WAFO

A MATLAB TOOLBOX FOR ANALYSIS
OF RANDOM WAVES AND LOADS

THE WAFO GROUP

Tutorial
August 2000, Version 2.0.02

LUND UNIVERSITY

Lund Institute of Technology
Centre for Mathematical Sciences
Mathematical Statistics

CENTRUM SCIENTIARUM MATHEMATICARUM

WAFO

a Matlab Toolbox for Analysis of Random Waves and Loads

–

Tutorial Version 2.0.02

The WAFO group

LUND INSTITUTE OF TECHNOLOGY
Lund University

Centre for Mathematical Sciences
Mathematical Statistics

Box 118, SE–221 00 Lund, SWEDEN

TABLE OF CONTENTS

FOREWORD

This is a preliminary manual and tutorial for the toolbox WAFO 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 as well as results avilable in the literature. References are given to the source of the algorithms whenever it is possible. These reference are given in the MATLABcode for all the routines and they are also listed in the last section of this tutorial. If the reference is not used explicitly in the tutorial it means that it is referred to in one of the MATLAB m-files.

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 and other load sequences. There is sectiona for fatigue analysis and for general extreme value analysis. The present toolbox represents a considerable development of two earlier toolboxes, the FAT and WAT toolboxes, for fatigue and wave analysis, respectively. These toolboxes were both Version 1; therefore WAFO has been named Version 2.

The persons that take actively part in creating this tutorial are:

- *Per Andreas Brodtkorb* [1], *Pär Johannesson* [2], *Georg Lindgren* [2], *Igor Rychlik* [2], *Jesper Rydén* [2], *Eva Sjö* [2], and *Martin Sköld* [2].

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, Professor Krzysztof Podgórski, Dept. of Mathematics IUPUI, USA. We would also like to particularly thank Michel Olagnon and Marc Provosto, at Institut Français de Recherches pour l'Exploitation de la Mer (IFREMER), Brest, who have contributed with many enlightening and fruitful discussions.

Other persons have put a great deal of effort into WAFO and its predecessors FAT and WAT are Mats Frendahl, Finn Lindgren, and Ulla Machado, all at Mathematical Statistics at Lund University, and Sylvie van Iseghem, IFREMER.

[1] Department of Marine Hydrodynamics, NTNU, Trondheim, Norway
[2] Centre for Mathematical Sciences, Lund University, Sweden.

TECHNICAL INFORMATION

- In this tutorial, the word `WAFO` when used in path specifications means the full name of the `WAFO` main catalogue, for instance `C:\MATLAB\toolbox\contrib\wafo`.
- For help on the toolbox, write `help wafo`.
- The MATLAB code used in this tutorial can be found in the WAFO catalogue `WAFO/docs`.
- The WAFO homepage `http://www.maths.lth.se/matstat/wafo/` contains information on how to download WAFO, updates, known errors, etc. It also contains links to exercises and articles using WAFO.

  Comments and suggestions are solicited - send to `wafo@maths.lth.se` .
- The first two digits in the version number of this tutorial follows that of the toolbox, thus Tutorial Version 2.0.xx goes with Toolbox Version 2.0.yy.
- Tutorial Version 2.0.02: This version of the Tutorial was printed and put on the Web August 9, 2000. It contains only minor changes compared to Version 2.0.01.
- The examples in the tutorial have been successfully tested on Toolbox Version 2.0.3 .

CHAPTER 1
INTRODUCTION

1.1 What is WAFO?

WAFO (Wave Analysis for Fatigue and Oceanography) is a toolbox of Matlab routines for statistical analysis and simulation of *random waves* and *random loads*. Using WAFO you can, for example, calculate theoretical distributions of wave characteristics from observed or theoretical power spectra of the sea or find the theoretical density of rainflow cycles from parameters of random loads. These are just two examples of variety of problems you can analyze using this toolbox.

There are three major audiences to which this toolbox can have a great of appeal. First, *ocean engineers* will find a very comprehensive set of computational tools for statistical analysis of random waves and ship's responses to them. Second, the toolbox contains a number of procedures of the prime importance for *mechanical engineers* working in the areas of *random loads* as well as *damage and fatigue analysis*. Finally, any *researcher* who is interested in *statistical analysis of random processes* will find here an extensive and up-to-date set of computational and graphical tools for her/his studies.

In a random wave model, like that for Gaussian or transformed Gaussian waves, the distribution of wave characteristics such as wave period and crest-trough wave height can be calculated by high accuracy for almost any spectral type. WAFO is a third-generation package of Matlab routines for handling statistical modelling, calculation and analysis of random waves and wave characteristics and their statistical distributions. The package also contains routines for cycle counting and computation in random load models, in particular the rainflow counting often used in fatigue life prediction.

Random wave distributions are notoriously difficult to obtain in explicit form from a random wave model, but numerical algorithms, based on the so-called regression approximation, work well. This method to calculate wave distributions is the only known method that gives correct answers valid for general spectra. The theoretical background is reviewed in [32] and computational aspects and algorithms in [52].

The algorithms are based on a specification of the random waves by means of their (unidirectional or directional) spectrum, and on the underlying assumption of linear wave theory and Gaussian distribution. However, a transformation of sea elevation data can be made to obtain a desired (horizontal) asymmetric marginal distribution.

A first complete toolbox FAT (Fatigue Analysis Toolbox) was presented in 1993, [14]. It was followed by WAT (Wave Analysis Toolbox) in 1995, [53] [1], being extended with routines for

[1] http://www.maths.lth.se/matstat/staff/georg/watinfo.html

probabilistic-modelling problems in oceanography. In WAFO, many new numerical routines have been introduced, and a considerable improvement in computational speed and accuracy has been achieved. WAFO allows treatment of more complicated problems; for example, spatial waves with time dynamics can be handled, thus extending the analysis to random fields. Algorithms for rainflow analysis of switching Markov chains are included, as well as for decomposition of the rainflow matrix. Many of the new tools are the result of recent research, e.g. [54], [42], [41], [22], and [7].

Further, WAFO has a modular structure, so users can easily add their own algorithms for special purposes. The modules of the toolbox handle

- wave/load data analysis and estimation,
- spectral distributions,
- transformation to Gaussian marginals (exact distributions),
- simple parametric approximations to wave characteristic distributions,
- simulation of Gaussian and Markovian wave/load time series,
- extreme value and other statistical analysis,
- cycle counting,
- rainflow cycle analysis and calculation,
- fatigue life calculation,
- smoothing and visualization.

In the following section, we discuss in more detail the idea of the modular structure. That section is followed by an overview of the organization of WAFO, presenting some of the capabilities of the toolbox. Finally, we give a number of examples to demonstrate the use of some of the tools in WAFO for analysis and modelling.

1.2 Philosophy – some features of WAFO

A common problem with research involving complex scientific (numerical) computations is that when researchers try to advance and leverage their colleagues work, they often spend a considerable amount of time just reproducing it.

Often after few months since the completion of their own work, authors are not capable of reproducing it without a great deal of agony, due to various circumstances such as the loss of the original input data or/and parameter values etc. Thus many scientific articles are reproducible in principle, but not in practice.

To deal with this and to organize computational scientific research and hence to conveniently transfer our technology, we impose a simple filing discipline on the authors contributing to the WAFO-toolbox. (A positive side effect of this discipline is a reduced amount of errors which are prone to occur in computational science.)

This philosophy is adopted from the article by Matthias Schwab et al “Making scientific computations reproducible” (`http://sepwww.stanford.edu/research/redoc/`).

The idea is to develop reproducible knowledge about the results of the computational experiments (research) done at Lund University and to make it available to other researchers for their inspection, modification, re-use and criticism.

As a consequence, WAFO is freely available through the Internet[2]. Other researchers can obtain here the Matlab code which generated figures in articles and reproduce them. They can if they wish modify the calculations by editing the underlying code, input arguments and parameter values. They can use the algorithms on other data sets or they can try their own methods on the same data sets and compare the methods in a fast and easy fashion.

This is the reason of existence for the `WAFO/paper` directory which contains subdirectories including scripts for recreating figures in published articles and technical reports. Each article has its own subdirectory. The directories contain demonstration scripts to generate individual figures and (possibly) specialized tools/functions not available in the official release of WAFO for generating these figures.

Just like the `WAFO/paper` directory the `WAFO/demos` directory also contains different subdirectories with scripts producing figures. The only difference is that it does not reproduce figures from published articles but merely test and demonstrate various methodologies, highlight some features of WAFO, and release code that approximately reproduces figures in other articles. The important thing for both directories is not the printed figures, but the underlying algorithm and code. In addition, the `paper` and `demos` scripts constitute an excellent starting point for the novel user to learn about WAFO.

The documentation directory `WAFO/docs` contains all the documentation available for the toolbox. The contents of any of these files may be examined by typing its name for ascii files or viewing in ghostview for postscript files. Also each function is well documented containing a help header describing how the function works with a detailed list of input and output arguments with examples of how to use the function.

The Matlab code to each function file also contains references to related functions and a complete reference to published articles from which the user can obtain further information if such exist.

One important enhancement of the new toolbox is the use of *structure arrays* in Matlab by which several types of data can be stored as one object. This significantly simplifies the passing of input and output arguments of functions and also makes the Matlab workspace much tidier when working with the new toolbox compared to the old one. Three structures or object classes are implemented and extensively used: the spectrum structure, covariance structure and probability density function (hereafter denoted pdf) structure. Thus the toolbox requires Matlab Version 5 or newer and is portable to any computational environment that supports Matlab, such as Unix or PC with MS Windows. See Section 1.5 for a description of the datastructures in WAFO.

All the files in the package are located in subdirectories under the main directory. The following directories are related to what has been discussed above. In the next section, we describe in more details the directories (or modules) which contain routines for application.

WAFO is the main directory containing different directories for the WAFO software, datasets and documentation.

WAFO/docs contains the documentation for the toolbox both in ascii and postscript format.

WAFO/paper is a subdirectory including scripts for reproducing figures in various articles and technical reports.

WAFO/demos contains different demonstrations that illustrates and highlights certain aspects of WAFO.

[2] http://www.maths.lth.se/matstat/wafo/

WAFO/data contains datasets used in the demo and paper scripts.

WAFO/source contains mex and Fortran source files.

WAFO/exec/... contains Fortran compiled executables for different computers and platforms.

1.3 Organization of WAFO

In this section, we make a brief presentation of each module. The text will not be a complete list of routines; such a list may be found at the web site for WAFO. We want to emphasize that all routines in WAFO work together – the division into sub-toolboxes is only to make it easier for the user to find the routines for his actual problem.

Data analysis

The routines in this category treat data in the form of time series. As examples of routines, we find procedures for extraction of so-called turning points, from which troughs and crests may be obtained, as well as procedures for estimation of autocovariance function and one-sided spectral density. One routine extracts wave heights and steepnesses. Numerous plotting routines are included.

Spectrum

Computation of spectral moments and covariance functions, given a spectrum, is a necessary step for calculation of exact probability distributions of wave characteristics. The spectrum structure mentioned in the previous section allows this calculation to be performed for directional spectra as well as encountered spectra. We present routines for calculations of commonly used frequency spectra $S(\omega)$, e.g. JONSWAP, Torsethaugen. The spectra can be expressed in frequency as well as wave number. Libraries of spreading functions $D(\theta)$, in some cases allowed to be also frequency dependent, cf. [23], are included.

Transformed Gaussian processes

WAFO is mainly intended to model linear, Gaussian waves. For this cathegory of waves, the exact distributions of wave characteristics can be calculated, given a spectrum; for example

- pdf for wavelength (period),
- joint pdf for wavelength (period) and amplitude,
- joint pdf of half wavelengths.

Routines for transformed Gaussian processes, cf. [54], are included.

Wave models

In WAFO, we have implemented certain models for distributions of wave characteristics found in the literature. For example, one finds

- approximations of the density (T_c, A_c) in a stationary Gaussian transformed process proposed by Cavanié, et al. [10], and Longuet-Higgins, [34],
- a model for the cdf/pdf of breaking limited wave heights proposed by Tayfun, [57],

- a model for the cdf/pdf of large wave heights by Tayfun, [58].

These are parametric models, where the calculation needs as input spectral moments, as opposed to the algorithms in the previous module, where the whole spectrum is required.

Simulation of random processes

Efficient simulation of a Gaussian process $X(t)$ and its derivative $X'(t)$, given the spectral density or the auto-correlation function, can be performed. A routine for simulation of a transformed Gaussian process (and its derivative) is also included. For fast and exact simulation, some routines use a technique with circulant embedding of the covariance matrix, [11]. More traditional spectral simulation methods (FFT) are also used. Simulation of discrete Markov chains, Markov chains of turning points, switching Markov chains etc. is possible.

Statistical tools and extreme value distributions

Certain probability distributions are extensively used in ocean engineering, e.g. Rayleigh, Gumbel, Weibull. The generalized extreme-value distributions (GEV) and generalized Pareto distributions (GPD) are also important. For the distributions mentioned, it is possible to estimate parameters, generate random variables, evalute pdf and cumulative distribution function, and plot in various probability papers. One cathegory of routines handles bivariate distributions. Besides having routines for estimation of parameters etc. for the two-dimensional Weibull distribution, bivariate modelling is possible.

Kernel-density-estimation tools

The routines in this cathegory complement the ones found in 'Data analysis' and, obviously, the routines in 'Statistical tools and extreme value distributions. They are, however, also applicable to multi-dimensional data, and hence very useful for smoothing purposes when comparing (theoretical) joint distributions of wave characteristics to data; cf. cf. [55], [61].

Discretization and cycle counting

After extraction of the so-called sequence of turning points (the sequence of local maxima and minima) from data, cycle counts can be obtained, e.g. max-to-min cycles, trough-to-crest cycles, rainflow cycles. For decriptive statistics, the counting distribution and the rainflow matrix are important; these can be obtained. Given a cycle matrix, one can obtain histograms for amplitude and range, respectively.

Markov models

If the sequence of turning points forms a Markov chain (MC), it is called a MC of turning points (MCTP). An important property is the Markov matrix: the expected histogram matrix of min2max and max-to-min cycles. Given a rainflow matrix of a MCTP, one can find its Markov matrix. In WAFO, algorithms are implemented to calculate the rainflow matrix for a MC and a MCTP; cf. [13].

In some applications, one wants to model data, whose properties change according to an underlying, often unobserved process, called the regime process. The state of the regime process controls which parameters to use and when to switch the parameter values. The regime process can be modelled by a Markov chain, and this is the fundamental basis for the set of routines presented. For an application with switching Markov models for fatigue problems; see [20, 22].

Fatigue and Damage

In WAFO, routines for calculation of the accumulated damage according to the Palmgren-Miner rule have been implemented. It is possible to compute the total damage from a cycle count as well as from a cycle matrix.

Miscellaneous routines

We find here various plot routines, algorithms for numerical integration, and functions for documentation of WAFO with modules.

1.4 Some applications of WAFO

In this section we demonstrate in examples some of the capabilities of WAFO. For further examples and knowledge about the algorithms used in the routines, we refer to the tutorial and the documentation in the routines. Note that the necessary Matlab code for generation of the figures in this paper is found in the directory `WAFO/paper`.

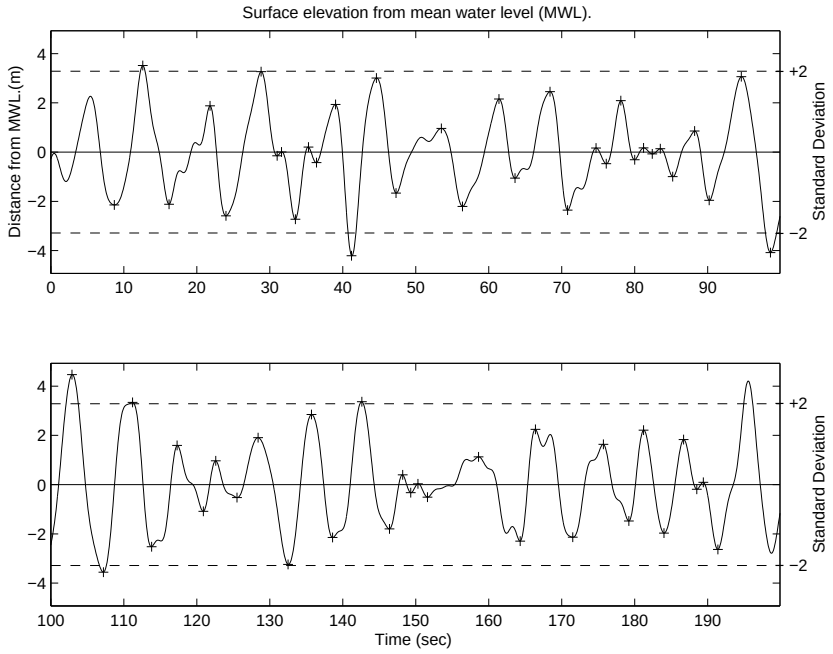


Figure 1.1: *Part of a simulation from $S(\omega)$, a Torsethaugen spectrum with $H_{m_0} = 6$ m, $T_p = 8$ s. Total number of points $= 2000$, $\Delta t = 0.1$ s.*

We start by defining a frequency spectrum, $S(\omega)$, which will be used in many of the examples; we choose a Torsethaugen spectrum with the parameters $H_{m_0} = 6$ m, $T_p = 8$ s, describing significant wave height and primary peak period, respectively. The energy is divided between two peaks, corresponding to contributions from wind and swell ([59]). WAFO allows spectra to be defined simply by their parameters H_{m_0} and T_p.

1.4.1 Simulation from spectrum, estimation of spectrum

In Figure 1.1, plotted using `waveplot`, we have simulated a sample path from $S(\omega)$. The user specifies the number of wanted points in the simulation. The following code in MATLAB generates the 200 seconds of data sampled with 10 Hz from the discussed spectrum. More on simulation can be found in Section 2.3.

```
S1=torsethaugen([],[6 8],1);
xs=spec2sdat(S1,[2000 1],0.1);
waveplot(xs,'-')
```

In a common situation data is given in form of a time series, for which one wants to estimate the related spectrum. We will now simulate 20 minutes signal sampled with 4 Hz, find an estimate $S_{\text{est}}(\omega)$ and compare the result to the original Torsethaugen spectrum $S(\omega)$. The following code in MATLAB was used to generate Figure 1.2, where the two spectra are displayed. The maximum lag size of the Parzen window function used (here 400) can be chosen by the user or automatically by WAFO.

```
xs=spec2sdat(S1,[20*60*4 1],0.25);
Sest = dat2spec2(xs,400)
wspecplot(Sest,1,'--'), hold on
wspecplot(S1,1), hold off
axis([0 3 0 5])
```

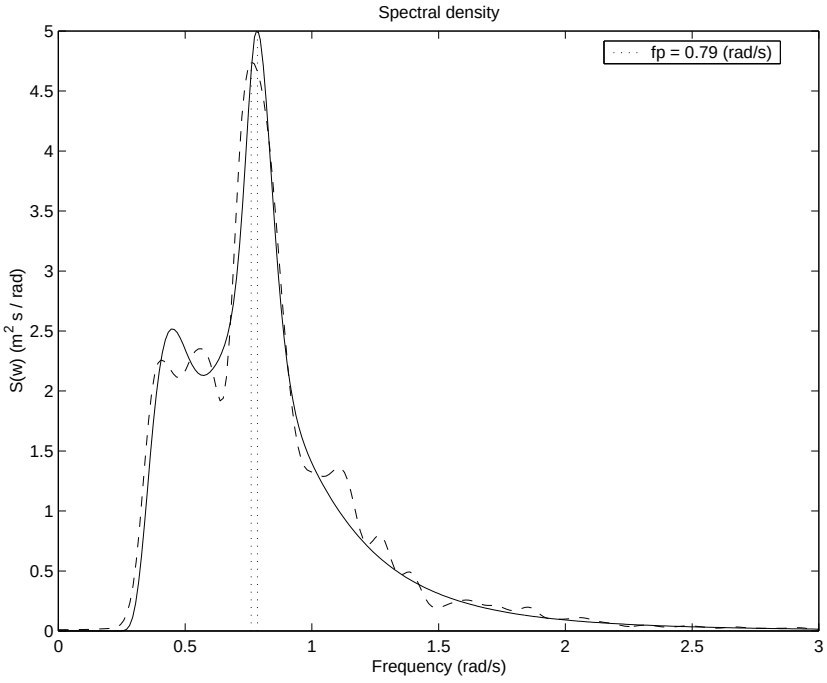


Figure 1.2: *Solid: original spectrum. Dashed: spectrum estimated from data (20 minutes of observations). Maximum lag size of the Parzen window* $= 400$.

1.4.2 Probability distributions of wave characteristics

WAFO gives the possibility to compute exact probability distributions for a number of wave characteristics, given a spectral density. A wave characteristic as, for example, wave period, can be

defined in several ways, see Table 3.1 in Chapter 3, and WAFO allows the user to choose between a number of definitions: trough-to-crest, down-to-up crossing, up-to-up crossing etc. Chapter 3 describes how to use WAFO to compute all wave characteristic distributions.

In the numerical example, we have chosen to consider the down-to-up crossing definition, rather a trough period. The wave periods can be extracted from the realization in Figure 1.1, and are shown as a histogram in Figure 1.3. This histogram may be compared to the theoretical density, calculated from $S(\omega)$, and the estimated density $S_{\text{est}}(\omega)$; see Figure 1.3. Recall that for this spectrum, $T_p = 8$ s. In Figure 1.3, the density for the half period is shown; the results are in good agreement with the original spectrum. The following are the code lines to produce the presented figure. The different steps are: first extract half periods from the data by means of the routine `dat2wa` and store in the variable `T`, then use `spec2tpdf` to calculate the theoretical distribution.

```
[T, index] = dat2wa(xs,0,'d2u');
whisto(T,25,1,1), hold on
dtyex = spec2tpdf(S1,[],[],[0 10 51],0,3);
dtyest = spec2tpdf(Sest,[],[],[0 10 51],0,3);
pdfplot(dtyex)
pdfplot(dtyest,'-.')
axis([0 10 0 0.35]), hold off
```

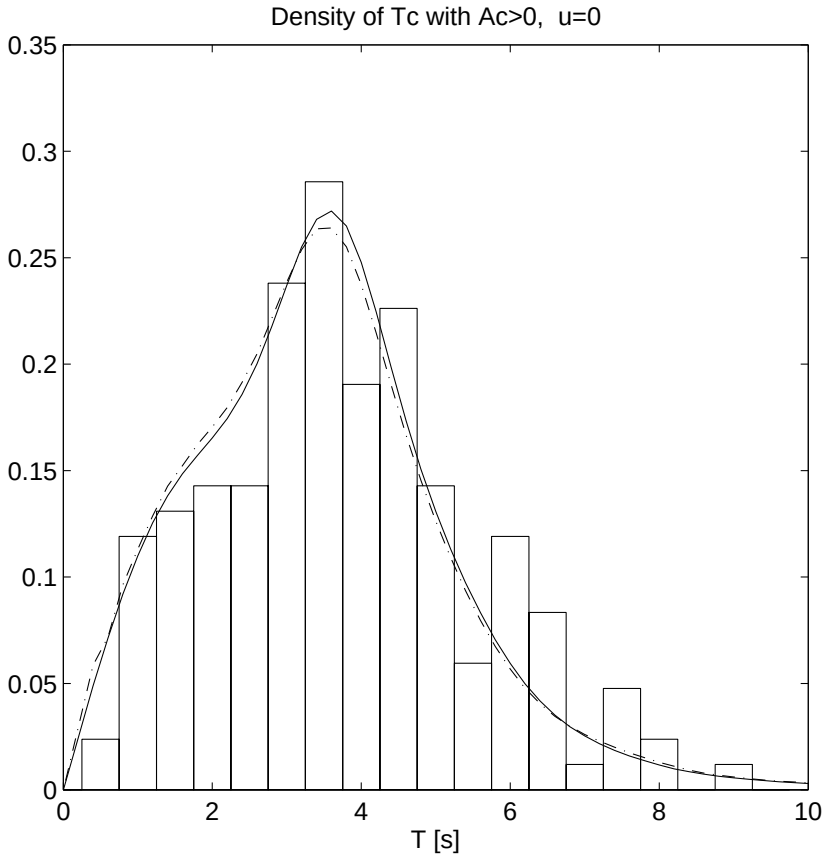


Figure 1.3: *Solid line: pdf for wave period, given $S(\omega)$. Dash-dotted line: pdf for wave period, given $S_{est}(\omega)$. The histogram shows the wave periods extracted from simulated data.*

1.4.3 Directional spectra

In WAFO one finds means for evaluation and visualization of directional spectra, that is

$$S(\omega,\theta) = S(\omega)D(\theta,\omega)$$

where $S(\omega)$ is a frequency spectrum and $D(\theta,\omega)$ is a spreading function. A number of common spreading functions can be chosen by the user.

One way of visualizing $S(\omega,\theta)$ is a polar plot. In Figure 1.4 we show the resulting directional spectrum (solid line) for the Torsethaugen spectrum used above. The spreading function is of the *cos-2s* type, that is (in the frequency dependent case)

$$D(\theta) = \frac{\Gamma(s+1)}{2\sqrt{\pi}\Gamma(s+1/2)}\cos^{2s}\left(\frac{\theta}{2}\right)$$

with s=15. Note that the two peaks can be distinguished. The dash-dotted line is the corresponding result when the spreading function is frequency dependent, cf. [23].

Here are a few lines of code which produce the graph of these directional spectra.

```
D1 = spreading(101,'cos',pi/2,[15],[],0);
D12 = spreading(101,'cos',0,[15],S1.w,1);
SD1 = mkdspec(S1,D1);
SD12 = mkdspec(S1,D12);
wspecplot(SD1,1), hold on, wspecplot(SD12,1); hold off
```

We finish this section with simulated sea surfaces on 128 [m] by 128 [m] for the sea with directional spectra `SD1` and `SD2`. The routine `seasim` simulates a sea surface with directional spectrum.

```
Y1=seasim(SD1,2^8,2^8,1,0.5,0.5,0.25,2,1);
Y12=seasim(SD12,2^8,2^8,1,0.5,0.5,0.25,2,1);
```

The results are shown in Figure 1.5 and one can sea that waves are coming from different direction. However, the frequency dependent spreading leads to a much more irregular surface so the orientation of waves is less transparent. From Figure 1.5 it is not easy to deduce that both sea surfaces have the same period distribution, but it is more obvious that the wavelength distributions are different.

1.4.4 Fatigue, Load cycles, and Markov models

In fatigue applications the exact sample path is not important, but only the tops and bottoms of the load, called the sequence of turning points (TP). From the turning points one can extract load cycles, from which damage calculations and fatigue life predictions can be performed. In WAFO there are numerous routines for evaluating fatigue measured loads, as well as making theoretical calculations of distributions that are important for fatigue evaluation. A powerful technique when analysing loads is to use Markov models as approximations, especially to model the sequence of turning points by a Markov chain. For such models there exist many explicit results. Here we will use this Markov approximation for computing the intensity of rainflow cycles and trough-to-crest cycles for the Gaussian model in Figure 1.2.

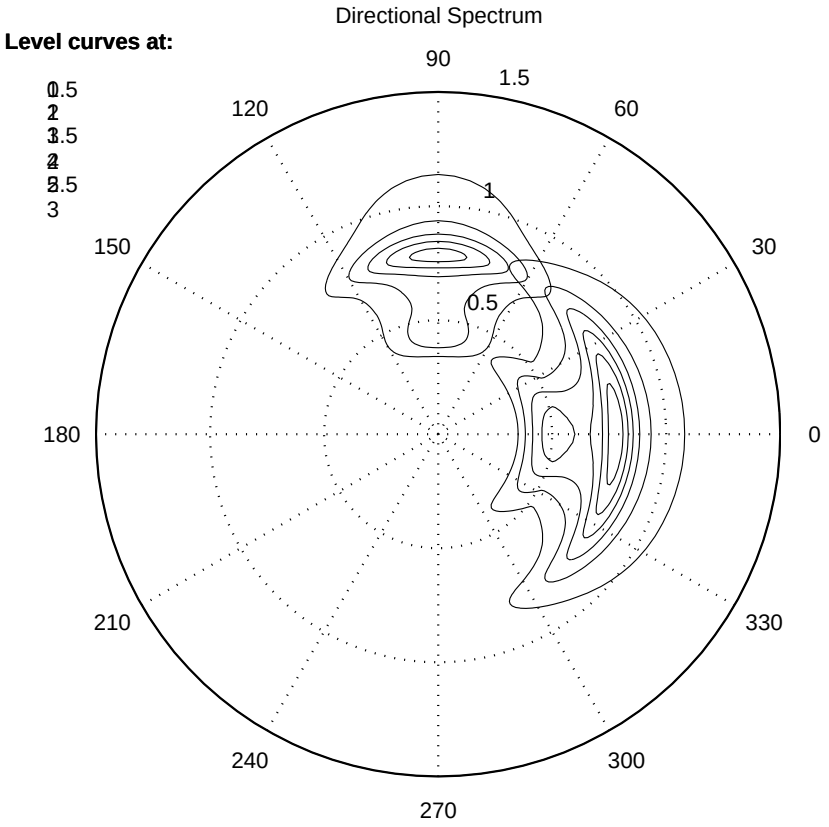


Figure 1.4: *Directional spectrum. The frequency spectrum is a Torsethaugen spectrum and the spreading function is of* cos-2s *type with* $s = 15$. *Solid line: directional spectrum. Dash-dotted line: directional spectrum, using frequency independent spreading function.*

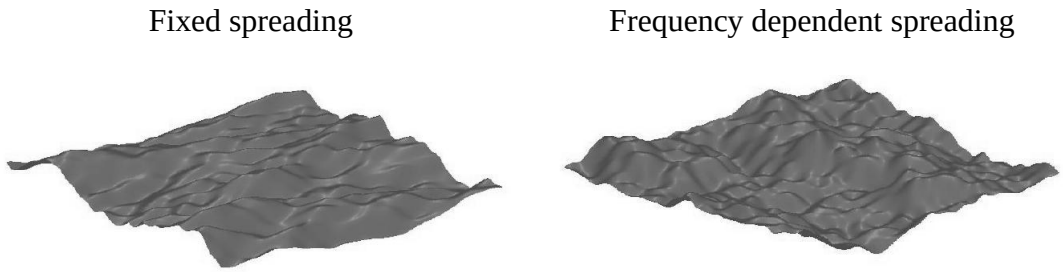


Figure 1.5: *Simulated sea surfaces on a rectangle of 128 [m] by 128 [m] with directional spectrum* `SD1`, *spreading independent of frequency (left), and* `SD12`, *frequency dependent spreading (right).*

For fatigue analysis the rainflow cycle, defined in Figure 4.1 in Chapter 4, is often used. The Markov model is defined by the min-to-max pdf, which is obtained from the power spectral density by using approximations in Slepian model processes, see e.g. [32] and references therein. Chapter 3 describes how WAFO routines can be used to find the min-to-max deistribution for Gaussian loads. For the Markov model there is an explicit solution for the intensity of rainflow cycles, see [13]. By using the routines in WAFO the intensity of rainflow cycles can be found using Markov

approximation; see Figure 1.6, where also the rainflow cycles found in the simulated load signal are shown. The figure has been plotted using the following MATLAB commands:

```
frfc=spec2cmat(S1,[],'rfc',[],[-6 6 61]);
pdfplot(frfc);
hold on
tp=dat2tp(xs);
rfc=tp2rfc(tp);
plot(rfc(:,2),rfc(:,1),'.')
hold off
```

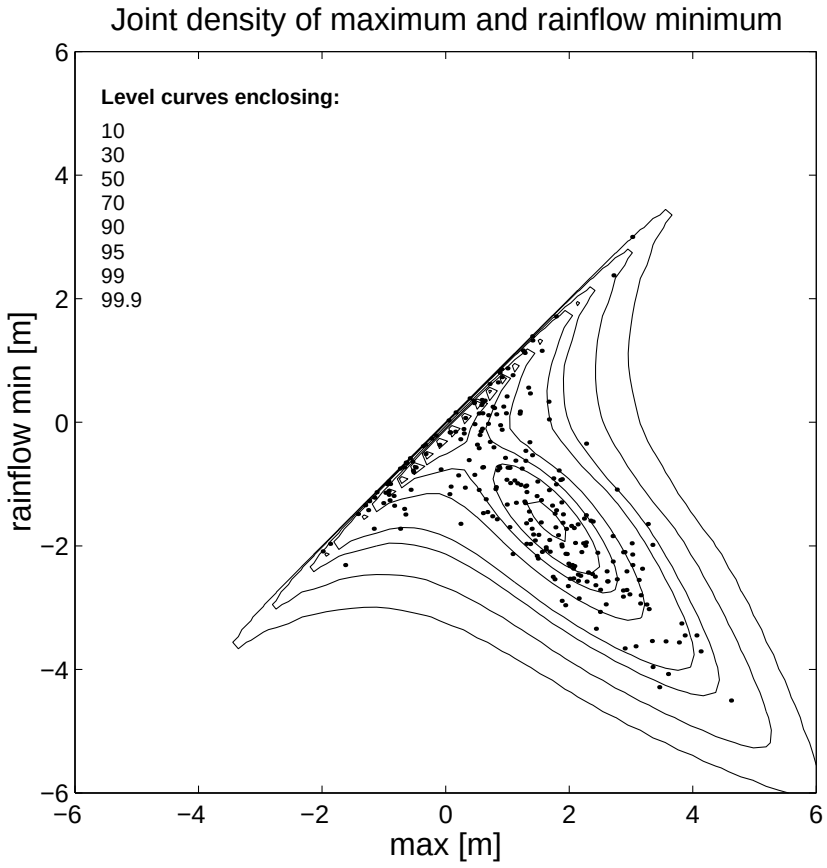


Figure 1.6: *Intensity of rainflow cycles computed from the psd through Markov approximation, compared with the cycles found in the simulation.*

The WAFO toolbox also contains routines for computing the intensity of rainflow cycles in more complex load processes, for example for a switching Markov chain of TP. Details on fatigue load analysis are given in Chapter 4.

1.5 Datastructures

```
help datastructures

  DATASTRUCTURES of spectrum, covariance function and density (pdf) in WAFO

  To represent spectra, covariance functions and probability density functions
  in WAFO, the MATLAB datatype 'structured array' is used. Here follows a list
  of the fields in the struct representing S, cvf and pdf, respectively.

  Spectrum structure
  ~~~~~~~~~~~~~~~~~~
   Requisite fields:
    .type  String: 'freq', 'dir', 'k2d', k1d', 'encdir', 'enc', 'rotdir',
                   'rotk2d' or 'rotk1d'.
    .S     Spectrum values (size=[nf 1] or [np nf]).
    .w OR .f OR .k Frequency/wave number lag, length nf.
    .tr    Transformation function (default [] (none)).
    .h     Water depth (default inf).
    .norm  Normalization flag, Logical 1 if S is normalized, 0 if not
    .note  Memorandum string.
    .date  Date and time of creation or change.
   Type-specific fields:
    .k2    Second dim. wave number lag, if .type='k2d' or 'rotk2d', length np.
    .theta Angular lags, if .type='dir', 'rotdir' or 'encdir', length np.
    .v     Ship speed, if .type = 'enc' or 'encdir'.
    .phi   Direction of ship if .type = 'enc', 'encdir',
           or rotation angle if .type = 'rotdir', 'rotk1d' or 'rotk2d'.

  See also: createspec, wspecplot

  Covariance function (cvf) structure
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    .R Covariance function values. Size [ny nx nt], all singleton dim. removed.
    .x     Lag of first space dimension, length nx.
    .y     Lag of second space dimension, length ny.
    .t     Time lag, length nt.
    .h     Water depth.
    .tr    Transformation function.
    .type  'enc', 'rot' or 'none'.
    .v     Ship speed, if .type='enc'
    .phi   Direction of ship if .type='enc', rotation angle if .type='rot'
    .norm  Normalization flag, Logical 1 if autocorrelation, 0 if covariance.
    .Rx ... .Rtttt   Obvious derivatives of .R.
    .note  Memorandum string.
    .date  Date and time of creation or change.

  See also: createcov, spec2cov, cov2spec, covplot

  Probability density function (pdf) structure
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  Describing a density of n variables:
    .f     Probability density function values (n-dimensional matrix)
    .x     Cell array of vectors defining grid for variables (n cells)
    .labx  Cell array of label strings for the variables (n cells)
    .title Title string
    .note  Memorandum string.

  See also: createpdf, pdfplot
```

CHAPTER 2
RANDOM LOADS AND STOCHASTIC WAVES - MODELING

In this chapter we present some tools for analysis of random functions with respect to their correlation, spectral, and distributional properties. We first give a brief introduction to the theory of Gaussian processes and then we present programs in WAFO which can be used to analyze random functions. The presentation will be organized in three examples; Example 1 is devoted to estimation of different parameters in the model, Example 2 deals with spectral densities and Example 3 presents the use of WAFO to simulate samples of a Gaussian process.

2.1 Introduction and preliminary analysis

The functions we shall analyze can be measured stresses or strains, which we call loads, or other measurements, where waves on the sea surface is one of the most important examples. We assume that the measured data are given by one of the following forms:

1. In the time domain, as measurements of a response function denoted by $x(t)$, $0 \leq t \leq T$, where t is time and T is the duration of the measurements. The $x(t)$-function is usually 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 in estimation of parameters or distributions. We assume that measured functions are saved as a two column ASCII or `mat` file.
2. In the frequency domain, as a power spectrum, which is an important mode in systems analysis. This means that the signal is represented by a Fourier series,

$$x(t) \approx m + \sum_{i=1}^{N} a_i \cos(\omega_i t) + b_i \sin(\omega_i t),$$

   where $\omega_i = i \cdot 2\pi/T$ are angular frequencies, m is the mean of the signal and a_i, b_i are Fourier coefficients.

Some general properties of measured functions can be summarized by using a few simple characteristics. Those are the *mean* m, defined as the average of all values, the standard deviation σ, and the *variance* σ^2, which measures the variability around the mean in linear and quatratic scale. These quantities are estimated by

$$m = 1/T \int_0^T x(t)\,dt, \tag{2.1}$$

$$\sigma^2 = 1/T \int_0^T (x(t) - m)^2\,dt. \tag{2.2}$$

Another important property is the *crossing spectrum* or *c*rossing intensity $\mu(u)$ defined as the intensity of upcrossings (= average number of upcrossings per time unit), of a level u by $x(t)$ as a function of u. The *mean frequency* f_0 is usually defined as the number of times $x(t)$ crosses upwards (upcrosses) the mean level m normalized by the length of the observation interval T, i.e. $f_0 = \mu(m)$. An alternative definition,[1] which we prefer to use is that $f_0 = \max(\mu(u))$, i.e. it is equal to the maximum of $\mu(u)$. The *irregularity factor* α, defined as the mean frequency f_0 divided by the intensity of local maxima (intensity of cycles) in $x(t)$. (Note, a small α means an irregular process, $(0 < \alpha \leq 1)$.)

Example 1. *(Sea data)* In this example we use a series with wave data `sea.dat` with time argument in the first column and function values in the second column. The data used in the examples are wave measurements at shallow water location, sampled with a sampling frequency of 4 Hz, and the units of measurement are seconds and meters, respectively. 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`.

