– a Matlab Toolbox for Analysis of Random Waves and Loads

Tutorial for WAFO version 2.5

by the WAFO group

Lund, March 2011

LUND UNIVERSITY

FACULTY OF ENGINEERING
CENTRE FOR MATHEMATICAL SCIENCES
MATHEMATICAL STATISTICS

Mathematical Statistics
Lund University
Box 118
SE-221 00 Lund
Sweden
http://www.maths.lth.se/

in the Hope that it is Useful

Part I

Front Matter

Foreword

This is a tutorial for how to use the MATLAB toolbox WAFO for analysis and simulation of random waves and random fatigue. The toolbox consists of a number of MATLAB m-files together with executable routines from FORTRAN or C++ source, and it requires only a standard MATLAB setup, with no additional toolboxes.

A main and unique feature of WAFO is the module of routines for computation of the exact statistical distributions of wave and cycle characteristics in a Gaussian wave or load process. The routines are described in a series of examples on wave data from sea surface measurements and other load sequences. There are also sections for fatigue analysis and for general extreme value analysis. Although the main applications at hand are from marine and reliability engineering, the routines are useful for many other applications of Gaussian and related stochastic processes.

The routines are based on algorithms for extreme value and crossing analysis, developed over many years by the authors as well as many results available in the literature. References are given to the source of the algorithms whenever it is possible. These references are given in the MATLAB-code for all the routines and they are also listed in the last section of this tutorial. If the references are not used explicitly in the tutorial; it means that it is referred to in one of the MATLAB m-files.

Besides the dedicated wave and fatigue analysis routines the toolbox contains many statistical simulation and estimation routines for general use, and it can therefore be used as a toolbox for statistical work. These routines are listed, but not explicitly explained in this tutorial.

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 routines in the tutorial are tested on WAFO-version 2.5, which was made available in beta-version in January 2009 and in a stable version in February 2011.

The persons that take actively part in creating this tutorial are (in alphabetical order): *Per Andreas Brodtkorb*[1], *Pär Johannesson*[2], *Georg Lindgren*[3], *Igor Rychlik*[4].

[1]Norwegian Defense Research Establishment, Horten, Norway.
[2]SP Technical Research Institute, Borås, Sweden.
[3]Centre for Mathematical Sciences, Lund University, Sweden.
[4]Mathematical Sciences, Chalmers, Göteborg, Sweden.

Many other people have contributed to our understanding of the problems dealt with in this text, first of all Professor Ross Leadbetter at the University of North Carolina at Chapel Hill and Professor Krzysztof Podgórski, Mathematical Statistics, Lund University. 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 who have put a great deal of effort into WAFO and its predecessors FAT and WAT are Mats Frendahl, Sylvie van Iseghem, Finn Lindgren, Ulla Machado, Jesper Ryén, Eva Sjö, Martin Sköld, Sofia Åberg.

This tutorial was first made available for the beta version of WAFO Version 2.5 in November 2009. In the present version some misprints have been corrected and some more examples added. All examples in the tutorial have been run with success on MATLAB up to 2010b.

Lund and Horten March 28, 2011

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:/wafo25`

- The MATLAB code used for the examples in this tutorial can be found in the WAFO catalogue

  `WAFO/papers/tutorcom/`

  The total time to run the examples is about one hour on a 64 bit, 2.93 GHz PC running Windows 7 and more than three hours on a 32 bit 2.2 GHz PC with Windows XP.

- WAFO is built of modules of platform independent MATLAB m-files and a set of executable files from `C++` and `Fortran` source files. These executables are platform and MATLAB-version dependent, and they have been tested with recent MATLAB and WINDOWS installations.

- If you have many MATLAB-toolboxes installed, name-conflicts may occur. Solution: arrange the MATLAB-path with `WAFO` first.

- WAFO Version 2.5, was released in beta version in January 2009 and in stable version in February 2011, and it can be downloaded from

  `http://code.google.com/p/wafo/`

  Older versions of the toolbox can be downloaded from the WAFO homepage

  `http://www.maths.lth.se/matstat/wafo/`

  There you can also find links to exercises and articles using WAFO, and notes about its history.

- For help on the toolbox, write `help wafo25`. Note, that in Windows, some of the routines in Chapter 4 do not work with MATLAB 2006 or earlier.

- Comments and suggestions are solicited — send to

  `wafo@maths.lth.se`

Contents

List of Figures

List of Tables

Nomenclature

Roman letters

A_c, A_t	Zero-crossing wave crest height and trough excursion.
a_i	Lower integration limit.
b_i	Upper integration limit.
c_0	Truncation parameter of truncated Weibull distribution.
$\mathsf{C}[X, Y]$	Covariance between random variables X and Y.
$D(\omega, \vartheta), D(\vartheta)$	Directional spreading function.
d	Water depth.
$dd_{crit}, d_{crit}, z_{crit}$	Critical distances used for removing outliers and spurious points.
$\mathsf{E}[X]$	Expectation of random variable X.
$E(\omega_i, \omega_j)$	Quadratic transfer function.
f	Wave frequency [Hz].
f_p	Spectral peak frequency.
$F_X(\cdot), f_X(\cdot)$	Cumulative distribution function and probability density function of variable X.
$G(\cdot),\ g(\cdot)$	The transformation and its inverse.
g	Acceleration of gravity.
$H,\ h$	Dimensional and dimensionless wave height.
$H_{m0},\ H_s$	Significant wave height, $4\sqrt{m_0}$.
H_c	Critical wave height.
$H_d,\ H_u$	Zero-downcrossing and -upcrossing wave height.
$h_{\max}$	Maximum interval width for simpson method.
H_{rms}	Root mean square value for wave height defined as $H_{m0}/\sqrt{2}$.
$K_d(\cdot)$	Kernel function.
k	Wave number or index.
L_p	Average wave length.
$L_{\max}$	Maximum lag beyond which the autocovariance is set to zero.
$M,\ M_k$	Local maximum.
M_k^{tc}	Crest maximum for wave no. k.

