrum using the `WAVE ANALYSIS TOOLBOX`-function `cross2tr`,

```

>> ma = mean(xx(:,2)); sa = std(xx(:,2));
>> [g test] = cross2tr(cross,ma,sa);

```

where `cross` contains the observed crossing spectrum of `xx` and `ma` and `sa` are the estimated mean and standard deviation in `xx`, respectively. Recall that we saved `sea.dat` in `xx` in Section 2.1 and the crossing spectrum `cross` in Section 2.3. (In this case `ma = 0`, since the mean has already been subtracted in `xx`.)

The transformation g is a continuously differentiable and one-to-one function derived from the expected crossing intensity of the sea level. However, in practice, we have only the observed crossing intensity, which, as we discussed before, is only a left continuous function and usually very irregular. Consequently it has to be smoothed before it can be used to compute the transformation `g`. The smoothing is done automatically by `cross2tr`. In order to check the smoothing `cross2tr` also makes a plot of an “empirical” transformation, obtained by taking `cross` (without smoothing) as an estimate of the expected crossing

intensity, together with \mathbf{g} , and a dashed straight line which represents a Gaussian model, see Figure 5.1(a).

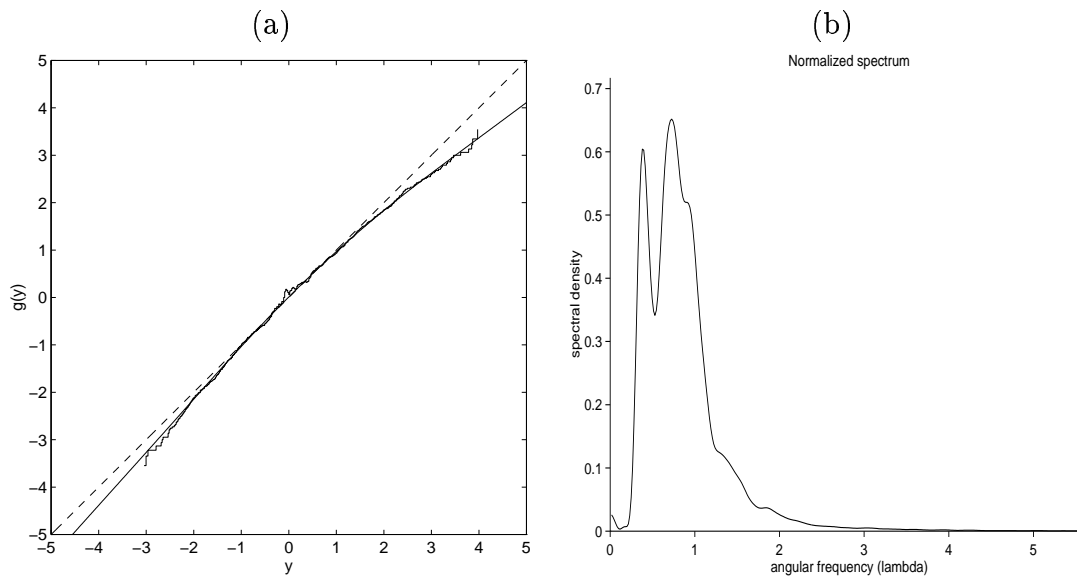


Figure 5.1: (a) *Estimated transformation $g(x)$ for sea.dat.* (b) *Estimated spectral density.*

Obviously, the more the transformation \mathbf{g} deviates from a straight line, the more non-Gaussian features are present in the data. This is measured by a single value, denoted in [21] by $e(g)$, which is an L^2 distance between \mathbf{g} and the dashed line. The $e(g)$ -value is obtained as the second output from the function `cross2tr`, i.e. `test` in this example.

5.2 Estimation of the spectral density S

The estimate of the spectral density `spec` is computed in the `WAVE ANALYSIS TOOLBOX` using the function `dat2spec`. There are many methods (and programs) to estimate the spectrum, mainly based on the FFT-algorithm and some smoothing operation. Any of the algorithms can be used. However, here we shall give our own program, which is more adapted for modelling of the sea level by a transformed Gaussian process. Since a transformed Gaussian model is used in the `WAVE ANALYSIS TOOLBOX` we need to estimate the spectral density from the transformed observations $\tilde{x}(t) = g(x(t))$, which under our assumptions is a sample of a Gaussian process, instead of the measured signal $x(t)$, which is a sample of transformed Gaussian model. (Usually the difference between the spectral densities estimated from $\tilde{x}(t) = g(x(t))$ and from $x(t)$ is negligible.) The procedure is as follows: First the program estimates the transformation g , next the data vector `xx` is transformed and saved as `yy`, and finally the spectral density is computed (from `yy`) using the FFT transform and Hamming window smoothing. In the following we illustrate the different uses of `dat2spec`, and start with the simplest one,

```
>> [S1 g] = dat2spec(xx);
```

where $S1$ is the estimated spectral density for yy and g is the estimated transformation. The spectrum is shown in Figure 5.1(b). Since the spectrum is too irregular we change the Hamming window width and obtain a smoother estimate of the spectrum,

```
>> [S1 g] = dat2spec(xx,50);
```

see Figure 5.2(a).

If we want to use a Gaussian model we can add a third input and set it to 2,

```
>> [S2 g] = dat2spec(xx,50,2);
```

to indicate that xx already is a sample of a Gaussian process. The only transformation needed in that case is linear, in order to standardize the transform to mean zero and variance one, and the g -output will get the form $g(:,1) = ma + sa*g(:,2)$.

One can also use the transformation g as input parameter into the program and compute a corresponding spectral density,

```
>> go = ochi; So = dat2spec(xx,50,3,10,go);
```

Here, the third input 3 indicates that xx is a sample of the transformed Gaussian process with known transformation go and that no transformation needs to be computed. The fourth input defines the size of the frequency grid on which the spectral density is to be computed (here the number of frequencies is 2^{10}).

Finally, we shall mention an important property of the `dat2spec` program. Since the signal xx usually contains time and space sampled values of a sea level $x(t)$ it can happen that xx is constant in some intervals. This adds high frequencies to the estimated spectral density. The program `dat2spec` resolves this problem by first estimating the irregularity factor from xx , and then truncating high frequencies in the estimated spectral density so that the transformed Gaussian process will have the same irregularity factor as observed in the data. Note that the spectrum is renormalized so that the first two spectral moments are one, and hence the average time between the mean level upcrossings of the transformed Gaussian process is 2π . In the following subsection we shall briefly address the problem of “oversmoothing” the spectral density.

5.3 The covariance function

In the WAVE ANALYSIS TOOLBOX-programs computing the theoretical wave characteristics, e.g. `wave_t`, `wave_th`, and `minmax`, the spectrum `spec` is always used as input. Internally,

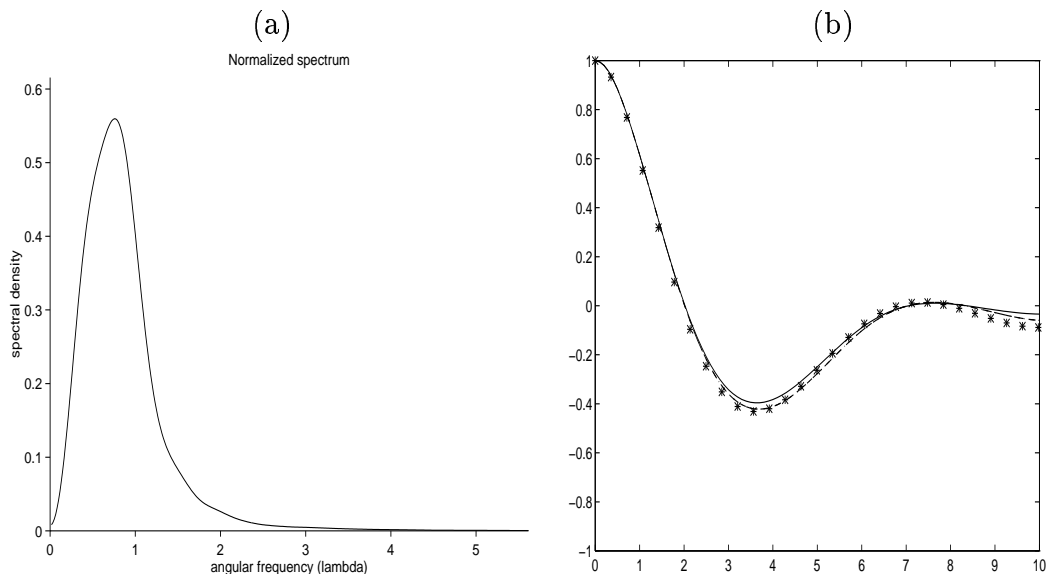


Figure 5.2: (a) Estimated normalized spectral density for `sea.dat`; Hamming window width = 50. (b) Estimated covariance function (stars) and covariance functions corresponding to estimated spectrum (solid curve: window width = 50, dashed curve: window width = 80).

the programs use the covariance function of the process and its derivatives. Consequently we have a program `s2c` which transforms the spectrum into the covariance function and up to four of its derivatives. The output of `s2c` is written on ASCII files which are then read by the calculating programs. The `s2c` program, which has no explicit MATLAB-output, is an internal function which is not ment to be used explicitly. However, we shall present a specific application of `s2c` for a rough check of the correctness of the estimated spectrum, and also illustrate the effects of normalization of the spectrum. The plot in Figure 5.2(b) is obtained from the following commands,

```
>> s2c([101 14 0.1],S1,0); load Cd0.in;
>> plot(Cd0(:,1),Cd0(:,2)); hold
```

It shows the autocorrelation function defined by the spectrum `S1`.