```
xx = load('sea.dat');
me = mean(xx(:,2))
sa = std(xx(:,2))
xx(:,2) = xx(:,2) - me;
lc = dat2lc(xx);
```

Here `me` and `sa` are the mean and standard deviation of the signal, respectively. The variable `lc` is a two column matrix with levels in the first column and the number of upcrossing of the level in the second. In Figure 2.1 the number of upcrossings of `xx` is plotted and compared with an estimation based on the assumption that `xx` is a realization of a Gaussian sea. The plot is automatically drawn by `dat2lc`.

Next we compute the mean frequency as the average number of upcrossings per time unit of the mean level (= 0); this may require interpolation in the crossing intensity curve, as follows.

```
T = max(xx(:,1))-min(xx(:,1))
f0 = interp1(lc(:,1),lc(:,2),0)/T
```

The process of fatigue damage accumulation depends only on the values and the order of the local extremes in the load. The sequence of local extremes is called the *sequence of turning points*. It is a two column matrix with time for the extremes in the first column and the values of `xx` in the second.

```
tp = dat2tp(xx);
alfa = f0/(length(tp)/(2*T))
```

Here `alfa` is the irregularity factor. Note that `length(tp)` is equal to the number of local maxima and minima and hence we have a factor 2 in the expression for `alfa`. □

[1] Still another definition, to be used in Chapter 4, is that f_0 is the average number of completed load cycles per time unit.

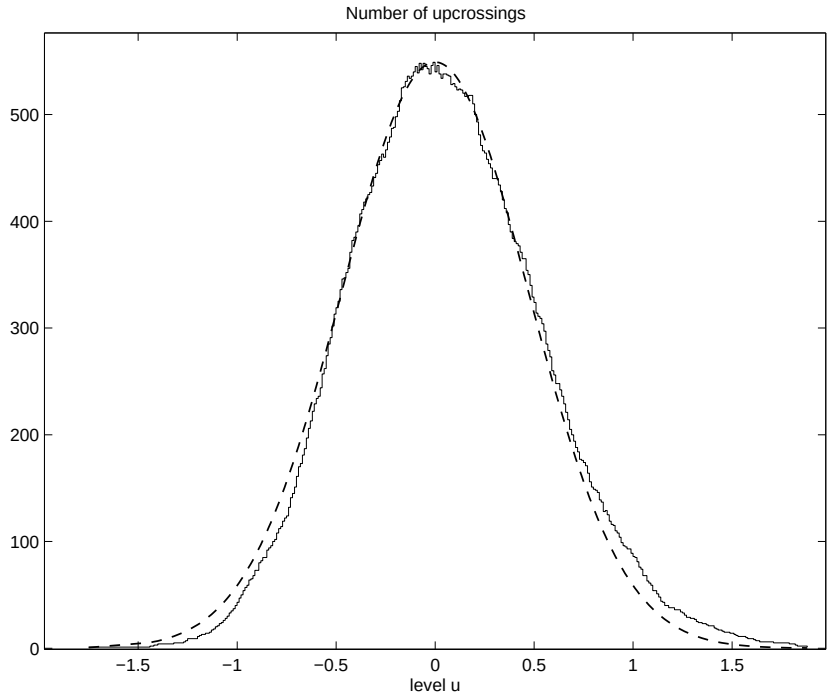


Figure 2.1: *The observed crossings compared with the theoretically expected for Gaussian signals, see (2.3).*

We finish this section with some remarks about the quality of the measured data. Especially sea surface measurements can be of poor quality. We shall now check the quality of the dataset `xx`. It is always good practice to visually examine the data before the analysis to get an impression of the quality, non-linearities and narrow-bandedness of the data.

Example 1. (contd.) First we shall plot the data and zoom in on a specific region. A part of the sea data is presented in Figure 2.2 obtained by the following commands.

```
clf
waveplot(xx,tp,'k-','*',1,1)
axis([0 100 -inf inf])
```

However, if the amount of data is too large for visual examination, or if one wants a more objective measure of the quality of the data one could use the following empirical criterion implemented in WAFO:

- $x'(t) < 5$ [m/s], since the raising speed of Gaussian waves rarely exceeds 5 [m/s],
- $x''(t) < 9.81/2$, $[m/s^2]$ which is the limiting maximum acceleration of Stokes waves,
- if the signal is constant in some intervals, then this will add high frequencies to the estimated spectral density; constant data may occur if the measuring device is blocked during some period of time.

To find possible spurious points of the dataset use the following commands.

```
dt = diff(xx(1:2,1));
dcrit = 5*dt;
ddcrit = 9.81/2*dt*dt;
zcrit = 0;
[inds indg] = findoutliers(xx,zcrit,dcrit,ddcrit);
```

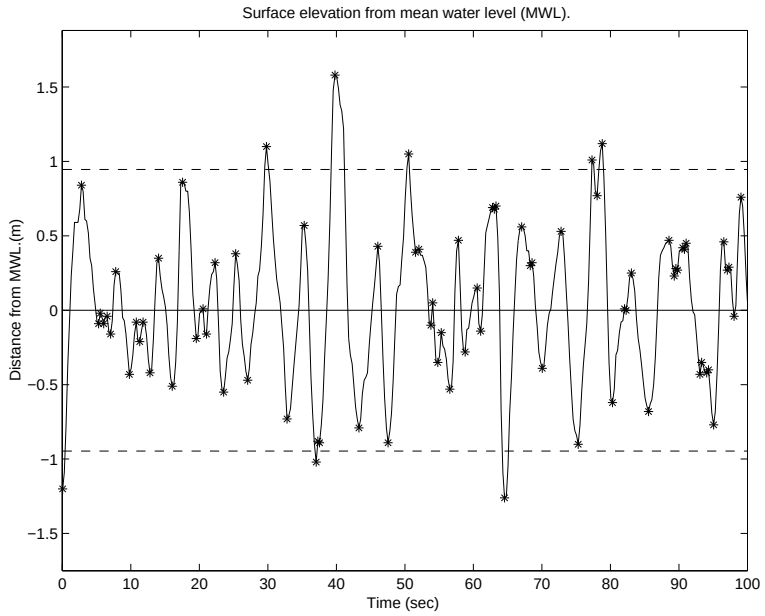


Figure 2.2: *A part of the sea data with turning points marked as stars.*

The program will give the following list when used on the sea data.

```
Found 0 missing points
Found 0 spurious positive jumps of Dx
Found 0 spurious negative jumps of Dx
Found 37 spurious positive jumps of D^2x
Found 200 spurious negative jumps of D^2x
Found 244 consecutive equal values
Found the total of 1152 spurious points
```

The values for `zcrit`, `dcrit` and `ddcrit` can be chosen more carefully. However, small changes of the constants are usually not so crucial. As seen from the transcripts from the program a total of 1152 points are found to be spurious which is approximately 12 % of the data. Based on this one may classify the datasets into good, reasonable, poor and useless. Obviously uncritical use of data may lead to unsatisfactory results. We return to this problem when discussing methods to reconstruct the data. □

2.2 Frequency Modeling of Load Histories

The most important characteristic of signals in frequency domain is their power spectrum

$$\hat{s}_i = (a_i^2 + b_i^2)/(2\Delta\omega),$$

where $\Delta\omega$ is the sampling interval in frequency domain, i.e. $\omega_i = i \cdot \Delta\omega$. The two-column matrix $\hat{s}(\omega_i) = (\omega_i, \hat{s}_i)$ will be called the power spectrum of $x(t)$.

The sequence $\theta_i = \arccos(a_i/\sqrt{2\hat{s}_i\Delta\omega})$ is called a sequence of phases and the Fourier series can be written as follows

$$x(t) \approx m + \sum_{i=1}^{N} \sqrt{2\hat{s}_i\Delta\omega}\cos(\omega_i t + \theta_i).$$

If the sampled signal contains exactly $2N + 1$ points then $x(t)$ is equal to its Fourier series at the sampled points. In the special case when $N = 2^k$, the so-called FFT (Fast Fourier Transform) can be used in order to compute the Fourier coefficients (and the spectrum) from the measured signal and in reverse the signal from the Fourier coefficients.

As mentioned, the Fourier coefficient to the zero frequency is just the mean of the signal, while the variance is given by $\sigma^2 = \Delta\omega \sum \hat{s}(\omega_i) \approx \int_0^\infty \hat{s}(\omega)\, d\omega$. The last integral is called the zero-order spectral moment m_0. Similarly, higher-order spectral moments are defined by

$$m_i = \int_0^\infty \omega^i \hat{s}(\omega)\, d\omega.$$

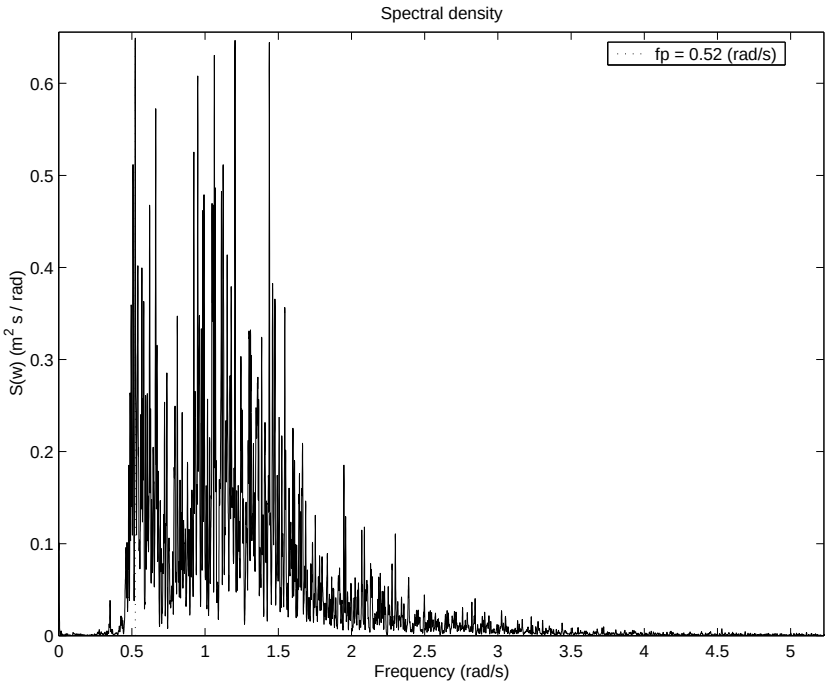


Figure 2.3: *The observed spectrum in the data set* `sea.dat`.

Example 1. (contd.) We give now the spectrum $\hat{s}(\omega)$ for the signal `xx`.

```
S = dat2spec2(xx,9500);
wspecplot(S)
```

In Figure 2.3 we can see that the spectrum is extremely irregular and probably it varies between measurements of the sea even under almost identical conditions. This will be further discussed in the following section. Next the spectral moments will be computed.

```
[mom text] = spec2mom(S,4)
[sa sqrt(mom(1))]
```

The vector `mom` now contains spectral moments m_0, m_2, m_4, which are the variances of the signal and its first and second derivative. We can speculate that the variance of the derivatives is too high partially because of spurious points. For example, if there are several points with the same value, the Gibb's phenomenon leads to high frequencies in the spectrum. □

2.2.1 Random Functions in Spectral Domain - Gaussian processes

In the previous section we studied the properties of one specific signal in frequency domain. Assume now that we get a new series of measurements of a signal, which we are willing to consider as equivalent with the first one. However, the two series are seldom identical and differ in some respects which it is natural to regard as purely random. Obviously it will have a different spectrum $\hat{s}(\omega)$ and the phases will be changed.

A useful mathematical model for such a situation is the random function (stochastic process) model which will be denoted by $X(t)$. Then $x(t)$ is seen as particular chosen randomly function. The simplest model for a stationary signal with a fixed spectrum $\hat{s}(\omega)$ is

$$X(t) = m + \sum_{i=1}^{N} \sqrt{\hat{s}_i \Delta\omega}\sqrt{2}\cos(\omega_i\, t + \Theta_i),$$

where Θ_i are independent uniformly distributed phases. However, this is not a very realistic model either, since in practice one often observes a variability in the spectrum $\hat{s}(\omega)$ between measured functions. Hhence $\hat{s}_i$ should also be modeled to include a certain randomness. The best way to accomplish this is to assume that there exists a deterministic function $S(\omega)$ such that the *average value* of $\hat{s}(\omega_i)\Delta\omega$ between different observed series can be approximated by $S(\omega_i)\Delta\omega$. In fact, in many cases one can model $\hat{s}_i$ as

$$\hat{s}_i = R_i^2 \cdot S(\omega_i)/2,$$

where R_i are independent random factors, all with a Rayleigh distribution. (Observe that the average value of R_i^2 is 2.) This gives the following random function as model for the series,

$$X(t) = m + \sum_{i=1}^{N} \sqrt{S(\omega_i)\Delta\omega}R_i\cos(\omega_i\, t + \Theta_i).$$

The process $X(t)$ has many useful properties that can be used for analysis. In particular, for any fixed t, $X(t)$ is normally (Gaussian) distributed. Then, the probability of any event defined for $X(t)$ can, in principal, be computed when the mean m and the spectral density S are known.

In sea modeling, the components in the sum defining $X(t)$ can be interpreted as individual waves. By the assumption that R_i and Θ_i are independent random variables one has that the individual waves are independent stationary Gaussian processes with mean zero and covariance function given by

$$r_i(\tau) = \Delta\omega\, S(\omega_i)\cos(\omega_i\, \tau).$$

Consequently the covariance between $X(t)$ and $X(t+\tau)$ is given by

$$r_X(\tau) = E[(X(t) - m)(X(t+\tau) - m)] = \Delta\omega \sum_{i=1}^{N} S(\omega_i)\cos(\omega_i\, \tau).$$

More generally, for a stationary stochastic process with spectral density $S(\omega)$, the correlation structure of the process is defined by its spectral density function, also called power spectrum,

$$r(\tau) = \mathrm{Cov}[X(t), X(t+\tau)] = \int_0^\infty \cos(\omega\tau)\, S(\omega)\, d\omega.$$

The spectral density represents a continuous distribution of the wave energy over a continuum of frequencies.

If $Y(t)$ is an output of a linear filter with $X(t)$ as input, then $Y(t)$ is also normally distributed and we need to derive the spectrum of $Y(t)$ to be able to analyze its properties. This is a simple task, since if the transfer function of the filter $H(\omega)$ is given, then the spectrum of $Y(t)$, denoted by S_Y, is given by

$$S_Y(\omega) = |H(\omega)|^2 S(\omega).$$

For example, the derivative $X'(t)$ is a Gaussian process with mean zero and spectrum $S_Y(\omega) = \omega^2 S(\omega)$. The variance of the derivative is equal to the second spectral moment,

$$\sigma^2_{X'} = \int S_Y(\omega)\, d\omega = \int \omega^2 S(\omega)\, d\omega = m_2.$$

The Gaussian process is a sum of cosine terms with amplitudes defined by the spectrum; hence, it is not easy to relate the power spectrum and the fatigue damage. The crossing intensity $\mu(u)$, which yields the average number of upcrossings of the level u, contains some information also on the fatigue properties. For a Gaussian process it is given by the celebrated *Rice's formula*,

$$\mu(u) = f_0 \exp(-(u-m)^2/2\sigma^2). \tag{2.3}$$

Using spectral moments we have that $\sigma^2 = m_0$ while $f_0 = \frac{1}{2\pi}\sqrt{\frac{m_2}{m_0}}$.

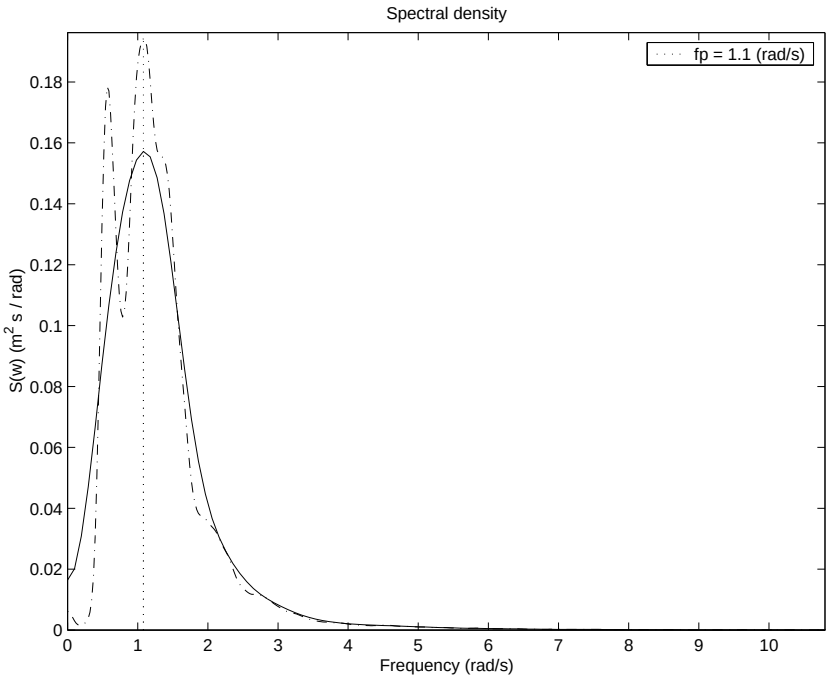


Figure 2.4: *The estimated spectrum in the data set* `sea.dat` *with different degree of smoothing.*

Example 1. (contd.) In order to estimate the spectrum of a Gaussian process one needs several realizations of the process. Then one spectrum estimate can be made for each realization which are then averaged. However, in many cases only one realization of the process is available. In such a case one is often assuming that the spectrum is a smooth function of ω and can use this information to improve the estimate. Practically it means that one has to use some smoothing techniques. For the sea data we shall estimate the spectrum by means of the WAFO function `dat2spec` which a second parameter defining the degree of smoothing.

```
S1 = dat2spec2(xx,200);
S2 = dat2spec2(xx,50);
wspecplot(S1,[],'-.')
hold on
wspecplot(S2)
hold off
```

In Figure 2.4 we see that with decreasing second input the spectrum estimate becomes smoother, and that it in the end becomes unimodal.

Obviously knowing the spectrum one can compute the covariance function. The following code in MATLAB will compute the covariance for the unimodal spectral density `S2` and compare it with estimated covariance in the signal `xx`.

```
R2 = spec2cov(S1,1);
Rest = dat2cov(xx,80,[],'- -');
covplot(R2,80,[],'.')
hold on
covplot(Rest)
hold off
```

We can see in Figure 2.5(a) that the covariance function corresponding to the spectral density `S2` significantly differs from the one estimated directly from data. It can be seen that the covariance corresponding to `S1` agrees much better with the estimated covariance function; see Figure 2.5(b), which is obtained using the same code with `S2` in `spec2cov` replaced by `S1`. □

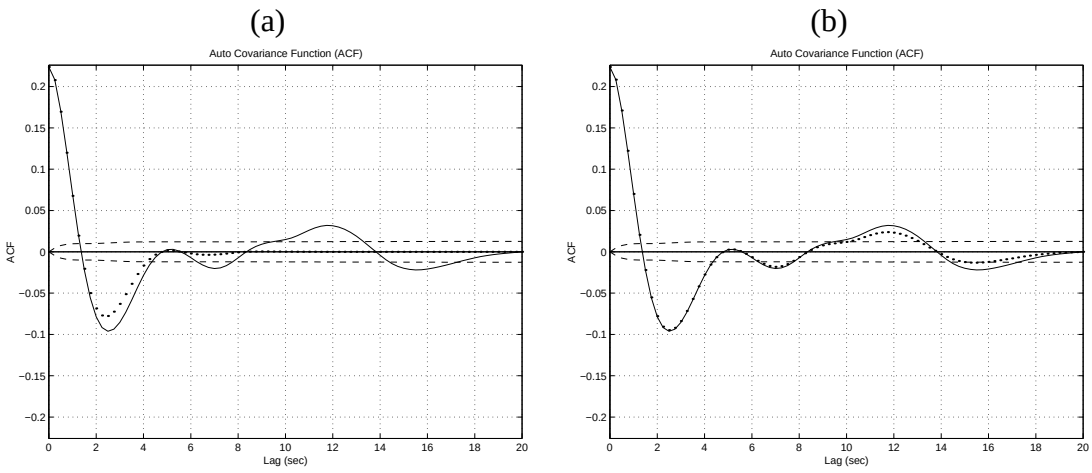


Figure 2.5: *The covariance function estimated in the data set* `xx`*, solid line, compared the theoretically computed covariance functions for the spectral densities* `S2` *in (a) and* `S1` *in (b).*

Observe that the WAFO function `spec2cov` can be used to compute a covariance structure which can contain covariances both in time and in space as well as that of the derivatives. The input can be any spectrum structure, e.g. wave number spectrum, directional spectrum or encountered directional spectrum; type `help spec2cov` for the detailed information.

2.2.2 Transformed Gaussian models

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 $S(\omega)$. The knowledge of which kind of spectral density $S(\omega)$ are suitable to describe sea state data is well established from experimental studies.

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 $S(\omega)$ for Gaussian processes, one often looks for a model of the sea state in the class of Gaussian processes. Furthermore, in previous work [54] 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 [54] that a simple transformed Gaussian process used to model $x(t)$ gave good approximations for those densities.

Consequently, in WAFO we shall model $x(t)$ by a process $X(t)$ which is a function of a single Gaussian process $\widetilde{X}(t)$, i.e.

$$X(t) = G(\widetilde{X}(t)), \tag{2.4}$$

where $G(\cdot)$ is a continuously differentiable function with positive derivative. We shall denote the spectrum of X by S, while the spectrum of $\widetilde{X}(t)$ by $\widetilde{S}$.

The transformation G performs all appropriate translation and scaling so that $\widetilde{X}(t)$ is always normalized to have mean zero and variance one, i.e. the first spectral moment of $\widetilde{S}$ is one.

Note that once the distributions of crests, troughs, amplitudes or wavelengths in a Gaussian process $\widetilde{X}(t)$ 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 $\widetilde{x}(t) = g(x(t))$ between the real sea data $x(t)$ and the transformed data $\widetilde{x}(t)$.

If the model (2.4) is correct, then $\widetilde{x}(t)$ 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.

There are several different ways to proceed when selecting a transformation. The simplest alternative is to estimate the function g directly from data by some parameteric or non-parametric means. A more physically motivated procedure is to use some of the parametric functions proposed in the literature, based on approximations of non-linear wave theory. The following options are programmed in the toolbox:

```
dat2tr        - non-parametric transformation g proposed by Rychlik,
hermitetr     - transformation g proposed by Winterstein,
ochitr2       - transformation g proposed by Ochi et al.
```

The transformation proposed by Ochi, [39], is a monotonic exponential function while Winterstein's model, [62], is a monotonic cubic Hermite polynomial. Both transformations use moments of $X(t)$ to compute g. Information about the moments of the process can be obtained by site specific data, laboratory measurements or by resort to physical models. Marthinsen and Winterstein (1992), [37], derived an expression for the skewness and kurtosis for narrow banded Stokes waves to the leading order and used these to define the transformation. The skewness and kurtosis (excess) of this model can also be estimated from data by the WAFO functions `wskewness` and `wkurtosis`.

Example 1. (contd.) We begin with computations of skewness and kurtosis for the data set `xx`. The commands

```
rho3 = wskewness(xx(:,2))
rho4 = wkurtosis(xx(:,2))
```

give the values `rho3 = 0.25` and `rho4 = 3.17`, respectively, compared to `rho3 = 0` and `rho4 = 3` for Gaussian waves. We can compute the same model using the spectrum $\tilde{S}$ using the second order wave approximation proposed by Winterstein. His approximation gives suitable values for skewness and kurtosis

```
[sk, ku ]=spec2skew(S1);
```

Here we shall use the Hermite transformation proposed by Winterstein and denote it by `gh` and compare it with the linear transformation, i.e. when $X(t)$ is Gaussian, denoted by `g`.

```
gh = hermitetr([],[sa sk ku me]);
g  = gh; g(:,2)=g(:,1)/sa;
trplot(gh)
```

These commands will result in two column matrices, `g, gh`, with equally spaced y-values in the first column and the values of $g(y)$ in the second column.

Since we have data we may estimate the transformation directly by a method proposed by Rychlik, [54].

```
[glc test0] = dat2tr(xx);
hold on
plot(gh(:,1),gh(:,2),'b-.')
hold off
```

The same transformation can be obtained from the crossing intensity by use of the WAFO function `lc2tr`.

In Figure 2.6 we compare the three transformations, the straight line is the Gaussian linear model, the dashed dotted line is the Hermite transformation based on higher moments of the response computed from the spectrum and the solid line is the direct transformation estimated from crossing intensity. (The unsmoothed line shows the estimation of the direct transformation from unsmoothed crossing intensity). We can see that the transformation derived from crossings will give the highest crest heights. It can be proved that asymptotically the transformation based on crossings intensity gives the correct density of crest heights.

The transformations indicates that data `xx` has a light lower tail and heavy upper tail compared to a Gaussian model. This is also consistent with second order wave theory where the crests are higher and the troughs shallower compared to Gaussian waves. Now the question is whether this difference is significant compared to the natural statistical variability due to finite length of the time series.

To determine the degree of departure from Gaussianity, we can compare an indicator of non-Gaussianity `test0` with calculated value obtained from Monte Carlo simulation. The value of `test0` is a measure of how munch the transformation `g` deviates from a straight line.

The significance test is done by simulating 50 independent samples of `test0` from a true Gaussian process with the same spectral density and length as the original data. This is accomplished by the WAFO program `mctrtest`. The output from the program is a plot of the ratio `test1` between the simulated (Gaussian) `test0`-values and the sample `test0`:

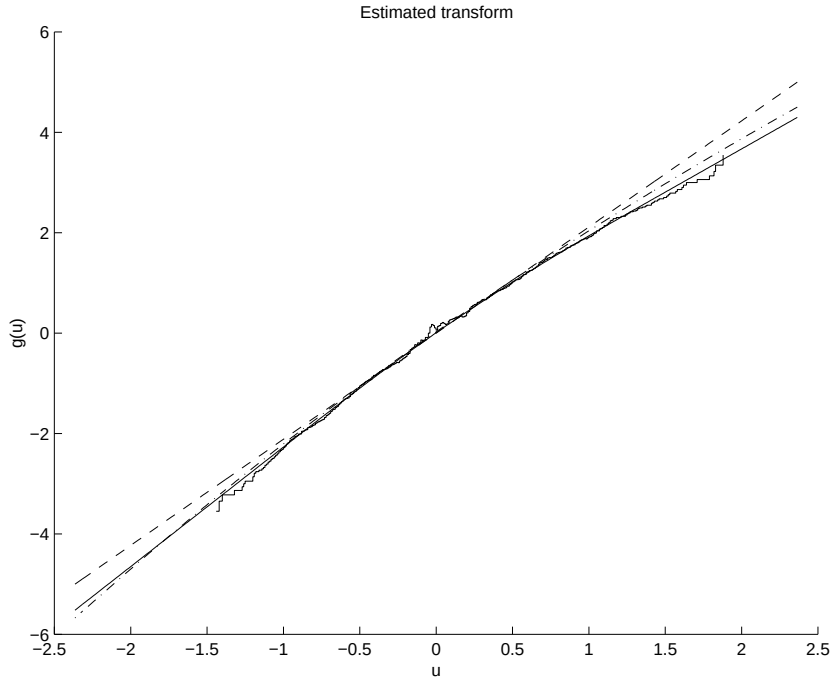


Figure 2.6: *Comparisons of the three transformations g, straight line - Gaussian model, dashed-dotted line the Hermite transformation* `gh` *and solid line the Rychlik method* `glc`.

```
N = length(xx);
test1 = mctrtest(S1,[N,50],test0);
```

The program gives a plot of simulated `test` values, see Figure 2.7. As we see from the figure none of the simulated values of `test1` is above 1.00. Thus the data significantly departs from a Gaussian distribution; see [54] for more detailed discussion of the testing procedure and the estimation of the transformation g from the crossing intensity.

We finish the tests for Gaussianity of the data by a more classical approach and simply plot the data on normal probability paper. Then N independent observation of identically distributed Gaussian variables forms a straight line in a normalplot. Now for a time series the data is clearly not independent. However, if the process is ergodic then the data forms straight line as N tends to infinity.

The command

```
wnormplot(xx(:,2))
```

produces Figure 2.8.

As we can see the normal probability plot is curved indicating than that the underlying distribution has a “heavy” upper tail and a “light” lower tail. □

2.2.3 Spectral densities of sea data

The knowledge of which kind of spectral density $S(\omega)$ 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. This formula is programmed in a WAFO function `jonswap`, which evaluates the spectral density $S(\omega)$ with specified wave characteristics. There are

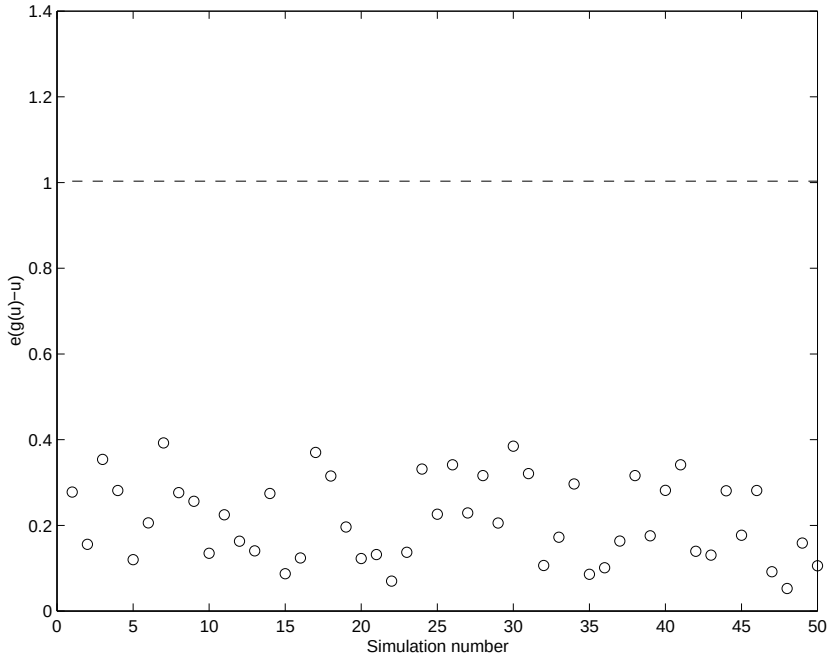


Figure 2.7: *The simulated 50 values of the* `test` *variable for the Gaussian process with spectrum* `S1` *compared with the observed value (= the dashed line).*

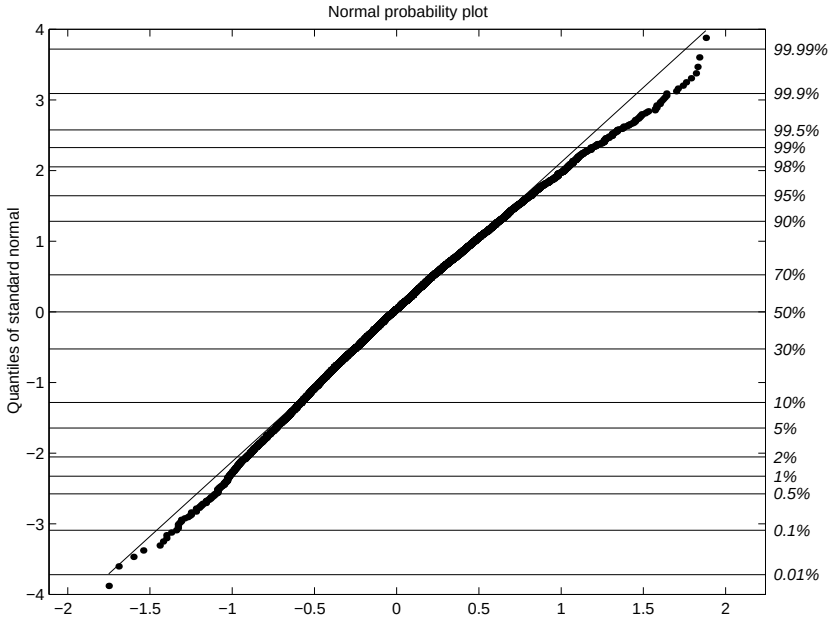


Figure 2.8: *The data* `xx` *on normal probability plot*

several other programmed spectral densities in WAFO to allow for bimodal and finite water depth spectra. The list includes the following spectra:

```
jonswap       - JONSWAP spectral density
wallop        - Wallop spectral density,
ohspec2       - Bimodal Ochi-Hubble spectral density,
torsethaugen  - Bimodal (swell + wind) spectral density,
pmspec        - Pierson-Moskowitz spectral density,
mccormick     - McCormick spectral density,
tmaspec       - JONSWAP spectral density for finite water depth,
```

WAFO also contains some different spreading functions; use `help spec` and help `spreading` for more detailed information.

The spectrum of the sea can be given in many different formats, that are interconnected by the dispersion relation. The spectrum can be given using frequencies, angular frequencies or wave numbers, and it can also be directional.

A related spectrum is the encountered spectrum for a moving vessel. The transformations between the different types of spectra are defined by means of integrals and variable change defined by the dispersion relation and the Doppler shift of individual waves. The function `spec2spec` makes all these transformations easily accessible for the user. (Actually many programs perform the appropriate transformations internally whenever it is necessary and for example one can compute the density of wave-length from an input that is the directional spectrum in frequency domain.)

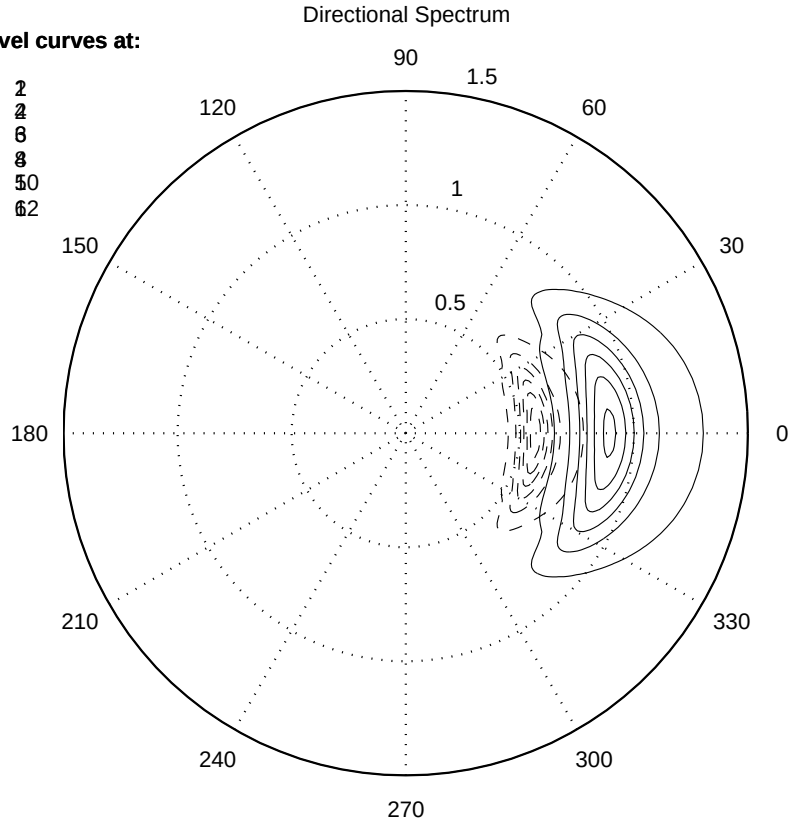


Figure 2.9: *The directional spectrum of JONSWAP sea (dashed line) compared with the encountered directional spectrum for heading sea, speed 10 [m/s] (solid line).*

Example 2. *(Different form of spectra)* In this example we have chosen a JONSWAP spectrum with parameters defined by significant wave height `Hm0 = 7` [m] and peak period `Tp = 11` [s]. This spectrum describes the measurements of sea evel at a fixed point (buoy).

```
Hm0 = 7; Tp = 11;
spec = jonswap([],[Hm0 Tp]);
spec.note
```

In order to include the space dimension, i.e. the direction in which the waves propagate, we compute a directional spectrum by adding spreading; see dashed curves in Figure 2.9.

```
D = spreading(101,'cos2s',0,[],spec.w,1)
Sd = mkdspec(spec,D)
```

Next we conside a vessel moving with speed 10 [m/s] against the waves. The sea measured from the vessel will have a different directional spectrum, called the encountered directional spectrum. The following code will compute the encountered directional spectrum and plot it on top of the original directional spectrum. The result is shown as the solid curves in Figure 2.9.

```
Se = spec2spec(Sd,'encdir',0,10);
wspecplot(Se), hold on
wspecplot(Sd,1,'--'), hold off
```

Obviously the periods of waves in the directional sea are defined by the JONSWAP spectrum (spreading is not affecting the sea level at a fixed point), but the encountered periods will be shorter. This can be seen by comparing the following three spectra:

```
S1 =spec2spec(Sd,'freq');
S2 = spec2spec(Se,'freq');
wspecplot(spec), hold on
wspecplot(S1,1,'.'),
wspecplot(S2),
hold off
```

We can see in Figure 2.10(a) that the spectra `spec` and `S1` are identical (in numerical sense) while spectrum `S2` contains more energy at higher frequencies.

A similar kind of question is how much the wave length differs between a longcrested JONSWAP sea and a JONSWAP sea with spreading. The wavenumber spectra for both cases can be computed by the following code, the result of which is shown in Figure 2.10(b).

```
Sk = spec2spec(spec,'k1d')
Skd = spec2spec(Sd,'k1d')
wspecplot(Sk), hold on
wspecplot(Skd,1,'--'), hold off
```

Finally we shall show how the JONSWAP spectrum can be corrected for a finite depth, see [9]. The following code computes spectrum for water of finite depth of 20 [m]. The resulting spectra are shown in Figure 2.11.