m, m_k	Local minimum.
m_k^{RFC}	Rainflow minimum no. k.
m_k^{tc}	Trough minimum for wave no. k.
m_n	n'th spectral moment, $\int_0^\infty \omega^n S_{\eta\eta}^+(\omega) d\omega$.
N	Number of variables or waves.
N_{c1c2}	Number of times to apply regression equation.
`NIT`	Order in the integration of wave characteristic distributions.
n_i, n	Sample size.
$O(\cdot)$	Order of magnitude.
Q_p	Peakedness factor.
$R_\eta(\tau)$	Auto covariance function of $\eta(t)$.
S_p	Average wave steepness.
S_s	Signuificant wave steepness.
$S_{\eta\eta}^+(f)$, $S_{\eta\eta}^+(\omega)$	One sided spectral density of the surface elevation η.
$S(\omega, \vartheta)$	Directional wave spectrum.
s	Normalized crest front steepness.
s_c	Critical crest front steepness.
s_{cf}	Crest front steepness.
s_N	Return level for return period N.
s_{rms}	Root mean square value for crest front steepness, i.e., $5/4\, H_{m0}/T_{m02}^2$.
T_c, T_{cf}, T_{cr}	Crest, crest front and crest rear period.
$T_{m(-1)0}$	Energy period
T_{m01}	Mean wave period.
T_{m02}	Mean zero-crossing wave period calculated as $2\pi\sqrt{m_0/m_2}$.
T_{m24}	Mean wave period between maxima calculated as $2\pi\sqrt{m_2/m_4}$.
T_{Md}	Wave period between maximum and downcrossing.
T_{Mm}	Wave period between maximum and minimum.
T_p	Spectral peak period.
T_z	Mean zero-crossing wave period estimated directly from time series.
T	Wave period.
U_{10}	10 min average of windspeed $10m$ above the watersurface.
U_i	Uniformly distributed number between zero and one.
V, v	Dimensional and dimensionless velocity.
$\mathsf{V}[X]$	Variance of random variable X.
V_{cf}, V_{cr}	Crest front and crest rear velocity.
V_{rms}	Root mean square value for velocity defined as $2H_{m0}/T_{m02}$.
W_{age}	Wave age.
$W(x, t)$	Random Gassian field.
$X(t)$	Time series.
X_i, Y_i, Z_i	Random variables.
x_c, y_c, z_c	Truncation parameters.

Greek letters

α	Rayleigh scale parameter or JONSWAP normalization constant.
α	Irregularity factor; spectral width measure.
$\alpha(h), \beta(h)$	Weibull or Gamma parameters for scale and shape.
α_i	Product correlation coefficient.
Δ	Forward difference operator.
$\delta_{i\|1}$	Residual process.
ε_2	Narrowness parameter defined as $\sqrt{m_0 m_2/m_1^2 - 1}$.
ε_4	Broadness factor defined as $\sqrt{1 - m_2^2/(m_0 m_4)}$.
ε	Requested error tolerance for integration.
ε_c	Requested error tolerance for cholesky factorization.
$\eta(\cdot)$	Surface elevation.
Γ	Gamma function.
γ	JONSWAP peakedness factor or Weibull location parameter.
λ_i	Eigenvalues or shape parameter of Ochi-Hubble spectrum.
$\mu_X(v)$	Crossing intensity of level v for time series $X(t)$.
$\mu_X^+(v)$	Upcrossing intensity of level v for time series $X(t)$.
$\Phi(\cdot)$, $\varphi(\cdot)$	CDF and PDF of a standard normal variable.
ψ_n	Phase function.
ρ_3, ρ_4	Normalized cumulants, i.e., skewness and excess, respectively.
ρ_{ij}	Correlation between random variables X_i and X_j.
Σ	Covariance matrix.
σ_X^2	Variance of random variable X.
τ	Shift variable of time.
τ_i	Parameters defining the eigenvalues of $\mathbf{\Sigma}$.
ω	Wave angular frequency $[rad/s]$.
ω_p	Wave angular peak frequency $[rad/s]$.

Abbreviations

AMISE	Asymptotic mean integrated square error.
CDF	Cumulative distribution function.
FFT	Fast Fourier Transform.
GEV	Generalized extreme value.
GPD	Generalized Pareto distribution.
HF	High frequency.
ISSC	International ship structures congress.
ITTC	International towing tank conference.
IQR	Interquartile range.
KDE	Kernel density estimate.
LS	Linear simulation.
MC	Markov chain.
MCTP	Markov chain of turning points.
ML	Maximum likelihood.
NLS	Non-linear simulation.
MISE	Mean integrated square error.
MWL	Mean water line.
PDF	Probability density function.
PSD	Power spectral density.
QTF	Quadratic transfer function.
SCIS	Sequential conditioned importance sampling.
TLP	Tension-leg platform.
TP	Turning points.
WAFO	Wave analysis for fatigue and oceanography.

Part II

WAFO tutorial

CHAPTER 1

Introduction to WAFO

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 the variety of problems you can analyze using this toolbox.

There are three major audiences to which this toolbox can have a great deal of appeal. First, *ocean engineers* will find a 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 prime importance for *mechanical engineers* working on *random loads* or *damage and fatigue analysis*. Finally, any *researcher* who is interested in *statistical analysis of random processes* will find 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 with 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 procedure 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 [35] and computational aspects and algorithms in [59].

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 appeared 1993, called the Fatigue Analysis Toolbox (FAT), [16]. It was followed by the Wave Analysis Toolbox (WAT[1]) in 1995, written by Rychlik and Lindgren, [60], being extended with routines for probabilistic modelling problems in oceanography. In WAFO, many new numerical routines were introduced, and a considerable improvement in computational speed and accuracy was 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. [57], [49], [48], [26], and [9].

WAFO, version 2.5, which appeared in beta-version January 2009, and in stable version February 2011, contains a great number of general statistical routines, making the toolbox useful also for statistical analysis and computation in many other areas than marine and mechanical engineering; see `help statistics`.