Now, using the program `dat2cov` we can estimate the correlation function of the signal $\tilde{x}(t) = g(x(t))$. (Since the spectrum is normalized, the correlation function is equal to the covariance function.) First we obtain $\tilde{x}(t)$ and save it in `yy`,

```
>> yy = gaus2dat(xx,[g(:,2) g(:,1)]); cov = dat2cov(yy(:,2),32);
```

(The program `gaus2dat` is also an internal function used in the simulation of the transformed Gaussian process. Here we are using it in the inverse direction to obtain a Gaussian process `yy` from the non-Gaussian input `xx`, hence the reversed order of columns in the `g`-argument.)

Before we can plot `cov` we need to find the correct time scale and from the first few elements in `xx` we can see that the sampling step of the signal is 0.25. Next we estimate the distance between zero-upcrossings in `xx`,

```
>> Tcross = max(xx(:,1))/max(cross(:,2));
```

Since we have normalized the spectrum so that the distance between the zero-upcrossings in the transformed Gaussian process is 2π we have to use the same time scale for the estimated covariance and hence we plot

```
>> plot(2*pi*(0:0.25:8)/Tcross,cov/cov(1),'*'); axis([0 10 -1 1]);
```

We are limiting the axis to the interval $[0, 10]$ since this is the range on which the programs work, see Figure 5.2(b). We can see some differences caused by smoothing with the Hamming window. Since the estimated correlations are based on a quite large amount of data one can ask whether this difference is significant. There is a large literature on this problem which shall not be discussed here and we only illustrate the problem by changing the width of the Hamming window and plot the corresponding correlation function as the dashed line in Figure 5.2(b),

```
>> [S1a g] = dat2spec(xx,80,1);
>> s2c([101 14 0.1],S1a,0); load Cd0.in;
>> plot(Cd0(:,1),Cd0(:,2),'--');
```

The estimated spectral density `S1a` in Figure 5.3(a) is more irregular, but the dashed line for the correlation in Figure 5.2(b) agrees better with the dots representing an estimate of the correlation function. The question which of the spectral densities `S1` or `S1a` is most appropriate to use is difficult to answer.

5.4 Validation of the model

In previous sections we have estimated the transformation `g` and the spectral density `S1`, and hence the transformed Gaussian model for the function `xx` is specified. A basic question is whether one needs to use the transformed Gaussian model at all instead of the simpler Gaussian model. In other words we may ask if the departure of the transformation `g` from the dashed line in Figure 5.1(a) is due to randomness or if it is significant.

First of all we have checked that the difference between the spectral densities `S1` and `S2` is negligible and hence we only need to show that the transformation `g` differs significantly from the straight line. Equivalently, we have to investigate whether the distance measure $e(g)$, which for the `sea.dat` was equal to 0.16, differs significantly from zero.

In order to check the significance of the observed $e(g)$ we use a Monte-Carlo method as follows. We assume a Gaussian model for the data with spectral density `S1`. Next we simulate

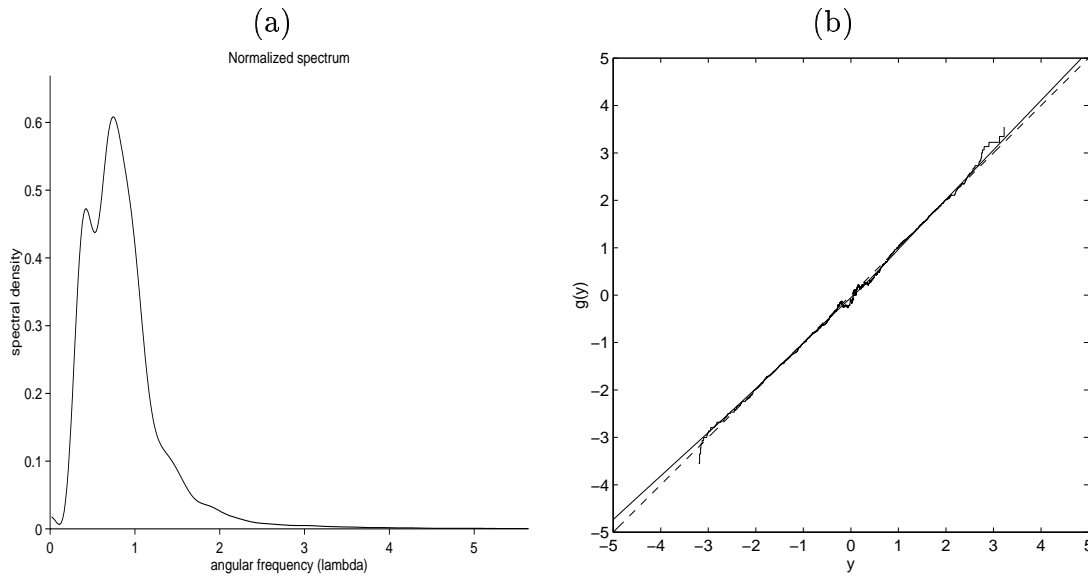


Figure 5.3: (a) Estimated normalized spectral density for `sea.dat`; Hamming window width = 80. (b) Transformation $g(x)$ estimated from 540 trough2crest waves in a simulated Gaussian process.

independent samples of a zero-mean Gaussian process with spectrum `S1` using the procedure `simspec`. The simulated samples have to contain the same number of waves as the data set on average. We then find a crossing spectrum `cross` in the simulated sample, estimate g and compute $e(g)$ using `cross2tr`.

Obviously for an ergodic Gaussian process $e(g)$ is zero for an infinitely long sample and hence the simulated $e(g)$ represents the variability of the estimator g due to the limited number of observed waves. We give now commands for one such iteration in the `WAVE ANALYSIS TOOLBOX`,

```
>> dt = 2*pi*max(cross(:,2))/2^15; LG = simspec([15 3 dt], S1);
>> TP = dat2tp(LG(:,1:2)); mM = tp2mm(TP); cross = mm2cross(mM);
>> [gG test] = cross2tr(cross,0,1); test
```

(Here we have used the fact that the spectrum is normalized and hence we have simulated a sample of a Gaussian process with mean zero and variance one.) Figure 5.3(b) shows the transformation gG . The `test` variable is equal to 0.03 (as compared to 0.16 for the observed data `sea.dat`). By repeating the procedure a few times we can directly conclude that the `sea.dat` can not be modelled by a Gaussian process.

The departure of `sea.dat` from the Gaussian distribution can also be seen from a plot of the marginal distribution. The `WAVE ANALYSIS TOOLBOX` contains a routine `normplot` for plotting of an empirical cumulative distribution function on normal probability paper. To get approximately independent observations we take the values in `xx` separated by 30 sampling steps. The command

```
>> norm = normplot(xx(1:30:length(xx(:,2)),2))
```

will give an estimate of the mean and standard deviation of \mathbf{xx} and the normal probability plot in Figure 5.4. As seen there is a clear deviation of the distribution from the straight line, which resembles a Gaussian distribution.

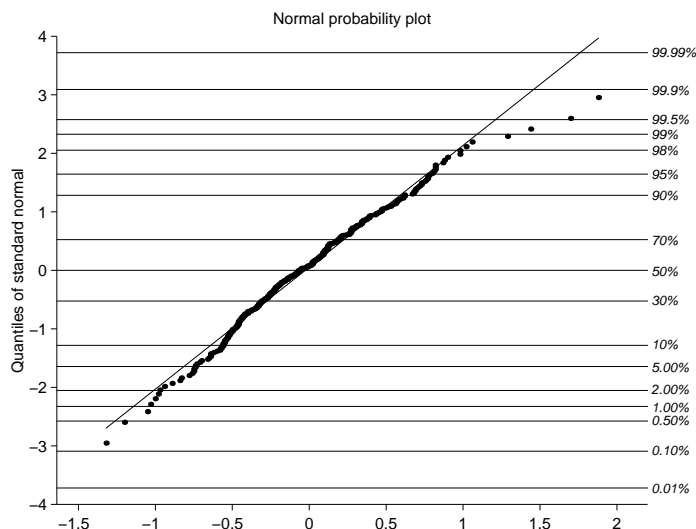


Figure 5.4: Plot of `sea.dat` values on normal probability paper.