```
wspecplot(spec,1,'--'), hold on
S20 = spec;
S20.S = S20.S.*phi1(S20.w,20);
S20.h = 20;
wspecplot(S20),  hold off
```

□

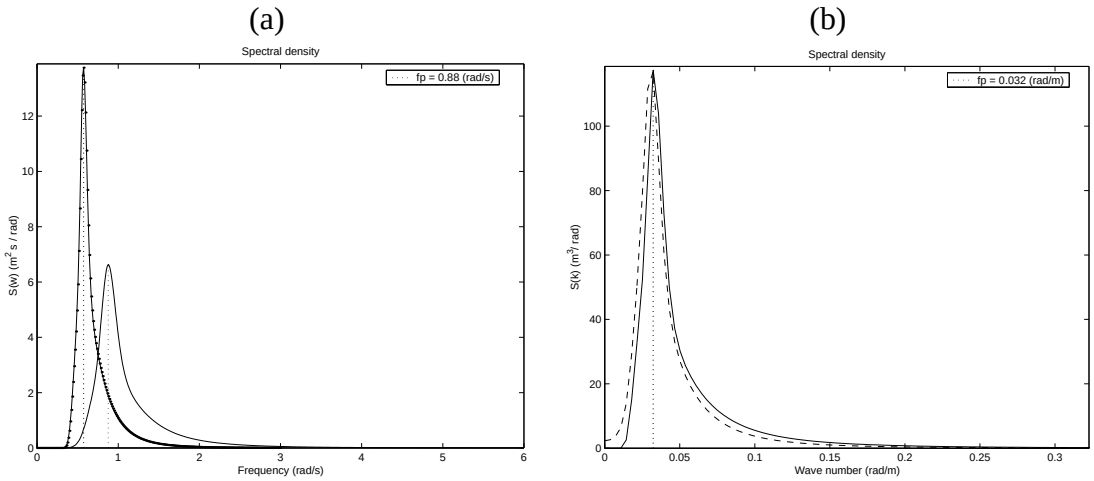


Figure 2.10: *(a) The frequency JONSWAP spectrum compared with encountered frequency spectrum for heading sea speed 10 [m/s] (solid line). (b) The wave number spectrum for longcrested JONSWAP sea (solid line) compared with wave number spectrum for JONSWAP with spreading.*

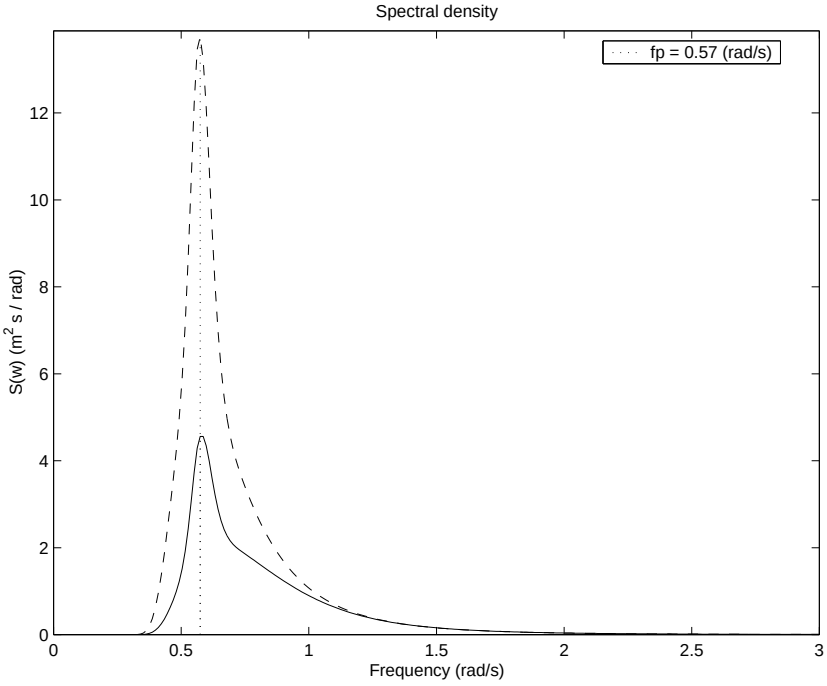


Figure 2.11: *Standard JONSWAP spectrum (dashed line) compared with the spectrum on finite depth of 20 [m] (solid line)*

2.3 Simulation of transformed Gaussian process

In this section we shall present some of the programs in WAFO that can be used to simulate random signals, loads and waves; type `help wsim` for the complete list. We shall be mostly concerned with simulation of the transformed Gaussian model for sea $X(t) = G(\widetilde{X})$.

The first important case is when we wish to reproduce random versions of the measured signal $x(t)$. Using `dat2tr` one can first estimates the transformation g. Next, using a function

`dat2gaus` one can compute $\tilde{x}(t) = g(x(t))$, which we assume is a realization of a Gaussian process. From $\tilde{x}$ we can then estimate the spectrum $\tilde{S}(\omega)$ by means of the function `dat2gaus`. The spectrum $\tilde{S}(\omega)$ and the transformation g will uniquely define the transformed Gaussian model. A random function that models the measured signal can then be obtained using the program `spec2sdat`. In the following example we shall illustrate this approach on the data set `sea.dat`.

Before we can start to simulate we need to put the transformation into the spectrum data structure, which is a MATLAB structure variable. Since WAFO is based on transformed Gaussian processes the entire process structure is defined by the spectrum and the transformation together. Therefore the transformation has been incorporated, as part of a model, into the spectrum structure, and is passed to other WAFO programs with the spectrum. If no transformation is given then the process is Gaussian.

Observe that the possible nonzero mean `m`, say, for the model is included in the transformation. The change of mean by for example 0.5 [m] is simply accomplished by modyfying the transformation, e.g. by executing the following command `g(:,2)=g(:,2)+0.5;`. Consequently the spectrum structure completely defines the model.

IMPORTANT NOTE: The simulation routine `spec2sdat` assumes that the input spectrum is the spectrum of a standardized process with spectral moment $m_0 = 1$, i.e. unit variance. The correct standard deviation for the output should normally be obtained via a transformation `spectrum.tr`. If you happen to use a transformation *together with* an input spectrum which does not have unit variance, then you get the double scale effect, both from the transformation and via the standard deviation from the spectrum. It is only the routine `spec2sdat` that works in this way. All other routines, in particular those which calculate cycle distributions, perform an internal normalization of the spectrum before the calculation, and then transforms back to the original scale at the end.

Example 3. *(Simulation of a random sea)* In Example 1 we have shown that the data set `sea.dat` contains a considerable amount of spurious points that we would like to omit or censor.

The program `reconstruct` replaces the spurious data by simulated data (one is assuming that no information about the removed points is known and one is filling up the gaps on the basis of the remaining data and fitted transformed Gaussian process; see [7] for more details. The reconstruction is performed as

```
[y grec] = reconstruct(xx,inds);
```

where `y` is the reconstructed data and `grec` is the transformation estimated from the signal `y`. In Figure 2.12 we can see the transformation (solid line) compared with the empirical smoothed transformation, `glc`, which is obtained from the original sequence `xx` without removing the spurious data (dash-dotted line). We can see that the new transformations has slightly smaller crests. Actually it is almost identical with the transformation `gh` computed from the spectrum of the signal, however it can be only a coincident (due to random fluctuations) and hence we do not draw any conclusions from this fact.

The value of the `test` variable for the transformation `grec` is 0.84 and, as expected, it is smaller than the value of `test0` = 1.00 computed for the transformation `glc`. However it is still significantly larger then the values shown in Figure 2.7, i.e. the signal `y` is not a Gaussian signal.

We turn now to estimation of the spectrum in the model. First transform data to obtain a sample $\tilde{x}(t)$.

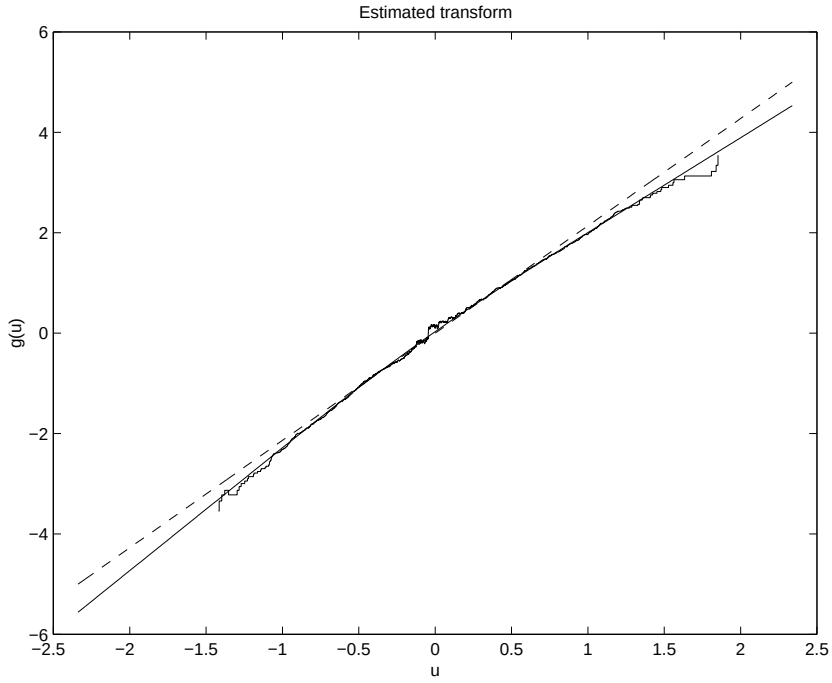


Figure 2.12: *The transformation computed from the reconstructed signal* `y` *(solid line) compared with the transformation computed from the original signal* `xx` *(dashed dotted line).*

```
x = dat2gaus(y,grec);
Sx = dat2spec(x,200);
```

The important remark here is that the smoothing of the spectrum defined by the parameter `L`, see `help dat2spec`, is removing almost all differences between the spectra in the three signals `xx`, `y` or `x`. (The spectrum `Sx` is normalized to have first spectral moment one and has to be scaled down to have the same energy as the spectrum `S1`.)

Next we shall simulate a random function equivalent to the reconstructed measurements `y`. The Nyquist frequency gives us the time sampling of the simulated signal,

```
dt = spec2dt(Sx)
```

and is equal to 0.25 seconds, since the data has been sampled with a sampling frequency of 4 Hz. We then simulate 2 minutes (2*60*4 points) of the signal, to obtain a synthetic wave equivalent to the reconstructed sea data, shown in Figure 2.13.

```
Sx.tr = grec;
ysim = spec2sdat(Sx,480);
waveplot(ysim,'-')
```

□

In the next example we consider a signal with a theoretical spectrum. Here we have a problem whether the theoretical spectrum is valid for the transformed Gaussian model, i.e. it is a spectrum $S(\omega)$ or is it the spectrum of the linear sea $\widetilde{S}$. In the previous example the spectrum of the transformed process was almost identical with the normalized spectrum of the original signal. In [54] it was observed that for sea data the spectrum estimated from the original signal and that for the transformed one do not differ significantly. Although more experiments should be done in order to

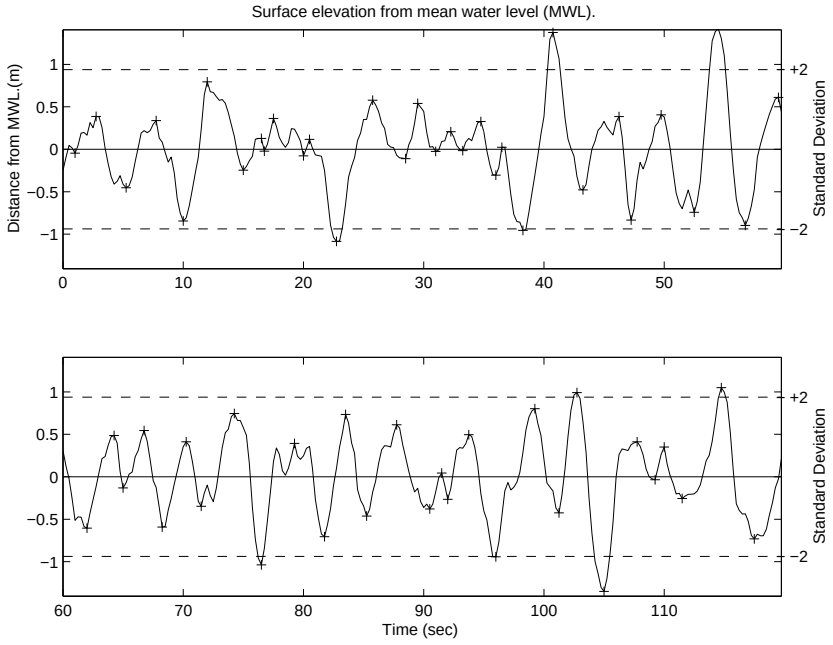


Figure 2.13: *Two minutes of simulated sea data, equivalent to the reconstructed data.*

recommend using the same spectrum in the two cases, here, if we wish to work with non-Gaussian models with a specified transformation, we shall derive the $\widetilde{S}$ spectrum by dividing the theoretical spectrum by the square root of the first spectral moment of S.

Example 3. (contd.) Since the spectrum `S1` is clearly two-peaked with peak frequency $T_p = 1.1$ [Hz] we choose to use the Torsethaugen spectra. (This spectrum is derived for a specific location and we should not expect that it will work for our case.) The inputs to the programs are T_p and H^s, which we now compute.

```
Tp = 1.1;
H0 = 4*sqrt(spec2mom(S1,1))
St = torsethaugen([0:0.01:5],[H0  2*pi/Tp]);
wspecplot(S1)
hold on
wspecplot(St,[],'-.')
```

In Figure 2.14 we can see that the Torsethaugen spectrum has too little energy on the swell peak. Despite this fact we shall use this spectrum in the rest of this example.

We shall now create the spectrum $\tilde{S}(\omega)$, i.e. the spectrum for the standardized geussian process $\widetilde{X}(t)$ with standard deviation equal to one.

```
Snorm = St;
Snorm.S = Snorm.S/sa^2;
dt = spec2dt(Snorm)
```

The sampling interval `dt` = 0.63 [s] is a consequence of our choice of cut off frequency in the definition of the `St` spectrum. This will however not affect our simulation where any sampling interval `dt` can be used.

Next we recompute the theoretical transformation `gh`.

```
[Sk Su] = spec2skew(St);
sa = sqrt(spec2mom(St,1));
gh = hermitetr([],[sa sk ku me]);
Snorm.tr = gh;
```

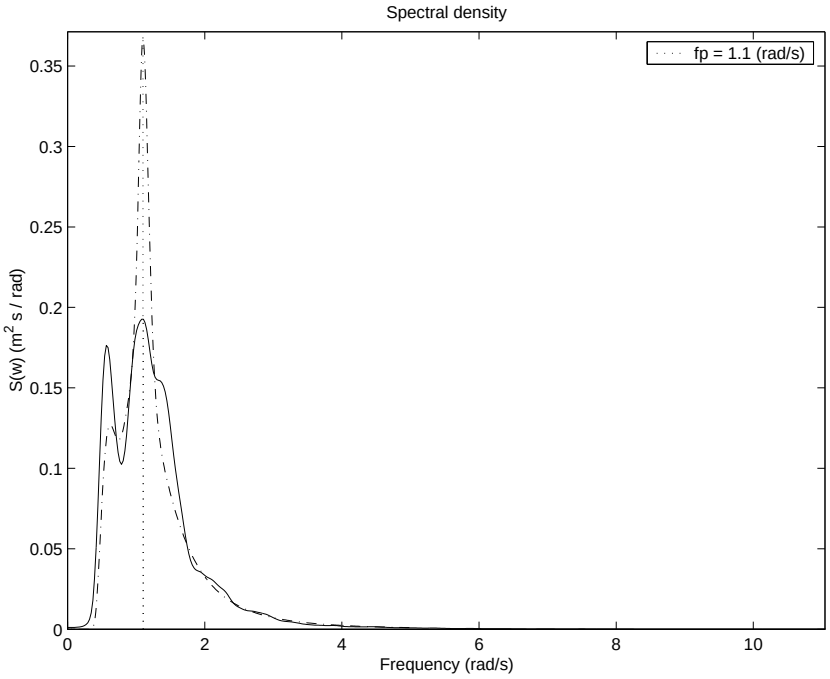


Figure 2.14: *Comparison between the estimated spectrum in the signal* `sea.dat` *(solid line) and the theoretical spectrum of the Torsethaugen type (dashed dotted line).*

The transformation is actually almost identical to `gh` for the spectrum `S1`, which can be seen in Figure 2.6, where it is compared to the Gaussian model `g`, given by a straight line. We can see from the diagram that the waves in a transformed Gaussian process $X(t) = G(\widetilde{X}(t))$, will have an excess of high crests and shallow troughs compared to waves in the Gaussian process $\widetilde{X}(t)$. The difference is largest for extreme waves with crests above 1.5 meters, where the excess is 10 cm, ca 7 % . Such waves, which have crests above three standard deviations `sa` are quite rare and for moderate waves the difference is negligible.

In order to illustrate the difference in distribution for extreme waves we will simulated a sample of 4 minutes of $X(t)$ with sampling frequency 2 Hz. The result are put into `ysim_t`. In order to obtain the corresponding sample path of the process $\widetilde{X}$ we use the transformation `gh` and put the result in `xsim_t`.

```
dt = 0.5;
ysim_t = spec2sdat(Snorm,240,dt);
xsim_t = dat2gaus(ysim_t,Snorm.tr);
```

Since the process $\tilde{X}(t)$ has always variance one, in order to compare the Gaussian and non-Gaussian models we need to scale the `xsim_t` to have the same first spectral moment as `ysim_t`, which will be done by the following commands.

```
xsim_t(:,2) = sa*xsim_t(:,2);
waveplot(xsim_t,ysim_t,5,1,sa,4.5,'r.','b')
```

In Figure 2.15 we have waves that are not extremely high and hence the difference between the two models is hardly noticeable in this scale. Only in the second subplot we can see that Gaussian waves (dots) has trough deeper and crest lower than the transformed Gaussian model (solid line). This also indicates that the amplitude estimated from the transformed Gaussian and

Gaussian models are practically identical. Using the transformation `glc` instead of `gh` would give errors of ca 11%, which for waves with higher significant wave heigh would give considerable underestimation of the crest heigh of more extreme waves. Even if the probability for observing an extreme wave during the period of 20 minutes is small, it is not negligible for safety analysis and therefore the choice of transformation is one of the most important questions in wave modeling.

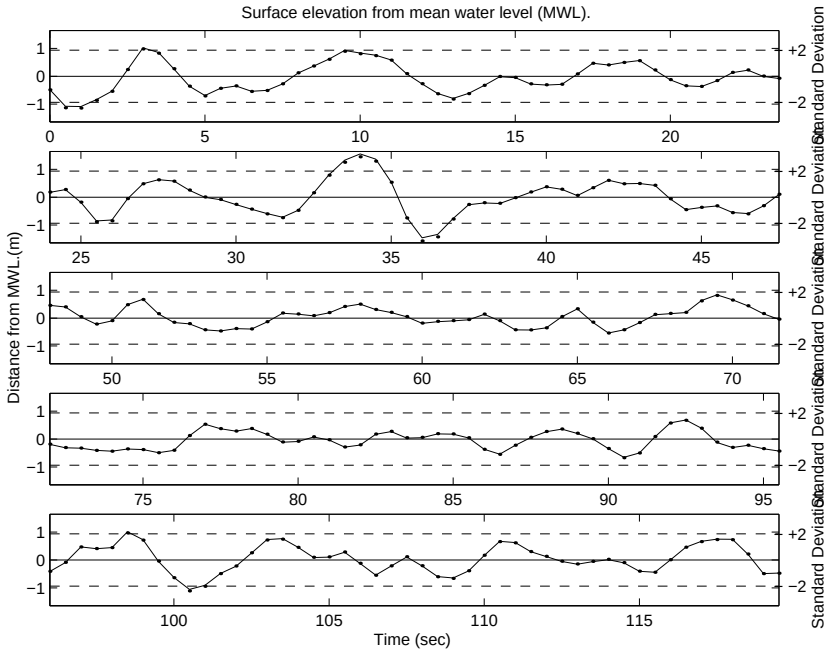


Figure 2.15: *Simulated $X(t) = G(\widetilde{X}(t))$ (solid line) compared with $\widetilde{X}(t)$ scaled to have the same H^s as $X(t)$ for a theoretical spectrum given by Torsethaugen spectrum* `St`.

Since the difference between Gaussian and non-Gaussian model is not so big we may ask a question whether 20 minutes of observation of a transformed Gaussian process presented in this example is long enough to be able to reject the Gaussian model. Using the function `mctrtest` we can sea rejection of Gaussian model would be very seldom. Observe that the `sea.dat` is 40 minutes long an that we clearly had rejected the Gaussian model. □

In WAFO there are several other programs to simulate the random functions or surfaces. Important class used in fatigue analysis and in modeling the long term variability of sea state parameters are Markov models. There is also a program to simulate the output of second order oscillators with nonlinear spring, when external force which is white noise. The nonlinear oscillators can be used to model nonlinear responses of sea structures.

CHAPTER 3

DISTRIBUTIONS OF APPARENT WAVE CHARACTERISTICS

3.1 Introduction

In the previous chapter we discussed modeling of random function by means of Fourier methods. The signal was represented as a sum of random cosine functions with random amplitudes and phases. In linear wave theory those cosine functions are waves traveling in water. Waves with different frequencies have different speeds, defined by the dispersion relation. This property causes the characteristic irregularity of sea surface. Even if it were possible to arrange a very particular combination of phases and amplitudes, so that the signal looks, for example, like a saw blade, it will, after a while, change shape totally. The phases will be almost independent and the sea would again look like a Gaussian random process. On the other hand an observer clearly can identify moving sea waves. The shape of those waves, which are often called the *apparent waves*, since theoretically, those are not mathematical waves, but are constantly changing up to the moment of disappearing.

The waves action on marine structures is often modeled using linear filters. Then the sea spectrum gives a complete characterization of the structures responses. However, often such models are too simplistic and non-linearities have to be considered to allow more complex responses. Then one may not wish to perform a complicated numerical analysis to derive the complete response but is willing to accept the simplification that the response is proportional to the waves. One may also wish to identify some properties of waves that are dangerous in some way for the particular ocean operation. Also the apparent waves themselves can be the reason for non-linear response. For example, for waves with crests of apparent waves higher then some threshold, water may fill a structure and change its dynamical properties. The combined effect of apparent waves, often described by its height and wave period, is therefore important in ocean engineering. These aspects are discussed in more detail in the textbook by Ochi (1998).

The apparent waves will be described by some geometric properties, which will be called wave characteristics, while frequencies of occurences of waves with specified characteristics will treated in the statistical sense and described by a probability distribution. Such distributions can then be used to estimate the frequency of occurrences of some events important in the design of floating marine systems, e.g. wave breaking, slamming, ringing, etc.

The wave surface is clearly a two-dimensional phenomenon that changes with time. Its study should naturally deal with moving two-dimensional objects (surfaces). Theoretical studies of random surfaces still face major difficulties and are a subject of ongoing research, for example see the PhD thesis by Sjö (2000), [56], where some results concerning the combined time-space aspects of

waves are presented. At present there are only few programs in WAFO that handle the space-time relations of waves, and hence in this tutorial, we shall not present any examples of waves in space and time but limit the presentation to simpler cases of waves in one-dimensional records. By this we mean the apparent waves extracted from functions (measured signals) with one-dimensional parameter, either in time or in space. These functions can be extracted from a photograph of the sea surface as, for example, the *instantaneous profile* along a line in some fixed horizontal direction on the sea, or can be obtained directly as a *record taken in time at a fixed position in space* as by means of a wave pole or distance meters. The *encountered sea*, another important one-dimensional record, can be collected at a moving point as by means of a ship-borne wave recorder.

To analyze collected wave data we need natural and operational definitions of an individual wave, its period, height, steepness, and possibly some other meaningful characteristic. There are several possible definitions of apparent wave, and here we shall concentrate mostly on zero down-crossing waves. Namely, the *apparent individual wave* at a fixed time or position is defined as the part of the record that falls between two consecutive down-crossings of the zero seaway level (the latter often more descriptively referred to as the still water level). For individual waves one can consider various natural characteristics, among them *apparent periods* and *apparent heights (amplitudes)*. The pictorial definitions of these two characteristics are given in Figure 3.1.

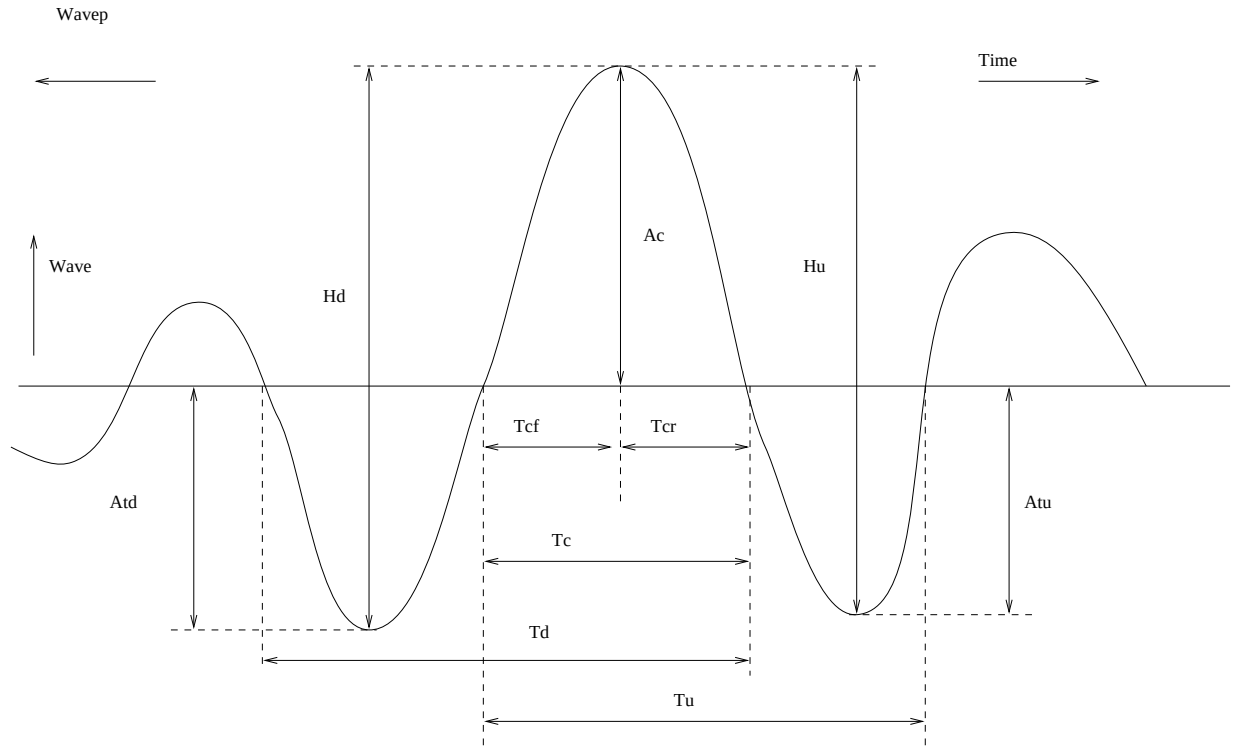


Figure 3.1: *Wave parameters, definitions. In the list below, the notation for the parameters used in our examples is given.*

The definitions of the most common wave characteristics are given in Table 3.1. In the WAFO toolbox the most important can be retrieved by the command `help wavedef`, `help periodef`, `help ampdef`, and `help crossdef`, producing the output in Section 3.5

Having precisely defined the characteristics of interest, one can extract their frequency (empirical) distributions from a typical sufficiently long record. For example, measurements of the

upcrossing wave		a wave between two successive mean level upcrossings
downcrossing wave		a wave between two successive mean level downcrossings
wave crest		the maximum value between a mean level upcrossing and the next downcrossing = the highest point of a wave
wave trough		the minimum value between a mean level downcrossing and the next upcrossing = the lowest point of a wave
crest front wave period	T_{cf}	time span from upcrossing to wave crest
crest rear wave period	T_{cr}	time from wave crest to downcrossing
crest period	T_c	time from mean level up- to downcrossing
trough period	T_t	time from mean level down- to upcrossing
upcrossing period	T_u	time between successive mean level upcrossings
downcrossing period	T_d	time between successive mean level downcrossings
crest-to-crest wave period	T_{cc}	time between successive wave crests
zero-downcrossing wave height	H_d	height between trough and following wave crest
crest amplitude	A_c	crest height above mean level
trough depth	A_t	through depth below mean level (note: $A_t > 0$)
upcrossing wave amplitude	H_u	crest-to-trough vertical distance
downcrossing wave amplitude	H_d	trough-to-crest vertical distance
wave steepness	S	Generic symbol for wave steepness
min-to-max wave period		time from local minimum to next local maximum
min-to-max wave amplitude		height between local minimum and the next local maximum
max-to-min wave period/amplitude		similar to min-to-max definitions

Table 3.1: *Wave characteristic definitions*

apparent period and height of waves could be taken over a sufficiently long observation time to form an empirical two-dimensional distribution. This distribution will represent some aspects of a given sea surface. Clearly, because of the irregularity of the sea, empirical frequencies will vary from record to record, however if the sea is in “steady” condition, which corresponds mathematically to the assumption that the observed random field is stationary and ergodic, their variability for sufficiently large records will be insignificant. Such limiting distributions (limiting with respect to observation time, for records measured in time, increasing without bound) are termed the *long-run distributions*. Obviously in real sea we seldom have a so long periods of ”steady” conditions that the limiting distribution will be reached. On average one may observe 400-500 waves per hour of measurements, while the stationary conditions may rest from 20 minutes to only few hours.

Despite of this, a fact that makes these long-run distributions particularly attractive is that they give probabilities of occurrence of waves that may not be observed in the short records but still are possible. Hence one can estimate the intensity of occurrence of waves with special properties and to extrapolate beyond the observed types of waves. What we shall be concerned with next is to show how to compute such distributional properties.

In the following sections we shall consider three different ways to obtain the wave characteristic probability densities (or distributions):

- To fit empirical distribution to the observed (or simulated) data in some parametric family of densities, and then relate the estimated parameters to some observed wave climate described by means of significant wave heigh and wave period. Algorithms to extract waves, estimate the densities and compute some simple statistics will be presented.
- To simplify the model for the sea surface into such degree that explicit computation of wave characteristic densities (in the simplified model) is possible. Some examples of proposed models from the literature will be given.
- To exactly compute the densities from the mathematical form of a random seaway. For zero-crossing waves there are explicit formulas for the densities of wave characteristics. The formulas are in principal infinite dimensional integrals that have to be computed numerically. In the toolbox there are efficient numerical algorithms to compute these integrals. The algorithms do not require any particular form of the spectrum to be used to model the sea surface. The method will be illustrated by computing densities of period, wavelength and amplitude, in many standard types of wave spectra.

3.2 Estimation of wave characteristics from data

In this section we shall extract the wave characteristics from a measured signal and then use non-parametric statistical methods to describe the data, i.e. empirical distributions, histograms, and kernel estimators. (In the last chapter of this Tutorial we presents some statistical tools to fit parametric models. That chapter is not included in this preliminary manuscript).

It is generally to be advised that, before analyzing sea wave characteristics, one should check the quality of the data by inspection and by the routine `findoutliers` used in Section 2.1. Then one usually should remove any present trend from the data. Trends could be due to tides or atmospheric pressure variations which affect the mean level. De-trending can be done using a WAFO functions `detrend` or `detrendma`.

Wave period

Example 1. (contd.) We begin with extracting the apparent waves and record their period. The signal `sea.dat` is recorded at 4 Hz sampling frequency. One of possible definition of a period is the distance between the consecutive wave crests. For this particular variable it may be convenient to have a higher resolution than 4 Hz and hence we shall interpolate signal to a denser grid. This will be obtained by giving an appropriate value to the variable `rate` which can be used as input to the WAFO-routine `dat2wa`. The following MATLAB code will return *crest2crest waveperiods*

T_{cc} in the variable `Tcrcr` and return the crest period T_c in `Tc`, i.e. the time from up-crossings to the following down-crossing.)

```
xx = load('sea.dat');
xx(:,2) = detrend(xx(:,2));
rate = 8;
Tcrcr = dat2wa(xx,0,'c2c','tw',rate);
Tc = dat2wa(xx,0,'u2d','tw',rate);
```

Next we shall use a kernel estimator (KDE) to estimate the probability density function (pdf) of the crest period and compare the resulting pdf with a histogram of the observed periods stored in `Tc`. In order to define a suitable scale for the density we first compute the mean and maximum of the observed crest periods.

```
mean(Tc)
max(Tc)
t = linspace(0.01,8,200);
L2 = 0;
ftc1 = kde(Tc,'epan',[],[],L2,t);
pdfplot(ftc1), hold on
whisto(Tc,[],[],1)
axis([0 8 0 0.5]), hold off
```

(The parameter `L2=0` is used internally in the function `kde`, and causes a logarithmic transformation of the data to ensure that the density is zero for negative values.)

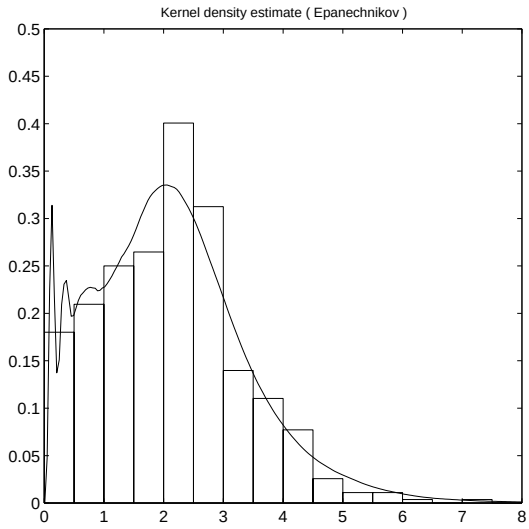


Figure 3.2: *Kernel estimate of the crest period density observed in* `sea.dat` *(solid line) compared with the histogram of the data.*

In Figure 3.2 we can sea that many short waves have been recorded (due to relatively high sampling frequency). The kernel estimate will be compared with the theoretically computed density in the last section of this chapter. □

Remark 3.1. Note that the program `kde` can be quite slow for large data sets. If a faster estimates of the pdf for the observations is preferred one can use `kdebin`, which is an approximation to the true kernel density estimator. An important input parameter in the program, that defines the degree of approximation, is `inc` which should be given a value between 100 and 500. (A value of `inc` below 50 gives fast execution times but can lead to inaccurate results.)

```
inc = 128;
ftc2 = kdebin(Tc,'epan',[],[],L2,inc);
```

□

Extreme waves – model check

We turn now to joint wave characteristics, e.g. the joint density of half period and crest height `(Tc,Ac)`, or waveheight and steepness `(Ac,S)`. The program `dat2speed` identifies waves and for each wave gives several wave characteristics (type `help dat2speed` for a list of computed variables).

We begin by examining profiles of waves having some special property, e.g. with high crests, or that are extremely steep.

Example 1. (contd.) The following MATLAB code will find a sequence of waves and their wave characteristics.

```
method = 0;
rate = 8;
[S, H, Ac, At, Tcf, Tcb, z_ind, yn] = ...
       dat2steep(xx,rate,method);
```

The first preliminary analysis of the wave data is to find the individual wave which is extreme by some specified criterion, e.g. the steepest or the highest wave etc. To do such an analysis one can use the function `spwaveplot(xx,ind)`, which plots waves in `xx` which are selected by the index variable `ind`. For example, let us look at the highest and the steepest waves.

```
[Smax indS]=max(S)
[Amax indA]=max(Ac)
spwaveplot(yn,[indA indS],'k.')
```

The two waves are shown in Figure 3.3 (a). The shape of the biggest wave reminds of the so called ”extreme” waves. In the following we shall examine whether this particular shape contradicts the assumption of a transformed Gaussian model for the sea.

This is done as follows. First we find the wave with the highest crest. Then we mark all positive values in that wave as missing. Next we reconstruct the signal, assuming the Gaussian model is valid, and compare the profile of the reconstructed wave with the actual measurements. Confidence bands for the reconstruction will be also plotted. In the previous chapter we have already used the program `reconstruct`, and here we shall need some additional output from the function, to be used to compute and plot the confidence bands.

```
inds1 = (5965:5974)';
Nsim = 10;
[y1, grec1, g2, test, tobs, mu1o, mu1oStd] = ...
       reconstruct(xx,inds1,Nsim);
spwaveplot(y1,indA-10)
hold on
plot(xx(inds1,1),xx(inds1,2),'+')
lamb = 2.;
muLstd = tranproc(mu1o-lamb*mu1oStd,fliplr(grec1));
muUstd = tranproc(mu1o+lamb*mu1oStd,fliplr(grec1));
plot (y1(inds1,1), [muLstd muUstd],'b-')
```

(Note that we have used the function `tranproc` instead of `gaus2dat`, since the last function require a two column matrix. Furthermore we have to use the index `indA-10` to identify the highest wave in `y1`. This is caused by the fact that interpolated signal `yn` has a few additional small waves that are not in `xx`.)

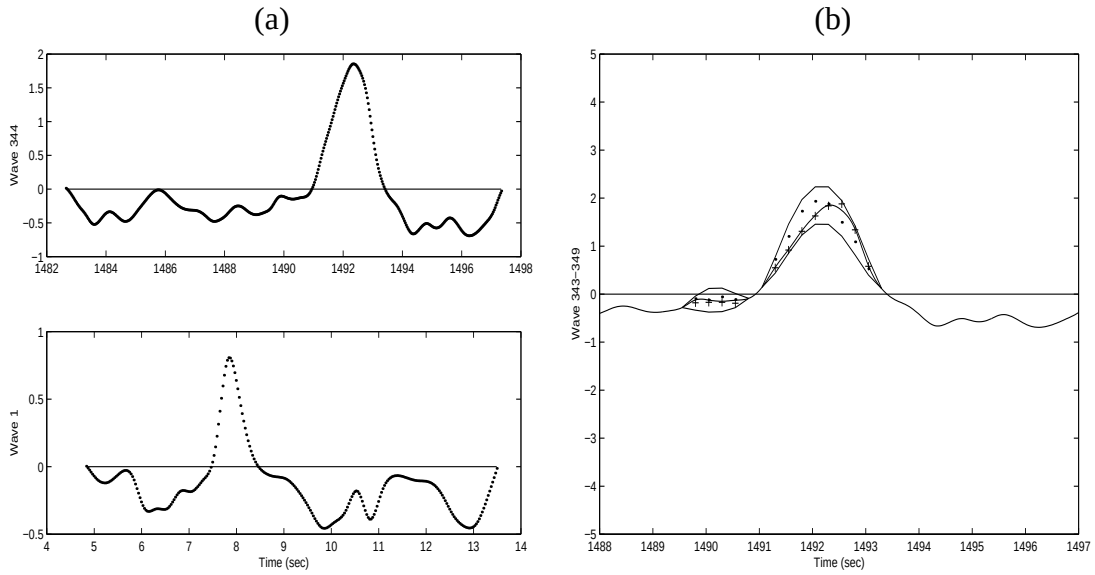


Figure 3.3: *(a): Two waves, the highest and the steepest, observed in* `sea.dat`. *(b): Crosses are observations removed from the highest wave, the reconstructed wave, using transformed Gaussian model is given by the middle solid line. Upper and lower curves give the confidence band defined as the conditional mean of the process plus minus two conditional standard deviations.*

In Figure 3.3 (b) the crosses are the removed values from the wave. The reconstructed wave, plotted by a solid line, is almost identical to the measured. (Observe that this is a simulated wave, using the transformed Gaussian model, and hence each time we execute the command the shape will change.) The confidence bands gives limits containing 95% of the simulated values, pointvise. From the figure we can deduce that ththis highest wave could have been even higher and that the height is determined by the particularly high values of the derivatives at the zero crossings which define the wave. The observed wave looks more asymmetric in time then the reconstructed one. Such asymmetry is unusual for the transformed Gaussian waves but not impossible. By executing the following commands we can see that actually the observed wave is close to the expected in transformed Gaussian model. We shall not investigate this question further in this tutorial.