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 and calculation of 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,
- general statistical analysis and computation.

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

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

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 the MATLAB code that 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/papers` 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/papers` directory the `WAFO/wdemos` directory also contains different subdirectories with scripts producing figures. The only difference is that these do 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 `papers` and `wdemos` 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 toolbox is the use of *structure arrays*, introduced in MATLAB, Version 5, 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 spec-

[2] `http://code.google.com/p/wafo/`

trum structure, covariance structure and probability density function (hereafter denoted pdf) structure. The toolbox is portable to any computational environment that supports MATLAB, such as Linux, Unix or PC with MS Windows. See Section 1.5 for a description of the datastructures in WAFO. Note that this tutorial uses the command naming convention introduced in WAFO, Version 2.5.

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 in ascii and postscript format.

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

`WAFO/wdemos` contains different demonstrations that illustrate and highlight certain aspects of 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 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 the 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 and Torsethaugen. The spectra can be expressed in frequency as well as wave number. Libraries of spreading functions $D(\vartheta)$, in some cases allowed to be also frequency dependent, cf. [27], are included.

Transformed Gaussian processes

WAFO is mainly intended to model linear, Gaussian waves. For this category 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. [57], are included. Contrary to what is often stated in the technical literature, these routines are very efficient and accurate and they can be used for engineering purposes; cf. [41, Sec. 4.4.1].

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 of crest period and amplitude, (T_c, A_c), in a stationary Gaussian transformed process proposed in [13], and [38],
- a model for the cdf/pdf of breaking limited wave heights proposed in [65],
- a model for the cdf/pdf of large wave heights in [66].

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 and fields

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, [15]. More traditional spectral simulation methods (FFT) are also used. Simulation of discrete Markov chains, Markov chains of turning points, switching Markov chains and Hidden Markov Models, etc, is possible. Other routines generate time-varying random (Gaussian or transformed Gaussian) wave fields with directional spectrum.

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 descriptive 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). The Markov matrix is the expected histogram matrix of min-to-max and max-to-min cycles. Given a rainflow matrix of a MCTP, one can find its Markov matrix, and vice versa. In WAFO, algorithms are implemented to calculate the rainflow matrix for a MC and a MCTP; cf. [17].

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. If the regime process is modelled by a Markov chain we have a Hidden Markov Model (HMM), and this is the fundamental basis for the set of routines presented. For an application with such switching Markov models for fatigue problems, see [25, 26].

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.

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 these and other popular distributions, used in reliability and life-span models, it is possible to estimate parameters, generate random variables, evaluate pdf and cumulative distribution function, and plot in various probability papers.

Kernel-density-estimation tools

The routines in this category 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. [63] and [72].

WAFO as a statistics toolbox

Besides the special statistical routines for extreme value analysis and kernel smoothing, WAFO contains statistical routines for handling univariate and multi-variate distribution functions, simulation, moments, likelihood estimation, regression and factor analysis, hypothesis testing and confidence intervals, bootstrap and jacknife estimation, and design of experiment.

Miscellaneous routines

We find here various plot routines, algorithms for numerical integration, and functions for documentation of WAFO with modules. Note, that the figures in this tutorial have beed edited with respect to font size, and some other properties.

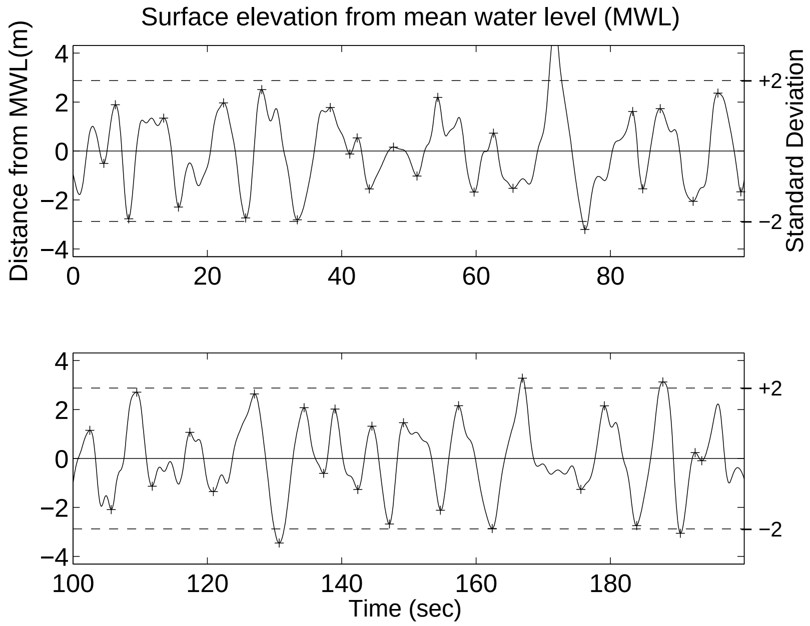


Figure 1.1: *A simulation from $S(\omega)$, a Torsethaugen spectrum with $H_{m_0} = 6$ [m], $T_p = 8$ [s]. Total number of points $= 2000$, $\Delta t = 0.1$ [s].*

1.4 Some applications of WAFO

In this section we demonstrate 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. The necessary MATLAB code for generation of the figures in this tutoriol is found in the directory `WAFO/papers/tutorcom/`. The commands for this chapter are collected in `Chapter1.m` and run in 25 seconds on a 2.93 GHz 64 bit PC.

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; [69]. 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 200 seconds of data sampled with 10 Hz from the discussed spectrum. More on simulation can be found in Section 2.3.

```
Hm0 = 6; Tp = 8; plotflag = 1;
S1 = torsethaugen([],[Hm0 Tp],plotflag);
dt = 0.1; N = 2000;
xs = spec2sdat(S1,N,dt);
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 of the 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 was used to generate Figure 1.2, where the original and estimated 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.