The `sea.dat` file is relatively long and it contains 540 trough2crest waves making it easy to verify the significance of the non-Gaussianity. However, often the data records are much shorter, and then model estimation and validation is more difficult. We shall illustrate this by simulating two samples of LG one fourth as long and compute the `test` variables for these samples. The results are shown in Figure 5.5 where we can see that `g` varies much more around the theoretical straight line. The `test` variables are equal to 0.12 and 0.13 for these two simulations. Consequently, if the sea level data are correctly modeled by the transformed Gaussian model with transformation g as in Figure 5.1(a), then if we had only recorded one fourth of the `sea.dat` data set, then it would not have been possible to distinguish the “true” model from the Gaussian one.

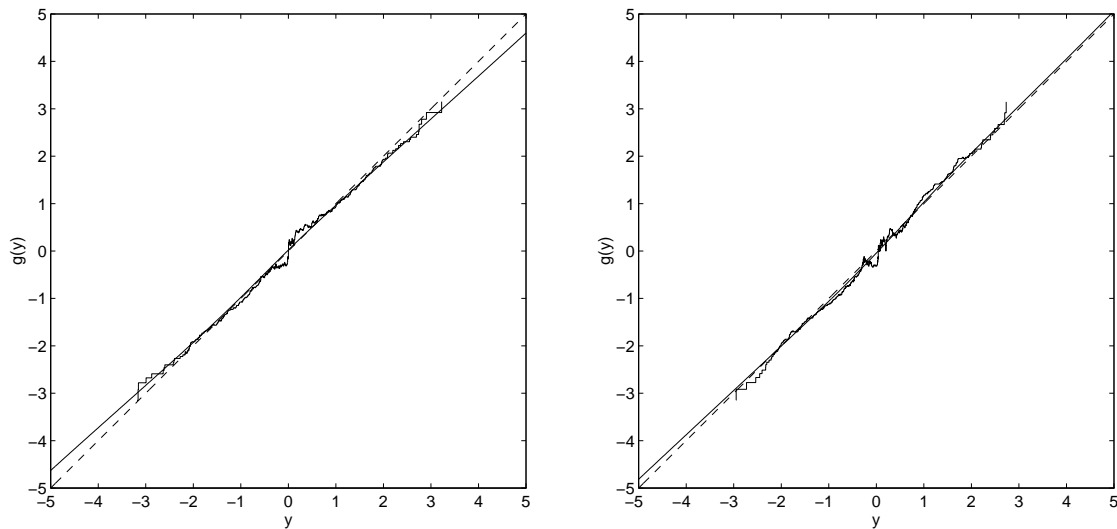


Figure 5.5: *Two examples of estimated $g(x)$ -transforms, estimated from 135 trough2crest waves in a simulated Gaussian process.*

5.5 Summary of estimation procedures

| | |
|-----------------------|---|
| <code>cross2tr</code> | <code>g = cross2tr(cross,ma,sa);</code> |
| | Estimates the transformation g from the observed crossing spectrum in the data \mathbf{xx} . Variables ma and sa are the mean and standard deviation of \mathbf{xx} , respectively. |
| <code>dat2cov</code> | <code>cov = dat2cov(xx,M)</code> |
| | Estimates $M+1$ points of covariance function of \mathbf{xx} . |
| <code>dat2spec</code> | <code>[S g] = dat2spec(xx)</code> |
| | Estimates the spectral density function S and the transformation g from the data set \mathbf{xx} . |
| <code>gaus2dat</code> | <code>xx = gaus2dat(yy,g)</code> |
| | Transforms the data set \mathbf{yy} (which is a sample of a Gaussian process) to $\mathbf{xx} = G(\mathbf{yy})$, where G is an inverse transformation of g . |
| <code>normplot</code> | <code>theta = normplot(data)</code> |
| | Gives estimates $\mathbf{theta} = [\text{mean std}]$ of the mean and standard deviation and a plot on normal probability paper of \mathbf{data} . |
| <code>s2c</code> | <code>s2c([N K dt],S,Nder)</code> |
| | Evaluates the covariance function and its $Nder$ derivatives from the spectral density function S . Here N is number of grid points and dt the step size where the covariance function is computed, and K is an input to the FFT-algorithm. |

CHAPTER 6

A MARKOV CHAIN MODEL FOR THE TROUGH-CREST SEQUENCE

6.1 Rainflow filtered sea wave data

In previous chapters we have presented models for sea wave data, treated as functions of time. The models can be used in response analysis for marine structures to sea forces or to compute wave characteristics for specified random wave models, e.g. those defined by their power spectrum. In particular, Gaussian models are very convenient as input to linear filters, since the output is again a Gaussian process with easily computable power spectral density function.

The measured signals are often very noisy and need to be smoothed before further analysis. A common practice is to use a bandpass filters to exclude high frequencies from the power spectrum and to filter out slow trends. If the function is modelled by a transformed Gaussian process \mathbf{xx} , such a filtration is performed on the inverse transformed signal $\mathbf{yy} = \mathbf{g}(\mathbf{xx})$. Obviously, one should not oversmooth data since that will affect the height of extreme waves. Consequently, if the signal is still too irregular even after smoothing, this is an indication that one should use the `trough2crest` waves, defined as in Definition 1.1, instead of the simpler `min2max` cycles, as in Definition 1.2. A significant part of this tutorial has aimed at showing how one can compute the `crest2trough` wave characteristics from a Gaussian or transformed Gaussian model.

The `trough2crest` wave concept is a nonlinear means to remove small irregularities from a wave series. Another nonlinear method to remove small waves from data is the rainflow filtering, introduced in Definition 1.3. For completeness, we again describe the rainflow filter.

In this tutorial we have used a simplified definition of rainflow cycles convenient for functions with finitely many local maxima and minima. However, rainflow filters and rainflow cycles can be defined for very irregular functions like a sample function of Brownian motion where there are infinitely many local extremes in any finite interval, regardless how small. This is done by defining the rainflow minimum $m^{rfc}(t)$ for all time points t of a function $x(t)$ in such a way that the rainflow amplitude $x(t) - m^{RFC}(t)$ is zero if the point $x(t)$ is not a strict local maximum of the function; see Rychlik [19] for more detailed discussion. Now, a *rainflow filter with threshold h* , extracts all rainflow cycles $(m^{RFC}(t), x(t))$ such that $x(t) - m^{RFC}(t) > h$. Consequently, if $h < 0$ then the signal is unchanged by the filter, if $h = 0$ we obtain a sequence of turning points, and finally, if $h > 0$, all small oscillations are removed, see Figure 2.2.

6.2 Oscillation count and rainflow matrix

In Chapter 1 we introduced the oscillation count $N^{osc}(u, v)$ which counts the number of times a signal crosses an interval $[u, v]$. We also remarked that the oscillation count is a counting distribution for the rainflow cycles. Consequently, if the matrix `N_osc` with elements $N^{osc}(u_j, u_i)$ is known we can compute the frequency (or rather histogram) matrix of the rainflow count by means of the `WAVE ANALYSIS TOOLBOX`-function `nt2fr` and obtain the matrix `F_rfc = nt2fr(N_osc)`, in fatigue practise called the *rainflow matrix*. Knowing the rainflow matrix of a signal one can compute the oscillation count by means of the function `fr2nt`.

The rainflow matrix will play an important role in the analysis of the rainflow filtered signals. Let $x(t)$ be a measured signal and denote by $x_h(t)$ the rainflow filtered version, filtered with threshold h . Now, if we know a rainflow matrix `F_rfc`, say, of x , then the rainflow matrix of x_h is obtained by setting some subdiagonals of `F_rfc` to zero, since there are no cycles in x_h with amplitudes smaller than h . Obviously, the oscillation count of x_h can then be derived from the oscillation count of x .

Note that extracting a sequence of troughs and crests (m_i^{tc}, M_i^{tc}) from the signal is closely related to rainflow filtering. Since given a reference level u^{tc} , the sequence (m_i^{tc}, M_i^{tc}) can be obtained by first removing all rainflow cycles (m_j^{RFC}, M_j) such that $M_j < u^{tc}$ or $m_j^{RFC} > u^{tc}$ and then finding the min2Max pairs in the filtered signal, as in Section 6.4.

Clearly, the oscillation count is an important characteristic of irregularity of a sea level function and the expected oscillation count, also called an oscillation intensity matrix, is an important characteristic of the random processes. Consequently we face two problems; how to compute the oscillation intensity, for a specified model; and if knowing the oscillation intensity, how can one find an explicit and easy to handle random process with this intensity. Note that by solving these two problems one increases the applicability of rainflow filters considerably. Since then, given a random process, one can find its oscillation intensity, and next one can compute the oscillation intensity of the rainflow filtered random process, and finally, find a random process model for the filtered signal.

6.3 Markov chain of turning points, Markov matrix

Since the oscillation intensity is closely related to the first passage problem it can be practically handled if some Markov structure of the process is assumed. While Gaussian processes are an important class of models for linear filtering, Markov processes are the appropriate models as far as rainflow filtering is concerned. In this section a class of models, the so called Markov chain of turnings points will be introduced.