```
clf
plot(xx(inds1,1),xx(inds1,2),'+'), hold on
mu = tranproc(mu1o,fliplr(grec1));
plot(y1(inds1,1), mu)
```

□

Crest height

We turn now to the kernel estimators of the crest height density. As it is well known that for Gaussian sea the tail of the density is well approximated by the Rayleigh distribution. Wand and

Jones (1995, Chap. 2.9) show that Gaussian distribution is one of the easiest distributions to obtain a good Kernel Density Estimate from. It is more difficult to find good estimates for distributions with skewness, kurtosis and multimodality. Here one can get help by transforming data. This can be done choosing different values of input `L2` into the program `kde`.

Example 1. (contd.) We shall continue with analysis of the crest height distribution. By letting `L2 = 0.6` we see that the normalplot of the transformed data is approximately linear. (Note: One should try out several different values for `L2`. It is also always good practise to try out several different values of the smoothing parameter; see the help text of `kde` and `kdebin` for further explanation)

```
L2 = 0.6;
wnormplot(Ac.^L2)
fac = kde(Ac,'epan',[],[],L2,linspace(0.01,3,200));
pdfplot(fac)
simpson(fac.x{1},fac.f)
```

The integral of the estimated density `fac` is 0.9675 but it should be one. Therefore, when we use the estimated density to compute different probabilities concerning the crest height the uncertity of the computed probability is at least 0.03. We suspect that the estimated density is erroneous for small amplitudes. In order to check this we compute the cumulative distribution using the following formula,

$$P(Ac \leq h) = 1 - \int_h^{+\infty} f_{Ac}(x)\, dx,$$

where $f_{Ac}(x)$ is the estimated probability density of Ac. For the pdf saved in `fac` the following MATLAB code gives an estimate of the cumulative distribution function (cdf) for crest height and compares it with the empirical distribution computed from data by means of function `empdistr`.

```
Fac = flipud(cumtrapz(fac.x{1},flipud(fac.f)));
Fac = [fac.x{1} 1-Fac];
Femp = empdistr(Ac,Fac);
axis([0 2 0 1])
```

□

Since a kernel density estimator KDE in principal is a smoothed histogram it is not very well suited to extrapolate the density to the region where no data are available, e.g. the heigh crests in such a case a model should be used. In WAFO there is a function `trraylpdf` that combines the nonparametric approach of KDE with a Rayleigh density. Simply, if the Rayleigh variable can be used to described the crests of Gaussian waves then a transformed Rayleig variable should be used for the crests of the transformed Gaussian waves. The method has several nice properties and will be described more in the following section. Here we just use it in order to compare with the nonparametric KDE method.

```
hold off
facr = trraylpdf(fac.x{1},'Ac',grec1);
Facr = cumtrapz(facr.x{1},facr.f);
hold on
plot(facr.x{1},Facr(:,2),'.')
axis([1.25 2.25 0.95 1])
```

Figure 3.4 (a) shows that our hypothesis that the `fac` pdf is slightly too low in the region of small crest seems to be correct. Next from Figure 3.4 (b) we can see that even the tail is resonably

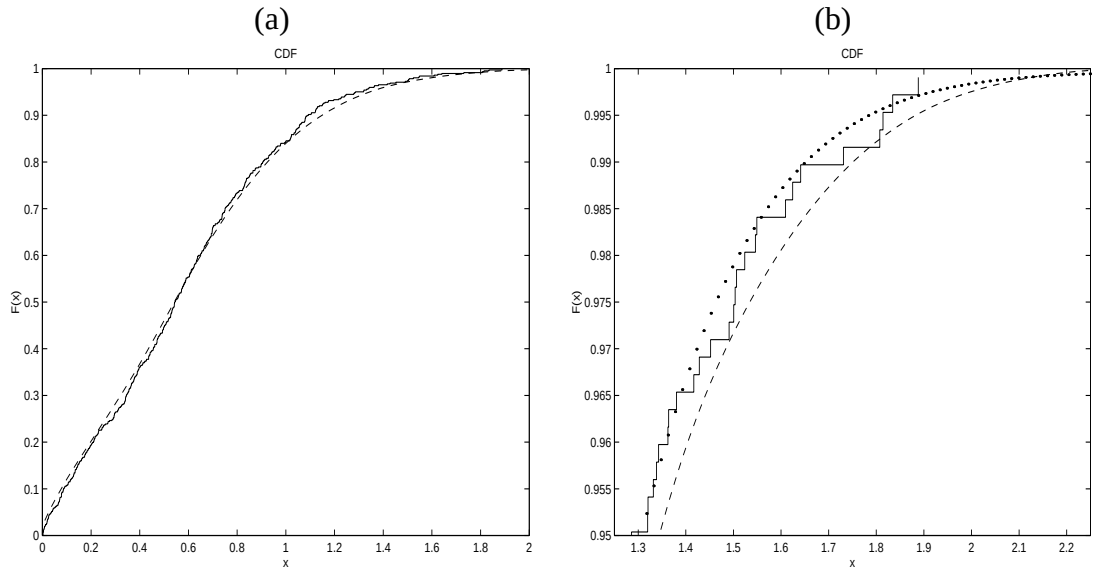


Figure 3.4: *(a) Comparison of the empirical distribution of the crest height with the cumulative distribution computed from the KDE estimator. (b) Zooming in on the tails of distributions in (a) together with the tail of the transformed Rayleigh approximation (dots) to the crest height distribution.*

modeled even if it is lighter than, i.e. gives smaller probabilities of high waves than, the one derived from the transformed Gaussian model.

Joint crest period and crest height distribution

We shall use the kernel density estimator to find a good estimator of the central part of the joint density of crest period and crest height. Usually, kernel density estimator gives poor estimates of the tail of the distribution, unless large amounts of data is available. However, a KDE gives qualitively good estimates in the regions of sufficient data i.e. in the main part of the distribution. This is good for visualization (`pdfplot`) and detecting modes, symmetries (anti-symmetry) of distributions.

Example 1. (contd.) The following command examines and plots the joint distribution of crest period and crest height in the data `sea.dat`.

```
L2 = 0.5;
hs = [];
inc = 256;
Tc = Tcf+Tcb;
fTcAc = kdebin([Tc Ac],'epan',hs,[],L2,inc);
fTcAc.labx={'Tc [s]'  'Ac [m]'}  % make labels for the plot
pdfplot(fTcAc)
hold on
plot(Tc,Ac,'k.')
hold off
```

In Figure 3.5 there are 544 pairs of crest period and height plotted. We can see that the kernel estimate describes the distribution of data quite well. It is also obvious that it can not be used to extrapolate outside the observation range. In the following section we shall compute the theoretical

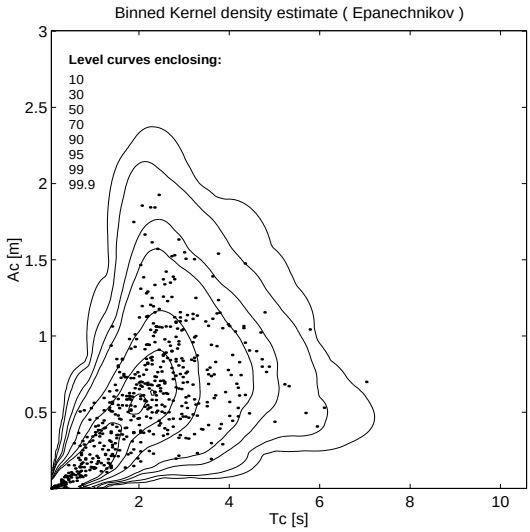


Figure 3.5: *Kernel estimate of the joint density of crest period* `Tc` *and crest height* `Ac` *in* `sea.dat` *compared with the observed data (dots). The countour lines are drawn in such a way that they contain specified (estimated) proportions of data.*

joint density of crest period and height from the transformed Gaussian model and compare with the KDE estimate. □

3.3 Explicit results - wave models

In this section we shall consider the Gassian sea. We assume that the reference level is zero and that the spectrum is known. We will present some of the explicit form results which are known and studied in the literature about wave characteristics. Some of them are exact, other are derived by simplification of the random functions describing the sea surface.

3.3.1 The average wave

For Gaussian waves the spectrum and the spectral moments contain exact information about the average behaviour of many of the wave characteristics. The WAFO programs `spec2char` and `spec2bw` can compute a long list of different wave characteristic parameters from the spectrum input. A list of available parameters can be found from `help spec2char`.

```
SPEC2CHAR Evaluates spectral characteristics and their variance

CALL: [ch r] = spec2char(S,fact,T)

      ch = vector of spectral characteristics
      r  = vector of the corresponding variances given T
      S  = spectral struct with angular frequency
    fact = vector with factor integers, see below. (default [1])
      T  = recording time (sec) (default 1200 sec = 20 min)

If input spectrum is of wave number type, output are factors for
```

```
corresponding 'k1D', else output are factors for 'freq'.
Input vector 'factors' correspondence:
   1 Hm0   = 4*sqrt(m0)          Significant wave height
   2 Tm01  = 2*pi*m0/m1          Mean wave period
   3 Tm02  = 2*pi*sqrt(m0/m2)    Mean zero-crossing period
   4 Tm24  = 2*pi*sqrt(m2/m4)    Mean period between maxima
   5 Tm_10 = 2*pi*m_1/m0         Energy period
   6 Tp    =                     Peak period
   7 Ss    = 2*pi*Hm0/(g*Tm02^2) Significant wave steepness
   8 Sp    = 2*pi*Hm0/(g*Tp^2)   Average wave steepness
   9 Ka    =                     Groupiness parameter
  10 Rs    =                     Quality control parameter
  11 Tp    = 2*pi*int S(w)^4 dw  Peak Period (more robust estimate)
             ------------------
             int w*S(w)^4 dw
Order of output is same as order in 'factors'
The variances are computed with a Taylor expansion technique
and is currently only available for factors 1,2 and 3.
```

Example 4. *(Simple wave characteristics obtained form spectral density)* We start by defining a JONSWAP spectrum, describing a sea state with `T_p = 10 s`, `H_{m_0} = 5 m`. Type `spec2mom` to see what spectrals moments are computed.

```
S = jonswap([],[5 10]);
[m  mt]= spec2mom(S,4,[],0);
```

The most basic information about waves is contained in the spectral moments. The variable `mt` now contains information about what kind of moments have been computed, in this case spectral moments up to order four ($m_0, \ldots, m_4$). Next, the irregularity factor α, significant wave height, zero crossing wave period, and peak period can be computed.

```
spec2bw(S)
[ch Sa2] = spec2char(S,[1  3])
```

The interesting feature of the program `spec2char` is that it also computes an estimate of the variance of the characteristics, given the length of observations (assuming the Gaussian sea), see [24], [60] and [64] for more detailed discussion. For example, for the JONSWAP Gaussian sea, the standard deviation of significant wave height estimated from 20 minutes of observations is approximately 0.25 meter. □

3.3.2 Explicit form approximations of wave characteristic densities

In the module `wavemodels` in WAFO, we have implemented some of the approximative models found in the literature. To get an overview of the routines in the module, type `help wavemodels`.

We will investigate three approximate models for the joint pdf of (T_c, A_c) (for the nomenclature, see the routines `perioddef` and `ampdef` in the module `docs`). Both of the functions need spectral moments as inputs.

Model by Longuet-Higgins

Longuet-Higgins derives his distribution by considering the joint distribution of the envelope amplitude and the time derivative of the envelope phase. The model is valid for narrow-band processes. It seams to give relatively accurate result for big waves, e.g. for waves with significant amplitudes.

The spectral width parameter[1] defined as

$$\nu = \frac{m_0 m_2}{m_1^2} - 1$$

appears in the model (for a narrow-band process, $\nu \approx 0$), given by

$$f^{\mathrm{LH}}_{T_c,A_c}(t,x) = c_{\mathrm{LH}} \left(\frac{x}{t}\right)^2 \exp\left\{-\frac{x^2}{8}[1+\nu^{-2}(1-t^{-1})^2]\right\},$$

where

$$c_{\mathrm{LH}} = \frac{1}{8}(2\pi)^{-1/2}\nu^{-1}[1+(1+\nu^2)^{-1/2}]^{-1}.$$

The density is calculated by the function `lh83pdf`.

Example 4. (contd.) For the Longuet-Higgins approximation we use the spectral moments just calculated.

```
t = linspace(0,15,100);
h = linspace(0,6,100);
flh = lh83pdf(t,h,[m(1),m(2),m(3)]);
```

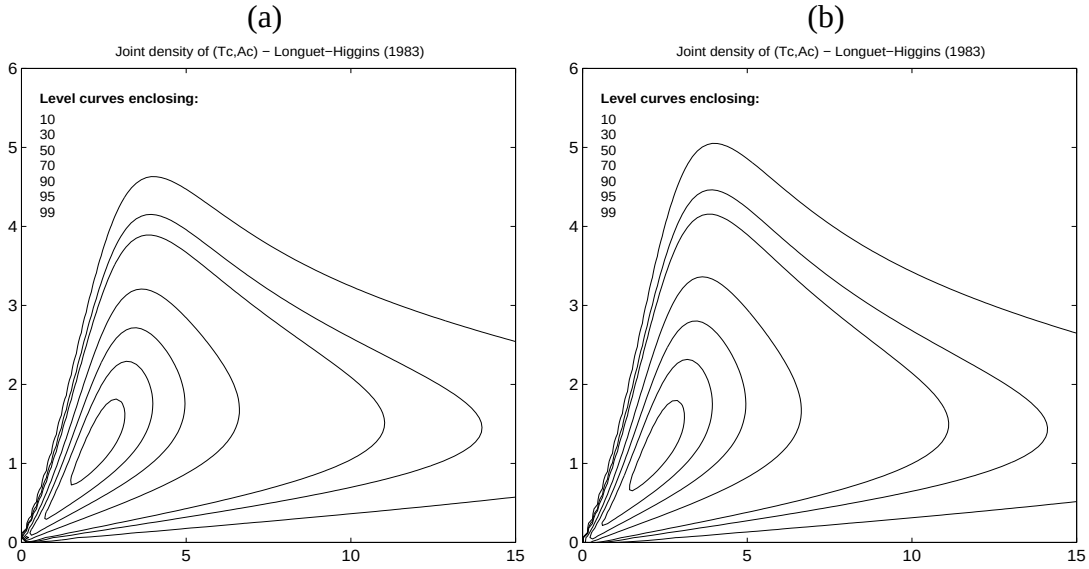


Figure 3.6: *Model by Longuet-Higgins for joint pdf of crest period T_c and crest height A_c. Spectrum: JONSWAP with $T_p = 10$ s, $H_{m_0} = 5$ m. (a) linear Gaussian sea, (b) transformed Gaussian sea.*

In WAFO we have modified the Longuet-Higgins density to be applicable for transformed Gaussian models. Following the examples from the previous chapter we compute the transformation proposed by Winterstein and combine it with the Longuet-Higgins model.

[1]The value of ν may be calculated by `spec2bw(S,'eps2')`

```
[sk, ku ]=spec2skew(S);
sa = sqrt(m(1));
gh = hermitetr([],[sa sk ku 0]);
flhg = lh83pdf(t,h,[m(1),m(2),m(3)],gh);
```

In Figure 3.6 the densities `flh` and `flhg` are compared. The countour lines are drawn in such a way that they contain predefined proportions of the total probability mass inside the countours. We can see that including some nonlinear effects gives somewhat higher waves for the JONSWAP spectrum. □

Model by Cavanié et al

Another explicit density for the crest height was propsed by Cavanié et al (1976). Here any positive local maximum is considered as a crest of a wave, then the second derivative (curvature) of the local maximum defines the wave period by means of a cosine function with the same height and curvature of its top.

The model uses the parameter ν and a bandwidth parameter[2] ϵ, defined by

$$\epsilon = \sqrt{1 - \frac{m_2^2}{m_0 m_4}};$$

hence, for a narrow-band process, $\epsilon \approx 0$. The Cavanié distribution is given by

$$f^{\mathrm{CA}}_{T_c,A_c}(t,x) = c_{\mathrm{CA}} \frac{x^2}{t^5} \exp\left\{ -\frac{x^2}{8\varepsilon^2 t^4} \left[\left(t^2 - \left(\frac{1-\varepsilon^2}{1+\nu^2} \right) \right)^2 + \beta^2 \left(\frac{1-\varepsilon^2}{1+\nu^2} \right) \right] \right\},$$

where

$$\begin{aligned} c_{\mathrm{CA}} &= \frac{1}{4}(1-\epsilon^2)(2\pi)^{-1/2}\epsilon^{-1}{\alpha_2}^{-1}(1+\nu^2)^{-2} \\ \alpha_2 &= \frac{1}{2}[1+(1-\epsilon^2)^{1/2}] \\ \beta &= \epsilon^2/(1-\epsilon^2) \end{aligned}$$

The density is computed by

```
t = linspace(0,10,100);
h = linspace(0,7,100);
fcav = cav76pdf(t,h,[m(1) m(2) m(3) m(5)],[]);
```

and a contour plot of the pdf is obtained by `pdfplot(fcav)`, cf. Figure 3.7.

Rayleigh approximation for wave crest height

There are several densities proposed in the literature to approximate the height of a wave crest or its amplitude. Some of them are programmed in WAFO; execute `help wavemodels` for a list of them. For Gaussian sea the most frequently used model is the Rayleigh density. The standardized Rayleigh variable R has the density given by $f(r) = r\exp(-r^2/2)$. It is well kown

[2] The value of ϵ may be calculated by `spec2bw(S,'eps4')`

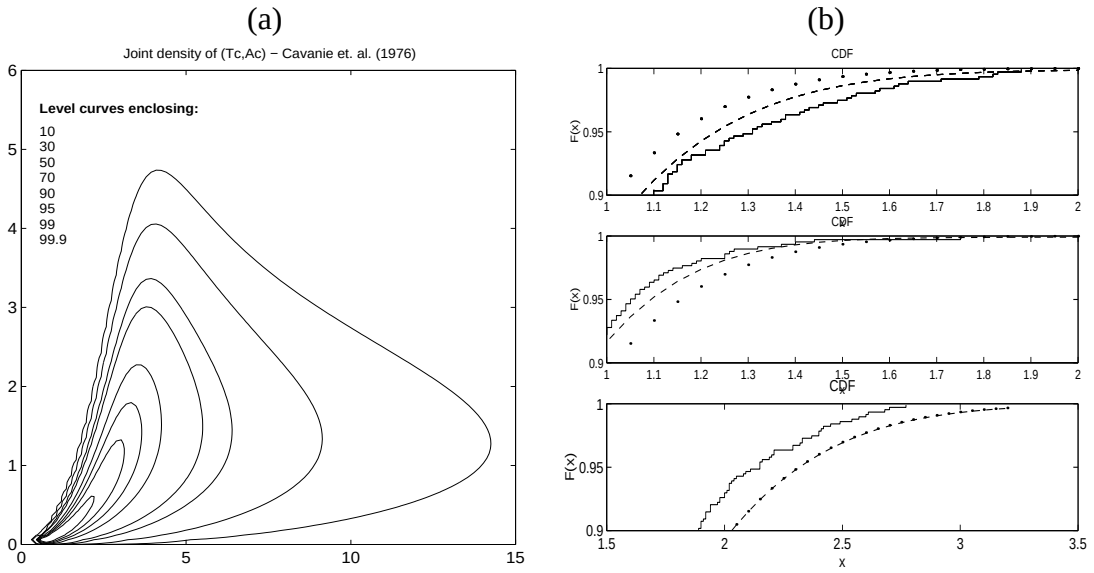


Figure 3.7: *(a) Contour lines of the joint density of crest period and crest height proposed by Cavanié et al, for Gaussian sea with JONSWAP spectrum ($T_p = 10$ s, $H_{m_0} = 5$ m). (b) The tail of the empirical distribution of crest height (top), trough height (middle) and amplitude (bottom) compared with Rayleigh approximation (dots) and transformed Rayleigh model with Hermite transformation.*

that for Gaussian sea the Rayleigh approximation works very well for high waves, and actually it is a conservative approximation since we have

$$P(A_c > h) \leq P(R > 4 * h/H_s) = e^{-8h^2/H_s^2},$$

see Rychlik and Leadbetter (1997). In the same paper it is shown that for any sea wave model with crossing intensity $\mu(u)$, one has $P(A_c > h) \leq \mu(u)/\mu(0)$. The approximation becomes more accurate as the level h increases.

The crossing intensity $\mu(u)$ is given by Rice's formula, Rice (1944), and it can be computed when the joint density of sea level $X(t)$ and its derivative $\dot{X}(t)$ is known,

$$\mu(u) = \int_0^{+\infty} z f_{X(t),\dot{X}(t)}(u, z)\, dz.$$

For Gaussian sea it can be computed explicitly

$$\mu(u) = \frac{1}{T_z} e^{-\frac{8u^2}{H_s^2}}.$$

For non-linear wave models with random Stoke's waves the crossing intensity has to be computed using numerical integration; see the Licentiate Thesis by Machado (2000).

Knowing the crossing intensity $\mu(u)$ one can compute the transformation g, by using the rouine `lc2tr`, such that the transformed Gaussian model has crossing intensity equal to $\mu(u)$. Consequently, we have that $P(A_c > h) \leq P(R > g(h)) = P(G(R) \leq h)$. The function `trraylpdf` computes the pdf of $G(R)$. (Obviously the function works for any transformation g.)

In previous examples we used the estimated crossing intensity to compute the transformation and then approximated the crest height density using the transformed Rayleigh variable. The accuracy of the approximation for the heigh crests in the data set `xx = sea.dat` was checked,

see Figure 3.4 (b). A more extensive study of the applicability of this approximation is done in Brodkorp et al. (2000).

Example 5. *(Rayleigh approximation of crest height from spectral density)* In this example we shall use a transformed Rayleigh approximation for crest height derived from the spectrum of sea. In order to check the accuracy of the approximations we shall use the estimated spectrum from the record `sea.dat`.

```
xx = load('sea.dat');
x = xx;
x(:,2) = detrend(x(:,2));
SS = dat2spec2(x);
[sk, ku, me, si ] = spec2skew(SS);
gh = hermitetr([],[si sk ku me]);
Hs = 4*si;
r = (0:0.05:1.1*Hs)';
fac_h = trraylpdf(r,'Ac',gh);
fat_h = trraylpdf(r,'At',gh);
h = (0:0.05:1.7*Hs)';
facat_h = trraylpdf(h,'AcAt',gh);
pdfplot(fac_h)
hold on
pdfplot(fat_h)
hold off
```

Next we shall compare the derived approximation with the observed crest heights in `x`. As before we could use the function `dat2steep` to find the crests. Here, for illustration only, we shall use `dat2tc` to find the crest heights `Ac` and trough depth `At`.

```
TC = dat2tc(xx, me); % Note: explanation for 'wdef' missing in help
tc = tp2mm(TC);
Ac = tc(:,2);
At = -tc(:,1);
AcAt = Ac+At;
```

Finally, the following commands will give the cumulative distributions for the computed densities.

```
Fac_h = [fac_h.x{1} cumtrapz(fac_h.x{1},fac_h.f)];
subplot(3,1,1)
Fac = empdistr(Ac,Fac_h);
hold on
plot(r,1-exp(-8*r.^2/Hs^2),'.')
axis([1. 2. 0.9 1])
Fat_h = [fat_h.x{1} cumtrapz(fat_h.x{1},fat_h.f)];
subplot(3,1,2)
Fat = empdistr(At,Fat_h);
hold on
plot(r,1-exp(-8*r.^2/Hs^2),'.')
axis([1. 2. 0.9 1])
Facat_h = [facat_h.x{1} cumtrapz(facat_h.x{1},facat_h.f)];
subplot(3,1,3)
Facat = empdistr(AcAt,Facat_h);
hold on
plot(r,1-exp(-2*r.^2/Hs^2),'.')
axis([1.5 3.5 0.9 1])
```

In Figure 3.7 (b) we can see some differences between the observed crest and trough distributions and the one obtained using the transformation `gh`. However, it still gives a much better approximation than using the standard Rayleigh approximation (dots). As it was shown before using the transformation computed from the crossing intensity, the transformed Rayleigh approach is giving a perfect fit. Finally, one can see that the Rayleigh and transformed Rayleigh variables give much too conservative approximations to the distribution of wave amplitude. □

3.4 Exact wave distributions in transformed Gaussian sea

In this section we shall demonstrate some functions for computation of exact probability densities, marginal and joint, of crest period `Tc`, crest length `Lc`, and crest height `Ac`. The same functions compute densities for trough period, lenght, and height, `Tt`, `Lt`, `At`, respectively. In WAFO there are also functions computing exact densities for other wave characteristics, which will not be presented here; make `help trgauss` to see a list of them all. The functions are the results of long time research at Lund University, see Podgórski et al, where a review of the historical development and the mathematical tools behind the algorithms are given.

```
help trgauss

  WAFO  Toolbox /trgauss
  Version 1.0.4   03-Jul-2000


  createpdf     - PDF class constructor
  dat2gaus      - Transforms  x  using the transformation  g.
  gaus2dat      - Transforms  xx  using the inverse of transformation  g.
  hermitetr     - Calculates the transformation g proposed by Winterstein
  initdata      - Initializes global constants used by the rind program
  iter          - Calculates a Markov matrix fmM  given a rainflow
                  matrix frfc;
  iter_mc       - Calculates a kernel f_xy of a MC given a rainflow
                  matrix f_rfc;
  ochitr        - Calculates the transformation g proposed by Ochi et al.
  ochitr2       - Calculates the transformation g proposed by Ochi et al.
  pdfplot       - Plot contents of pdf structures
  rind          - Computes  E[Jacobian*Indicator|Condition ]*f_{Xc}(xc(:,ix))
  spec2acat+    - Evaluates survival function R(h1,h2)=P(Ac>h1,At>h2).
  spec2acdf+    - Evaluates cdf of crests Ac (trough At) P(Ac<=h) (P(At<=h)).
  spec2cmat+    - Joint intensity matrix for (max,min)-, rainflow- and
                  (crest,trough)-cycles
  spec2mmtpdf+  - Calculates joint density of Maximum, minimum and period.
  spec2skew     - Estimates the moments of 2'nd order waves due to
                  Marthinsen and Winterstein
  spec2tccpdf+  - Evaluates densities of wave period Tcc, wave lenght Lcc.
  spec2thpdf    - Joint density of amplitude and period/wave-length
                  characteristics
  spec2tpdf     - Evaluates densities for crest-,trough-period, length.
  spec2tpdf2    - Evaluates densities for various wave periods or wave lengths
  spec2vhpdf    - Joint density of amplitude and crest front
                  velocity Vcf=Ac/Tcf
  trangood      - Makes a version  f  of the transformation  ff  that is
```

```
tranproc      - Transforms process  x  and up to four derivatives
trraylpdf     - Calculates transformed Rayleigh approximation for amplitudes.
```

3.4.1 Density of crest period, crest length or encountered crest period

All of the three densities can be computed using the function `spec2tpdf`. The function computes also the densities for waves with crest above a specified height `h`. This is a useful option allowing computation of the probability that a crest is higher than a specified threshold. It can also be used to provide information about the distribution of the period (lenght) of such high waves; see Brodkorb et al (2000) for detailed presentation.

The function `spec2tpdf` performs all necessary transformations scalings etc. making it very flexible. It handles different spectra as inputs. Which kind of the density is computed (output) is defined by the variable `def` that takes values `'Tc'` for crest period, `'Lc'` for crest length, `'Tt'` for trough period and `'Lt'` for trough length. The transformation is only affecting the value of the still water level `u` and the treshold `h`. The function `spec2tpdf` allows any value for the still water level; if `u` it is not equal to the most frequently crossed level then the densities of `Tc` and `Tt` are not identical.

Example 6. *(Crest period and crest length)* We start by defining a frequency spectrum, $S(\omega)$, which was used in the Introduction; we choose a Torsethaugen spectrum with parameters $H_{m_0} = 6$ m, $T_p = 8$ s, describing significant wave height and primary peak period, respectively. The energy is divided between two peaks, corresponding to contributions from wind and swell. We shall also use the two directional spectra from the Introduction.

```
S1 = torsethaugen([],[6 8],1);
D1 = spreading(101,'cos',pi/2,[15],[],0);
D12 = spreading(101,'cos',0,[15],S1.w,1);
SD1 = mkdspec(S1,D1);
SD12 = mkdspec(S1,D12);
```

We begin with the density of crest period, which (obviously) is identical for all three spectra `S1`, `SD1`, and `SD12`. The computed density is a result of a numerical integration of the theoretically derived formula. The algorithm gives upper (and if requested lower bound too) bound for the density. Consequently, if the integral of the computed density, over all periods, is close to one it implies that the density is computed with high accuracy.

```
f_tc = spec2tpdf(S1,[],'Tc',[0 11 56],[],4);
pdfplot(f_tc)
simpson(f_tc.x{1},f_tc.f)
```

The crest period density is shown in Figure 3.8 (a). The integral of the density `f_tc` computed using the function `simpson` is 1.0012 showing the accuracy of the approximation. The computation time is 105 seconds on a PC, Pentium 450 Mhz. The plot of the density is shown in Figure 3.8. The computation time depends on the required accuracy and how broad banded the spectrum is. For example, the same accuracy is achieved for JONSWAP spectrum, used in the previous section, in less than 5 seconds. The computation time increases if there is a considerable probability for long waves with low crests.

We then turn to the density of crest length for the Torsethaugen spectrum. It can be computed using the same function `spec2tpdf`, we just change the input `'Tc'` to `'Lc'`.

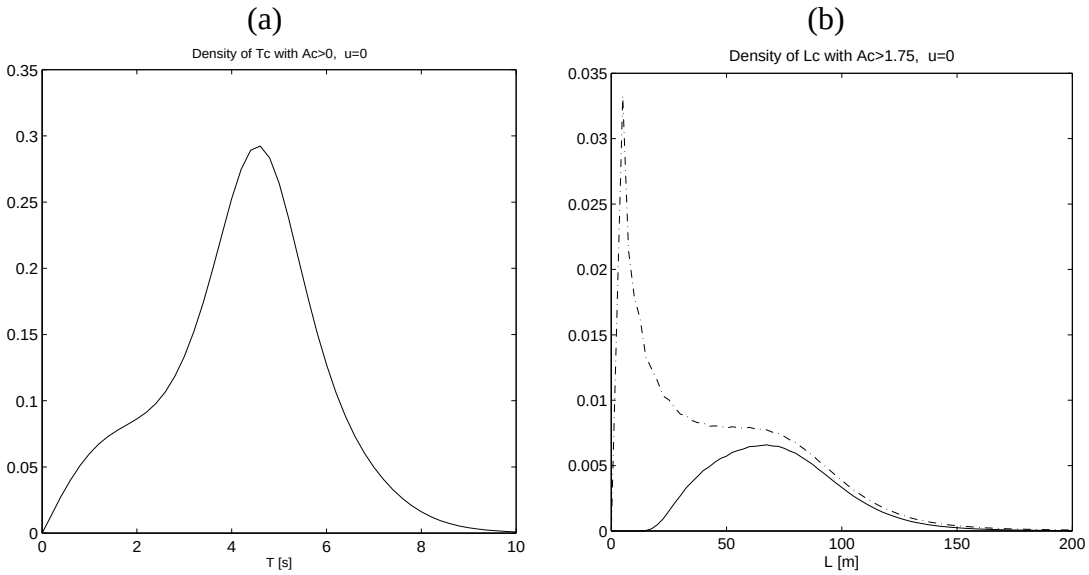


Figure 3.8: *(a) The density of crest period* `Tc`*, (b) Densities of crest length (dashed dotted line) compared to the density of crest length for waves with crest above 1.5 [m] (one standard deviation above the still water level, for Gaussian sea with the Torsethaugen spectrum.*

```
f_Lc = spec2tpdf(S1,[],'Lc',[0 200 81],[],5);
pdfplot(f_Lc,'-.')
```

The crest length density has a sharp peak for very short waves – the wave-number spectrum is much more broad banded that the frequency spectrum. However, the short waves have small crests and should be considered as 'noise' rather than as apparent waves. Consequently we may wish to compute the proportion of waves that has crest higher than one standard deviation ($H_s/4$) = 1.5 [m] and give the density of the crest length for these waves.

```
f_Lc_1 = spec2tpdf(S1,[],'Lc',[0 200 81],1.5,5);
hold on
pdfplot(f_Lc_1)
```

The result is presented in Figure 3.8 (b) and we can see that all short waves were small. The proprtion of waves with crests above 1.5 [m] (one standard deviation) is computed next.

```
simpson(f_Lc.x{1},f_Lc.f)
simpson(f_Lc_1.x{1},f_Lc_1.f)
```

As we can see more than half of the waves are small, more precisely 46% of the waves have crests above 1.5 [m].

We finish this example by considering the Torsethaugen spectrum with the two different spreading functions `SD1` and `SD12`. In Figure 1.5 we have presented simulations of the sea surfaces with the spertrums. From the figure we expect that the two crest length distributions should be different. (Obviously the crest period densities are identical). In the directional sea we have to define the azimuth of the line for which the crest length should be computed (the default value is zero). Now the directional spectra `SD1` and `SD12` has different main wave directions, 90^o, 0^o degrees, respectively, and hence we shall choose different azimuths for the spectra. More precisely for both cases we shall consider heading waves, this is achieved using function `spec2spec`.

```
f_Lc_d1 = spec2tpdf(spec2spec(SD1,'rotdir',pi/2),[],...
          'Lc',[0 300 121],[],5);
pdfplot(f_Lc_d1,'-.')
hold on
f_Lc_d12 = spec2tpdf(SD12,[],'Lc',[0 200 81],[],5);
pdfplot(f_Lc_d12)
hold off
```

(The last input in `spec2tpdf`, also called `nit` is defining the dimensionality of the computed integral. Higher `nit` is required to get good approximation for long waves, but that also takes long computing time. With low values of this parameter we obtain useful approximations that are computed in 10 seconds or so.)

As expected, after examination of Figure 1.5, the crest lenght for the two directional spectra are different. The sea with frequency dependent spreading seems to be more irregular. We can see in Figuren 3.9 that waves are only slightly longer than the waves in unidirectional sea but the crest lenght of both seas are much shorter than for frequency independent spreading. □

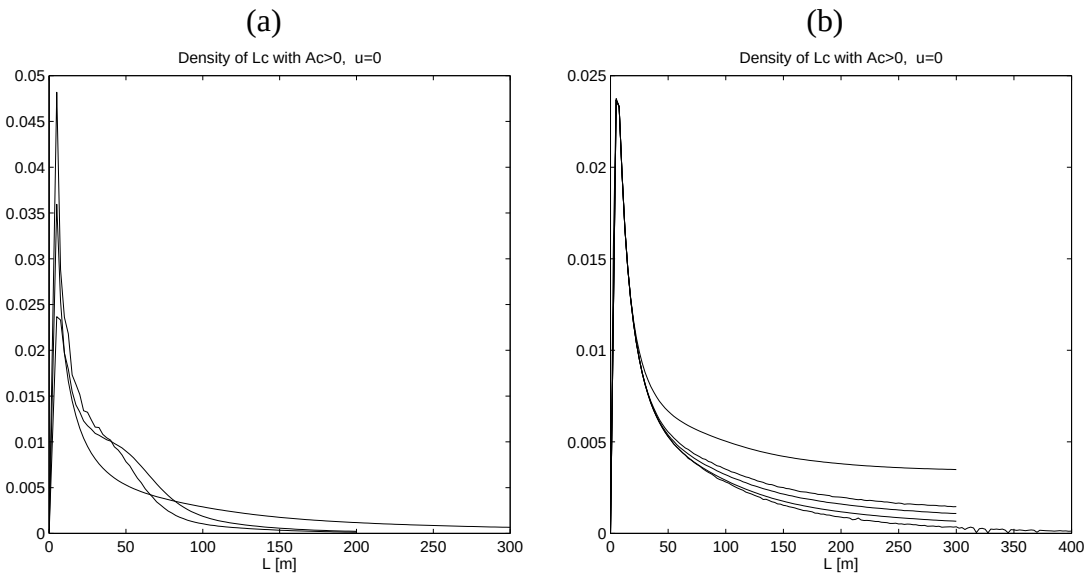


Figure 3.9: *(a) Comparison of crest length densities, heading waves, for unidirectional sea with Torsethaugen spectrum (shortest waves) and two different spreading functions the frequency independent spreading (longest waves), the frequency dependent spreading (intermegiate waves). (b) A sequence of approxiamtions of the crest lenght for the directional spectrum with frequency independent spreading is presented. The density with range* $(0, 400)$ *is computed with negative* `nit`.

Numerical accuracy

We finish this section with some comments about the program `spec2tpdf`, which is a MATLAB interface to a FORTRAN 95 program. All programs computing ecact densities of different wave characteristics can be reformulated in such a way that the density is written as a certain multidimensional integral of a function of Gaussian variables. This integral is computed using a FORTRAN

module called RIND. There is also a MATLAB interface called `rind` which can be used to test programs for new wave characteristics (before writting a more optimal code in FORTRAN).

An example is a function `spec2tpdf2` which uses the program `rind`. The program is slower than the function `spec2tpdf` (and does not have an option that allows to choose waves with crest above some level) but on the other hand it is easier to use for experimentation, and it can also be used to learn how to create own programs.

In the examples presented in this section the last input into `spec2tpdf` was equal to 5. In the help for `spec2tpdf` is called `nit` and besides the input `speed` it is the main tool to control the accuracy of the computation. Using program `simpson` we can deduce that the density for crest length above 250 meters in `f_Lc_d1` is just slightly too large. In order to increase the accuracy of the computed density one could increaae `nit` to six, but then the computation time probably becomes un-acceptably long for a laptop computer.

The program also allows negative values for `nit`, values that switches to another technique (based on importance sampling) to integrate Gaussian functions, see Brodkorb (2000) for a review of different methods. (The important other references are [48], [3], [15].) Although the method is based on simulation the accuracy is still controlled. If the number of simulation is too small to achive the required accuracy the program gives an error statement with an estimate of the possible error in the computed density. Different negative `nit` values, representing different integration method and error messages with predicted errors, can be used as indicators to switch the method. The method with positive `nit` values is very reliable and has been tested on different wave problems since 14 years (the first version was already used in [45]).

The integration methods corresponding to negative `nit` values are still under tests and modifications. However, as will be shown next, those are often much faster and more accurate in cases when the previous method is going into troubles with too long execution times.

Example 6. (contd.) We shall exemplify the use of the parameter `NIT` by computing the crest length density for the directional spectrum with frequency independent spreading. We shall also use the slower program `spec2tpdf2` for illustration.