```
plotflag = 1; Fs = 4;
dt = 1/Fs; N  = fix(20*60*Fs);
xs = spec2sdat(S1,N,dt);
Sest = dat2spec(xs,400)
plotspec(S1,plotflag), hold on
plotspec(Sest,plotflag,'--')
axis([0 3 0 5]), hold off
```

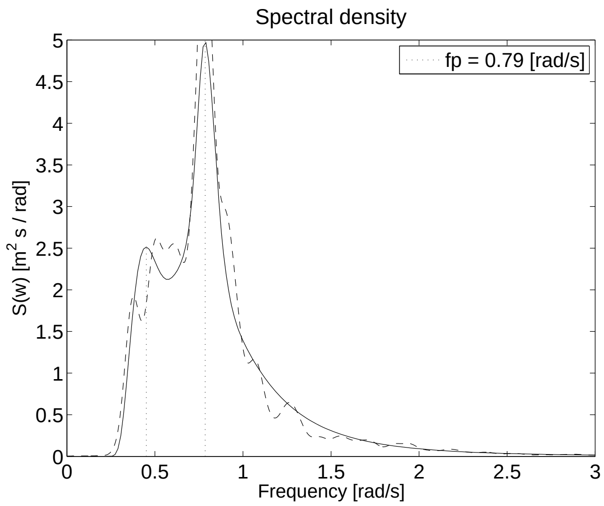


Figure 1.2: *Solid: original Thorsethaugen spectrum. Dashed: spectrum estimated from data (20 minutes of observations). Maximum lag size of the Parzen window* $= 400$.

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, page 42, 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. In Chapter 3 we analyse wave characteristics from observed date, and present some commonly used apprimative distributions. Chapter 4 describes how to use WAFO to compute the exact theoretical distributions for all these wave characteristics in a Gaussian or transformed Gaussian model.

In the numerical example, we consider the trough period, i.e. the down-to-up crossing definition. 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 the original spectrum $S(\omega)$, and from the estimated spectrum $S_{\text{est}}(\omega)$; see Figure 1.3. Recall that, for this spectrum, $T_p = 8$ s. The figure shows the density for the half period; the results are in good agreement with that from the original spectrum. The following code lines to produced 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. The parameter `NIT` determines the accuracy of the calculation.

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

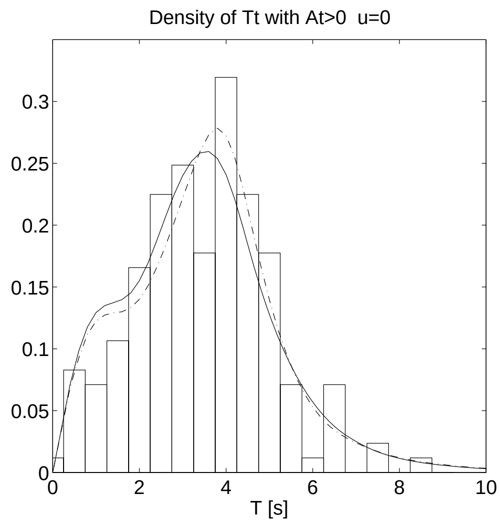


Figure 1.3: *Pdf for wave trough period given $S(\omega)$ (solid line) and $S_{est}(\omega)$ (dash-dotted line). The histogram shows the wave periods extracted from the simulated data in Figure 1.1.*

1.4.3 Directional spectra

In WAFO one finds means for evaluation and visualization of directional spectra to model sea states with waves coming from many different directions, that is

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

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

One way of visualizing $S(\omega, \vartheta)$ 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 independent case),

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

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. [27].

Here are a few lines of code, which produce the graph of these directional spectra with frequency independent and frequency dependent spreading. The main directions are 90^o and 0^o, respectively.

```
plotflag = 1;
Nt = 101;   % number of angles
th0 = pi/2; % primary direction of waves
Sp = 15;    % spreading parameter
D1 = spreading(Nt,'cos',th0,Sp,[],0); %frequency independent
D12 = spreading(Nt,'cos',0,Sp,S1.w,1); %frequency dependent
SD1 = mkdspec(S1,D1);  SD12 = mkdspec(S1,D12);
plotspec(SD1,plotflag), hold on
plotspec(SD12,plotflag,'-.'), hold off
```

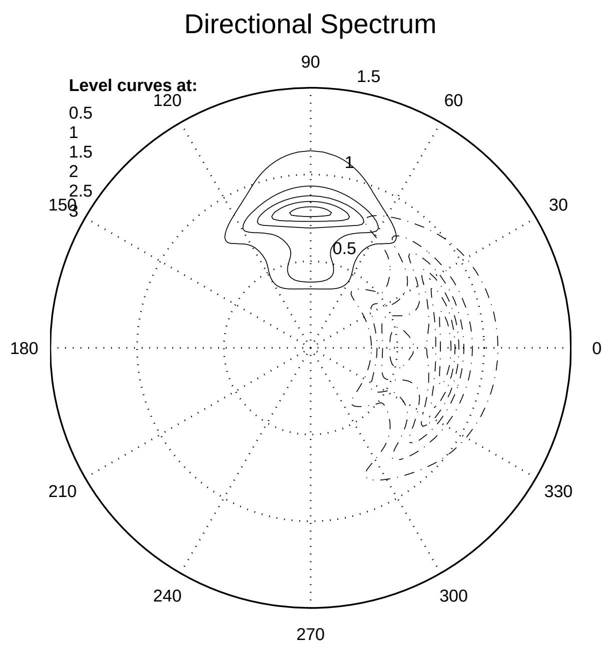


Figure 1.4: *Directional spectrum. The frequency spectrum is a Torsethaugen spectrum and the spreading function is of* cos-2s *type with* $s = 15$. *Solid line: directional spectrum with frequency independent spreading. Dash dotted line: directional spectrum, using frequency dependent spreading function.*

We finish the section with simulated sea surfaces on 128[m] by 128[m] for a sea with directional spectra `SD1` and `SD12`. The routine `seasim` is used for simulation.

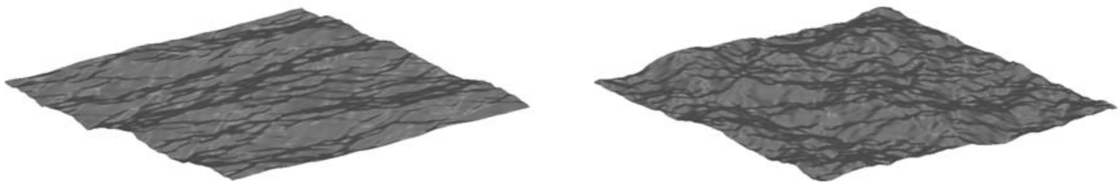


Figure 1.5: *Simulated sea surfaces on a rectangle of 128 [m] by 128 [m]. Left: with directional spectrum* SD1*, spreading independent of frequency; Right: with directional spectrum* SD12*, frequency dependent spreading.*