Let \mathbf{xx} be the sea level function and `TP` the sequence of turning points. The sequence `TP` will be called a Markov chain of turning points if it forms a Markov chain, i.e. if the distribution of a local extremum depends only on the value of the previous extremum. The elements in the histogram matrix of min2Max cycles and Max2min cycles are equal to the observed number of transitions from a minimum (maximum) to a maximum (minimum) of specified height. Consequently, the probabilistic structure of the Markov chain of turning

points is fully defined by the expected histogram matrix of min2Max and Max2min cycles; sometimes called *Markov matrices*. Note that for transformed Gaussian process a Markov matrix for min2Max cycles was computed by means of the WAVE ANALYSIS TOOLBOX-function `minmax`. The Max2min matrix is obtained by symmetry. Next, the function `mctp2tc` (= Markov Chain of Turning Points to Trough Crests), used to compute the trough2crest intensity for transformed Gaussian model, uses a Markov matrix to approximate the sequence of turning points by a Markov chain. This approximation method is called a *Markov method*.

Finding the expected rainflow matrix is a difficult problem and explicit results are known only for special classes of processes, e.g. if x is a stationary diffusion, a Markov chain or a function of a vector valued Markov chain. Markov chains are very useful in wave analysis since they form a broad class of processes and for several sea level data, as well as for transformed Gaussian processes, one can observe a very good agreement between the observed or simulated rainflow matrix and that computed by means of the Markov method. Furthermore, Markov chains can be simulated in a very efficient way. However, the most important property is that, given a rainflow matrix or oscillation count of a Markov chain of turning points one can find its Markov matrix. This means that a Markov chain of turning points can be defined by either a Markov matrix `F_mM` or by its rainflow matrix `F_rfc`, which are connected by the following nonlinear equation

$$\mathbf{F_rfc} = \mathbf{F_mM} + \mathcal{F}(\mathbf{F_mM}), \quad (6.1)$$

where \mathcal{F} is a matrix valued function, defined in [19], where also an algorithm to compute $(\mathcal{I} + \mathcal{F})^{-1}$ is given. The WAVE ANALYSIS TOOLBOX-function for computing `F_rfc` from `F_mM` is `mctp2rfc` while the inverse, i.e. `F_mM` as a function of `F_rfc`, is computed by `rfc2mctp`.

6.4 Rainflow filtered transformed Gaussian model

Assume that we have a transformed Gaussian model `xx` for sea data, and that the power spectrum `spec` and the transformation `g` are given. Then we wish to smooth `xx` by eliminating small oscillations. In this section we shall describe a method to handle rainflow filtered data for this theoretical model.

Let `g` be the transformation given by Ochi and Ahn [11] and suppose the spectrum `spec` is of the JONSWAP type,

```
>> g = ochi; spec = jonswap; param = [-3 5 81];
>> L = simspec([15 3 0.1],spec,g); xx=L(:,1:2);
>> xx_05 = dat2tp(xx,0.5);
>> plot(xx(:,1),xx(:,2)); hold;
>> plot(xx_05(:,1),xx_05(:,2),'--'); axis([0 50 -3 5])
```

The rainflow filtered data `xx_05` contains the turning points of `xx` with rainflow cycles less than 0.5 removed.

In order to analyse the statistical properties of `xx_05` we need to find its oscillation intensity, or equivalently the rainflow matrix, and we start with the rainflow intensity of the

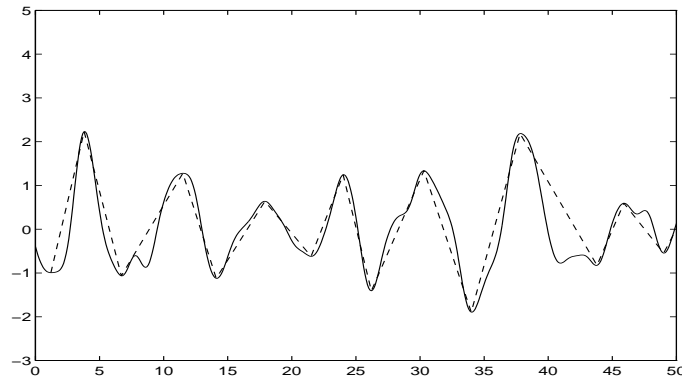


Figure 6.1: *Ochi transformed wave data and its rainflow filtered sequence of turning points, $h = 0.5$.*

process \mathbf{xx} . As in Section 4.2 we computed the approximation $\mathbf{f_mM3}$ of the min2Max density and approximate the sequence of turning points by a Markov chain with transitions between extrema defined by $\mathbf{f_mM3}$. Then, by (6.1) we compute the rainflow matrix.

```
>> load Chap_4
>> F_rfc = mctp2rfc(F_mM);
```

The rainflow matrix $\mathbf{F_rfc}$ is shown in Figure 6.2(a), obtained using

```
>> lev = levels(param);
>> contour(lev,lev,NmM*flipud(F_rfc),clevels), axis('square')
```

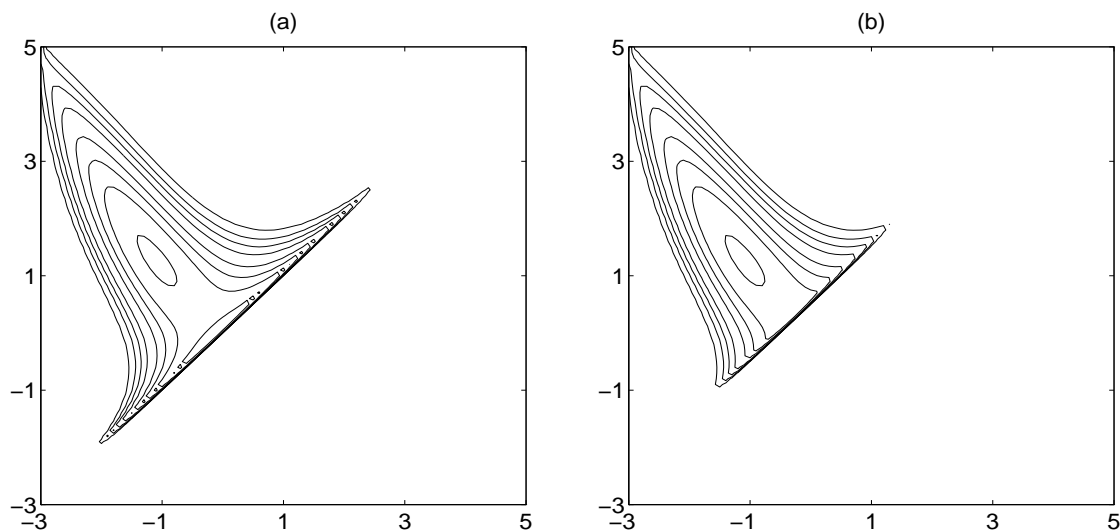


Figure 6.2: *Rainflow matrices for original (a) and rainflow filtered (b) waves.*

The rainflow matrix $\mathbf{F_rfc1}$ of the filtered process $\mathbf{xx_05}$ can be computed using the following sequence of commands,

```

>> h = 0.5;
>> dh = (param(2)-param(1))/(param(3)-1);
>> n = floor(h/dh); N = length(F_rfc);
>> fact = ones(N,N);
>> for i = 1:n, fact = fact - rot90(diag(ones(N-i,1),i),1); end
>> F_rfc1 = F_rfc.*fact;

```

The rainflow matrix `F_rfc1` is shown in Figure 6.2(b), obtained by

```

>> contour(lev,lev,flipud(F_rfc1),clevels), axis('square')

```

Finally, the joint density of minimum and the following maximum `F_mM1` in the signal `xx_h` is computed by

```

>> F_mM1 = rfc2mctp(F_rfc1);

```

(Note that `F_mM1` can be computed by `F_mM1 = rfc2mctp(F_rfc,h)`.) The densities `F_mM`, `F_mM1` are shown in Figure 6.3(a), (b), respectively.

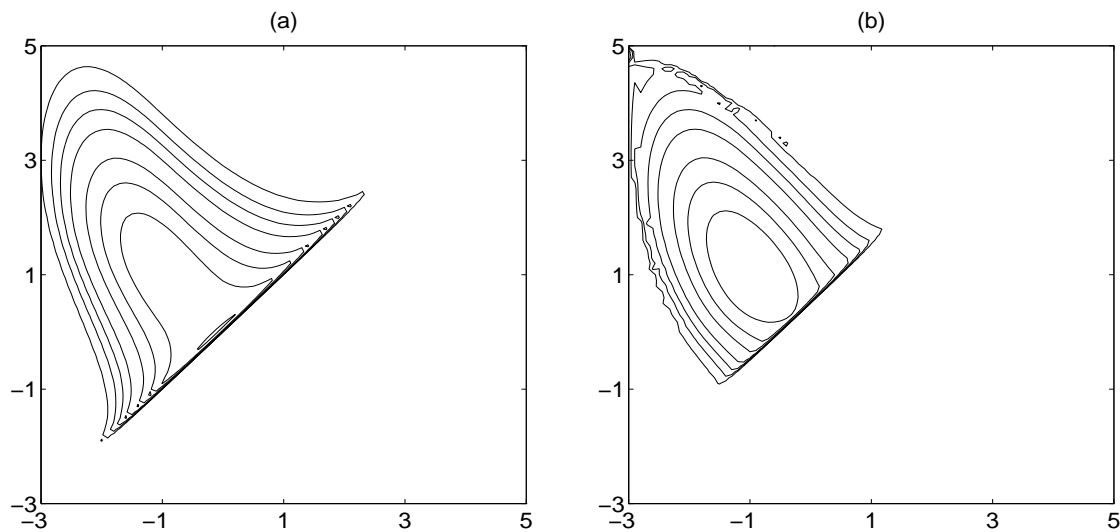


Figure 6.3: *Isolines of intensity of min2Max cycles in original signal (a) and in rainflow filtered signal (b); $h = 0.5$.*

One of the important uses of the density `F_mM1` is for simulation of a Markov chain of turning points which is an approximation of a sequence of turning points of `xx_05`,

```

>> TP = mctp2tp(param,100,f_mM);
>> TP_h = mctp2tp(param,100,F_mM1);

```

The sequences `TP` and `TP_h` contain 201 local extremes and are good approximations of the turning points of `xx` and `xx_05`.