```
f_Lc_d1_5 = spec2tpdf(spec2spec(SD1,'rotdir',pi/2),[],...
            'Lc',[0 300 121],[],5);
f_Lc_d1_3 = spec2tpdf(spec2spec(SD1,'rotdir',pi/2),[],...
            'Lc',[0 300 121],[],3);
f_Lc_d1_2 = spec2tpdf(spec2spec(SD1,'rotdir',pi/2),[],...
            'Lc',[0 300 121],[],2);
f_Lc_d1_0 = spec2tpdf(spec2spec(SD1,'rotdir',pi/2),[],...
            'Lc',[0 300 121],[],0);
f_Lc_d1_n4 = spec2tpdf2(spec2spec(SD1,'rotdir',pi/2),[],...
             'Lc',[0 400 161],-4);
pdfplot(f_Lc_d1_5)
hold on
pdfplot(f_Lc_d1_2)
pdfplot(f_Lc_d1_0)
pdfplot(f_Lc_d1_n4)
simpson(f_Lc_d1_n4.x{1},f_Lc_d1_n4.f)
```

The execution time was 1 hour 25 minutes, 3 minutes 20 seconds, 40 seconds, 8 seconds, and 10 minutes, respectively. The total probability mass of `f_Lc_d1_n4` is 0.991, which is suprisingly close to one, considering the complicated spectrum. In Figure 3.9 (b) the different approximation are presented and we can see how with the increasing positive `nit` the density decreases. The

negative `nit` involves some random number integration methods, but we can hardly see that the computed density is actually a random function. Most of problems are less numerical demanding and `nit=2` suffices, here clearly the negative `nit` is preferable. □

3.4.2 Density of wave period, wave length or encountered wave period

In the previous sections we have considered the densities of crest period `Tc` and trough period `Tt`. We could also limit the population to such waves that the crest (trough) amplitude are above some predescribed treshold.

The wave period `Tcc = Tc+Tt` can be computed using the function `spec2tccpdf`, giving the probability density for the distance between upcrossings of the still water level `u`. The wave length `Lcc` or encounterd wave period can also be computed using `spec2tccpdf`, with just a few inputs to be modified. Hence, these variables shall not be discussed here any more. The computations using `spec2tccpdf` are slower than using `spec2tpdf`, since one needs to compute the joint density of `Tc` and `Tt` and then change variable (integrate the convolution) to get `Tcc = Tc+Tt`. See also the discussion in the remark in the previous section about speed of programs. In addition to the methods to reduce computation mentioned in the remark, one of the best methods to speed speed up computations is to cut off high frequencies in the spectrum.

The syntax of `spec2tccpdf` is almost identical to that of `spec2tpdf`, and hence we limit ourselves to a few examples. In order to be able to make comparisons with observations `sea.dat` we shall use the estimated spectrum SS, see Example 1.

Example 7. *(Crest period of high-crest waves)* In this example we will compute the wave period density for waves with significant crest, i.e. with `Ac > Hs/2`. We shall compare it with the density of crest periods for the same type of waves. By integrating both densitities we obtain the proportion of waves with significant crest. These two numbers should be the same, but for numerical reasons, they will usually differ somewhat. The difference will be a measure of the accuracy of the computation of the convolution `Tcc = Tc+Tt`. We can also compare the calculated proportion of significant crest with the proportion observed in data and with linear approximation (Rayleigh model). Finally we estimate the density using KDE from data and compare to the theoretically computed one, based on the transformed Gaussian model. We finish the example with an even more intresting case, the density of wave period of waves with both significant crest and significant trough, i.e. really big waves.

For completeness we again estimate the transformation and find wave characteristics in the signal. The estimated and computed densities for the crest period T_c are almost identical, see Figure 3.10 (a).

```
xx = load('sea.dat');
x = xx;
x(:,2) = detrend(x(:,2));
SS = dat2spec2(x);
si = sqrt(spec2mom(SS,1));
SS.tr = dat2tr(x);
Hs = 4*si
method = 0;
rate = 2;
[S, H, Ac, At, Tcf, Tcb, z_ind, yn] = dat2steep(x,rate,method);
t = linspace(0.01,8,200);
L2 = 0;
```

```
ftc1 = kde(Tc,'epan',[],[],L2,t);
pdfplot(ftc1)
hold on
f_t = spec2tpdf(SS,[],'Tc',[0 8 81],0,4);
simpson(f_t.x{1},f_t.f)
pdfplot(f_t,'-.')
hold off
```

We next consider computation of the density of crest period and wave period, but now for waves with significant crest height, i.e. waves for which $A_c > H_s/2$. We start with crest period. In the following call to `spec2tpdf` the restriction to $A_c > H_s/2$ is indicated by the argument `[Hs/2]`.

```
f_t2 = spec2tpdf(SS,[],'Tc',[0 8 81],[Hs/2],4);
Pemp = sum(Ac>Hs/2)/sum(Ac>0)
simpson(f_t2.x{1},f_t2.f)
index = find(Ac>Hs/2);
ftc1 = kde(Tc(index),'epan',[],[],L2,t);
ftc1.f = Pemp*ftc1.f;
pdfplot(ftc1)
hold on
pdfplot(f_t2,'-.')
hold off
```

The observed frequency of significant crests is 0.1788 which is remarcably close to the theoretically computed value 0.1800. (Observe that the Rayleigh approximation would give a probability equal to 0.1353. This is not suprising since crests in non-Gaussian sea tend to be higher than those in Gaussian sea.) Clearly, by canging the input `Hs/2` to any other fixed level `h`, say, and integrating the resulting density we obtain the approximation of the probability $P(Ac > h)$.

If h is a vector then it is more efficient to use a program `spec2Acdf` to computes $P(Ac \leq h)$. However, before using the program it is important to use first `spec2tp` and check that the computed density integrates to one. If not, the inputs `param` and `nit` have to be changed.

Observe that in this section we are analysing apparent waves in time. If the input `'Tc'` in `spec2tpdf` is replaced by `'Lc'` then we would consider waves in space and the proportion of significant crest would probably be different.

The computed crest period density for wave with $A_c > H_s/2$ agrees also with the estimated distribution, obtained by using KDE, from data, see Figure 3.10 (b). □

Example 8. *(Wave period for high-crest waves)* We turn now to the more difficult problem of wave period density with wave with significant crest height. As mentioned, this differs from Example 7 in that it involves the distribution of the sum $T_c + T_t$ of two dependent random variables, with the same marginal distribution. Since the computations need to be done with high accuracy (the computed density is different for the two densities, the wave period density and wave period density for waves with crest below a given treshold, see [8] for more detailed discussion), we need to use a high `nit` value, so that the total sum of the density is 0.18. We begin with negative `nit` which gives faster results very close to the true density.

```
f_tcc2 = spec2tccpdf(SS,[],'t>',[0 12 61],[Hs/2],[0],-1);
simpson(f_tcc2.x{1},f_tcc2.f)
f_tcc3 = spec2tccpdf(SS,[],'t>',[0 12 61],[Hs/2],[0],3,5);
simpson(f_tcc3.x{1},f_tcc3.f)
pdfplot(f_tcc2,'-.')
```

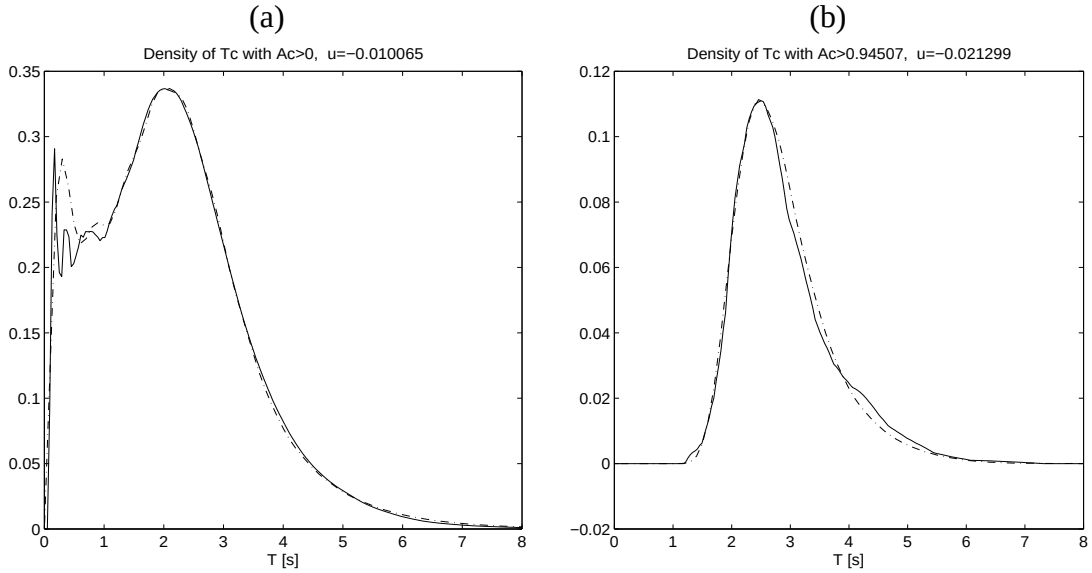


Figure 3.10: *(a) Estimated density (KDE) of crest periods in* `sea.dat` *(solid line) compared with theoretically computed using* `spec2tpdf` *(dashed line). (b) The same for the waves with significant crest, i.e.* `Ac>Hs/2`.

```
hold on
pdfplot(f_tcc3)
hold off
```

The integral of the density `f_tcc2` is 0.1789 which agrees with the previously computed. However the execution time was 40 minutes. We have checked the program with `nit=3` and the integral was 0.1664 (execution time 2 hours 30 minutes), indicating that the `nit` has to be increased. The densities are shown in Figures 3.11 (a). We can see that the density computed using `nit=-1` (dashed-dotted line) is slightly wiggly (it is a random function with very small variance), and errors compensate each other giving almost perfect total probability mass. Note that another call of the program would give slightly different values and the total mass would also be changed. One example gave the the value 0.1776.

Finally, we shall consider the case of waves with both significant crest and significant trough higher then `Hs/2` [m]. We first estimate the probability of such waves in the data.

```
[TC tc_ind v_ind] = dat2tc(yn,[],'dw');
N = length(tc_ind);
t_ind = tc_ind(1:2:N);
c_ind = tc_ind(2:2:N);
Pemp = sum(yn(t_ind,2)<-Hs/2 & yn(c_ind,2)>Hs/2)/length(t_ind)
ind = find(yn(t_ind,2)<-Hs/2 & yn(c_ind,2)>Hs/2);
spwaveplot(yn,ind(2:4))
Tcc = yn(v_ind(1+2*ind),1)-yn(v_ind(1+2*(ind-1)),1);
t = linspace(0.01,14,200);
L2 = 0;
ftcc1 = kde(Tcc,'epan',[],[],L2,t);
ftcc1.f = Pemp*ftcc1.f;
pdfplot(ftcc1,'-.')
```

The probability is estimated to be 0.0368, which is slightly higher than what we could expect if high crests and low troughs occur independently of each other (probability would then be less

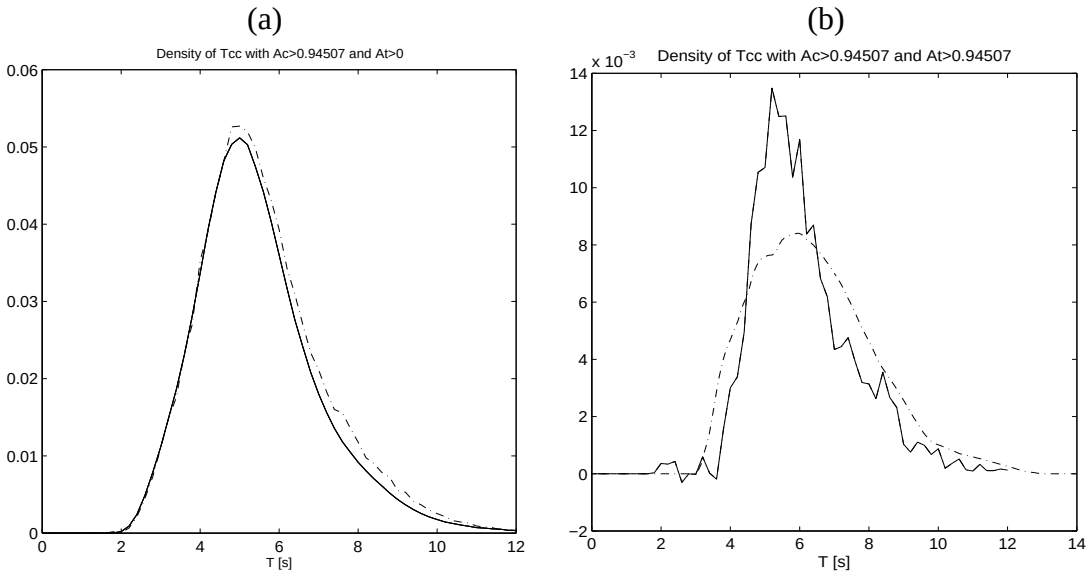


Figure 3.11: *(a) Densities of period* `Tcc` *for waves with significant crest in the transformed Gaussian model of the sea recorded in* `sea.dat` *computed with different degree of accuracy; (dashed dotted line)* `nit=-1`*; the two solid lines are computed for* `nit=3,4`*. (b) Densities of period* `Tcc` *for waves with significant crest and trough in the same model (dashed dotted line)* `nit=-1`*; the solid line is estimated from the data with KDE.*

than 0.025). We turn now to computation of the probability using `spec2tccpdf` with `nit=-1`. However, we are here in a situation when the error in computations is of the order 10^{-3}, which is comparable to the values of the density itself. Hence the computed function will look very noisy.

```
f_tcc22_1 = spec2tccpdf(SS,[],'t>',[0 12 61],[Hs/2],[Hs/2],-1);
simpson(f_tcc22_1.x{1},f_tcc22_1.f)
hold on
pdfplot(f_tcc22_1)
hold off
```

The computed probability is 0.0348 which is well in agreement with the estimated number. In Figure 3.11 (b) we see the density of wave period for these big waves. Those are well concentrated around the mean value. It is compared with KDE estimator. We have not tried to tune up the estimator that is based only on 20 values and hardly can be considered as accurate. However the agreement with the computed density is good. Next one could compute the same density using `nit=4` but we go instead to some new problems. □

3.4.3 Joint density of crest period and crest height

In this section we shall present programs for joint characteristics of apparent waves. We shall be mostly concerned with crest period, crest position, and crest height. Since we also want to compare the theoretically derived densities with observations we wish to study a longer record of measurements than we did in the previous section. By doing so we will have more reliable statistical estimates of the densities, but on the other hand we face the problem that the sea state can change during the measured period of time – the process is simply not stationary.

The data we are choosing is from Gullfaks C platform, see Figure 3.12 (a); `help gfaksr89` gives a detailed description of the data and `help northsea` for the instructions how the map showing location of the measurements place was drawn.

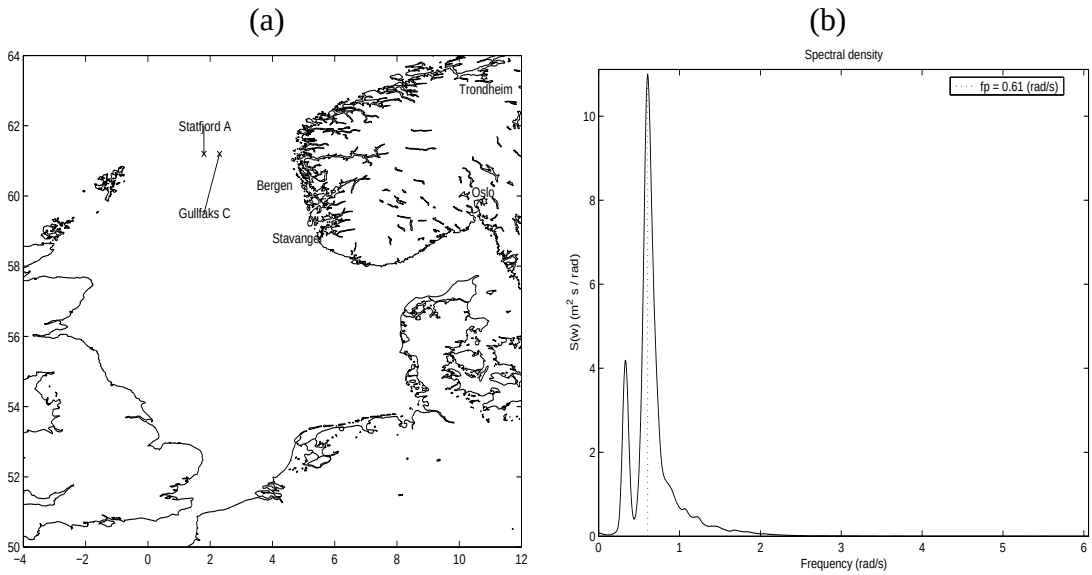


Figure 3.12: *Location of Gullfaks C platform (a). The estimated spectrum (b).*

WARNING: In the following examples we run the programs with maximum accuracy and hence we have long execution times. Usually one should use simpler and faster approximations at first experiments with complicated distributions. When one is satisfied with the results, one should compute the densities with the desired high accuracy. For testing own problems we recommend to start execution of programs with input parameter `speed=9,8` (maximal speed is 9, the default is 4) and `nit=0,1` (default is 2). These choices will produce fast but still useful approximations.

Example 9. *(Some preliminary analysis of the data)* We begin with loading the data, estimating spectrum, finding the transformation g, and checking crest period density. Observe that the data is sampled with 2.5 [Hz], what may cause some interpolation errors in the estimated densities.

```
yy = load('gfaksr89.dat');
SS = dat2spec(yy);
si = sqrt(spec2mom(SS,1));
SS.tr = dat2tr(yy);
Hs = 4*si
v = gaus2dat([0 0],SS.tr);
v = v(2)
```

The spectrum has two peaks, see Figure 3.12 (b). We are not checking different options to estimate the spectrum, but use the default parameters.

We shall now extract some simple wave characteristics `Tc,Tt,Tcf,Ac,At`. All these are column vectors containing crest period, trough period, position of crest, crest height and trough height, respectively. All vectors are ordered by number of a wave, i.e. all vectors contain characteristic of the i'th wave in their position `i` .

```
[TC tc_ind v_ind] = dat2tc(yy,v,'dw');
N = length(tc_ind);
t_ind = tc_ind(1:2:N);
c_ind = tc_ind(2:2:N);
v_ind_d = v_ind(1:2:N+1);
v_ind_u = v_ind(2:2:N+1);
T_d = yy(v_ind_d,1)- yy(v_ind_d,2)* ...
     (yy(2,1)-yy(1,1))./(yy(v_ind_d+1,2)-yy(v_ind_d,2));
T_u = yy(v_ind_u,1)- yy(v_ind_u,2)* ...
     (yy(2,1)-yy(1,1))./(yy(v_ind_u+1,2)-yy(v_ind_u,2));
Tc = T_d(2:end)-T_u(1:end);
Tt = T_u(1:end)-T_d(1:end-1);
Tcf = yy(c_ind,1)-T_u;
Ac = yy(c_ind,2)-v;
At = v-yy(t_ind,2);
```

We then turn now to computation of crest period density and compare it with that observed in data.

```
t = linspace(0.01,15,200);
L2 = 0;
clf
ftc1 = kde(Tc,'epan',[0.25],[],L2,t);
ftt1 = kde(Tt,'epan',[0.25],[],L2,t);
pdfplot(ftt1,'k')
hold on
pdfplot(ftc1,'k-.')
f_tc4 = spec2tpdf(SS,[],'Tc',[0 12 81],0,4,5);
f_tc = spec2tpdf(SS,[],'Tc',[0 12 81],0,-1);
pdfplot(f_tc,'b')
hold off
```

We do not present the graphical result for this computations but simply comment that the agreement between these three densities is very good, except for observed long waves, which have somewhat longer periods (about 0.25 s) than theoretically computed. It is not much for a signal with 2.5 Hz sampling frequency. There is also the possibility that the swell peak in the spectrum is too much smoothed. □

We turn now to the joint density for the variables which describe the wave crest `Tc,Tcf,Ac`. We shall estimate the densities from the observations and compute them from for the transformed Gaussian process with estimated spectrum and the transformation using function the WAFO function `spec2thpdf`. This function computes many joint characteristics of the half wave, i.e. the part of the signal between the consecutive crossings of a still water level – most of them are simply functions of the tripple `Tc,Tcf,Ac`. (Execute `help spec2thpdf` for a complete list).

In a special case ,when the so called crest velocity is of interest, `Vcf=Ac/Tcf`, the joint density of `Vcf,Ac` is computed by the program `spec2vhpdf`, which is a simplified and modified `spec2thpdf` program.

Example 10. *(Joint characteristics of a half wave - position and height of a crest for a wave with given period)* We shall first consider crest period, i.e. consider only waves with crest period `Tc` $\approx$ 4.5 seconds. Obviously the position of the crest of such waves is not constant, but varies from wave to wave. The following commands estimates the density of crest position and height for waves with `Tc` $\approx$ 4.5 seconds.

```
clf
ind = find(4.4<Tc & Tc<4.6);
f_AcTcf = kde([Tcf(ind) Ac(ind)],'epan',[],[],0.5);
plot(Tcf(ind), Ac(ind),'.');
hold on
pdfplot(f_AcTcf)
```

Next, we compare the observed distribution with the theoretically computed joint density of `Tc, Tcf, Ac` for a fixed value of `Tc`. By this we mean that if we integrate the result we shall obtain the value of the density. Note that `Tc` can be computed using the program `spec2tpdf`.

```
f_tcfac1 = spec2thpdf(SS,[],'TcfAc',[4.5 4.5 46],[0:0.25:8],-1);
simpson(f_tcfac1.x{1},simpson(f_tcfac1.x{2},f_tcfac1.f,1))
f_tcfac2=spec2thpdf(SS,[],'TcfAc',[4.5 4.5 46],[0:0.25:8],2);
simpson(f_tcfac2.x{1},simpson(f_tcfac2.x{2},f_tcfac2.f,1))
f_tcf4=spec2tpdf(SS,[],'Tc',[4.5 4.5 46],[0:0.25:8],6);
f_tc4.f(46)
f_tcac1=spec2thpdf(SS,[],'TcAc',[0 12 81],[0:0.25:8],-1);
plot(Tcf(ind), Ac(ind),'.');
hold on
pdfplot(f_tcfac1,'-.')
pdfplot(f_tcfac2)
```

First we conclude that the densities `f_tcfac1` and `f_tcfac2` really integrate to the marginal density of `Tc` (`f_tc4.f(46)`) demonstrating the accuracy of the computed densities `f_tcfac1` and `f_tcfac2`.

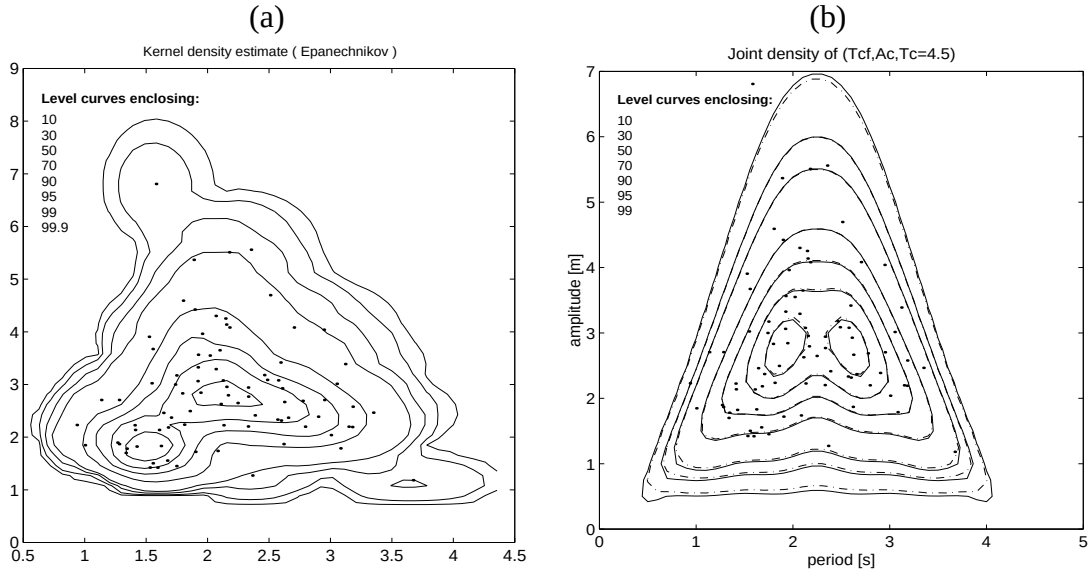


Figure 3.13: *(a) The estimated (KDE) density of crest position and height together with obsevations (dots). (b) The theoretically computed density with* `nit = -1, 2` *and the data.*

In Figure 3.13 (a) the estimated (KDE) joint density is given and it should be compared with Figure 3.13 (b) where the theoretical density is presented. Here we can really see the advantage of the theoretically computed densities. Even if we have here used a long record of wave data, there is not enough of waves to make a reliable estimate of the joint density, and in a standard 20 minutes records there would be far too few observations. □

As we have mentioned already the integral over the position of the computed densities is equal to the joint density of crest period and height. So in order to get the whole density of `Tc, Ac` one needs to execute the previous program to obtain the density of `Tc, Tcf, Ac` for different values of `Tc` and integrate out the variable `Tcf`. Clearly it will take some time. However the most time is spent on the computation of density of long and small waves that are not interesting. Hence we can start to compute the joint density of `Tc, Ac` for significant waves.

Example 10. (contd.) We compute the joint density of `Tc, Ac` of significant waves in the Gullfaks data in order to compare the distribution with the Longuet-Higgins approximation; see Section 3.3.2.

```
f_tcac_s = spec2thpdf(SS,[],'TcAc',[0 12 81],[Hs/2:0.1:2*Hs],-1);
```

Next we find the modified Longuet-Higgins (L-H)-density, i.e. the density with transformed crest heights. The original (L-H)-density is underestimating the high crest with up to one meter. We can see that for significant waves and the present spectrum the Longuet-Higgins density is quite accurate.

```
mom = spec2mom(SS,4,[],0);
t = f_tcac_s.x{1};
h = f_tcac_s.x{2};
flh_g = lh83pdf(t',h',[mom(1),mom(2),mom(3)],SS.tr);
clf
ind=find(Ac>Hs/2);
plot(Tc(ind), Ac(ind),'.');
hold on
pdfplot(flh_g,'k-.')
pdfplot(f_tcac_s)
```

In Figure 3.14 (a) the theoretical density is plotted with solid lines while the transformed L-H density is a dashed dotted line. We can see that the simple approximation is working very well, even if it gives slightly too short periods.

Finally, we compute the density for all wave heights. In Figure 3.14 (b) the theoretical density is compared with the data, and as we see, the agrement is quite good.

```
f_tcac = spec2thpdf(SS,[],'TcAc',[0 12 81],[0:0.2:8],-1);
pdfplot(f_tcac)
```

□

3.4.4 Joint density of crest and trough height

In previous sections we presented programs that compute joint densities of different wave characteristics. We started with marginal densities of `Tc` crest, `Tt` trough periods, and then joint densities of `Tc, Tt` were derived in order to get the wave period `Tcc`. Next we considered `Tc, Tcr, Ac` - crest period, crest position, and crest height. (The same is possible for `Tt, Ttb, At`.) However, in order to fully describe a wave we should compute the joint density of `Tc, Tac, Ac, Tt, Tat, At`. It is possible to write a program that computes such six dimensional densities and it would not take more then 10 minutes of computer time to compute the density for 200, say, different combinations of the characteristics. But in order to describe a six dimensional density one needs may be 100 000 combinations of values and this is not practically possible yet.

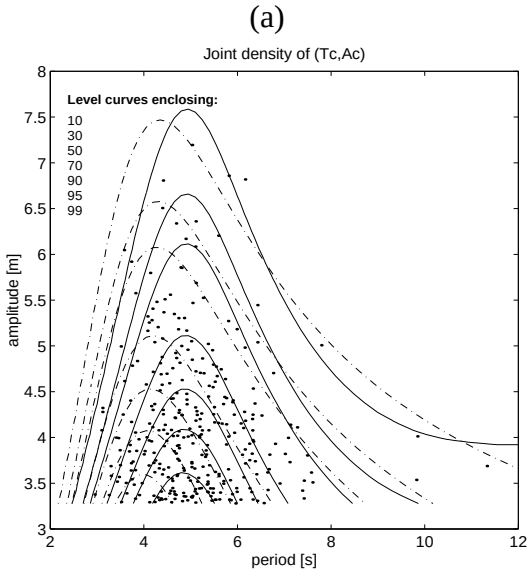


Figure 3.14: *The joint density of* `Tc` *and* `Ac` *for the transformed Gaussian model of the sea measurements from Gullfaks C platform (solid line) compared with the transformed Longuet-Higgins density (dashed dotted line) and the data (dots) for waves with significant crest (a).*

Observe that using numerical derivatives one can compute the joint density of `Tc, Ac, Tt, At` using the program `spec2tccpdf` (or `spec2AcAt`) but it would still take one or two days to do such computations.

There are however some alternatives. From previous studies we know that very high crests (troughs) occur at the local maximum (minimum) closest to a zero crossing. We also know that it is the derivative at the crossing that mainly determines the hight of the wave crest. Consequently, the steepness of a wave is mainly determined by the hight and location of *the last minimum before* and *the first maximum after* an upcrossing of the still water level. This particular type of min-to-max wave is called a *mean separated minimum-to-maximum* wave. In general, we can introduce a v*-level separated min-to-max wave* to be the last minimum before and the first maximum after a level v upcrossing. The distance between the mean-level separated mininim and maxima, denoted `TmM` can be used to compute steepnes of a wave, see Brodkorb [6] for details. The function `spec2mmtpdf` computes the joint density of `v`-separated wave length and other characteristics of the `v`-separated minima and maxima. It also computes the coint density of all pairs of local minima, maxima and the distance inbetween.

3.4.5 Min-to-max distributions – Markov method

We shall now investigate another of wave characteristic, namely the min-to-max wave distribution, including the min-to-max period and amplitude. This requires the joint density of the height of a local minimum (maximum) and the following maximum (minimum). The WAFO routine that handles this is called `spec2mmtpdf`, and calculates, i.a. the joint density of the height of a maximum and the following minimum; see `help spec2mmtpdf`.

One important application of the min-to-max distribution is for approximation of the joint density of `Ac, At`, the crest and trough amplitudes, by approximating the sequence of local extremes

in transformed Gaussian model by a Markov chain; see [54] for detailed description of the algorithm. The approximation has been checked on many different sea data givning very accurate results. It is also relatively fast.

Remark 3.2. There is also another program `spec2cmat` which is a function adaptet from WAT. It is somewhat less accurate but even faster. It is used to compute the so called Markov matrices and rainflow matrices used in fatigue. □

Example 11. *(min-max problems with Gullfaks data)* In this example we continue the analysis of the Gullfaks C platform data. First we shall retrive the sequence of turning points, i.e. the minima and maxima, in `yy` and calculate the theoretical distribution. In Figure 3.15 we can see that the theoretically computed density agrees very well with the estimated one.

```
tp = dat2tp(yy);
Mm = fliplr(tp2mm(tp));
fmm = kde(Mm,'epan');
f_mM = spec2mmtpdf(SS,[],'mm',[],[-7 7 51],1);
clf
pdfplot(f_mM,'-.')
hold on
pdfplot(fmm,'k-')
hold off
```

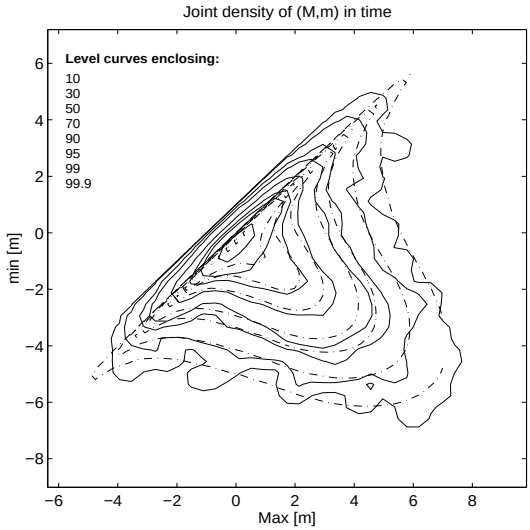


Figure 3.15: *The joint density of maximum and the following minimum for the transformed Gaussian model of the sea measurements from Gullfaks C platform (dashed-dotted line) compared with the estimated (KDE) density from data.*

We turn now to the joint density of crest and trough. We shall first compute the exact distribution with the help of `spec2mmtpdf`, and then compare the obtained distribution with that obtained from the min-to-max distribution by means of the Markov approximation for the min-max sequence. As we have mentioned before we do not use the full min-to-max distribution but instead the ”still water separated” minima and maxima.

```
ind = find(Mm(:,1)>v & Mm(:,2)<v);
Mmv = abs(Mm(ind,:)-v);
fmmv = kde(Mmv,'epan');
f_vmm = spec2mmtpdf(SS,[],'vmm',[],[-7 7 51],1);
clf
pdfplot(fmmv,'k-')
hold on
pdfplot(f_vmm,'-.')
hold off
```

Then we compute the joint density of crest and trough using the Markov approximation to the sequence of local extremes (sequence of turning points `tp`).

```
facat = kde([Ac At],'epan');
f_acat = spec2mmtpdf(SS,[],'AcAt',[],[-7 7 51],1);
clf
pdfplot(f_acat,'-.')
hold on
pdfplot(facat,'k-')
hold off
```

Now we are in the position to check our two methods, the Markov method, where the min-to-max sequence is approximated by a Markov chain, and the replcement of the true min-to-max transition probabilities by the transition probabilities which are valid for the "still water separated" min-to-max values. The results are presented in Figure 3.16. We see in (a) that the "still water separated" min-to-max distribution miss a considerable number of min-to-max values which fall on the same side of the still water level. On the other hand, figure (b) indicates that the Markov assumption is acceptable.

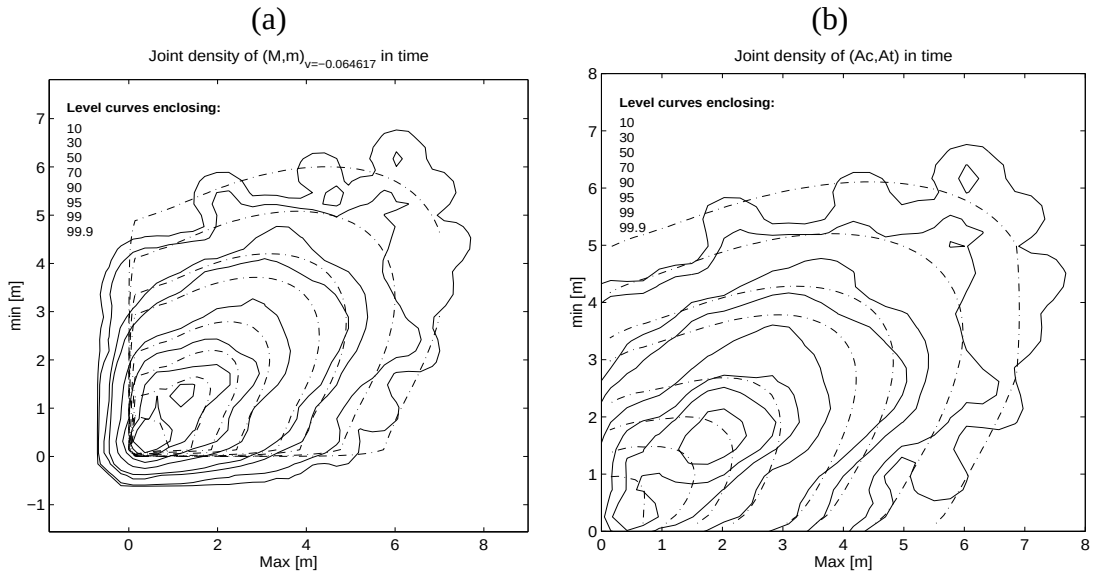


Figure 3.16: *(a) The joint density of "still water separated" min-to-max values for the transformed Gaussian model for the measurements from Gullfaks C (dashed-dotted line) compared to the estimated density from data (KDE, solid line). (b) Markov approximation for the joint density of crest and trough height* `Ac, At` *compared with the (KDE) estimator of the density.*

□

3.5 WAFO wave characteristics

3.5.1 wavedef

```
help wavedef
```

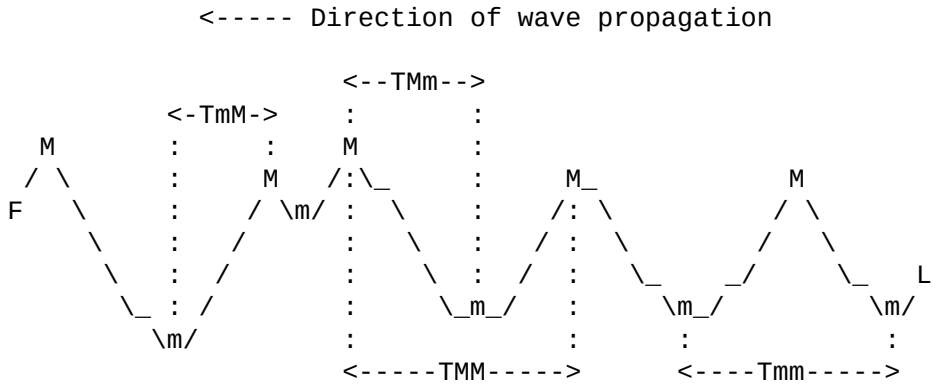
```
  WAVEDEF wave definitions and nomenclature

  Definition of trough and crest:
 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

  A trough (t) is defined as the global minimum between a
  level v down-crossing (d) and the next up-crossing (u)
  and a crest (c) is defined as the global maximum between a
  level v up-crossing and the following down-crossing.