```
plotflag = 1; iseed = 1; Nx = 2^8; Ny = Nx; Nt = 1;
dx = 0.5; dy = dx; dt = 0.25; fftdim = 2;
randn('state',iseed)
Y1 = seasim(SD1,Nx,Ny,Nt,dx,dy,dt,fftdim,plotflag);
randn('state',iseed)
Y12 = seasim(SD12,Nx,Ny,Nt,dx,dy,dt,fftdim,plotflag);
```

The results are shown in Figure 1.5 and one can see that waves are coming from different directions. However, frequency dependent spreading leads to a 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 peaks and troughs of the load, called the turning points (TP). From these 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 with spectrum from Figure 1.2.

For fatigue analysis the rainflow cycle, defined in Figure 5.1 in Chapter 5, 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. [35] and references therein. Chapter 4 describes how WAFO routines can be used to find the min-to-max distribution for Gaussian loads. For the Markov model, there is an explicit solution for the intensity of rainflow cycles, see [17]. 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 commands:

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

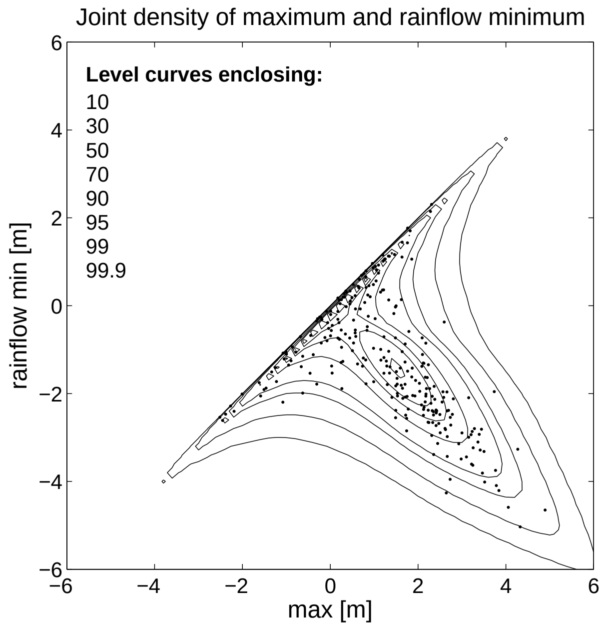


Figure 1.6: *Intensity of rainflow cycles computed from the power spectral density through Markov approximation, compared with the cycles found in the simulation.*

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 5.

1.4.5 Statistical extreme value analysis

The WAFO-toolbox contains almost 600 routines for general statistical analysis, description, plotting, and simulation. In Chapter 6 we describe some routines which are particularly important for wave and fatigue analysis, related to statistics of extremes. These are based on the generalized extreme value (GEV) and generalized Pareto distribution (GPD), combined with the peaks over threshold (POT) method. As an example we show an analysis of wave elevation data from the Poseidon platform in the Japan Sea. Data from about 23 hours of registration are stored in the data set yura87, taken with a 1 Hz sampling rate. We first load and plot, in Figure 1.7, part of the data and calculate the maximum over 5 minute periods.

```
xn = load('yura87.dat'); subplot(211);
plot(xn(1:30:end,1)/3600,xn(1:30:end,2),'.')
title('Water level'), ylabel('m')
yura = xn(1:85500,2);
yura = reshape(yura,300,285);
```

```
maxyura = max(yura); subplot(212)
plot(xn(300:300:85500,1)/3600,maxyura,'.')
xlabel('Time (h)'), ylabel('m')
title('Maximum 5 min water level')
```

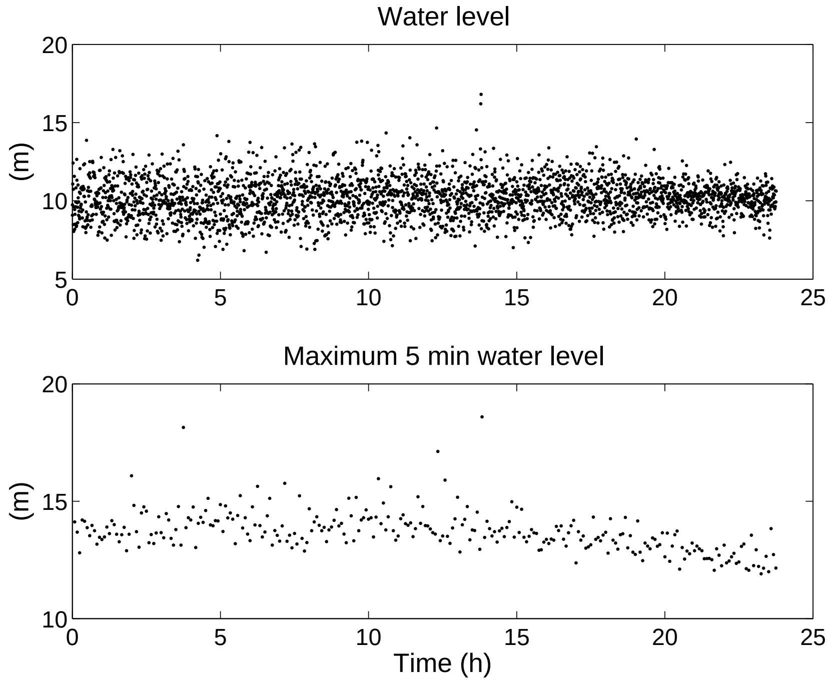


Figure 1.7: *Water level variation in the Japan Sea from the data set* yura87 *and maxima over 5 minute periods.*

It is clear from the figures that there is a trend in the data, with decreasing spreading with time. In Chapter 5 we will deal with that problem; here we make a crude extreme value analysis, by fitting a GEV distribution to the sequence of 5 minute maxima, simply by issuing the commands