Finally, we shall show that the sequence of troughs and crests (m_i^{tc}, M_i^{tc}) is actually a special form of rainflow filtering of \mathbf{x} . More precisely, take the reference level $u^{tc} = 0$ and let $\mathbf{F_rfc2}$ be the rainflow matrix of the sequence of trough2crest waves computed as follows,

```
>> utc = 0; u = levels(param);
>> udisc = fliplr(u);
>> ntc = sum(udisc>utc); ntc1 = sum(u<utc);
>> fact = zeros(N,N); fact(1:ntc,1:ntc1) = ones(ntc,ntc1);
>> F_rfc2 = fact.*F_rfc;
>> contour(lev,lev,NmM*flipud(F_rfc2),clevels)
```

The density $\mathbf{F_rfc2}$ is presented in Figure 6.4(a), which should be compared to Figures 6.2.

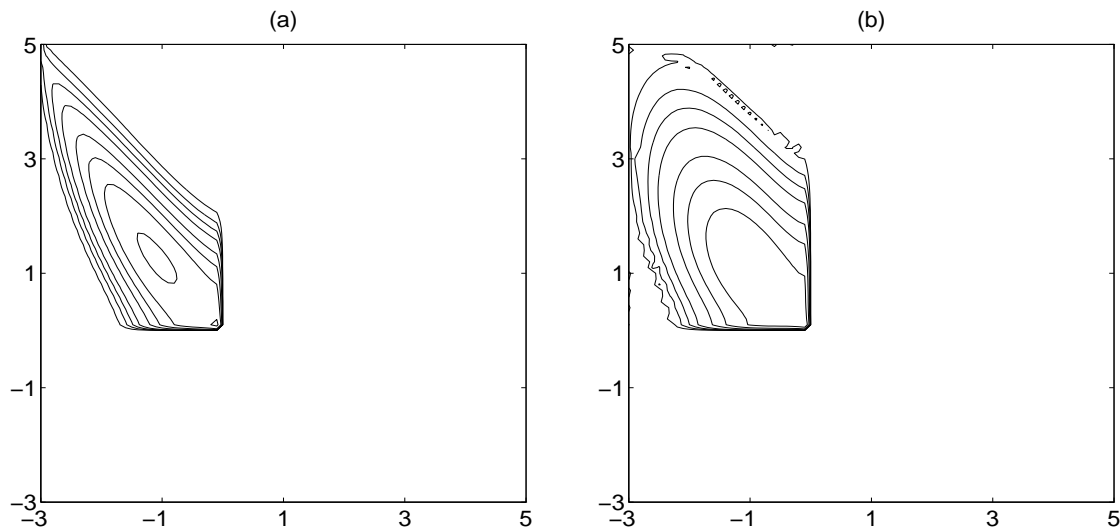


Figure 6.4: (a) Isolines of mean separated rainflow distribution (with negative minima and positive maxima); (b) Isolines of trough2crest distribution obtained by inversion of the distribution in (a).

Now the sequence of trough2crest waves is a sequence of turning points which has rainflow matrix $\mathbf{F_rfc2}$. Consequently, the joint density of troughs and crests $\mathbf{F_tc}$, say, can be computed by inverting equation (6.1),

```
>> F_tc = rfc2mctp(F_rfc2,0,param,0.00001);
>> contour(lev,lev,NmM*flipud(F_tc),clevels)
```

The density $\mathbf{NmM} \cdot \mathbf{F_tc}$ with $\mathbf{NmM} = 763.15$ taken from Section 4.3, is shown in Figure 6.4(b) which should be compared to Figure 4.4(b).¹ Simply, the $\mathbf{F_tc}$ density computed by `rfc2mctp` and that computed by `mctp2tc` are identical up to numerical accuracy. However the algorithm `mctp2tc` is much faster, since it does not involve solutions of nonlinear equations.

¹The irregular pattern in Figures 6.3b and 6.4b is due to numerical instability in the presently used algorithm.

6.5 Summary of estimation procedures

| | |
|----------|--|
| mctp2rfc | $F_rfc = mctp2rfc(F_mM);$ |
| | Gives the rainflow matrix F_rfc from the Markov matrix F_mM of turning points. |
| mctp2tc | $F_tc = mctp2tc(F_mM)$ |
| | Calculates the intensity of trough2crest cycles from the Markov matrix F_mM . |
| rfc2mctp | $F_mM = rfc2mctp(F_rfc)$ |
| | Inverts the rainflow matrix to give a Markov matrix of turning points. |

CHAPTER 7

EXTREME VALUE ANALYSIS

Of particular interest in wave analysis is how to find extreme quantiles and extreme significant values for a wave series. Often this implies going outside the range of observed data, i.e. to predict, from a limited number of observations, how large the extreme values might be. Such analysis is commonly known as *Weibull analysis* or *Gumbel analysis*, from the names of two familiar extreme value distributions. The `WAVE ANALYSIS TOOLBOX` contains routines for fitting of such distributions, both for the Weibull and Gumbel distributions, and for two more general classes of distributions, the *Generalized Pareto Distribution* (GPD) and the *Generalized Extreme Value distribution* (GEV).

7.1 Weibull and Gumbel papers

The Weibull and Gumbel distributions, the latter also called the *extreme value distribution*, are two extreme value distributions with distribution functions

$$\text{Weibull: } F_W(x; a, c) = 1 - e^{-(x/a)^c}, \quad x > 0, \quad (7.1)$$

$$\text{Gumbel: } F_G(x; a, b) = \exp\left(-e^{-(x-b)/a}\right), \quad -\infty < x < \infty. \quad (7.2)$$

The Weibull distribution is often used as distribution for random quantities which are the minimum of a large number of independent (or weakly dependent) identically distributed random variables. It is often used as a model for random strength of material, in which case it was originally motivated by the principle of *weakest link*. Similarly, the Gumbel distribution is used as a model for values which are maxima of a large number of independent variables.

Since one gets the minimum of variables x_1, x_2, \dots, x_n by changing the sign of the maximum of $-x_1, -x_2, \dots, -x_n$, one realises that distributions suitable for the analysis of maxima can also be used for analysis of minima. Both the Weibull and the Gumbel distribution are members of the class of Generalized Extreme Value distributions (GEV), which we shall describe in Section 7.2.

We begin here with an example of Weibull and Gumbel analysis, where we plot data and empirical distribution and also estimate the parameters a, b, c in (7.1) and (7.2). We take the sequence of min2Max wave-lengths `T_mM` in the `sea.dat` and try to fit a Weibull distribution,

```
>> wei = weibplot(T_mM);
```

This will result in a two element vector `wei = [a* c*]` with estimated values of the parameters (a, c) in (7.1). The empirical distribution function of the input data is plotted automatically in a Weibull diagram with scales chosen to make the distribution function equal to a straight line. The horizontal scale is logarithmic in the observations x , and the vertical scale is linear in the *reduced variable* $\log(-\log(1 - F(x)))$; see Figure 7.1(a). Obviously, a Weibull distribution is not very well suited to describe the wavelength data.