  Definition of down- and up -crossing waves:
 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

  A level v-down-crossing wave (dw) is a wave from a
  down-crossing to the following down-crossing.
  Similarly a level v-up-crossing wave (uw) is a wave from a up-crossing
  to the next up-crossing.

  Definition of trough and crest waves:
 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

  A trough to through wave (tw) is a wave from a trough (t) to the
  following trough. The crest to crest wave (cw) is defined similarly.

  Definition of min2min and Max2Max wave:
 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

  A min2min wave (mw) is defined starting from a minimum (m) and
  ending in the following minimum.
  Similarly a Max2Max wave (Mw) is thus a wave from a maximum (M)
  to the next maximum (all waves optionally rainflow filtered).

            <----- Direction of wave propagation


    <------Mw-----> <----mw---->
    M             : :  c       :
   / \            M : / \_     :     c_           c
  F   \          / \m/    \    :    /: \         /:\
------d--------u----------d-------u----d--------u---d-------- level v
       \      /:           \  :  /: :  :\_    _/  : :\_   L
        \_   / :            \_t_/ : :  :  \t_/    : :  \m/
          \t/  <-------uw---------> :  <-----dw----->
           :                  :     :             :
           <--------tw-------->     <------cw----->

  (F= first value and L=last value).

  See also: tpdef, crossdef, dat2tc, dat2wa, dat2crossind
```

3.5.2 perioddef

```
help perioddef
```

```
  PERIODDEF wave periods (lengths) definitions and nomenclature

  Definition of wave periods (lengths):
 ---------------------------------------


           <----- Direction of wave propagation

               <-------Tu--------->
               :                  :
               <---Tc----->       :
               :          :       : <------Tcc---->
   M           :      c   :       : :              :
  / \          : M   / \_ :       : c_             c
 F   \         :/ \m/    \:       :/  \           / \
------d--------u----------d-------u----d--------u---d-------- level v
       \      /            \     /     :\_    _/:   :\_   L
        \_   /              \_t_/      :  \t_/  :   :  \m/
          \t/                 :        :        :   :
           :                  :        <---Tt--->   :
           <--------Ttt------->        :            :
                                      <-----Td----->
  Tu   = Up crossing period
  Td   = Down crossing period
  Tc   = Crest period, i.e., period between up crossing and
         the next down crossing
  Tt   = Trough period, i.e., period between down crossing and
         the next up crossing
  Ttt  = Trough2trough period
  Tcc  = Crest2crest period


           <----- Direction of wave propagation

                <--Tcf->                              Tuc
                :      :              <-Tcb->         <->
   M            :      c              :     :         : :
  / \           : M   / \_            c_    :         : c
 F   \          :/ \m/    \          /  \___:         :/ \
------d---------u----------d---------u-------d--------u---d-------- level v
      :\_      /            \     __/:        \_    _/     \_   L
      :  \_   /              \_t_/   :          \t_/         \m/
      :    \t/                 :     :
      :     :                  :     :
      <-Ttf->                  <-Ttb->


  Tcf  = Crest front period, i.e., period between up crossing and crest
  Tcb  = Crest back period, i.e., period between crest and down crossing
  Ttf  = Trough front period, i.e., period between down crossing and trough
  Ttb  = Trough back period, i.e., period between trough and up crossing

 Also note that Tcf and Ttf can also be abbreviated by their crossing
 marker, e.g. Tuc (u2c)  and Tdt (d2t), respectively. Similar applies to all the
 other wave periods and wave lengths.
```

(The nomenclature for wave length is similar, just substitute T and period with L and length, respectively)



See also: wavedef, ampdef, crossdef, tpdef

3.5.3 ampdef

```
help ampdef
```

```
  AMPDEF wave heights and amplitude definitions and nomenclature

  Definition of wave amplitude and wave heights:
 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

              <----- Direction of wave propagation


             ...............c_..........
             |              /| \          |
         Hd  |            _/ |  \         |  Hu
    M        |           /   |   \        |
   / \       |      M   / Ac |    \_      |     c_
  F   \      |     / \m/     |      \     |    /  \
 ------d-----|---u-------------------d---|---u----d------ level v
        \    |  /|                    \  |  /      \L
         \_  | / | At                  \_|_/
           \|/..|                       t
            t

   Ac   = crest amplitude
   At   = trough amplitude
   Hd   = wave height as defined for down crossing waves
   Hu   = wave height as defined for up crossing waves

  See also: wavedef, ampdef, crossdef, tpdef
```

3.5.4 crossdef

```
help crossdef

  CROSSDEF level v crossing definitions and nomenclature

  Definition of level v crossing:
 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  Let the letters 'm', 'M', 'F', 'L','d' and 'u' in the
  figure below denote local minimum, maximum, first value, last
  value, down- and up-crossing, respectively. The remaining
  sampled values are indicated with a '.'. Values that are identical
  with v, but do not cross the level is indicated with the letter 'o'.

  We have a level up-crossing at index, k, if

            x(k) <  v and v < x(k+1)
  or if
            x(k) == v and v < x(k+1) and x(r) < v for some di < r <= k-1

  where di is  the index to the previous downcrossing.
  Similarly there is a level down-crossing at index, k, if

            x(k) >  v and v > x(k+1)
   or if
            x(k) == v and v > x(k+1) and x(r) > v  for some ui < r <= k-1

  where ui is  the index to the previous upcrossing.

  The first (F) value is a up crossing if  x(1) = v and x(2) > v.
  Similarly, it is a down crossing if      x(1) = v and x(2) < v.

       M
     .   .                 M                   M
   .      . .             .                   .   .
 F            d               .             .       L
  ----------------------u-------d-------o---------------------- level v
                 .     .           .   .   u
                   .                 m
                    m

  See also: perioddef, wavedef, tpdef, findcross, dat2tp
```

CHAPTER 4
FATIGUE LOAD ANALYSIS AND RAIN-FLOW CYCLES

4.1 Random fatigue

4.1.1 Random load models

This section is intended to present some tools from WAFO for analysis of random loads in order to assess the fatigue damage. A complete list of fatigue routines can be obtained by `help fatigue`.

We shall assume that the load is given by one of three possible forms:

1. As measurements of the stress or strain function with some given sampling frequency in Hz. Such loads will be called measured loads and denoted by $x(t)$, $0 \leq t \leq T$, where t is time and T is the duration of the measurements.
2. In the frequency domain (that is important in system analysis) as a power spectrum. This means that the signal is represented by a Fourier series

$$x(t) \approx m + \sum_{i=1}^{N} a_i \cos(\omega_i\, t) + b_i \sin(\omega_i\, t)$$

   where $\omega_i = i \cdot 2\pi/T$ are angular frequencies, m is the mean of the signal and a_i, b_i are Fourier coefficients.
3. In the rainflow domain, i.e. the measured load is given in the form of a rainflow matrix.

We shall now review some simple means to characterize and analyze loads which are given in any of the forms (1)–(3), and show how to derive characteristics, important for fatigue evaluation and testing.

We assume that the reader has some knowledge about the concept of cycle counting, in particular rainflow cycles, and damage accumulation using Palmgren-Miners linear damage accumulation hypotheses. The basic definitions are given in the end of this introduction. Another important property is the crossing spectrum $\mu(u)$ defined as the intensity of upcrossings of a level u by $x(t)$ as a function of u.

The process of damage accumulation depends only on the values and the order of the local extremes in the load. The sequence of local extremes is called the *sequence of turning points*. The irregularity factor α measures how dense the local extremes are relatively to the mean frequency f_0. For a completely regular function there would be only one local maximum between upcrossings of the mean level, giving irregularity factor equal to one. In the other extreme case, there are infinitely many local extremes giving irregularity factor zero. However, if the crossing intensity $\mu(u)$ is finite,

most of those local extremes are irrelevant for the fatigue and should be disregarded by means of some smoothing device.

A particularly useful filter is the so-called rainflow filter that removes all local extremes that builds rainflow cycles with amplitude smaller than a given threshold. We shall always assume that the signals are rainflow filtered; see Section 4.2.1.

If more accurate predictions of fatigue life are needed then more detailed models are required for the sequence of turning points. Here the Markov chain theory has shown to be particularly useful. There are two reasons for this:

- the Markov models constitute a broad class of processes that can accurately model many real loads
- for Markov models, the fatigue damage prediction using rainflow method is particularly simple, Rychlik [47] and Johannesson [22]

In the simplest case, the necessary information is the intensity of pairs of local maxima and the following minima (the so-called Markov matrix or min-max matrix). The dependence between other extremes is modeled using Markov chains, see Frendahl & Rychlik [13].

4.1.2 Damage accumulation in irregular loads

In laboratory experiments, one often subjects a specimen of a material to a constant amplitude load, e.g. $L(t) = s\sin(\omega t)$ where s and ω are the constant amplitude and frequency, and one counts the number of cycles (periods) until the specimen breaks. The number of load cycles $N(s)$ as well as the amplitudes s are recorded. For small amplitudes, $s < s_\infty$, the fatigue life is often very large, and is set to infinity, $N(s) \approx \infty$, i.e. no damage will be observed even during an extended experiment. The amplitude s_∞ is called *the fatigue limit* or *the endurance limit*. In practice, one often uses a simple model for $N(s)$,

$$N(s) = \begin{cases} K^{-1}s^{-\beta} & s > s_\infty, \\ \infty & s \le s_\infty, \end{cases} \tag{4.1}$$

where K is a material dependent random variable, usually lognormally distributed, i.e. with $K^{-1} = E\epsilon^{-1}$ where $\ln(E) \in \mathrm{N}(0, \sigma_E^2)$, and ϵ, β are fixed constants.

For irregular loads, also called variable amplitude loads, one often combines the S-N curve with a cycle counting method by means of the *Palmgren-Miner linear damage accumulation theory*, to predict fatigue failure time. A cycle counting procedure is used to form equivalent load cycles, which are used in the life prediction.

If the $k:th$ cycle has amplitude s_k then it is assumed that it causes a damage equal to $1/N(s_k)$. The total damage at time t is then

$$D(t) = \sum_{t_k \le t} \frac{1}{N(s_k)} = K \sum_{t_k \le t} s_k^\beta = K D_\beta(t), \tag{4.2}$$

where the sum contains all cycles which have been completed up to time t. Then, the fatigue life time T^f, say, is shorter than t if the total damage at time t exceeds 1, i.e. if $D(t) > 1$. In other words, T^f is defined as the time when $D(t)$ crosses level 1 for the first time.

A very simple predictor of T^f is obtained by replacing $K = E^{-1}\epsilon$ in Eq. (4.2) by a constant, for example the median value of K, which is equal to ϵ. For high cycle fatigue, the time to failure is long, more than $10^5/f_0$, and then for stationary (and ergodic and some other mild assumptions) loads, the damage $D_\beta(t)$ can be approximated by its mean $E[D_\beta(t)] = d_\beta \cdot t$. Here d_β is the damage intensity, i.e. how much damage is accumulated per time unit. This leads to a very simple predictor of fatigue life time

$$\hat{T}^f = \frac{1}{\epsilon d_\beta}. \tag{4.3}$$

4.1.3 Rainflow cycles and hysteresis loops

The now commonly used cycle counting method is the rainflow counting, which was introduced by Endo [36] in 1968. It was designed to catch both slow and rapid variations of the load by forming cycles by pairing high maxima with low minima even if they are separated by intermediate extremes. Each local maximum is used as the maximum of a hysteresis loop with an amplitude that is computed by the rainflow algorithm. A new definition of the rainflow cycle, equivalent to the original definition, was given by Rychlik [46]. The formal definition is also illustrated in Figure 4.1.

Definition 4.1 (Rainflow cycle) *From each local maximum M_k one shall try to reach above the same level, in the backward (left) and forward (right) directions, with an as small downward excursion as possible. The minima, m_k^- and m_k^+, on each side are identified. That minimum which represents the smallest deviation from the maximum M_k is defined as the corresponding rainflow minimum m_k^{RFC}. The k:th rainflow cycle is defined as $(m_k^{\mathrm{RFC}}, M_k)$.*

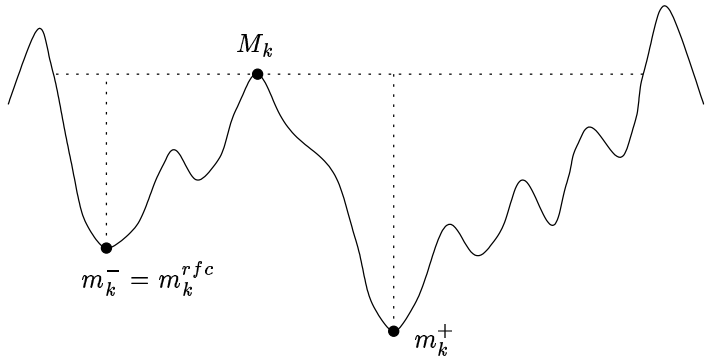


Figure 4.1: *Definition of the rainflow cycle, as given by Rychlik [46].*

If t_k be the time of the k:th local maximum and the corresponding rainflow amplitude is $s_k^{\mathrm{RFC}} = M_k - m_k^{\mathrm{RFC}}$, the amplitude of the attached hysteresis loop, then the total damage by

$$D(t) = \sum_{t_k \le t} \frac{1}{N(s_k^{\mathrm{RFC}})} = K \sum_{t_k \le t} (s_k^{\mathrm{RFC}})^\beta = K D_\beta(t), \tag{4.4}$$

where the sum contains all rainflow cycles which have been completed up to time t.

To use (4.3) to predict the fatigue life we need the damage intensity d_β i.e. the damage per time unit caused by the rainflow cycles. If there are on the average f_0 maxima[1] per time unit, and

[1] We have defined f_0 as the mean level upcrossing frequency, i.e. the mean number of times per time unit that the load upcrosses the mean level. Thus there are in fact at least f_0 local maxima per time unit. Since the rainflow filter reduces the number of cycles, we let f_0 here be *defined as* the averege number of rainflow cycles per time unit.

equally many rainflow cycles, and each rainflow cycle causes an expected damage $\epsilon E(1/N_{S^{\mathrm{RFC}}})$ it is clear that the damage intensity is equal to

$$d_\beta = f_0\, E\left((S^{\mathrm{RFC}})^\beta\right).$$

Thus, an important parameter for prediction of fatigue life is the distribution of the rainflow amplitudes and in particular the expected value of the β-power of its inverted value. WAFO contains a number of routines for handling the rainflow cycles in observed load data and in theoretical load models.

4.2 Load cycle characteristics

4.2.1 Rainflow filtered load 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 modeled by a transformed Gaussian process `xx`, such a filtration is performed on the inverse transformed signal `yy = g(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 trough-to-crest wave concept, defined as in Table 3.1, instead of the simpler min-to-max cycles. Chapter 3 of this tutorial was aimed at showing how one can compute the crest-to-trough wave characteristics from a Gaussian or transformed Gaussian model.

The trough-to-crest 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 by Rychlik, [51], and included in the WAFO toolbox. For completeness, we describe the algorithm of the rainflow filter.

In this tutorial we have used a simple definition of rainflow cycles which is 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 accomplished by defining the rainflow minimum $m^{\mathrm{RFC}}(t)$ for all time points t of a function $x(t)$ in such a way that the rainflow amplitude $x(t) - m^{\mathrm{RFC}}(t)$ is zero if the point $x(t)$ is not a strict local maximum of the function; see Rychlik [51] for more detailed discussion. Now, a *rainflow filter with threshold h*, extracts all rainflow cycles $(m^{\mathrm{RFC}}(t), x(t))$ such that $x(t) - m^{\mathrm{RFC}}(t) > h$. 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 4.7 for an example.

4.2.2 Oscillation count and rainflow matrix

The rainflow count is a generalization of the crossing count. The crossing spectrum counts the number of times a signal upcrosses any level u. More important for fatigue damage is the *o*scillation count, $N^{\mathrm{OSC}}(u,v)$ which counts the number of times a signal upcrosses an interval $[u,v]$. The oscillation count is thus a function of the two variables u and v, and is plotted as a bivariate count. The oscillation count is a counting distribution for the rainflow cycles. Consequently, if the matrix `Nosc` with elements $N^{\mathrm{OSC}}(u_j,u_i)$ is known we can compute the frequency (or rather histogram) matrix of the rainflow count by means of the WAFO-function `nt2fr` and obtain the matrix `Frfc = nt2fr(Nosc)`, in fatigue practice 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 `Frfc`, say, of x, then the rainflow matrix of x_h is obtained by setting some subdiagonals of `Frfc` 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 $(m_i^{\mathrm{TC}}, M_i^{\mathrm{TC}})$ from the signal is closely related to rainflow filtering. Since given a reference level u^{TC}, the sequence $(m_i^{\mathrm{TC}}, M_i^{\mathrm{TC}})$ can be obtained by first removing all rainflow cycles $(m_j^{\mathrm{RFC}}, M_j)$ such that $M_j < u^{\mathrm{TC}}$ or $m_j^{\mathrm{RFC}} > u^{\mathrm{TC}}$ and then finding the min-to-max pairs in the filtered signal.

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.

4.2.3 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.

For any load sequence we shall denote by `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 min-to-max cycles and max-to-min 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 min-to-max and max-to-min cycles; sometimes called *Markov*

matrices. Note that for transformed Gaussian process a Markov matrix for min-to-max cycles was computed by means of the WAFO-function `minmax`. The max-to-min 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.*

Figure 4.2 shows the general principle of a Markov transition count between turning points of local maxima and minima. The values have been discretized to levels `1, ..., n`, from smallest to largest.

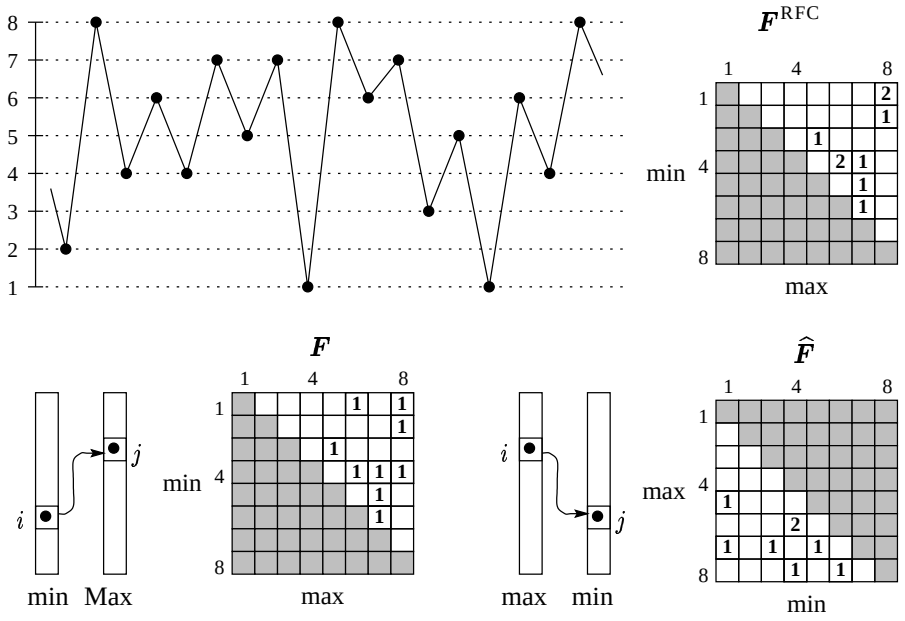


Figure 4.2: *Part of a discrete load process where the turning points are marked with •. The scale to the left is the discrete levels. The transitions from minimum to maximum and the transitions from maximum to minimum are collected in the min-max matrix, $\boldsymbol{F}$ and max-min matrix, $\widehat{\boldsymbol{F}}$, respectively. The rainflow cycles are collected in the rainflow matrix, $\boldsymbol{F}^{\mathrm{RFC}}$. The numbers in the squares are the number of observed cycles and the grey areas are by definition always zero.*

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 `FmM` or by its rainflow matrix `Frfc`, which are connected by the following nonlinear equation

$$\texttt{Frfc} = \texttt{FmM} + \mathcal{F}(\texttt{FmM}), \tag{4.5}$$

where $\mathcal{F}$ is a matrix valued function, defined in [51], where also an algorithm to compute $(\mathcal{I}+\mathcal{F})^{-1}$ is given. The WAFO-function for computing `Frfc` from `FmM` is `mctp2rfc` while the inverse, i.e. `FmM` as a function of `Frfc`, is computed by `rfc2mctp`.

4.3 Cycle analysis with WAFO

In this section we shall demonstrate how WAFO can be used to extract rainflow cycles from a load sequence, and how fatigue life can be estimated. The Markov method is used for simulation and approximation of real load sequences. We shall use three load examples, the deep water sea load, a simulated transformed Gaussian model, and a load sequence generated from an special Markov structure.

4.3.1 Crossing intensity

Basic to the analysis is the crossing intensity function $\mu(u)$, i.e. the number of times per time unit that the load up-crosses the level u, considered as a function of$u$. We illustrate the computations on the deep water sea waves data.

```
load sea.dat; xx_sea=sea;
tp_sea = dat2tp(xx_sea);
lc_sea = tp2lc(tp_sea);
T_sea = xx_sea(end,1)-xx_sea(1,1);
lc_sea(:,2) = lc_sea(:,2)/T_sea;
subplot(221), plot(lc_sea(:,1),lc_sea(:,2))
title('Crossing intensity, (u, \mu(u))')
subplot(222), semilogx(lc_sea(:,2),lc_sea(:,1))
title('Crossing intensity, (log \mu(u), u)')
```

The routine `dat2tp` and`tp2lc` take a load sequence and extracts the turning points, and from this calculates the number of upcrossings as a function of level. The plots produced, Figure 4.3, show the crossing intensity plotted in two common modes, lin-lin of $(u, \mu(u))$ and log-lin of $(\log \mu(u), u)$.

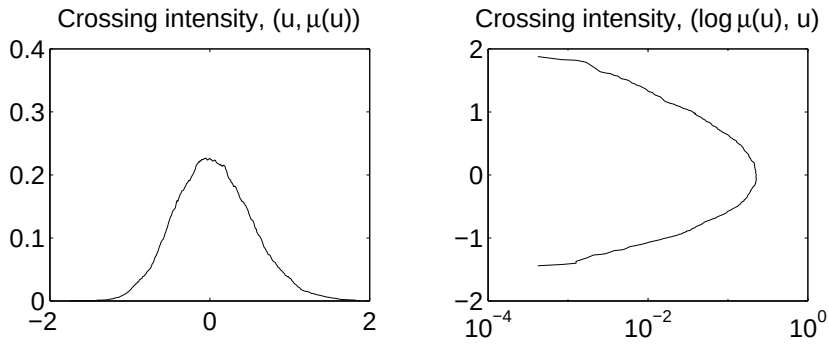


Figure 4.3: *Level crossing intensity for sea data.*

We shall also have use for the *mean frequency* f_0, i.e. the number of mean level upcrossings per time unit, and the irregularity index, α, which is the mean frequency divided by the mean number

of local maxima per time unit. Thus $1/\alpha$ is the average number of local maxima that occur between the mean level upcrossings.

To compute f_0 we use the Matlab function `interp1`, see `help interp1`.

```
m_sea = mean(xx_sea(:,2));
f0_sea = interp1(lc_sea(:,1),lc_sea(:,2),m_sea,'linear');
f0_sea
extr_sea = length(tp_sea)/(2*T_sea);
alfa_sea = f0_sea/extr_sea
```

4.3.2 Extraction of rainflow cycles

We start by a study of rainflow cycles in the deep water sea data. Recall the definition of rainflow and min-max cycle counts. The demo program `democc` illustrates these definitions. To use it to identify the first few rainflow and min-max cycles, just use,

```
proc = xx_sea(1:500,:);
democc
```

Two windows will appear. In Demonstration Window 1, first mark the turning points by the button TP. Then choose a local maximum (with the buttons marked $+1, -1, +5, -5$) and find the corresponding cycle counts, using the buttons RFC, PT. The cycles are visualized in the other window.

We shall now examine cycle counts in the load `xx_sea`. From the sequence of turning points `tp` we find the rainflow and min-max cycles in the data set,

```
RFC_sea = tp2rfc(tp_sea);
mM_sea = tp2mm(tp_sea);
```

Since each cycle is a pair of a local maximum and a local minimum in the load, a cycle count can be visualized as a set of pairs in the $\mathbf{R}^2$-plane. Compare the min-max and rainflow counts in the load in Figure 4.4.

```
clf
subplot(121), ccplot(mM_sea)
title('min-max cycles')
subplot(122), ccplot(RFC_sea)
title('Rainflow cycles')
```

Observe that `RFC` contains more cycles with high amplitudes, compared to `mM`. This becomes more evident in an amplitude histogram as seen in Figure 4.5.

```
ampmM_sea = cc2amp(mM_sea);
ampRFC_sea = cc2amp(RFC_sea);
clf
subplot(221), hist(ampmM_sea,25);
title('min-max amplitude distribution')
subplot(222), hist(ampRFC_sea,25);
title('Rainflow amplitude distribution')
```

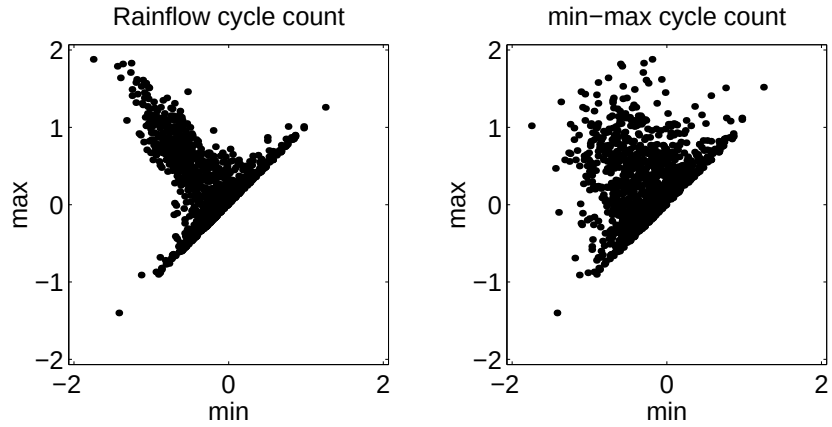


Figure 4.4: *min-max and rainflow cycle plots for sea data.*

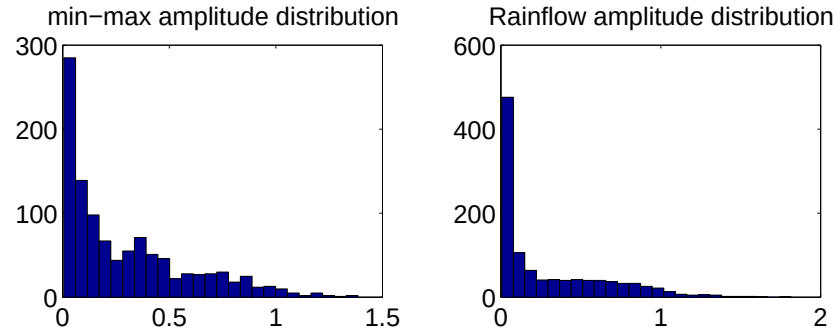


Figure 4.5: *min-max and rainflow cycle distributions for sea data.*

4.3.3 Simulation of rainflow cycles

Simulation of cycles in a Markov model

The most simple cycle model assumes that the sequence of turning points forms a Markov chain. Then the model is defined by the min-max matrix, `G`. The matrix has dimension $n \times n$, where n is the number of discrete levels (e.g. 32 or 64). In this example the discrete levels `u` are chosen in the range from -1 to 1. The matrix `G` will contain the probabilities of transitions between the different levels in `u`.

```
n = 41; param_m = [-1 1 n]; param_D = [1 n n];
u_markov = levels(param_m);
G_markov = mktestmat(param_m,[-0.2 0.2],0.15,1);
```

The model is easy to simulate and this is performed by the simulation routine `mctpsim`. This routine simulates only the sequence of turning points and not the intermediate load values.

```
T_markov = 5000;
xxD_markov = mctpsim({G_markov []},T_markov);
xx_markov = [(1:T_markov)' u_markov(xxD_markov)'];
```

Here `xxD_markov` takes values $1, \ldots, n$, and by changing the scale we get the load `xx_markov` which takes values between -1 and 1. The first 50 samples of the simulation is plotted in Figure 4.6 by

```
plot(xx_markov(1:50,1),xx_markov(1:50,2))
```

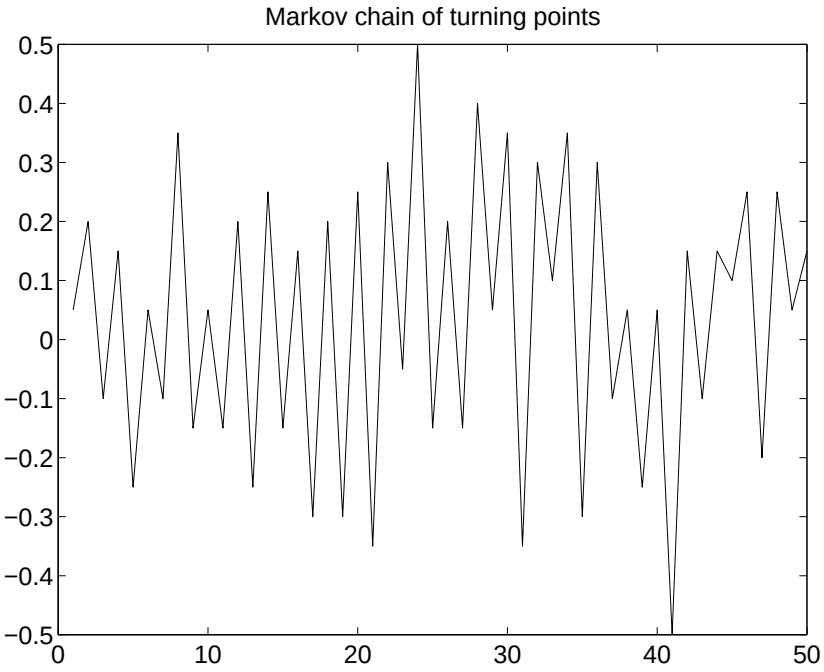


Figure 4.6: *Simulated Markov sequence of turning points.*

We shall later use the matrix `G_markov` to calculate the theoretical rainflow matrix, but first we construct a similar sequence of turning points from a transformed Gaussian model.

Rainflow cycles in a transformed Gaussian model

In this example we shall consider a sea-data-like series obtained as a transformed Gaussian model with JONSWAP spectrum. Since the JONSWAP spectrum contains also rather high frequencies a JONSWAP load will contain many cycles with small amplitude. These are often uninteresting and can be removed by a rainflow filter as follows.

Let `g` be the Hermite transformation proposed by Winterstein, which we used in Chapter 2. Suppose the spectrum `spec` is of the JONSWAP type. To get the transform we need as input the approximative higher moments, skewness and kurtosis, which are automatically calculated from the spectrum by the routine `spec2skew`. We define the spectrum structure, including the transformation, and simulate the tranformed Gaussian load `xx_herm`. The routine `dat2dtp` extracts the turning points discretized to the levels specified by the parameter vector `param`.

Note that when calling the simulation routine `spec2sdat` with a transformation the input spectrum must be normalized to have standard deviation 1, i.e. one must divide the spectral values by the variance `saˆ2`.

```
me = mean(xx_sea(:,2));
sa = std(xx_sea(:,2));
Hm0_sea = 4*sa;
Tp_sea = 1/max(lc_sea(:,2));
spec = jonswap([],[Hm0_sea Tp_sea]);

[sk, ku] = spec2skew(spec);
spec.tr = hermitetr([],[sa sk ku me]);
param_h = [-1.5 2 51];
spec_norm = spec;
spec_norm.S = spec_norm.S/saˆ2;
```

```
xx_herm = spec2sdat(spec_norm,[2^15 1],0.1);
h = 0.2;
[dtp,u_herm,xx_herm_1]=dat2dtp(param_o,xx_herm,h);
clf
plot(xx_herm(:,1),xx_herm(:,2),'k','LineWidth',2);
hold on;
plot(xx_herm_1(:,1),xx_herm_1(:,2),'k--','Linewidth',2);
axis([0 50 -4 6]), hold off;
title('Rainflow filtered wave data')
```

The rainflow filtered data `xx_herm_1` contains the turning points of `xx_herm` with rainflow cycles less than `h=0.2` removed. In Figure 4.7 the dashed curve connects the remaining turning points after filtration.

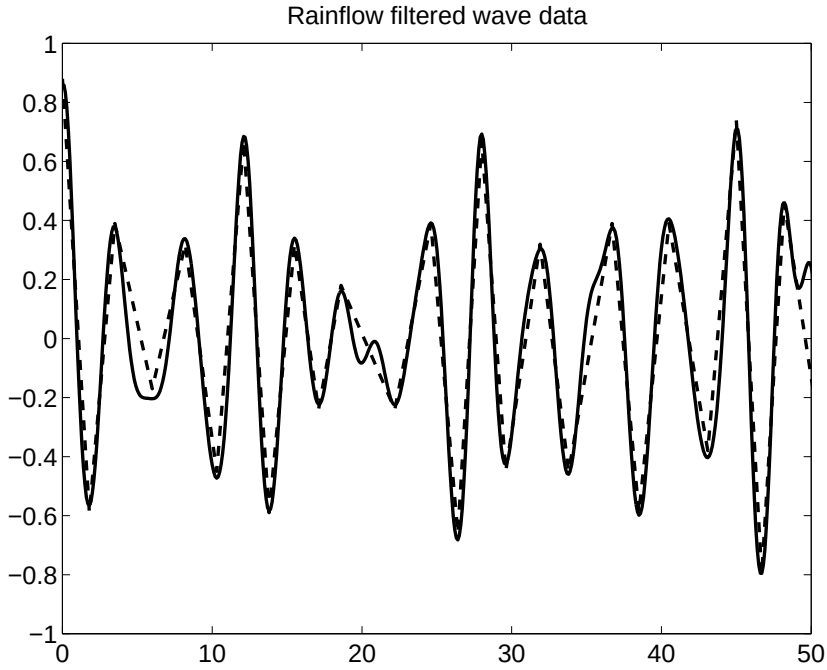


Figure 4.7: *Hermite transformed wave data and rainflow filtered turning points,* `h = 0.2`.

Try different degree of filtering on the Ochi transformed sequence and see how it effects the min-max cycle distribution. You can use the following sequence of commands, with different `h` -values; see Figure 4.8 for the results. Note that the rainflow cycles have their original values in the left figure but that they have been discretized to the discrete level defined by `param_o` in the right figure.

```
tp_herm=dat2tp(xx_herm);
RFC_herm=tp2rfc(tp_herm);
mM_herm=tp2mm(tp_herm);
h=1;
[dtp,u,tp_herm_1]=dat2dtp(param_o,xx_herm,h);
RFC_herm_1 = tp2rfc(tp_herm_1);
clf
subplot(121), ccplot(RFC_herm)
title('h=0')
subplot(122), ccplot(RFC_herm_1)
title('h=1')
```

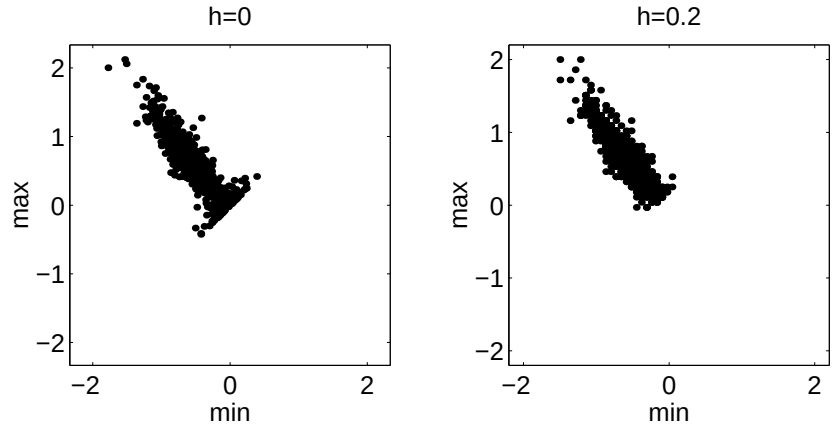


Figure 4.8: *Rainflow cycles and rainflow filtered rainflow cycles in the transformed Gaussian process.*

4.3.4 Calculating the Rainflow Matrix

We have now shown how to extract rainflow cycles from a load sequence and to perform rainflow filtering in measured or simulated load sequences. Next we shall demonstrate how the expected (theoretical) rainflow matrix can be calculated in any random load or wave model defined either as a Markov chain of turning points, or as a stationary random process. We do this by means of the Markov method based on the max-min transition matrix for the sequence of turning points. This matrix can either be directly estimated from or assigned to a load sequence, or it can be calculated from the correlation or spectrum structure of a transformed Gaussian model.

Calculation of rainflow matrix in the Markov model

The theoretical rainflow matrix `Grfc` for the Markov model is calculated in WAFO by the routine `mctp2rfm`. Let `G_markov` be as in Section 4.3.3 and calculate the theoretical rainflow matrix by

```
Grfc_markov=mctp2rfm({G_markov []});
```

A cycle matrix, e.g. a min-max or rainflow matrix, can be plotted by `cmatplot`. Now we will compare the min-max and the rainflow matrices.