```
phat = fitgev(maxyura,'plotflag',1);
```

This results in Figure 1.8, which shows cumulative distribution and density of the fitted GEV distribution together with diagnostic plots of empirical and model quantiles. We see that the non-stationarity gives a very bad fit in the upper tail of the distribution. The fitted GEV has shape parameter 0.1, with a 95% confidence interval $(0.01, 0.18)$.

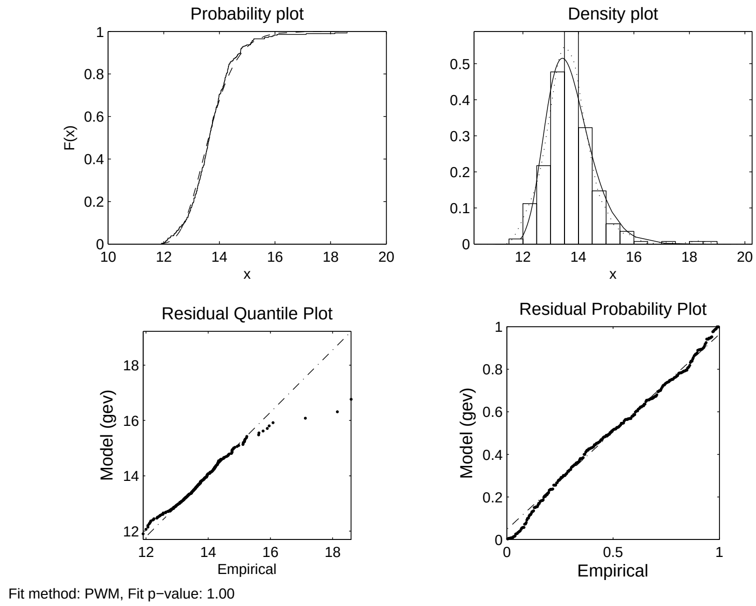


Figure 1.8: *Diagnostic plots of extreme value analysis of* `yura87` *with GEV distribution.*

1.5 Datastructures

```
help datastructures

 DATASTRUCTURES of spectrum (S), covariance function (cvf)
 and probability 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'.
    .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', 'rotk2d', length np.
    .theta Angular lags, if .type='dir', 'rotdir' or 'encdir',
           length np.
    .v     Ship speed, if .type = 'enc' or 'encdir'.
    .phi   angle of rotation of the coordinate system
           (counter-clockwise) e.g. azimuth of a ship.

  See also  createspec, plotspec

  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   Rotation of coordinate system, e.g. direction of ship.
```

```
  .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

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. The commands, collected in `Chapter2.m`, run in less than 10 seconds on a 2.93 GHz 64 bit PC.

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.

   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 measure the variability around the mean in linear and quadratic scale. These quantities are estimated by

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

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

for a continuous recording or by corresponding sums for a sampled series.

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,

$$x(t) \approx m + \sum_{i=1}^{N} a_i \cos(\omega_i\, t) + b_i \sin(\omega_i\, t), \tag{2.3}$$

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. Another important way to represent a load sequence is by means of the *crossing spectrum* or *crossing intensity*, $\mu(u)$ = the intensity of upcrossings = average number of upcrossings per time unit, of a level u by $x(t)$ as a function of u, see further in Section 2.2.3. 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", i.e. the average number of local maxima per time unit) 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);
plotflag = 2;
lcplot(lc,plotflag,0,sa)
```

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.

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.

[1] Still another definition, to be used in Chapter 5, is that f_0 is the average number of completed load cycles per time unit.

```
T = max(xx(:,1))-min(xx(:,1))
f0 = interp1(lc(:,1),lc(:,2),0)/T
   % zero up-crossing frequency
```

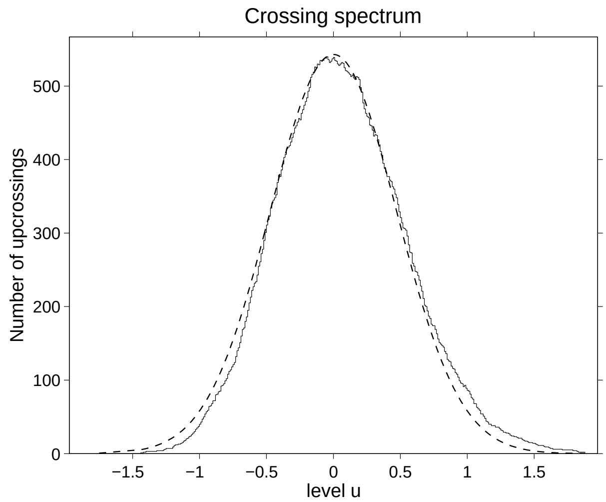


Figure 2.1: *The observed crossings intnsity compared with the theoretically expected for Gaussian signals, see (2.5).*

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);
fm = length(tp)/(2*T)            % frequency of maxima
alfa = f0/fm
```