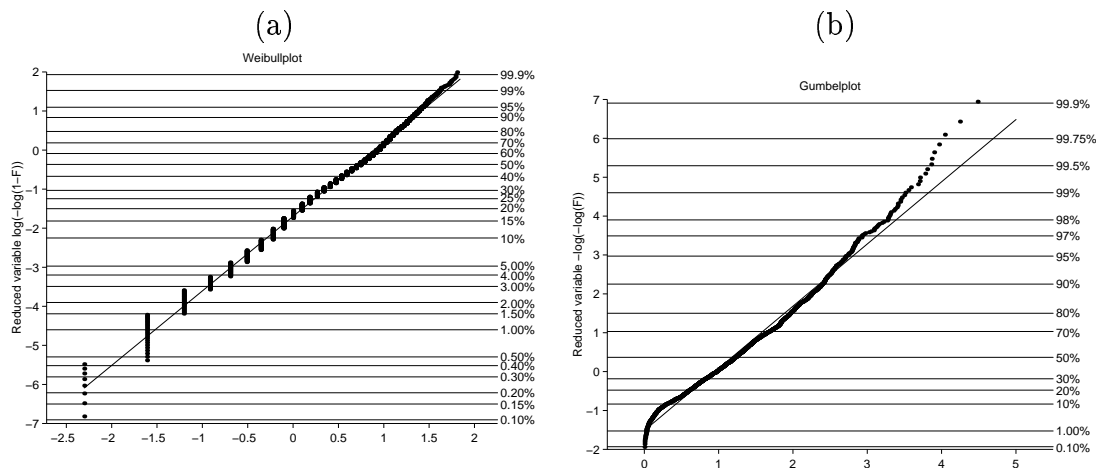


Figure 7.1: (a) *min2Max* wavelengths on Weibull paper. (b) *Crest* values on Gumbel paper.

To illustrate the use of the Gumbel distribution we plot and estimate the parameters (a, b) in the Gumbel distribution (7.2) for the crest values saved in `TC(:,2)`. The command

```
>> gum = gumbplot(TC(:,2));
```

results in a vector `gum` with estimated values `[a* b*]` and the plot in Figure 7.1(b). Here the horizontal axis is linear in the observations x and the vertical axis carries the reduced variable $-\log(-\log(F(x)))$. The fit to a Gumbel distribution is rather bad.

Both `weibplot` and `gumbplot` can take a matrix of observations as input. Then the output will be a two column matrix with one row of estimated parameters for each column of data in the input series.

The parameter estimation in `gumbplot` and `weibplot` is done by fitting a straight line to the empirical distribution functions in the diagrams; see the help text for the relations between the parameter values and the intercept and slope of the fitted line. In the following section we shall describe some more statistical techniques for parameter estimation in the Generalized Extreme Value distribution.

7.2 Generalized Pareto and extreme value distribution

The Generalized Pareto Distribution (GPD) has the distribution function

$$\text{GPD: } F(x; k, \sigma) = \begin{cases} 1 - (1 - kx/\sigma)^{1/k}, & \text{if } k \neq 0, \\ 1 - \exp\{-x/\sigma\}, & \text{if } k = 0, \end{cases} \quad (7.3)$$

for $0 < x < \infty$ if $k \leq 0$ and for $0 < x < \sigma/k$ if $k > 0$. The Generalized Extreme Value distribution (GEV) has distribution function

$$\text{GEV: } F(x; k, \mu, \sigma) = \begin{cases} \exp\left\{-\left(1 - k(x - \mu)/\sigma\right)^{1/k}\right\}, & \text{if } k \neq 0, \\ \exp\left\{-\exp\left\{-(x - \mu)/\sigma\right\}\right\}, & \text{if } k = 0, \end{cases} \quad (7.4)$$

for $k(x - \mu) < \sigma$, $\sigma > 0$, k, μ arbitrary. The case $k = 0$ is interpreted as the limit when $k \rightarrow 0$ for both distributions.

Note that the Gumbel distribution is a GEV distribution with $k = 0$ and that the Weibull distribution is equal to a reversed GEV distribution with $k = 1/c$, $\sigma = a/c$, and $\mu = -a$, i.e. if W has a Weibull distribution with parameters (a, c) then $-W$ has a GEV distribution with $k = 1/c$, $\sigma = a/c$, and $\mu = -a$.

The estimation of parameters in the GPD and GEV distributions is not a simple matter, and no general method exists which has uniformly good properties for all parameter combinations. The `WAVE ANALYSIS TOOLBOX` contains algorithms for plotting of distributions and estimation of parameters with four different methods, suitable in different regions.

7.2.1 Generalized Extreme Value distribution

For the Generalized Extreme Value (GEV) distribution the estimation methods used in the `WAVE ANALYSIS TOOLBOX` are the Maximum Likelihood (ML) method and the method with Probability Weighted Moments (PWM), described in [13, 5]. The programs have been adapted to `MATLAB` from a package of `S-Plus` routines described in [1].

We start with the crest values in `sea.dat` saved in `TC(:,2)`. The command

```
>> [gev cov] = gev_est(TC(:,2)');
```

will give estimates `gev = [k* sigma* mu*]` of the parameters (k, σ, μ) in the GEV distribution (7.4) based on data `TC(:,2)`. The optional output matrix `cov` will contain the estimated covariance matrix of the estimates. The program also gives a plot of the empirical distribution together with the best fitted distribution; see Figure 7.2.¹

The default estimation algorithm for the GEV distribution is the method with Probability Weighted Moments (PWM). An optional second argument, `gev_est(TC(:,2), method)`, allows a choice between the PWM-method (when `method = 1`) and the alternative ML-method (when `method = 3`). The variances of the ML estimates are usually smaller than those of the PWM estimates. However, it is recommended that one first uses the PWM method, since it works for a wider range of parameter values.

¹We have used the crest values only as an illustration of how one can use the GEV distribution, and we do not propose that one should use that distribution without more detailed study. One should, e.g. not use a distribution which allows negative values.

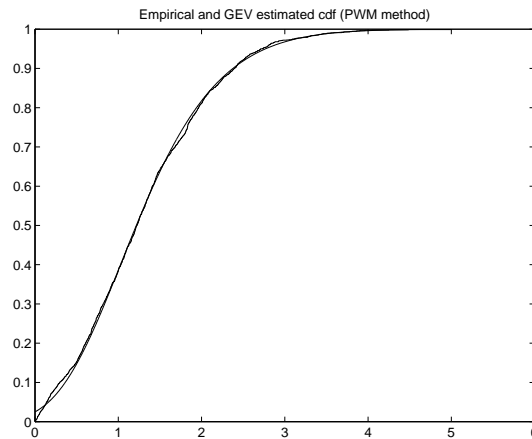


Figure 7.2: *Empirical distribution of crest values with estimated Generalized Extreme Value distribution.*

7.2.2 Generalized Pareto distribution

For the Generalized Pareto distribution (GPD) the `WAVE ANALYSIS TOOLBOX` uses the method with Probability Weighted Moments (PWM), described in [6], and the standard method of Moments (M), as well as a general method suggested by Pickands in [12]. S-Plus routines for these methods are described in [1].

The GPD is often used for exceedances over high levels, and it is well suited as a model for significant wave heights. To fit a GPD to the crest values in `sea.dat` saved in `TC(:,2)`, one uses the command

```
>> [gpd cov] = gpd_est(TC(101:200,2)');
```

This will give estimates `gpd = [k* sigma*]` of the parameters (k, σ) in the Generalized Pareto distribution (7.3) based on data `TC(:,2)`. The optional output matrix `cov` will contain the estimated covariance matrix of the estimates. The program also gives a plot of the empirical distribution together with the best fitted distribution; see Figure 7.3.²

The choice of estimation method is rather dependent on the actual parameter values. The default estimation algorithm in the `WAVE ANALYSIS TOOLBOX` for estimation in the Generalized Pareto distribution is the Pickands' (P) estimator. This estimator gives generally good estimates of the parameter k in (7.3) for $-5 \leq k \leq 5$, and it is also recommended for estimation of σ if $k < -0.5$. The optional second argument, `gev_est(TC(:,2), method)`, gives a choice between Pickands' method (when `method = 4`), the Moment method (when `method = 2`), and the PWM-method (when `method = 1`). It is recommended that one first uses Pickands' method to get an estimate of k , and then if necessary, improves the estimates by means of the PWM or Moment method.

It is possible to simulate independent GEV and GPD observations in the `WAVE ANALYSIS TOOLBOX`. The commands

²Again we want to stress that this example only serves illustrational purposes, and that we do not advocate the use of this particular distribution as a model for crest values.