```
clf
subplot(121),cmatplot(u_markov,u_markov,G_markov),axis('square')
subplot(122),cmatplot(u_markov,u_markov,Grfc_markov),axis('square')
```

Both 2D- and 3D-plots can be drawn; see `help cmatplot`. It is also possible to plot many matrices in one call.

```
cmatplot(u_markov,u_markov,{G_markov Grfc_markov},3)
```

A plot with `method=4` gives contour lines; see Figure 4.9. Note that for high maxima and low minima, the rainflow matrix has a pointed shape while the min-max matrix has a more rounded shape.

```
cmatplot(u_markov,u_markov,{G_markov Grfc_markov},4)
subplot(121), axis('square'), title('min-to-max transition matrix')
subplot(122), axis('square'), title('Rainflow matrix')
```

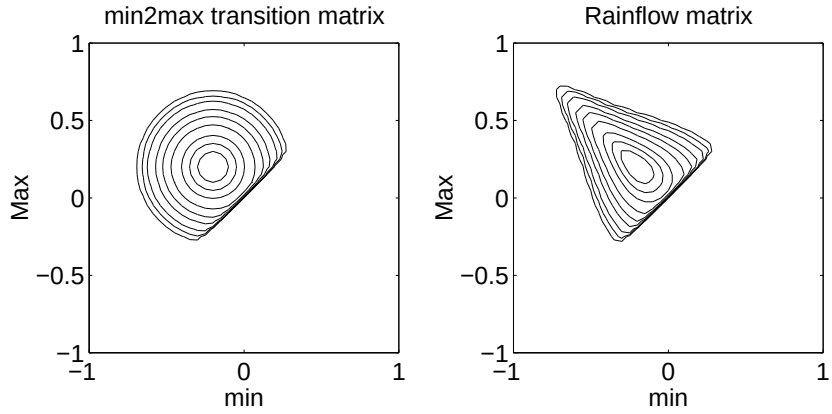


Figure 4.9: *min-max-matrix and theoretical rainflow matrix for test Markov sequence.*

We now compare the theoretical rainflow matrix with an observed rainflow matrix obtained in the simulation. In this case we have simulated discrete Markov chain of turning points with states `1,...,n` and put them in the variable `xxD_markov`. It is turned into a rainflow matrix by the WAFO-routine `dtp2rfm`). The comparison in Figure 4.10 between the observed rainflow matrix and the theoretical one is produced as follows.

```
n = length(u_markov);
Frfc_markov = dtp2rfm(xxD_markov,n);
clf
cmatplot(u_markov,u_markov,{Frfc_markov Grfc_markov*T/2},3)
subplot(121), axis('square'), title('Observed rainflow matrix')
subplot(122), axis('square'), title('Theoretical rainflow matrix')
```

Note that in order to compare the observed matrix `Frfc_markov` with the theoretical matrix `Grfc_markov` we have to multiply the latter by the number of cycles in the simulation which is equal to `T/2`.

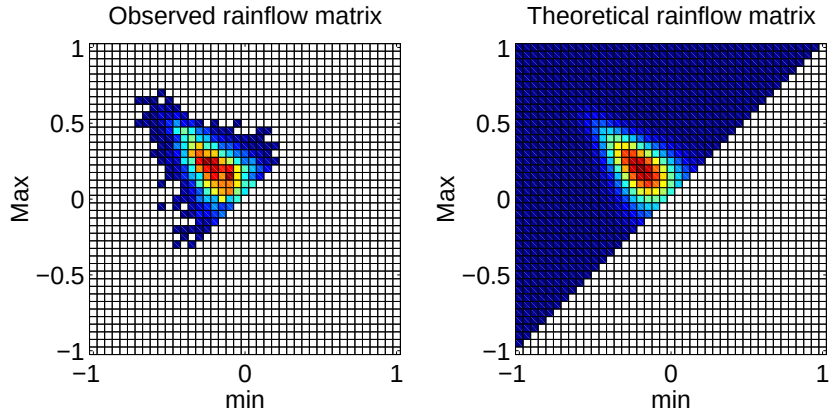


Figure 4.10: *Observed and theoretical rainflow matrix for test Markov sequence.*

We end this section by an illustration of the rainflow smoothing operation. The observed rainflow matrix is rather irregular, due to the statistical variation in the finite sample. To facilitate comparison with the theoretical rainflow matrix we smooth it by the built in smoothing facility in the routine `cc2cmat`; do `help cc2cmat`. To see how it works for different degrees of smoothing we calculate the rainflow cycles by `tp2rfc`.

```
tp_markov = dat2tp(xx_markov);
RFC_markov = tp2rfc(tp_markov);
h = 1;
Frfc_markov_smooth = cc2cmat(param_m,RFC_markov,[],1,h);
clf
cmatplot(u_markov,u_markov,{Frfc_markov_smooth Grfc_markov*T/2},4)
subplot(121), axis('square'), title('Smoothed observed rainflow matrix')
subplot(122), axis('square'), title('Theoretical rainflow matrix')
```

Here, the smoothing is done as a kernel smoother with a bandwidth parameter `h = 1`. The effect of the smoothing is shown in Figure 4.11.

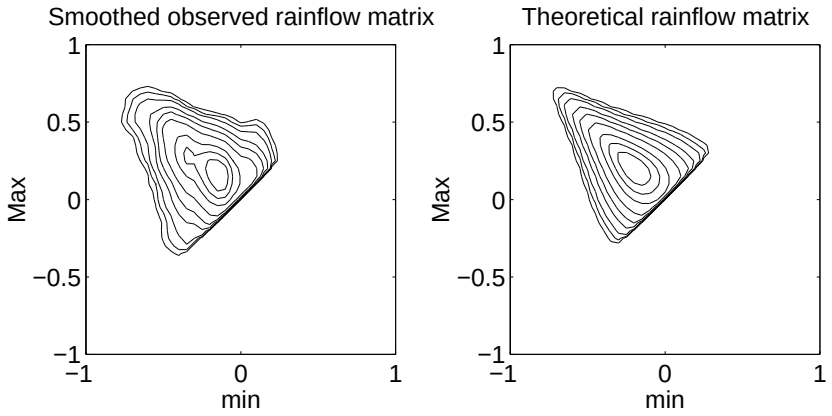


Figure 4.11: *Smoothed observed and calculated rainflow matrix for test Markov sequence.*

Rainflow matrix from spectrum

We are now ready to demonstrate how the rainflow matrix can be calculated in a load or wave model defined by its correlation or spectrum structure. We chose the sequence transformed Gaussian model with the Hermite transform `xx_herm` which was studied in Section 4.3.3. This model was defined by its JONSWAP spectrum and the standard Hermite transform for asymmetry.

We first need to find the structure of the turning points which is defined by the min-to-max density by the methods in Section 3.4.5. We start by computing an approximation `GmM3_herm` of the min-max density by means of the cycle distribution routine `spec2cmat`. The type of cycle is specified by a cycle parameter, in this case `'Mm'`.

```
GmM3_herm = spec2cmat(spec,[],'Mm',[],[],2);
```

The result is seen in Figure 4.12.

Then we approximate the distribution of the turning points by a Markov chain with transitions between extrema calculated from `GmM3_herm`, and by (4.5) compute the rainflow matrix.

```
Grfc_herm = mctp2drfm({GmM3_herm.f,[]});
```

In WAFO, the rainflow matrix can be calculated directly from the spectrum by the cycle distribution routine `spec2cmat` by specifying the cycle parameter to `'rfc'`.

```
Grfc_direct_herm = spec2cmat(spec,[],'rfc',[],[],2);
```

The output is a structure array which contains the rainflow matrix in the cell `.f`.

The min-max matrix `GmM3_herm` and the rainflow matrix `Grfc_herm` are shown together in Figure 4.12, obtained using the following commands.

```
clf
u_herm = levels(param_o);
cmatplot(u_herm,u_herm,{GmM3_herm.f Grfc_herm},4)
subplot(121), axis('square'), title('min-max matrix')
subplot(122), axis('square'), title('Theoretical rainflow matrix')
```

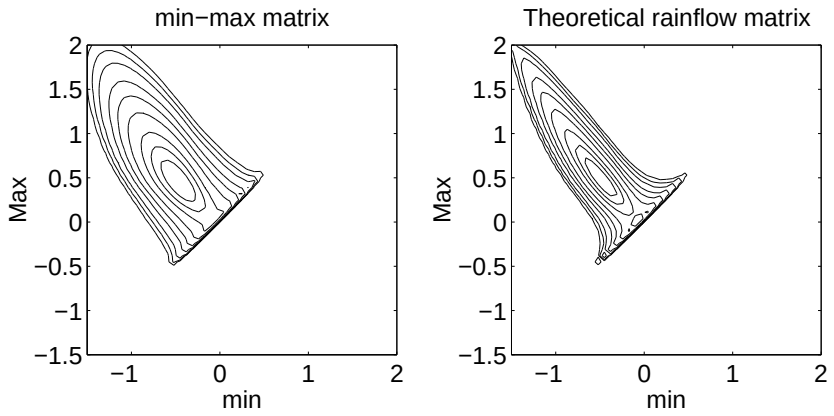


Figure 4.12: *min-max matrix and theoretical rainflow matrix for Hermite-transformed Gaussian waves.*

We can also compare the theoretical min-max matrix with the observed cycle count and the theoretical rainflow matrix with the observed one. In both comparisons we smooth the observed matrix to get more regular structure. We also illustrate the multi-plotting capacity of the routine `cmatplot`.

```
tp_herm=dat2tp(xx_herm);
RFC_herm=tp2rfc(tp_herm);
mM_herm=tp2mm(tp_herm);
h = 1;
FmM_herm_smooth = cc2cmat(param_o,mM_herm,[],1,h);
Frfc_herm_smooth = cc2cmat(param_o,RFC_herm,[],1,h);
T_herm=xx_herm(end,1)-xx_herm(1,1);
clf
cmatplot(u_herm,u_herm,{FmM_herm_smooth GmM3_herm.f*T_herm/2;...
   Frfc_herm_smooth Grfc_herm*T_herm/2},4)
subplot(221), axis('square'), title('Observed smoothed min-max matrix')
subplot(222), axis('square'), title('Theoretical min-max matrix')
subplot(223), axis('square'), title('Observed smoothed rainflow matrix')
subplot(224), axis('square'), title('Theoretical rainflow matrix')
```

4.3.5 Simulation from crossings or rainflow structure

In fatigue experiments it is important to generate load sequences with a prescribed rainflow or other crossing properties. Besides the previously used simulation routines for Markov loads and spectrum loads WAFO contains algorithm for generation of random load sequences which has a specified average rainflow distribution or a specified irregularity and crossing spectrum. These routines are `rfc2load` and `lc2sdat`, respectively.

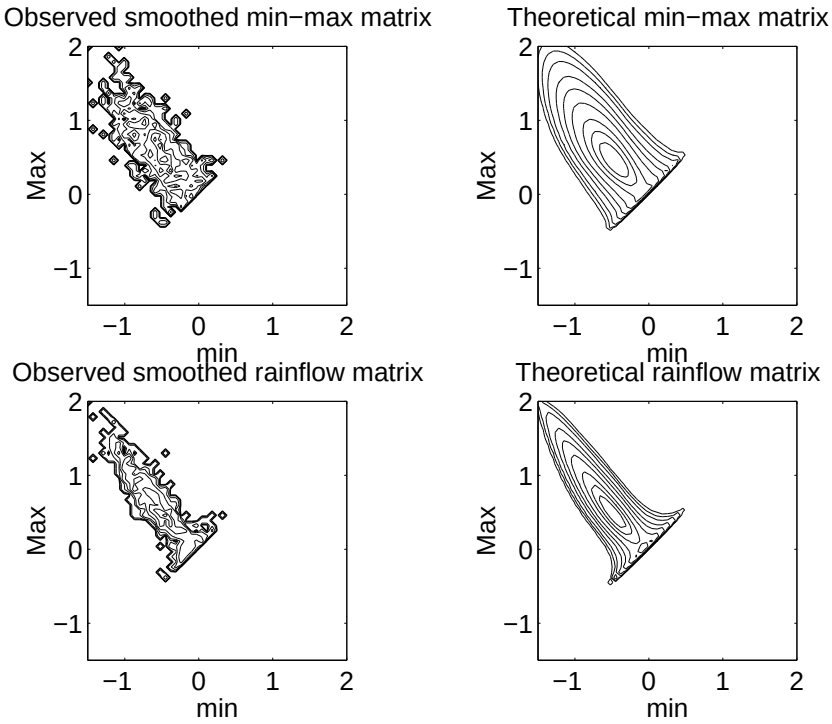


Figure 4.13: *Observed smoothed and theoretical min-max matrix, (and bserved smoothed and theoretical rainflow matrix for Hermite-transformed Gaussian waves.*

Simulation from crossing structure

The routine `lc2sdat` simulates a load with specified irregularity factor and crossing spectrum. We first estimate these quantities in the simulated Hermite transformed Gaussian load, and then simulate series with the same crossing spectrum but with varying irregularity factor. The sampling variability increases with decreasing irregularity factor, as is seen in Figure 4.14. The figures were generated by the following commands.

```
clf
cross_herm=dat2lc(xx_herm);
alpha1=0.25;
alpha2=0.75;
xx_herm_sim1=lc2sdat(500,alpha1,cross_herm);
cross_herm_sim1=dat2lc(xx_herm_sim1);
subplot(211)
plot(cross_herm(:,1),cross_herm(:,2)/max(cross_herm(:,2)))
hold on
stairs(cross_herm_sim1(:,1),...
    cross_herm_sim1(:,2)/max(cross_herm_sim1(:,2)))
hold off
title('Crossing intensity, \alpha = 0.25')
subplot(212)
plot(xx_herm_sim1(:,1),xx_herm_sim1(:,2))
title('Simulated load, \alpha = 0.25')

xx_herm_sim2=lc2sdat(500,alpha2,cross_herm);
cross_herm_sim2=dat2lc(xx_herm_sim2);
subplot(211)
plot(cross_herm(:,1),cross_herm(:,2)/max(cross_herm(:,2)))
```

```
hold on
stairs(cross_herm_sim2(:,1),...
    cross_herm_sim2(:,2)/max(cross_herm_sim2(:,2)))
hold off
title('Crossing intensity, \alpha = 0.75')
subplot(212)
plot(xx_herm_sim2(:,1),xx_herm_sim2(:,2))
title('Simulated load, \alpha = 0.75')
```

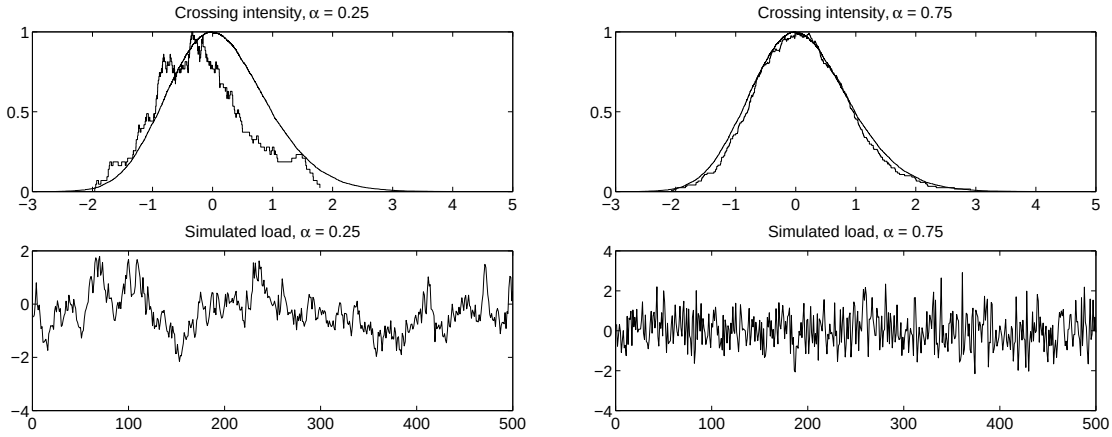


Figure 4.14: *Upper figures show target crossing spectrum (smooth curve) and obtained spectrum (wiggled curve) for simulated process shown in lower figures. Irregularity factor: left* $\alpha = 0.25$*, right* $\alpha = 0.75$*.*

4.4 Fatigue damage and fatigue life distribution

4.4.1 Introduction

We shall now give a more detailed account of how WAFO can be used to estimate and bound the fatigue life distribution under random loading. The basic assumptions are the Wöhler curve (4.1) and the Palmgren-Miner damage accumulation rule (4.2),

$$N(s) = \begin{cases} K^{-1}s^{-\beta} & s > s_\infty, \\ \infty & s \le s_\infty, \end{cases} \tag{4.6}$$

$$D(t) = \sum_{t_k \le t} \frac{1}{N(s_k)} = K \sum_{t_k \le t} s_k^\beta = K D_\beta(t). \tag{4.7}$$

Here $N(s)$ is the fatigue life expected from constant amplitude test with amplitude s, and $D(t)$ is the total damaged at time t caused by variable amplitude cycles s_k, which have been completed before time t. The damage intensity $d_\beta = D(t)/t$ for large t is the amount of damage per time unit.

Most information is contained in the cycle amplitude distribution, in particular in the rainflow cycles, in which case (4.7) becomes,

$$D(t) = \sum_{t_k \le t} \frac{1}{N_{s_k}} = \sum_{t_k \le t} K \left(S_k^{\mathrm{RFC}}\right)^\beta, \qquad S_k^{\mathrm{RFC}} = \left(M_k - m_k^{\mathrm{RFC}}\right)/2.$$

The rainflow cycle count RFC can directly be used for prediction of expected fatigue life. The expression (4.3) gives the expected time to fatigue failure in terms of the material constant ϵ and the expected damage per cycle d_β. The parameters ϵ and β can be estimated from an S-N curve. In the examples here we will use $\epsilon = 5.5 \cdot 10^{-10}$, $\beta = 3.2$; see Section 4.4.4. For our sea load xx_sea the computations go directly from the rainflow cycles as follows:

```
beta=3.2; gam=5.5E-10; T_sea=xx_sea(end,1)-xx_sea(1,1);
d_beta=cc2dam(RFC_sea,beta)/T_sea;
time_fail=1/gam/d_beta/3600
```

giving the time to failure 5.9693e+006 when time to failure is counted in hours (= 3600 sec). Obviously this load causes little damage to the material with the specified properties, since the failure time is almost 700 years.

4.4.2 Level Crossings

We have in Section 4.3.5 seen how the crossings intensity contains information about the load sequence and how it can be used for simulation. We shall now investigate the relation between the crossing intensity, the rainflow cycles and the expected fatigue life.

We use the Markov model from Section 4.3.3 for the sequence of turning points as an example. First we go from the rainflow matrix to the crossing intensity.

```
mu_markov = cmat2lc(param_m,Grfc_markov);
muObs_markov = cmat2lc(param_m,Frfc_markov/(T_markov/2));
clf
plot(mu_markov(:,1),mu_markov(:,2),...
   muObs_markov(:,1),muObs_markov(:,2),'--')
title('Theoretical and observed crossing intensity ')
```

The plot, shown in Figure 4.15, compares the theoretical upcrossing intensity mu_markov with the observed upcrossing intensity muObs_markov, as calculated from the theoretical and observed rainflow matrices.

4.4.3 Damage

The WAFO toolbox contains a number of routines to compute and bound the damage, as defined by (4.7), inflicted by a load sequence. The most important routines are cc2dam and cmat2dam which give the total damage from a cycle count and from a cycle matrix, respectively. More detailed information is given by cmat2dmat, which gives a damage matrix, separated for each cycle, from a cycle matrix. An upper bound for total damage from level crossings is given by lc2dplus.

We first calculate the damage by the routines cc2dam for a cycle count (e.g. rainflow cycles) and cmat2dam for a cycle matrix (e.g. rainflow matrix).

```
beta = 4;
Dam_markov = cmat2dam(param_m,Grfc_markov,beta)
DamObs1_markov = cc2dam(u_markov(RFC_markov),beta)/(T_markov/2)
DamObs2_markov = cmat2dam(param_m,Frfc_markov,beta)/(T_markov/2)
```

Here Dam_markov is the theoretical damage per cycle in the assumed Markov chain of turning points, while DamObs1 and DamObs2 give the observed damage per cycle, calculated from the cycle count and from the rainflow matrix, respectively. For this model the result should be

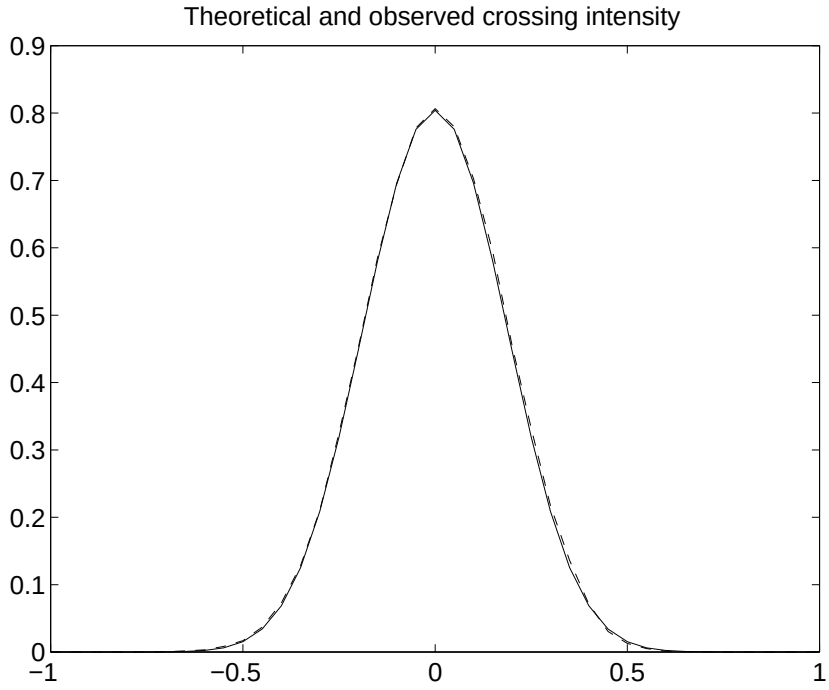


Figure 4.15: *Crossing intensity as calculated from the Markov matrix (solid curve) and from the observed rainflow matrix (dashed curve).*

`Dam_markov = 0.0073` for the theoretical damage and very close to this value for the simulated series.

The damage matrix is calculated by `cmat2dmat`. It shows how the damage is distributed among the different cycles as illustrated in Figure 4.16. The sum of all the elements in the damage matrix gives the total damage.

```
Dmat_markov = cmat2dmat(param_m,Grfc_markov,beta);
DmatObs_markov = cmat2dmat(param_m,Frfc_markov,beta)/(T_markov/2);}
clf
subplot(121), cmatplot(u_markov,u_markov,Dmat_markov,4)
title('Theoretical damage matrix')
subplot(122), cmatplot(u_markov,u_markov,DmatObs_markov,4)
title('Observed damage matrix')
sum(sum(Dmat_markov))
sum(sum(DmatObs_markov))
```

It is possible to calculate an upper bound on the damage intensity from the crossings intensity only, without using the rainflow cycles. This is done by the WAFO routine `lc2dplus`, which works on any theoretical or observed crossing intensity function.

```
Damplus_markov = lc2dplus(mu_markov,beta)
```

4.4.4 Estimation of S-N curve

WAFO contains routines for computation of parameters in the basic S-N curve (4.1), for the relation between the load cycle amplitude s and the fatigue life $N(s)$ in fixed amplitude tests.

$$N(s) = \begin{cases} K^{-1}s^{-\beta} & s > s_\infty, \\ \infty & s \leq s_\infty, \end{cases} \tag{4.8}$$

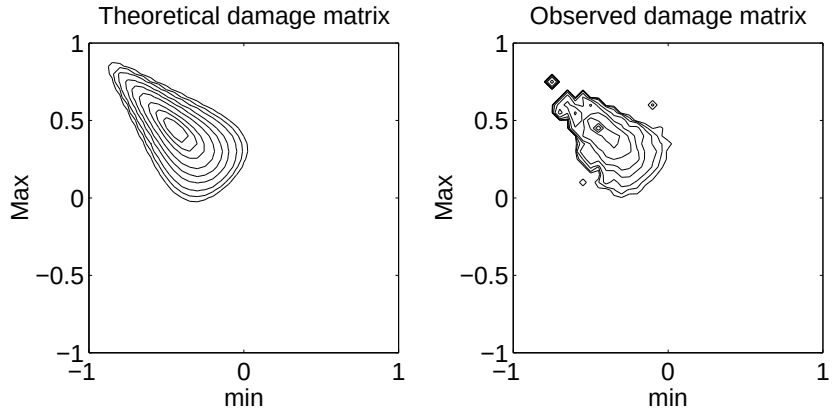


Figure 4.16: *Distribution of damage from different RFC cycles, from calculated theoretical and from observed rainflow matrix.*

where K is a material dependent random variable. The variation of K is often taken to be lognormal,

$$K = E\epsilon^{-1},$$

where ϵ is a fixed parameter, depending on material, and $\ln E$ has a normal distribution with mean 0 and standard deviation σ_E. Thus, there are three parameter, ϵ, β, σ_E, to be estimated from an S-N experiment. Taking logarithms in (4.1) the problem turns into a standard regression problem,

$$\ln N(s) = -\ln E - \ln \epsilon - \beta \ln s,$$

in which the parameters can easily be estimated.

The toolbox contains a data set `SN` with fatigue lives from 40 experiments with s = 10, 15, 20, 25, and 30 MPa, in groups of five. The estimation routine is called `snplot`, which performs both estimation and plotting; see `help snplot`.

First load SN-data and plot in log-log scale.

```
load SN
clf
loglog(N,s,'o'), axis([0 14e5 10 30])
```

To further check the assumptions of the S-N-model we plot the results for each s-level separately on normal probability paper. As seen from Figure 4.17 the assumptions seems acceptable since the data fall of almost parallel straight lines.

```
wnormplot(reshape(log(N),8,5))
```

The estimation is performed and fitted lines plotted in Figure 4.18, with linear and log-log plotting scales:

```
[e0,beta0,s20] = snplot(s,N,12)
title('S-N-data with estimated N(s)','FontSize',20)
set(gca,'FontSize',20)
```

gives linear scale and

```
[e0,beta0,s20] = snplot(s,N,14)
title('S-N-data with estimated N(s)','FontSize',20)
set(gca,'FontSize',20)
```

gives log-log scales.

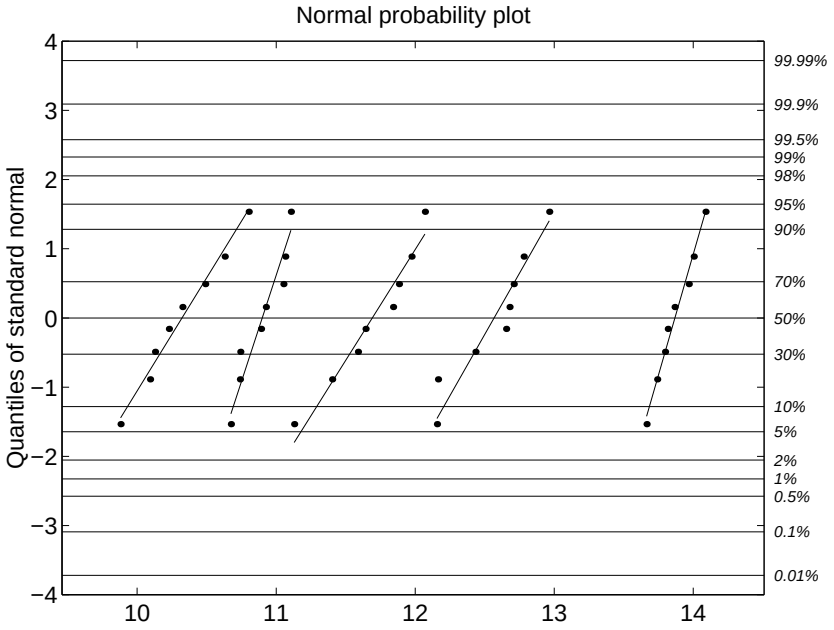


Figure 4.17: *Check of S-N-model on normal probability paper.*

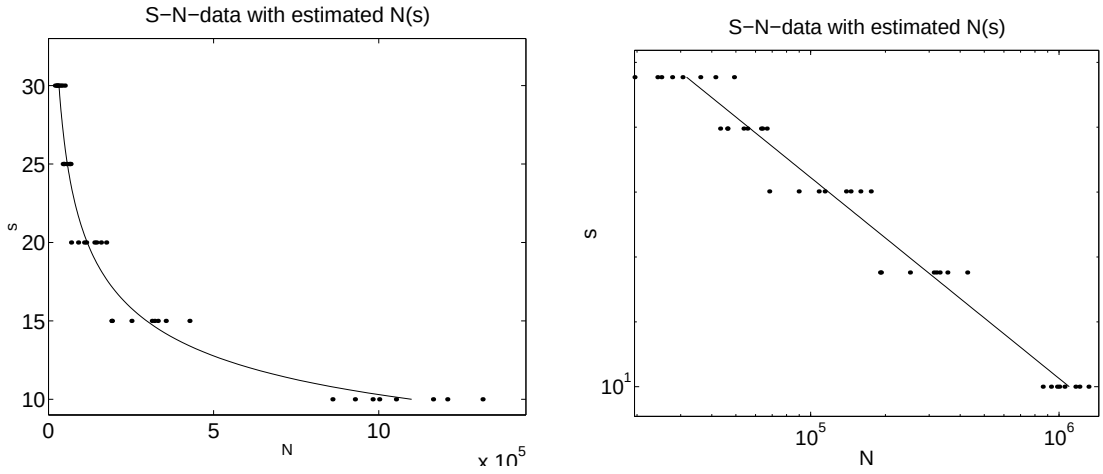


Figure 4.18: *Estimation of S-N-model on linear and log-log scale.*

4.4.5 From S-N-curve to fatigue life distribution

The Palmgren-Miner hypothesis states that fatigue failure occurs when the damage exceeds one, $D(t) > 1$. Thus, if the fatigue failure time is denoted by T_f then

$$P(T_f \leq t) = P(D(t) \geq 1) = P(K \leq \epsilon D_\beta(t)).$$

Here $K = E^{-1}\epsilon$ takes care of the uncertainty in the material. In the previous section we used and estimated a lognormal distribution for the variation of K around ϵ, when we assumed that $\ln K = \ln \epsilon - \ln E$ is normal with mean $\ln \epsilon$ and standard deviation σ_E.

The cycle sum $D_\beta(t)$ is the sum of a large number of damage terms, only dependent on the cycles. For loads with short memory one can assume that $D_\beta(t)$ is approximately normal,

$$D_\beta(t) \approx N(d_\beta t,\, \sigma_\beta^2\, t),$$

where

$$d_\beta = \lim_{t\to\infty} \frac{D_\beta(t)}{t} \qquad \text{and} \qquad \sigma_\beta^2 = \lim_{t\to\infty} \frac{V(D_\beta(t))}{t}.$$

Thus the fatigue life distribution can be computed by combining the lognormal distribution for K with the normal distribution for $D_\beta(t)$. Denoting the standard normal density and distribution functions by $\phi(x)$ and $\Phi(x)$, respectively, an approximate explicit expression for the failure probability within time t is

$$P(T^f \le t) \approx \int_{-\infty}^{\infty} \Phi\left(\frac{\ln\epsilon + \ln d_\beta t + \ln(1 + \frac{\sigma_\beta}{d_\beta\sqrt{t}} z)}{\sigma_E} \right) \phi(z)\, dz. \tag{4.9}$$

We have already estimated the material parameters ϵ = `e0`, β = `beta0`, and σ_E^2 = `s20`, in the S-N data, so we need the damage intensity d_β and its variability σ_β for the acting load.

We first investigate the effect of uncertainty in the β-estimate.

```
beta = 3:0.1:8;
DRFC = cc2dam(RFC_sea,beta);
dRFC = DRFC/T_sea;
plot(beta,dRFC), axis([3 8 0 0.25])
title('Damage intensity as function of \beta')
```

The plot in Figure 4.19 shows the increase in damage with increasing β.

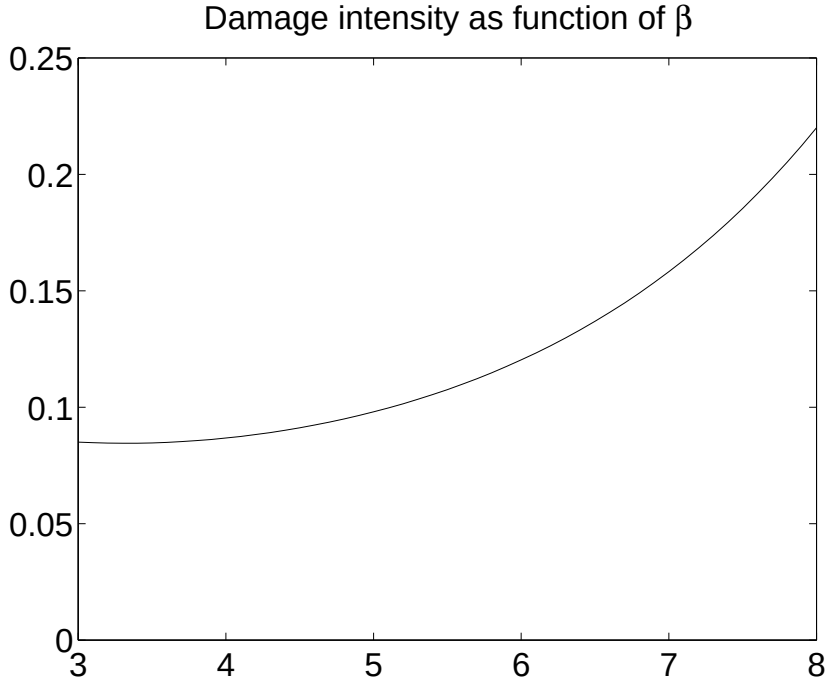


Figure 4.19: *Increasing damage intensity from sea-load with increasing β.*

Next, we shall see how the load effect variability affects the fatigue life. We use three different values for σ_β^2, namely 0, 0.5, and 5. With `beta0`, `e0`, `s20` estimated in Section 4.4.4, we compute and plot the following three possible fatigue life distributions.