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 `fm`. □

We finish this section with some remarks about the quality of the measured data. Especially sea surface measurements can be of poor quality. 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 xx and zoom in on a specific region. A part of the sea data is presented in Figure 2.2 obtained by the following commands.

```
waveplot(xx,tp,'k-','*',1,1)
axis([0 2 -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 criteria:

- $x'(t) < 5$ [m/s], since the raising speed of Gaussian waves rarely exceeds 5 [m/s],

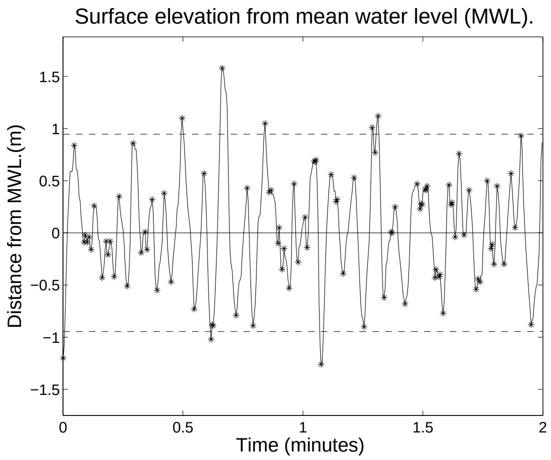


Figure 2.2: *A part of the sea data with turning points marked as stars.*

- $x''(t) < 9.81/2$, $[m/s^2]$ 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);
```

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. One must be careful using the criteria for extreme value analysis, because it might remove extreme waves that belong to the data and are not spurious. 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. □

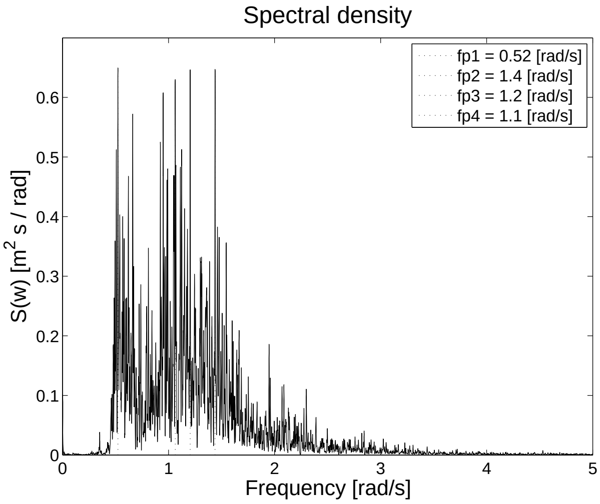


Figure 2.3: *The observed, unsmoothed, spectrum in the data set* `sea.dat`.

2.2 Frequency modeling of load histories

2.2.1 Power spectrum, periodogram

The most important characteristic of signals of the form (2.3) 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 alternative term *periodogram* was introduced as early as 1898 by A. Schuster, [62].

The sequence ϑ_i such that $\cos\vartheta_i = a_i/\sqrt{2\,\hat{s}_i\,\Delta\omega}$ and $\sin\vartheta_i = -b_i/\sqrt{2\,\hat{s}_i\,\Delta\omega}$, is called a sequence of phases and the Fourier series can be written as follows:

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

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 to compute the Fourier coefficients (and the spectrum) from the measured signal and in reverse the signal from the Fourier coefficients.

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

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

Example 1. (contd.) We now calculate the spectrum $\widehat{s}(\omega)$ for the sea data signal `xx`.

```
Lmax = 9500;
S = dat2spec(xx,Lmax);
plotspec(S)
```

In Figure 2.3 we can see that the spectrum is extremely irregular with sharp peaks at many distinct frequencies. In fact, if we had analysed another section of the sea data we had found a similar general pattern, but the sharp peaks had been found at some other frequencies. It must be understood, that the observed irregularities are random and vary between measurements of the sea even under almost identical conditions. This will be further discussed in the following section, where we introduce smoothing techniques to get a stable spectrum that represents the "average randomness" of the sea state.

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 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. □

2.2.2 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 to the first one. However, the two series are seldom identical and differ in some respect that 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 randomly chosen function. The simplest model for a stationary signal with a fixed spectrum $\hat{s}(\omega)$ is

$$X(t) = m + \sum_{i=1}^{N} \sqrt{\hat{s}_i\,\Delta\omega}\,\sqrt{2}\cos(\omega_i\,t + \Theta_i), \tag{2.4}$$

where the phases Θ_i are random variables, independent and uniformly distributed between 0 and 2π. 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. Hence, $\hat{s}_i$ 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 $\widehat{s}(\omega_i)\Delta\omega$ over many 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, with probability density function $f_R(r) = r\exp(-r^2/2), r > 0$. (Observe that the average value of R_i^2 is 2.) This gives the following random function as a 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[2] 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

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

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,

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

Since $\mathsf{V}[X(t)] = r_X(0) = \int_0^\infty S(\omega)\, d\omega$, the spectral density represents a continuous distribution of the wave energy over a continuum of frequencies.

The Gaussian process model is particularly useful in connection with linear filters. If $Y(t)$ is the output of a linear filter with the Gaussian process $X(t)$ as input, then $Y(t)$ is also normally distributed. Further, the spectrum of $Y(t)$ is related to that of $X(t)$ in a simple way. If the filter has *transfer function* $H(\omega)$, also called *frequency function*, then the spectrum of $Y(t)$, denoted by S_Y, is given by

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

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

$$\sigma_{X'}^2 = \int S_{X'}(\omega)\, d\omega = \int \omega^2 S_X(\omega)\, d\omega = m_2.$$

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. In practice, it means that one has to use some smoothing techniques. For the `sea.dat` we shall estimate the spectrum by means of the WAFO function `dat2spec` with a second parameter defining the degree of smoothing.

[2] A *Gaussian* stochastic process $X(t)$ is any process such that all linear combinations $\sum a_k X(t_k)$ have a Gaussian distribution; also derivatives $X'(t)$ and integrals $\int_a^b X(t)\, dt$ are Gaussian.