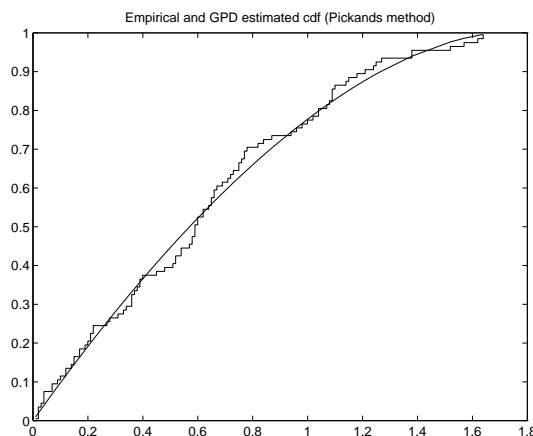


Figure 7.3: Empirical distribution of crest values with estimated Generalized Pareto Distribution.

```
>> gev_samp = rgev(100,0.4,1,2);
>> gx = min(gev_samp):0.01:max(gev_samp);
>> gev_para_pwm = gev_est(gev_samp,1);
>> gy_pwm = pgev(gx,gev_para_pwm(1),gev_para_pwm(2),gev_para_pwm(3));
>> gev_para_ml = gev_est(gev_samp,3); clf
>> gy_pwm = pgev(gx,gev_para_pwm(1),gev_para_pwm(2),gev_para_pwm(3));
>> plot(gx,gy_pwm), hold on, plot(gx,gy_ml,'-')
>> stairs(sort(gev_samp),((1:length(gev_samp))-0.5)/length(gev_samp));
```

and

```
>> gpd_samp = rgpd(100,0.4,1);
>> gx = min(gpd_samp):0.01:max(gpd_samp);
>> gpd_para_pwm = gev_est(gpd_samp,1);
>> gy_pwm = pgev(gx,gev_para_pwm(1),gev_para_pwm(2),gev_para_pwm(3));
>> gpd_para_m = gev_est(gpd_samp,2);
>> gy_m = pgev(gx,gev_para_m(1),gev_para_m(2),gev_para_m(3));
>> gpd_para_pick = gev_est(gpd_samp,4);
>> gy_pick = pgev(gx,gev_para_pick(1),gev_para_pick(2),gev_para_pick(3));
>> plot(gx,gy_pwm), hold on, plot(gx,gy_m,'-'), plot(gx,gy_pick,'-.')
>> stairs(sort(gpd_samp),((1:length(gpd_samp))-0.5)/length(gpd_samp));
```

simulates and estimates 100 observations of GEV and GPD variables with shape parameter $k = 0.5$, $\mu = 1$, and $\sigma = 2$. The empirical distributions are shown in Figure 7.4(a) and (b). Estimates by means of the alternative methods are also shown in the diagrams.

The WAVE ANALYSIS TOOLBOX also contains a program `genrand` which can simulate variables from a broad variety of distributions; see the help text to `genrand`.

7.3 POT-analysis

Peaks Over Threshold analysis (POT) is a systematic way to analyse the distribution of the exceedances over high levels in order to estimate extreme quantiles outside the range

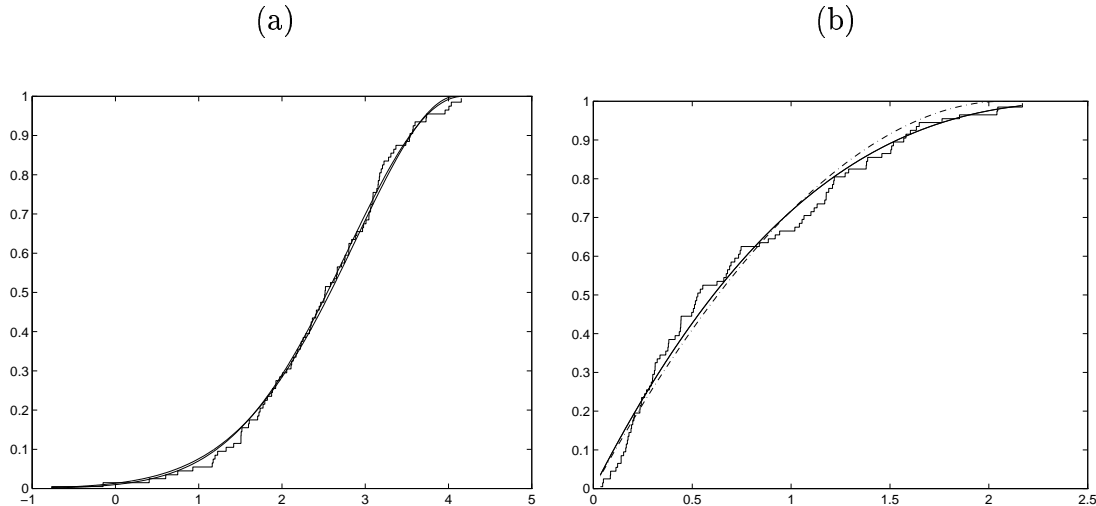


Figure 7.4: Empirical distributions and estimated distribution functions for 100 observations of GEV and GPD variables.

of observed values. The method is based on the observation that the extreme tail of a distribution often has a rather simple and standardized form regardless of the shape of the more central parts of the distribution. One then fits such a simple distribution only to those observations which exceed some suitable level, with the hope that this fitted distribution gives an accurate fit to the real distribution also in the more extreme parts. The level should be chosen high enough for the tail to have approximately the standardized form, but not so high that there remains too few observations above it. After fitting a tail distribution one estimates the distribution of the (random) number of exceedances over the level, and then combines the tail distribution of the individual exceedances with the distribution for the number of exceedances to find the total tail distribution.

The POT method is related to the technique to look at the α -significant values in a wave series, i.e. to study the average value of those values which exceed the α -quantile; see Section 2.4.

The simplest distribution to fit to the exceedances over a level u is the Generalized Pareto distribution, GPD, with distribution function

$$F(x; k, \sigma) = \begin{cases} 1 - (1 - kx/\sigma)^{1/k}, & \text{if } k \neq 0, \\ 1 - \exp\{-x/\sigma\}, & \text{if } k = 0. \end{cases}$$

Note that if a random variable X follows a Generalized Pareto distribution $F(x; k, \sigma)$ then the exceedances over a level u is also GPD with distribution function $F(x; k, \sigma - ku)$ with the same k -parameter and with scale parameter $\sigma - ku$,

$$P(X > u + y \mid X > u) = \frac{\left(1 - k \frac{u+y}{\sigma}\right)^{1/k}}{\left(1 - k \frac{u}{\sigma}\right)^{1/k}} = \left(1 - k \frac{y}{\sigma - ku}\right)^{1/k}.$$

Another important property of the Generalized Pareto Distribution is that if $k > -1$, then the mean exceedance over a level u is a linear function of u :

$$E(X - u | X > u) = \frac{\sigma - ku}{1 + k}.$$

In wave analysis this means that if the α -significant crest heights $C_s = E(X|X>C_q)$ are a linear function of the α -quantiles C_q , then one could try to fit a Generalized Pareto distribution. The slope of the linear function should be equal to $1/(1+k)$,

$$C_s = \frac{C_q + \sigma}{1 + k}$$

The following commands illustrate this,

```
>> Cs = vc2sign(TC(:,2));
>> Cq = Cs(:,3); Cs = Cs(:,2);
>> [gpd cov] = gpd_est(TC(:,2))
>> k = gpd(1); s = gpd(2);
>> plot(Cq, Cs); hold on
>> plot(Cq, (Cq+s)/(1+k));
```

As seen in Figure 7.5 the crest values exhibit an almost linear relationship between significant values and the quantiles. The straight line constructed by means of the estimated parameter values does not give a perfect fit in the right part of the diagram. The fit is better in the left part, where there are more observations available, but it is by no means perfect.

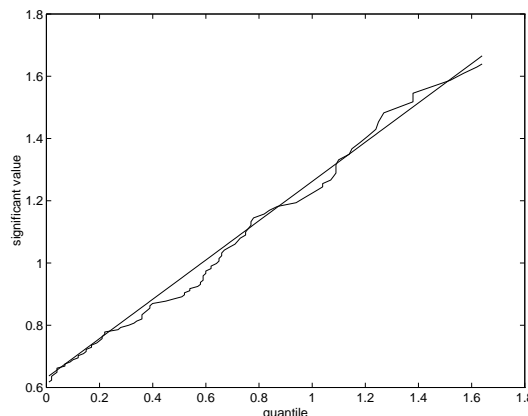


Figure 7.5: *Linear relation between α -significant crest values and α -quantiles. Straight line from fitted Generalized Pareto distribution.*

If one is successful in fitting a Generalized Pareto distribution to the tail of a set of data, one would like to use the GPD to predict how extreme values might occur over a certain period of time. One could e.g. want to predict the most extreme wave height that will appear in a storm during a year. If the distribution of the individual wave height exceedances in a storm is GPD one can easily find e.g. the distribution of the largest value in a fixed number of storms. However, the number of storms in a year is not fixed but random, and then one has to combine the distribution of the random maximum in individual storms with the random number of storms in a year before one can say anything about the total maximum in a year.

There is a nice relationship between the Generalized Pareto distribution and the Generalized Extreme Value distribution in this respect: *the maximum of a Poisson distributed*

number of independent GPD variables has a GEV distribution. This follows by simple summation of probabilities: if ν is a Poisson distributed random variable with mean μ , and $M_\nu = \max(X_1, X_2, \dots, X_\nu)$ is the maximum of ν independent GPD variables then,

$$\begin{aligned} P(M_\nu \leq x) &= \sum_{n=0}^{\infty} P(\nu = n) \cdot P(X_1 \leq x, X_2 \leq x, \dots, X_n \leq x) \\ &= \sum_{n=0}^{\infty} e^{-\mu} \frac{\mu^n}{n!} \cdot \left(1 - \left(1 - k \frac{x}{\sigma}\right)^{1/k}\right)^n \\ &= \exp \left\{ -\left(1 - k(x - a)/b\right)^{1/k} \right\}, \end{aligned}$$

which is the Generalized Extreme Value distribution with $b = \sigma/\mu^k$ and $a = \sigma(1 - \mu^{-k})/k$.