```
dam0 = cc2dam(RFC_sea,beta0)/T_sea;
[t0,F0] = ftf(e0,dam0,s20,0.5,1);
[t1,F1] = ftf(e0,dam0,s20,0,1);
[t2,F2] = ftf(e0,dam0,s20,5,1);
plot(t0,F0,t1,F1,t2,F2)
```

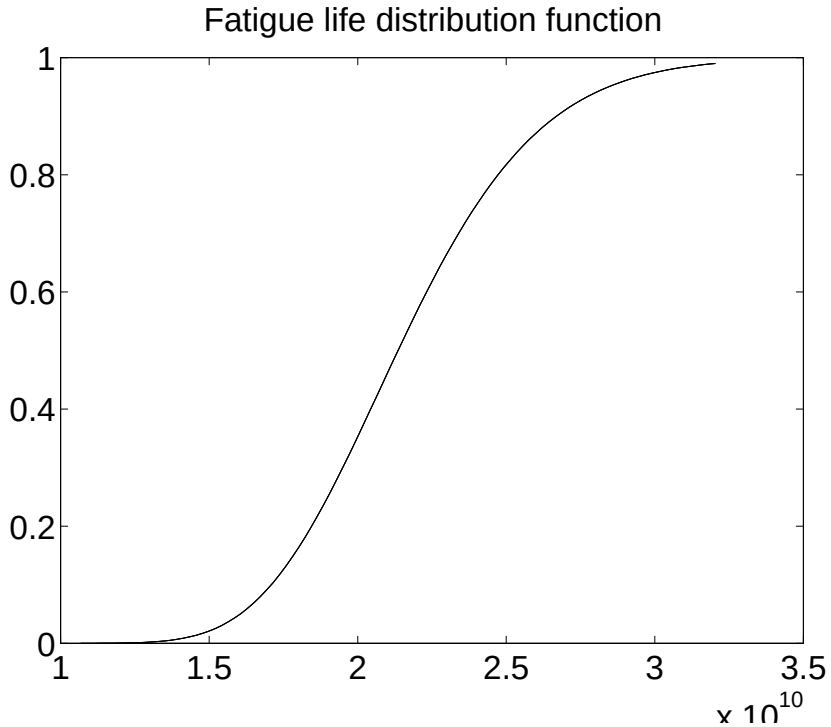


Figure 4.20: *Fatigue life distribution with sea load.*

Here, the fourth parameter is the value of σ_β^2 used in the computation; see `help ftf`.

The resulting fatigue life distribution function is shown in Figure 4.20. As seen, the curves are identical, indicating that the correct value of σ_β^2 is not important for such small ϵ-values as are at hand here. Hence, one can use $\sigma_\beta^2 = 0$, and assume that the damage accumulation process is proportional to time.

4.4.6 Fatigue analysis of complex loads

Loads which cause fatigue are rarely of the homogeneous and stationary character as the loads used in the previous sections. On the contrary, typical load characteristics often change their value during the life time och a structure, for example, load spectra on an airplane part have very different fatigue properties during the different stages of an air mission. Marin loads on a ship are quite different during the loading and unloading phase, compared to a loaded ocean voyage, and the same holds for any road vehicle.

The WAFOcan be used to analyze also loads of complex structure and we shall illustrate some of these capabilities in this section. The To be eligible for WAFO-analysis the loads has to have a piecewise stationary character, for example the mean level or the standard deviation may take two distinct levels and change abruptly, or the frequency content can alternate between two modes, one irregular and one more regular. Such processes are called *switching processes*. A flexible family of switching loads are those where the change between the different stationary states is governed by a Markov chain. WAFO contains a special package of routines for analysis of such switching Markov loads, based on methods from [20, 22].

In the following example the load alternates between two different mean levels, corresponding to one heavy-load state (1) and one light-load state (2). In Figure 4.21 the observed load is shown

in the upper part. The alternating curve in the lower part shows where the load switches between the two states.

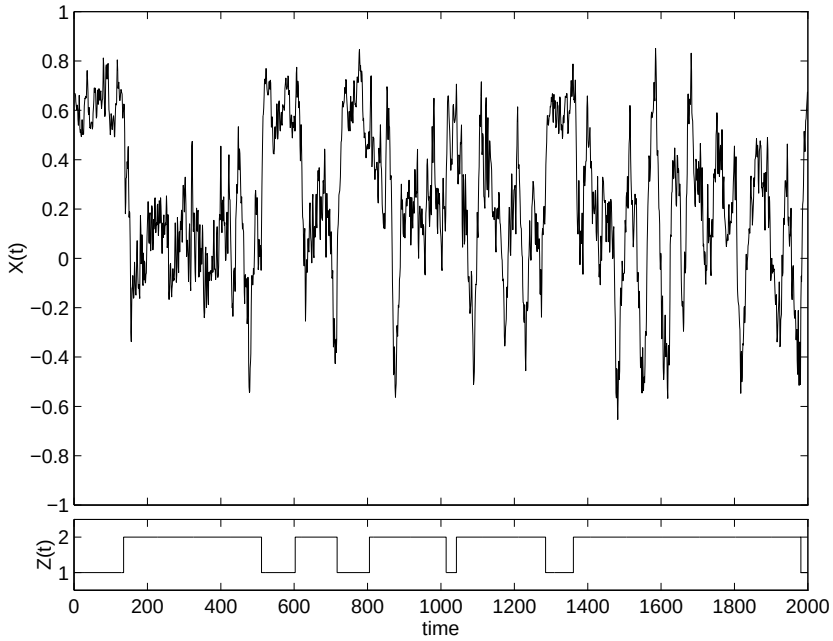


Figure 4.21: *Simulated switching load with two states. Upper graph shows the load, and the states are indicated in the lower graph.*

As long as the load is in one of the states the rainflow cycles are made up of alternations between turning points belonging only to that part of the load. When the state changes there is introduced extra rainflow cycle with larger amplitude amplitude. These extra cycles can be seen in the total rainflow matrix, shown in Figure 4.22. The two large groups of cycles around (min,max) = (0.5, 0.75) and (min,max) = (0, 0) come from states (1) and (2), respectively. The contribution from the switching is seen in the small assembly of cycles around (min,max) = (-0.5, 1).

More details on how to analyse and model switching loads can be found in [19].

(a) Rainflow matrix (b) Rainflow matrix

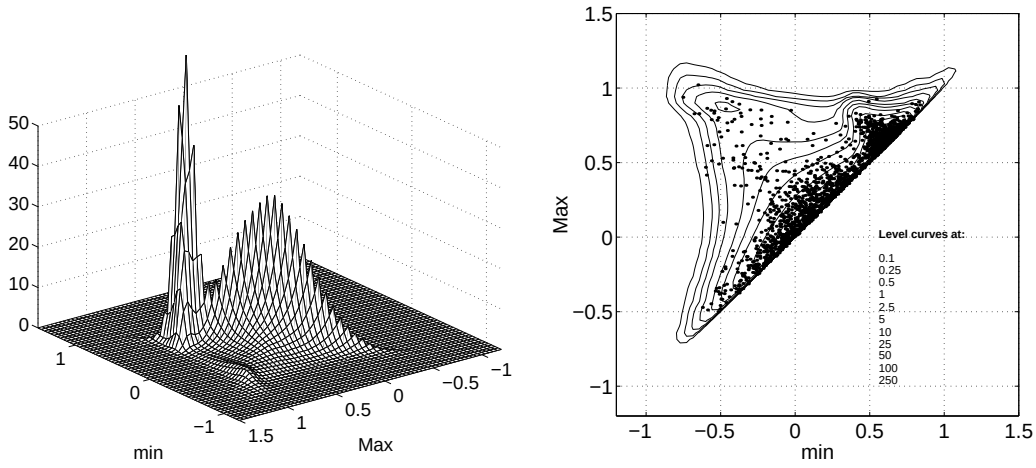


Figure 4.22: *3D-plot (left) and isolines (right) of calculated rainflow matrix for switching load in Figure 4.21. The dots in the right figure are the observed rainflow cycles.*

CHAPTER 5
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. WAFO 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).

5.1 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

$$\text{Weibull:} \qquad F_W(x;a,c) = 1 - e^{-(x/a)^c}, \quad x > 0, \tag{5.1}$$

$$\text{Gumbel:} \qquad F_G(x;a,b) = \exp\left(-e^{-(x-b)/a}\right), \quad -\infty < x < \infty. \tag{5.2}$$

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 $x_1, x_2, \ldots, x_n$ by changing the sign of the maximum of $-x_1, -x_2, \ldots, -x_n$, 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 5.2.

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 (5.1) and (5.2). The file `atlantic.dat` contains significant wave-height data recorded approximately 14 times a month in the Atlantic Ocean in December – February during seven years and at two locations. The data is stored in the vector `Hs`. We try to fit a Weibull distribution to this data set:

```
Hs = load('atlantic.dat');
wei = wweibplot(Hs)
```

This will result in a two element vector `wei = [ahat chat]` with estimated values of the parameters (a, c) in (5.1). 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 5.1(a). Obviously, a Weibull distribution is not very well suited to describe the significant wave-height data. To illustrate the use of the Gumbel distribution we plot and estimate the parameters (a, b) in the Gumbel distribution (5.2) for the data in `Hs`. The command

```
gum=wgumbplot(Hs)
```

results in a vector `gum` with estimated values `[ahat bhat]` and the plot in Figure 5.1(b). Here the horizontal axis is linear in the observations x and the vertical axis carries the reduced variable $-\log(-\log(F(x)))$. The data shows a better fit to the Gumbel than to a Weibull distribution. A distribution which is often hard to distinguish from the Gumbel distribution is the Lognormal distribution, and making a Normal probability plot of the logarithm of `Hs` in Figure 5.1(c) also shows a good fit:

```
wnormplot(log(Hs),1,0);
```

The parameter estimation in `wgumbplot` and `wweibplot` is done by fitting a straight line to the empirical distribution functions in the diagrams and using the relations

$$\log\{-\log[1 - F_W(x; a, c)]\} = c\log(x) - c\log(a) \tag{5.3}$$

and

$$-\log\{-\log[F_G(x; a, b)]\} = x/a - b/a, \tag{5.4}$$

to relate parameters to intercepts and slopes of the estimated lines. In the following section we shall describe some more statistical techniques for parameter estimation in the Generalized Extreme Value distribution.

5.2 Generalized Pareto and Extreme Value distributions

The Generalized Pareto Distribution (GPD) has the distribution function

$$\text{GPD:} \qquad F(x; k, \sigma) = \begin{cases} 1 - (1 - kx/\sigma)^{1/k}, & \text{if } k \neq 0, \\ 1 - \exp\{-x/\sigma\}, & \text{if } k = 0, \end{cases} \tag{5.5}$$

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

$$\text{GEV:} \qquad F(x; k, \mu, \sigma) = \begin{cases} \exp\left\{-(1 - k(x - \mu)/\sigma)^{1/k}\right\}, & \text{if } k \neq 0, \\ \exp\left\{-\exp\{-(x - \mu)/\sigma\}\right\}, & \text{if } k = 0, \end{cases} \tag{5.6}$$

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

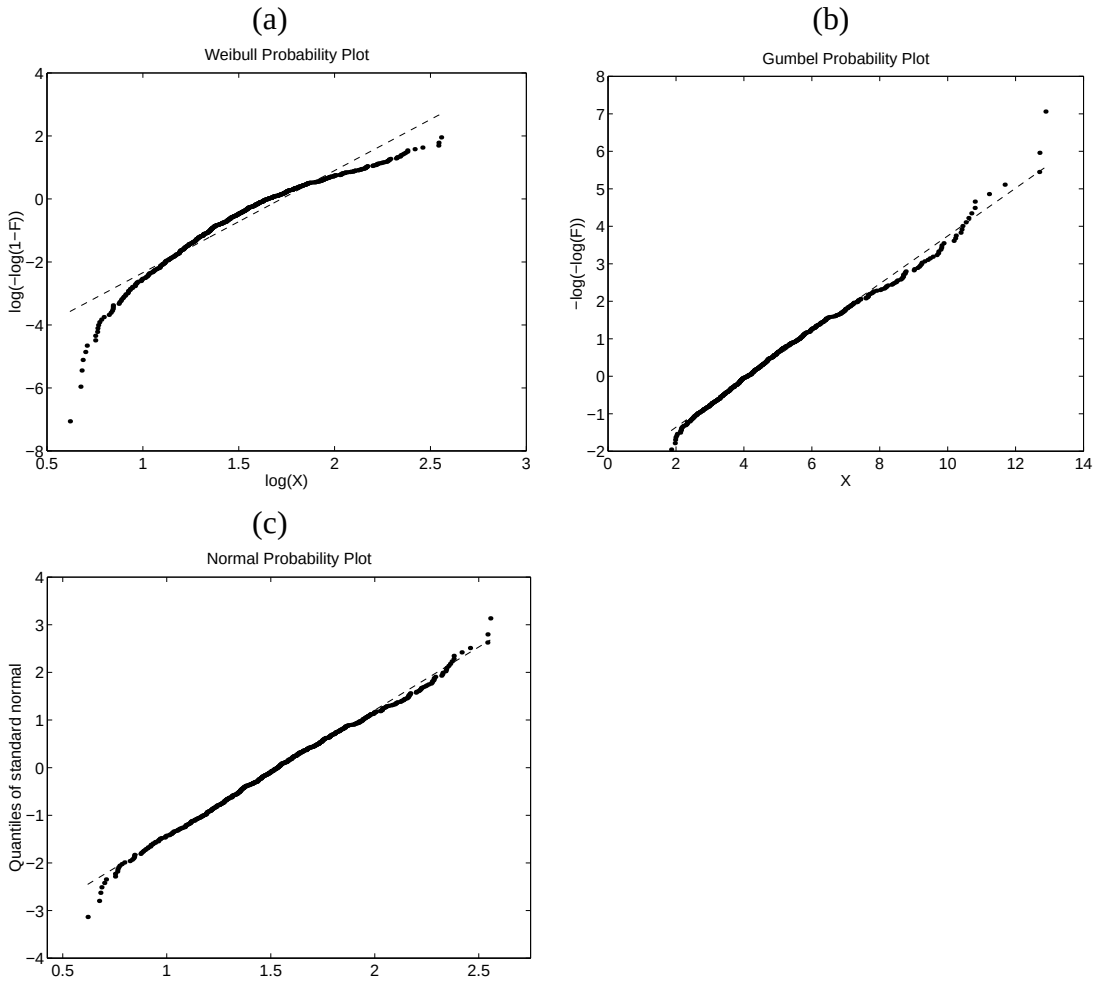


Figure 5.1: *(a) Significant wave-height data on Weibull paper, (b) on Gumbel paper and (c) logarithm of data on Normal probability paper.*

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. WAFO contains algorithms for plotting of distributions and estimation of parameters with four different methods, suitable in different regions.

5.2.1 Generalized Extreme Value distribution

For the Generalized Extreme Value (GEV) distribution the estimation methods used in WAFO are the Maximum Likelihood (ML) method and the method with Probability Weighted Moments (PWM), described in [43, 17]. The programs have been adapted to MATLAB from a package of S-Plus routines described in [4].

We start with the significant wave-height data in `Hs`. The command

```
[gev cov]=wgevfit(Hs);
```

will give estimates `gev = [khat sigmahat muhat]` of the parameters (k, σ, μ) in the GEV distribution (5.6) based on data `Hs`. 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 5.2.

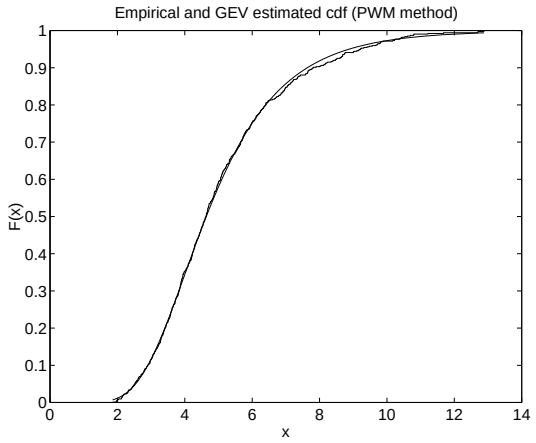


Figure 5.2: *Empirical distribution of significant wave-height with estimated Generalized Extreme Value distribution.*

The default estimation algorithm for the GEV distribution is the method with Probability Weighted Moments (PWM). An optional second argument, `wgevfit(Hs, method)`, allows a choice between the PWM-method (when `method = 'pwm'`) and the alternative ML-method (when `method = 'ml'`). 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.

5.2.2 Generalized Pareto distribution

For the Generalized Pareto distribution (GPD) the WAFO uses the method with Probability Weighted Moments (PWM), described in [18], and the standard method of Moments (MOM), as well as a general method suggested by Pickands in [40]. S-Plus routines for these methods are described in [4].

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 exceedances of thresholds 3 and 7 of values in `Hs`, one uses thecommands

```
[gpd3  cov] = wgpdfit(Hs(Hs>3)-3);
figure
[gpd7  cov] = wgpdfit(Hs(Hs>7)-7);
```

This will give estimates `gpd = [khat sigmahat]` of the parameters (k, σ) in the Generalized Pareto distribution (5.5) based on exceedance data `Hs(Hs>u)-u`. 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 5.3. Here the fit is better for exceedances over level 7 but there are less data available.

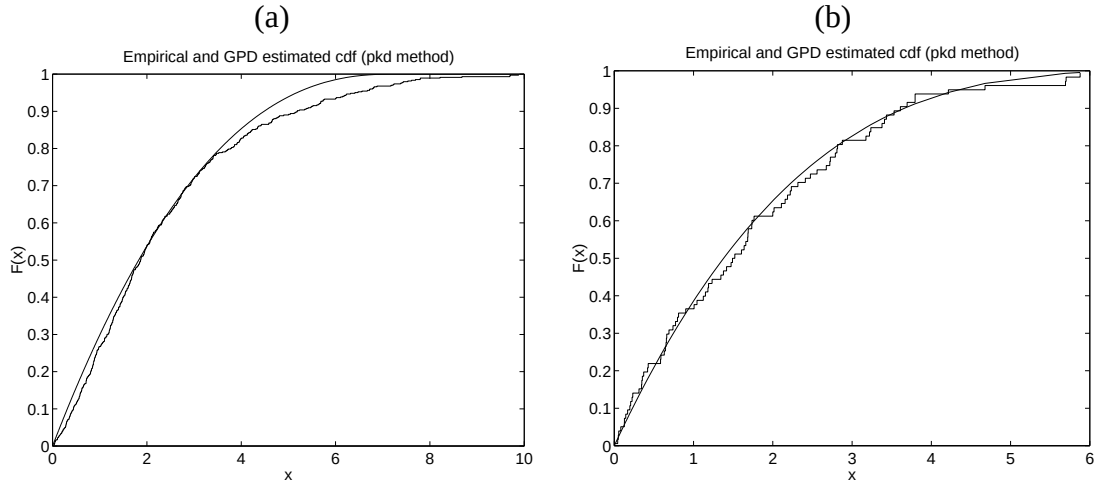


Figure 5.3: *(a) Exceedances of significant wave-height data over level 3, (b) over level 7.*

The choice of estimation method is rather dependent on the actual parameter values. The default estimation algorithm in the WAFO for estimation in the Generalized Pareto distribution is the Pickands' (PKD) estimator. This estimator gives generally good estimates of the parameter k in (5.5) for $-5 \leq k \leq 5$, and it is also recommended for estimation of σ if $k < -0.5$. The optional second argument, `wgpdfit(Hs(Hs>u)-u, method)`, gives a choice between Pickands' method (when `method = 'pkd'`), the Moment method (when `method = 'mom'`), and the PWM-method (when `method = 'pwm'`). 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 WAFO. The commands

```
Rgev = wgevrnd(0.3,1,2,1,100);
empdistr(Rgev);
hold on
gp = wgevfit(Rgev,'pwm',[],0);
x=sort(Rgev);
plot(x,wgevcdf(x,gp(1),gp(2),gp(3)))
gm = wgevfit(Rgev,'ml',gp,0);
plot(x,wgevcdf(x,gm(1),gm(2),gm(3)),'--')
hold off
```

simulates 100 values from the GEV distribution with parameters $(0.3, 1, 2)$, then estimates the parameters using two different methods and plots the estimated distribution functions together with the empirical distribution. Similarly for the GPD distribution;

```
Rgpd = wgpdrnd(0.4,1,1,100);
empdistr(Rgpd);
hold on
gp = wgpdfit(Rgpd,'pkd',0);
x=sort(Rgpd);
plot(x,wgpdcdf(x,gp(1),gp(2)))
gm = wgpdfit(Rgpd,'mom',0);
plot(x,wgpdcdf(x,gm(1),gm(2)),'--')
gw = wgpdfit(Rgpd,'pwm',0);
plot(x,wgpdcdf(x,gw(1),gw(2)),':')
hold off
```

with the three different methods of parameter estimation. The results are shown in Figure 5.4(a) and (b).

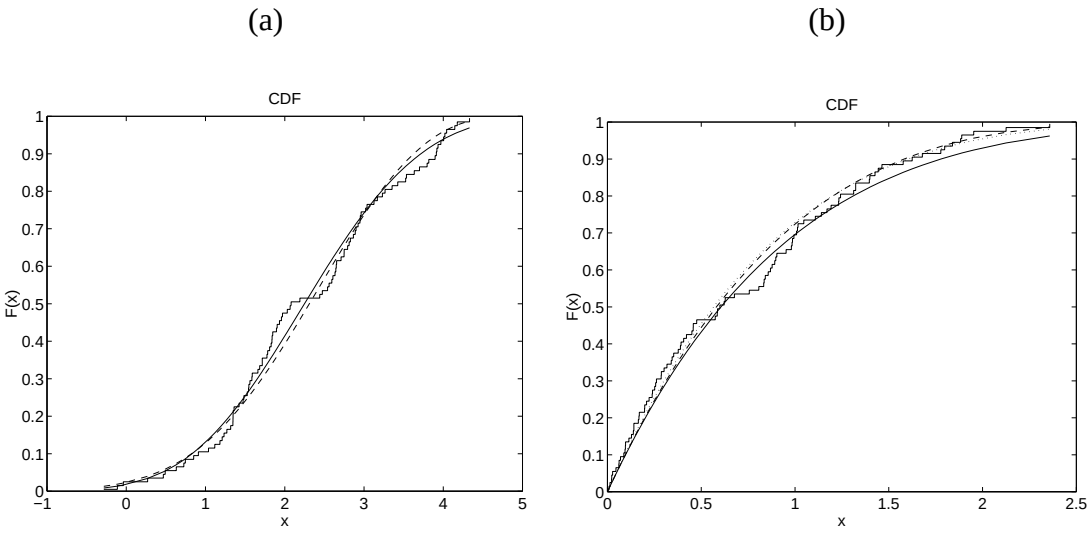


Figure 5.4: *Empirical distributions and estimated distribution functions for 100 observations of GEV (a) and GPD (b) variables.*

WAFO contains random number generators from a broad variety of distributions; see `help Contents` and Section 5.4.

5.3 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 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 simplest distribution to fit to the exceedances over a level u is the Generalized Pareto distribution, GPD, with distribution function (5.5). 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:

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

The following commands illustrate this for the significant wave height data:

```
u=linspace(2,10,200);
for i=1:length(u)
  m(i)=mean(Hs(Hs>u(i)));
end
plot(u,m-u)
xlabel('u')
title('Mean exceedance over level u')
```

The result is plotted in Figure 5.5, and seems for $u \geq 7$ exhibit an almost linear relationship.

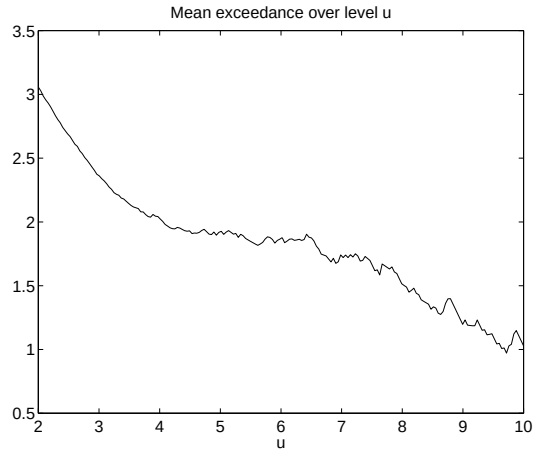


Figure 5.5: *Estimated expected exceedance over level u as function of u.*

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 during a year. If the distribution of the individual significant wave height exceedances is GPD one can easily find e.g. the distribution of the largest value of a fixed number of exceedances. However, the number of exceedances is not fixed but random, and then one has to combine the distribution of the random individual exceedances with the random number of exceedances N before one can say anything about the total maximum. If the level u is high we can, due to the Poisson approximation of the Binomial distribution and neglecting the dependence of nearby values, assume N to have an approximate Poisson distribution.

Now 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 N is a Poisson distributed random variable with mean μ, and $M_N = \max(X_1, X_2, \ldots, X_N)$ is the maximum of N independent GPD variables then,

$$P(M_N \leq x) = \sum_{n=0}^{\infty} P(N = n) \cdot P(X_1 \leq x, X_2 \leq x, \ldots, X_n \leq x)$$

$$= \sum_{n=0}^{\infty} e^{-\mu} \frac{\mu^n}{n!} \cdot \left(1 - (1 - k\frac{x}{\sigma})^{1/k}\right)^n$$
$$= \exp\left\{-(1 - k(x-a)/b)^{1/k}\right\},$$

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

This means that we can estimate the distribution of the maximum significant wave height during a winter (December – February) month from our data set `Hs` by fitting a GPD to the exceedances over some level u, estimating μ by the number of exceedances N divided by the number of months ($7 \times 3 \times 2 = 42$) and use the above relation to fit a GEV distribution:

```
gpd7=wgpdfit(Hs(Hs>7)-7,'pwm',0);
khat=gpd7(1);
sigmahat=gpd7(2);
muhat=length(Hs(Hs>7))/(7*3*2);
bhat=sigmahat/muhat^khat;
ahat=7-(bhat-sigmahat)/khat;
x=linspace(5,15,200);
plot(x,wgevcdf(x,khat,bhat,ahat))
```

We have here used the threshold $u = 7$ since the exceedances over this level seems to fit well to a GPD distribution in Figures 5.3(b) and 5.5. A larger value will improve the Poisson approximation to the number of exceedances but give us less data to estimate the parameters.

Since we have data to compute the monthly maxima `mm` over 42 months we can also try to fit a GEV distribution directly:

```
for i=1:41                          % Last month is not complete
 mm(i)=max(Hs(((i-1)*14+1):i*14)); % Approx. 14 values each month
end
gev=wgevfit(mm);
hold on
plot(x,wgevcdf(x,gev(1),gev(2),gev(3)),'--')
empdistr(mm)
hold off
```

The results of the two methods agree very well in this case as can be seen in Figure 5.6 where the estimated distributions are plotted together with the empirical distribution of `mm`.

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.

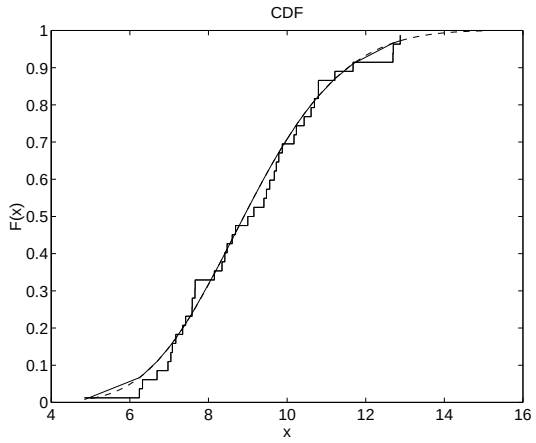


Figure 5.6: *Estimated distribution functions of monthly maxima with the POT method (solid), fitting a GEV (dashed) and the empirical distribution.*

5.4 Summary of extreme value procedures

```
help wstats

  WAFO  Toolbox /wstats
  Version 1.0.2   20-June-2000

  Parameter estimation
    wgevfit    - Parameter estimates for GEV data
    wgevlike   - Is an internal routine for wgevfit
    wgpdfit    - Parameter estimates for GPD data
    wgumbfit   - Parameter estimates and confidence intervals for Gumbel data
    wgumblike  - Is an internal routine for wgumbfit
    wraylfit   - Parameter estimates and confidence intervals for Rayleigh data
    wweibfit   - Parameter estimates for Weibull data
    wweibcfit  - Is an internal routine for wweibfit
    wkurtosis  - Computes sample kurtosis
    wskewness  - Computes sample skewness

  Probability density functions (pdf)
    wchi2pdf   - Chi squared probability density function
    wgampdf    - Gamma probability density function
    wgevpdf    - Generalized Extreme Value probability density function
    wgpdpdf    - Generalized Pareto probability density function
    wgumbpdf   - Gumbel probability density function
    wnormpdf   - Normal probability density function
    wraylpdf   - Rayleigh probability density function
    wweibpdf   - Weibull probability density function

  Cumulative distribution functions (cdf)
    wchi2cdf   - Chi squared cumulative distribution function
    wgamcdf    - Gamma cumulative distribution function
    wgevcdf    - Generalized Extreme Value cumulative distribution function
    wgpdcdf    - Generalized Pareto cumulative distribution function
    wgumbcdf   - Gumbel cumulative distribution function
    wnormcdf   - Normal cumulative distribution function
    wraylcdf   - Rayleigh cumulative distribution function
    wweibcdf   - Weibull cumulative distribution function

  Inverse cumulative distribution functions
    wchi2inv   - Inverse of the Chi squared distribution function
    wgaminv    - Inverse of the Gamma distribution function
    wgevinv    - Inverse of the Generalized Extreme Value distribution function
    wgpdinv    - Inverse of the Generalized Pareto distribution function
    wgumbinv   - Inverse of the Gumbell cumulative distribution function
    wnorminv   - Inverse of the Normal distribution function
    wraylinv   - Inverse of the Rayleigh distribution function
    wweibinv   - Inverse of the Weibull distribution function

  Random number generators
    walpharnd  - Random matrices from a symmetric alpha-stable distribution
    wchi2rnd   - Random matrices from a Chi squared distribution
    wgamrnd    - Random matrices from a Gamma distribution
    wgevrnd    - Random matrices from a Generalized Extreme-Value distribution
    wgpdrnd    - Random matrices from a Generalized Pareto Distribution
    wgumbrnd   - Random matrices from the Gumbel distribution.
    wmnormrnd  - Random vectors from the multivariate Normal distribution
    wnormrnd   - Random matrices from the Normal distribution
```

```
  wraylrnd   - Random matrices from a Rayleigh distribution
  wweibrnd   - Random matrices from the Weibull distribution

Statistical plotting
  wgumbplot  - Plots data on a Gumbel distribution paper
  wnormplot  - Plots data on a normal distribution paper
  wqqplot    - Plots empirical quantile vs empirical quantile
  wraylplot  - Plots data on a Rayleigh distribution paper
  wweibplot  - Plots data on a Weibull distribution paper
  whisto     - Plots a histogram
  empdistr   - Computes and plots the empirical CDF
  cempdistr  - Computes and plots the conditional empirical CDF

Statistics
  wchi2stat  - Mean and variance for the Chi squared distribution
  wgamstat   - Mean and variance for the Gamma distribution
  wgevstat   - Mean and variance for the GEV distribution
  wgpdstat   - Mean and variance for the Generalized Pareto distribution
  wgumbstat  - Mean and variance for the Gumbel distribution
  wnormstat  - Mean and variance for the Normal distribution
  wraylstat  - Mean and variance for the Rayleigh
  wweibstat  - Mean and variance for the Weibull distribution

Hypothesis Tests
  wgumbtest  - Tests whether the shape parameter in a GEV is equal to zero
```

REFERENCES

[1] Aage, C., Allan, T., Carter, D.J.T, Lindgren, G. and Olagnon, M. (1998): *Oceans from Space – A textbook for Offshore Engineers and Naval Architects*, Repères océans 16, IFREMER, Brest, pp. 1 - 163.

[2] Adler, R. J. (1981). *The Geometry of Random Fields*. J. Wiley & Sons.

[3] R. Ambartzumian, A. Der Kiureghian, V. Ohanian and H. Sukiasian (1998) Multinormal probabilities by sequential conditioned importance sampling: theory and application *Probabilistic Engineering Mechanics*, **13**, No 4. pp 299-308

[4] Borg, S. (1992): XS - a statistical program package in Splus for extreme-value analysis. Dept. of Mathematical Statistics, Lund University, 1992:E2.

[5] Brodtkorb, P.A. (2000): Numerical evaluation of singular multivariate normal expectations. (In preparation)

[6] Brodtkorb, P.A. (2000): The probability of occurrence of dangerous wave situations at sea. PhD thesis in preparation.

[7] Brodtkorb, P.A., Myrhaug D., Rue, H. (1999). Joint Distribution of Wave Height and Wave Crest Velocity from Reconstructed Data. *Proceedings of the 9th ISOPE Conference*, Vol III, pp. 66-73.

[8] Brodtkorb, P.A., van Ishegem, S., Olagnon, M., Provosto, M., and Rychlik, I. (2000). In preparation.

[9] Buows, E., Gunther, H., Rosenthal, W. and Vincent, C.L. (1985) 'Similarity of the wind wave spectrum in finite depth water: 1 spectral form.' *J. Geophys. Res.*, **90**, No. C1, pp 975-986.

[10] Cavanié, A., Arhan, M. and Ezraty, R. (1976). A statistical relationship between individual heights and periods of storm waves. *Proceedings, BOSS'76*, Trondheim, pp. 354–360.

[11] Dietrich C.R. and Newsam G.N. (1997). Fast and exact simulation of stationary Gaussian process through circulant embedding of the Covariance matrix. *SIAM J. Sci. Compt.*, **18**, pp. 1088-1107.

[12] Falk, M., Hüsler, J. and Reiss, R.-D. (1994): *Laws of Small Numbers: Extremes and rare events*. Birkhäuser, Basel.

[13] Frendahl, M. and Rychlik, I. (1993). Rainflow analysis: Markov method. *Int. J. Fatigue*, **15**(4), 265-272.

[14] Frendahl, M., Lindgren G. and Rychlik I. (1993). *Fatigue Analysis Toolbox*. Dept. of Math. Stat., Lund University.

[15] Genz, A. (1992): Numerical Computation of Multivariate Normal Probabilites. *J. Computational Graphical Statistics*, Vol.1, pp 141–149

[16] Holm S. and de Maré J. (1985): Generation of random processes for fatigue testing. *Stoch. Proc. Appl.* **20**, 149–156.

[17] 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.

[18] Hosking, J.R.M. and Wallis, J.R. (1987): Parameter and quantile estimation for the generalized Pareto distribution. *Technometrics*, **29**, pp. 339-349.

[19] P. Johannesson. *Matlab Toolbox: Rainflow Cycles for Switching Processes, V. 1.0*. Manual TFMS–7008, Department of Mathematical Statistics, Lund Institute of Technology, 1997.

[20] Johannesson, P. (1998). Rainflow Cycles for Switching Processes with Markov Structure. *Probability in the Engineering and Informational Sciences* **12**(2), pp. 143-175.

[21] P. Johannesson. Rainflow matrix for switching random loads. In B. F. Spencer, Jr. and E. A. Johnson, editors, *Stochastic Structural Dynamics*, pages 281–286. Balkema, Rotterdam, 1999.

[22] Johannesson, P. (1999). *Rainflow Analysis of Switching Markov Loads*. PhD thesis, Mathematical Statistics, Centre for Mathematical Sciences, Lund University.

[23] Krogstad, H.E. and Barstow, S.F. (1999). Directional Distributions in Ocean Wave Spectra. *Proceedings of the 9th ISOPE Conference*, Vol III, pp. 79-86.

[24] Krogstad, H.E., Wolf, J., Thompson, S.P., and Wyatt, L.R. (1999) Methods for intercomparison of wave measurements. *Coastal Enginering*, Vol. 37, pp. 235–257

[25] Leadbetter, M. R., Lindgren, G. and Rootzén, H. (1983): *Extremes and related properties of random sequences and processes.* Springer Verlag, New York.

[26] Lindgren, G. (1972): Some properties of a normal process near a local maximum. *Ann. Math. Statist.* **41**, pp. 1870-1883.

[27] Lindgren, G. (1972): Wave-length and amplitude in Gaussian noise. *Adv. Appl. Probab.* **4**, pp. 81-108.

[28] Lindgren, G. (1972): Wave-length and amplitude for a stationary Gaussian process after a high maximum. *Z. Wahrscheinlichkeitstheorie verw. Geb.* **23**, pp. 293-326.

[29] Lindgren, G. (1972): Local maxima of Gaussian fields. *Ark. Matematik*, **10**, pp. 195-218.

[30] Lindgren, G. (1973): Discrete wave-analysis of continuous stochastic processes. *Stoch. Processes Appl.* **1**, pp. 83-105.

[31] Lindgren, G. and Rychlik, I. (1982): Wave characteristic distributions for Gaussian waves – wave-length, amplitude and steepness. *Ocean Engineering*, **9**, pp. 411-432.

[32] Lindgren, G. and Rychlik, I. (1991): Slepian models and regression approximations in crossings and extreme value theory. *International Statistical Review*, **59**, pp. 195-225.

[33] Longuet-Higgins, M.S. (1975) On the joint distribution wave periods and amplitudes of sea waves, *J. geophys. Res.* **80**, pp 2688–2694

[34] Longuet-Higgins, M.S. (1983). On the joint distribution wave periods and amplitudes in a random wave field. *Proc. R. Soc.* **A389**, pp 24–258.

[35] Machado, U. (2000) Licentiat Thesis, in preparation.

[36] Matsuishi M. & Endo T.: *Fatigue of metals subject to varying stress*. Paper presented to Japan Soc. Mech. Engrs, Jukvoka, Japan (1968).

[37] Marthinsen, T. and Winterstein, S.R (1992) On the skewness of random surface waves. In *Proceedings of the 2nd ISOPE Conference,* San Francisco, 14-19 june.

[38] Michel K. Ochi (1998), *OCEAN WAVES, The stochastic approach,* OCEAN TECHNOLOGY series 6, Cambridge, pp 255-275.

[39] Ochi, M.K. and Ahn, K. (1994): Probability distribution applicable to non-Gaussian random processes. *Probabilistic Engineering Mechanics*, **9**, 255-264.

[40] Pickands III, J. (1975): Statistical inference using extreme order statistics. *Annals of Statistics*, **3**, pp. 119-131.

[41] Podgórski, K., Rychlik I. and Machado, U. E. B. (2000). Exact Distributions for Apparent Waves in Irregular Seas. *Ocean. Engng.* **27**(9), 979-1016.

[42] Podgórski, K., Rychlik, I., Rydén, J. and Sjö, E. (1999). How big are the big waves? *Proceedings of the 9th ISOPE Conference*, Vol III, pp. 53-60.

[43] Prescott, P. and Walden, A.T. (1980): Maximum likelihood estimation of the parameters of the generalized extreme-value distribution. *Biometrika*, **67**, pp. 723-724.

[44] Rice, S.O. (1944-45): Mathematical analysis of random noise. *Bell Syst. Techn. J.* **23-24**, pp. 282-332 and 46-156.

[45] Rychlik, I. (1987): Regression approximations of wavelength and amplitude distributions. *Adv. Appl. Probab.* **19**, pp. 396-430.

[46] Rychlik I.: A new definition of the rainflow cycle counting method. *Int. J. Fatigue*, **9** (1987) 119-121.

[47] I. Rychlik. Rain-Flow-Cycle distribution for ergodic load processes. *SIAM Journal on Applied Mathematics*, 48:662–679, 1988.

[48] Rychlik I. (1992) Confidence bands for linear regressions, *Commun. Statist. -simula.*, **21**, No 2, pp 333–352

[49] Rychlik I. (1993): Note on cycle counts in irregular loads. *Fatigue Fract. Eng. Mater. and Struc.*, **16**, pp. 377-390.

[50] Rychlik I. (1995): A note on significant wave height. To appear in *Ocean Engineering.*

[51] Rychlik, I. (1995): Simulation of load sequences from Rainflow matrices: Markov method. Stat. Research Report, Dept. of Mathematical Statistics, Lund, 1995:29, 1-23.

[52] 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.

[53] Rychlik I. and Lindgren G. (1995). *WAVE Analysis Toolbox – a Tutorial.* Dept. of Math. Stat., Lund University.

[54] Rychlik I., Johannesson P., Leadbetter M. R. (1997). Modelling and Statistical Analysis of Ocean-Wave Data using transformed Gaussian processes. *Marine Structures*, **10**, 13-47.

[55] Silverman, B.W. (1986). *Density estimation for statistics and data analysis*. Monographs on statistics and applied probability, Chapman and Hall.

[56] Sjö, E. (2000): *Crossings and maxima in Gussian fields and seas*. PhD thesis, Mathematical Statistics, Centre for Mathematical Sciences, Lund University.

[57] Tayfun, M.A. (1981). Breaking limited waveheights. *Journal of the Waterway, port and coastal and ocean division* **107**, pp. 59-69.

[58] Tayfun, M.A. (1990). Distribution of large waveheights. *Journal of the Waterway, port and coastal and ocean division* **116**, pp. 686-707

[59] Torsethaugen, K. (1996). Model for a doubly peaked wave spectrum. Report No. STF22 A96204. SINTEF Civil and Environm. Engineering, Trondheim.

[60] Tucker, M.J. (1993) Recommended standard for wave data sampling and near-real-time processing. *Ocean Engineering*, **20**, No.5, pp. 459–474

[61] Wand, M.P., and Jones, M.C. (1995). *Kernel smoothing*. Chapman and Hall.

[62] Winterstein, S.R. (1988) Nonlinear vibration models for extremes and fatigue. *J. Engng. Mech.*, ASCE, **114**, No 10, pp 1772-1790

[63] Winterstein, S.R, Ude, T.C. and Kleiven, G. (1994) *Springing and slow drift responces: predicted extremes and fatigue vs. simulation* In Proc. 7th International behaviour of Offshore structures, (BOSS) **3**, pp.1-15

[64] Young, I.R. (1999) *Wind generated ocean waves* Elsevier Ocean Engineering Book Series, Vol. 2, pp 239

INDEX

August 2000, Version 2.0.02

Mathematical Statistics
Centre for Mathematical Sciences
Lund University
Box 118, SE-221 00 Lund, Sweden
http://www.maths.lth.se/