In practice, one does not always find a Poisson distribution for the number of exceedances. Since extreme values sometimes have a tendency to cluster, some declustering algorithm could be applied to identify the largest value in each of the clusters, and then use a Poisson distribution for the number of clusters.

7.4 Summary of extreme value procedures

| | |
|-----------------------|--|
| <code>gev_est</code> | <code>[theta cov] = gev_est(data,method)</code> |
| | Estimates parameters <code>theta = [shape scale location]</code> and their covariance matrix <code>cov</code> in a Generalized Extreme Value distribution from the data set <code>data</code> . When <code>method = 1,3</code> we get the Probability Weighted Moments and Maximum Likelihood estimator, respectively. |
| <code>gpd_est</code> | <code>[theta cov] = gpd_est(data,method)</code> |
| | Estimates parameters <code>theta = [shape scale]</code> and their covariance matrix <code>cov</code> in a Generalized Pareto Distribution from the data set <code>data</code> . When <code>method = 1,2,4</code> we get the Probability Weighted Moments, Moment, and Pickands' estimator, respectively. |
| <code>gumbplot</code> | <code>theta = gumbplot(data)</code> |
| | Estimates parameters <code>theta = [scale location] = [a b]</code> in a Gumbel distribution from the data set <code>data</code> . |
| <code>random</code> | <code>r = random(n, theta, type)</code> |
| | Simulates <code>n</code> values from the distribution <code>type</code> with parameters <code>theta</code> . |
| <code>rgev</code> | <code>sample = rgev(n, shape, scale, location)</code> |
| | Simulates <code>n</code> values from a Generalized Extreme Value distribution. |
| <code>rgpd</code> | <code>sample = rgpd(n, shape, scale)</code> |
| | Simulates <code>n</code> values from a Generalized Pareto Distribution. |
| <code>weibplot</code> | <code>theta = weibplot(data)</code> |
| | Estimates parameters <code>theta = [scale shape] = [a c]</code> in a Weibull distribution from the data set <code>data</code> . |

APPENDIX A

SOME SIMULATION PROGRAMS FOR RANDOM PROCESSES

In this tutorial we have used two functions to generate random processes; `simspec` gives a sample of the transformed Gaussian process; `mctp2tp` gives a sample of a time reversible Markov chain of turning points. However, in order to test our programs we have developed some additional means to generate random loads which we attach to the toolbox for convenience of the user. We do not systematically cover the methods to simulate random processes, neither are the programs optimized. Here follows a short list of functions and we refer to their help texts for description of how to use them.

- `irregsim` - simulates a random sequence given irregularity factor and crossing spectrum, the algorithm proposed in [4],
- `mc2dat` - simulates a discrete Markov chain, $X_i, i = 0, \dots, N$,
- `mctp2tp` - simulates a discrete Markov chain of turning points, $m_0, M_0, m_1, M_1, \dots, m_N$,
- `simduff` - simulates the solution to different oscillators, e.g. harmonic or nonlinear Duffing oscillator, driven by white noise, Gaussian or α -stable having heavy tails,
- `simo_u` - simulates a continuous Markov chain (discretized Gaussian Ornstein-Uhlenbeck process),
- `simosc` - simulates a two-dimensional continuous Markov chain (discretized $Y(t), Y'(t)$ where Y is the solution to a harmonic oscillator equation driven by Gaussian noise.)
- `simspec` - simulates a Gaussian or transformed Gaussian processes with specified power spectrum.

REFERENCES

- [1] Borg, S. (1992): XS - a statistical program package in Splus for extreme-value analysis. Dept. of Mathematical Statistics, Lund University, 1992:E2.
- [2] Matsuishi M. & Endo T.: *Fatigue of metals subject to varying stress*. Paper presented to Japan Soc. Mech. Engrs, Jukvoka, Japan (1968).
- [3] Falk, M., Hüsler, J. and Reiss, R.-D. (1994): *Laws of Small Numbers: Extremes and rare events*. Birkhäuser, Basel.
- [4] Holm S. and de Maré J. (1985): Generation of random processes for fatigue testing. *Stoch. Proc. Appl.* **20**, 149–156.
- [5] Hosking, J.R.M., Wallis, J.R. and Wood, E.F. (1985): Estimation of the generalized extreme-value distribution by the method of probability-weighted moments. *Technometrics*, **27**, pp. 251-261.
- [6] Hosking, J.R.M. and Wallis, J.R. (1987): Parameter and quantile estimation for the generalized Pareto distribution. *Technometrics*, **29**, pp. 339-349.
- [7] Leadbetter, M. R., Lindgren, G. and Rootzén, H. (1983): *Extremes and related properties of random sequences and processes*. Springer Verlag, New York.
- [8] Lindgren, G. (1972): Some properties of a normal process near a local maximum. *Ann. Math. Statist.* **41**, pp. 1870-1883.
- [9] Lindgren, G. and Rychlik, I. (1982): Wave characteristic distributions for Gaussian waves – wave-length, amplitude and steepness. *Ocean Engineering*, **9**, pp. 411-432.
- [10] Lindgren, G. and Rychlik, I. (1991): Slepian models and regression approximations in crossings and extreme value theory. *International Statistical Review*, **59**, pp. 195-225.
- [11] Ochi, M.K. and Ahn, K. (1994): Probability distribution applicable to non-Gaussian random processes. *Probabilistic Engineering Mechanics*, **9**, 255-264.
- [12] Pickands III, J. (1975): Statistical inference using extreme order statistics. *Annals of Statistics*, **3**, pp. 119-131.
- [13] Prescott, P. and Walden, A.T. (1980): Maximum likelihood estimation of the parameters of the generalized extreme-value distribution. *Biometrika*, **67**, pp. 723-724.
- [14] Rice, S.O. (1944-45): Mathematical analysis of random noise. *Bell Syst. Techn. J.* **23-24**, pp. 282-332 and 46-156.
- [15] Rychlik, I. (1987): Regression approximations of wavelength and amplitude distributions. *Adv. Appl. Probab.* **19**, pp. 396-430.
- [16] Rychlik I.: A new definition of the rainflow cycle counting method. *Int. J. Fatigue*, **9** (1987) 119-121.

- [17] Rychlik I. (1993): Note on cycle counts in irregular loads. *Fatigue Fract. Eng. Mater. and Struc.*, **16**, pp. 377-390.
- [18] Rychlik I. (1995): A note on significant wave height. To appear in *Ocean Engineering*.
- [19] Rychlik, I. (1995): Simulation of load sequences from Rainflow matrices: Markov method. Stat. Research Report, Dept. of Mathematical Statistics, Lund, 1995:29, 1-23.
- [20] Rychlik, I. and Lindgren, G. (1993): CROSSREG - A computer package for extreme value and wave analysis, with reliability applications. *Probability in the Engineering and Informational Sciences*, **7**, pp. 125-148.
- [21] Rychlik, I., Johannesson P., and Leadbetter, M.R. (1995): *Modelling and statistical analysis of ocean wave data using transformed Gaussian processes*. Stat. Research Report, Stat. Res. Report, Dept. of Math. Statist. Lund University, 1995:13, pp. 1-43.

INDEX

cc2fr, 15
ccplot, 9
cl2fr, 21
cocc, 16
cowa, 38
cross2tr, 45
dat2cov, 48
dat2hwa, 9
dat2spec, 46
dat2tp, 8
dc2cc, 22
empdistr, 32
filt, 16
fr2amp, 17
fr2nt, 16
fr2pmax, 18
gaus2dat, 48
genrand, 65
gev_est, 63
gpd_est, 64
gumbplot, 62
help wave, 5
hs2sign, 20
irregsim, 71
jonswap, 26
levels, 15
mc2dat, 71
mctp2rfc, 55
mctp2tc, 31, 55
minmax, 28
mkdisc, 21
mm2cross, 10
normplot, 50
normspec, 30
nt2fr, 16
ochi, 26
param, 15
pickdiag, 20
rfc2mctp, 55
rgev, 65
rgpd, 65
s2c, 48
sea.dat, 6
simduff, 71
simo_u, 71
simosc, 71
simspec, 27
test, 46
tp2mm, 9
tp2rfc, 9
tp2tc, 9
tp2wa, 9
vc2sign, 12
wattutor, 6
wave_th, 38
wave_t, 35
wavepath, 5
weibplot, 62

counting distribution, 4
crest front amplitude, 2
crossing spectrum, 4, 10
cycle intensity, 29

extreme value distribution, 61

Generalized Extreme Value distribution, GEV,
20, 63
Generalized Pareto distribution, GPD, 20,
62
Gumbel distribution, 61

half wavelength, 2
histogram matrix, 15

irregularity factor, 9

JONSWAP spectrum, 26

Markov chain of turning points, 31
Markov matrix, 55
mean separated min2Max cycle, 2
min2Max cycle, 2
NIT, 34
oscillation count, 4
parameter vector, 15
rainflow cycle, 3
rainflow filtering, 8
rainflow matrix, 54
significant wave characteristics, 11, 20
trough2crest cycle, 2
turning point, 2
Weibull distribution, 61