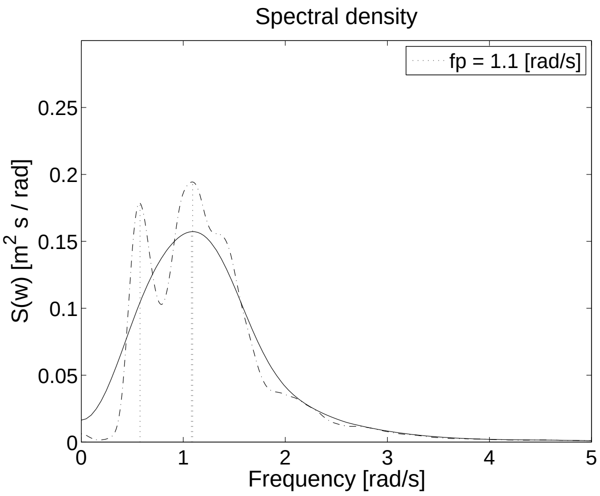


Figure 2.4: *Estimated spectra in the data set* `sea.dat` *with varying degree of smoothing.*

```
Lmax0 = 200; Lmax1 = 50;
S1 = dat2spec(xx,Lmax0);
S2 = dat2spec(xx,Lmax1);
plotspec(S1,[],'-.'), hold on
plotspec(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.

Knowing the spectrum one can compute the covariance function by means of the *Fourier inversion* formula, which for a time-continuous signal reads,

$$S(\omega) = \frac{2}{\pi}\int_0^\infty \cos(\omega\tau) r(\tau)\, d\tau.$$

The following code in WAFO will compute the covariance for the unimodal spectral density `S1` and compare it with estimated covariance in the signal `xx`.

```
Lmax = 80;
R1 = spec2cov(S1,1);
Rest = dat2cov(xx,Lmax);
covplot(R1,Lmax,[],'.'), hold on
covplot(Rest), hold off
```

We can see in Figure 2.5(a) that the covariance function corresponding to the spectral density `S2` differs significantly 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 `S1` in `spec2cov` replaced by `S2`. □

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 detailed information.

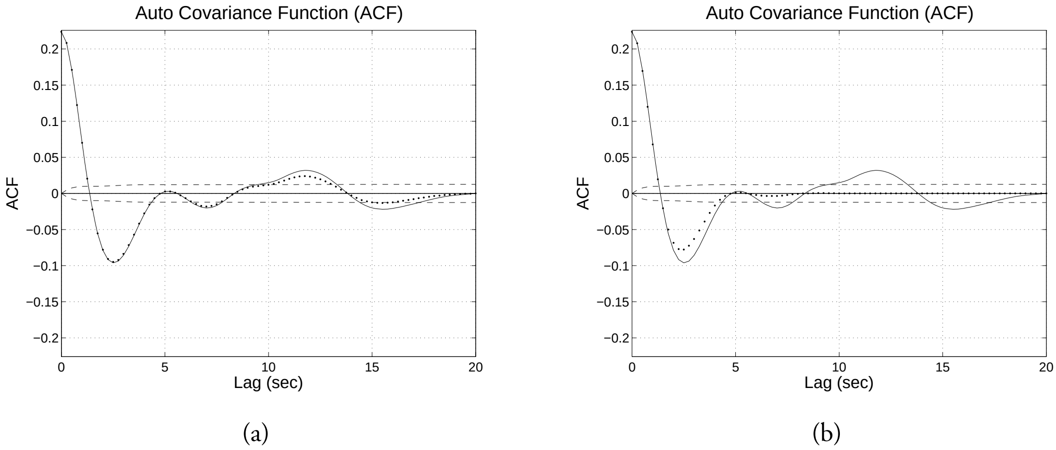


Figure 2.5: *The covariance function estimated in the data set* `sea.dat`, *solid line, compared to the theoretically computed covariance functions for the spectral densities* `S2` *in (a) and* `S1` *in (b).*

2.2.3 Crossing intensity – Rice's formula

The Gaussian process is a sum of cosine terms with amplitudes defined by the spectrum, and the instantaneous value $X(t)$ has a normal distribution with mean 0 and variance $\sigma^2 = \int S(\omega)\,d\omega$. In wave analysis and fatigue applications there is another quantity that plays a central role, namely the *upcrossing intensity* $\mu(u)$, which yields the average number, per time or space unit, of upcrossings of the level u. It contains important information on the fatigue properties of a load signal and also of the wave character of a random wave.[3]

For a Gaussian process the crossing intensity is given by the celebrated *Rice's formula*,

$$\mu(u) = f_0 \exp(-(u-m)^2/2\sigma^2). \tag{2.5}$$

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

2.2.4 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 with a spectral density $S(\omega)$. The knowledge of which kind of spectral densities $S(\omega)$ are suitable to describe different 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, [57], we have found that for many sea wave data, even such that are clearly non-Gaussian, the

[3]The general expression for the upcrossing intensity for a stationary process $X(t)$ with derivative $X'(t)$, is $\mu(u) = \int_{z=0}^{\infty} z f_{X(0),X'(0)}(u,z)\,dz$, where $f_{X(0),X'(0)}(u,z)$ is a joint probability density function.

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 [57] 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.

$$X(t) = G(\widetilde{X}(t)), \tag{2.6}$$

where $G(\cdot)$ is a continuously differentiable function with positive derivative. We shall denote the spectrum of X by S, and the spectrum of $\widetilde{X}(t)$ by $\widetilde{S}$. The transformation G performs the appropriate non-linear 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 a simple variable transformation 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 in Eq. (2.6) is correct, then $\widetilde{x}(t)$ should be a sample function of a process with Gaussian marginal distributions.

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 parametric or non-parametric techniques. 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,
ochitr    - transformation g proposed by Ochi et al.
```

The transformation proposed by by Ochi et al., [44], is a monotonic exponential function, while Winterstein's model, [73], 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 from physical considerations. Rychlik's non-parametric method is based on the crossing intensity $\mu(u)$; see [57]. Martinsen and Winterstein, [40], 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 `skew` and `kurt`.

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

```
rho3 = skew(xx(:,2))
rho4 = kurt(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 for 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 Winterstein's Hermite transformation and denote it by `gh`, and compare it with the linear transformation, denoted by `g`, that only has the effect to standardize the signal, assuming it is already Gaussian,

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

These commands will result in two 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 the method proposed by Rychlik et al., in [57]:

```
[glc test0 cmax irr gemp] = dat2tr(xx,[],'plotflag',1);
hold on
plot(glc(:,1),glc(:,2),'b-')
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`.

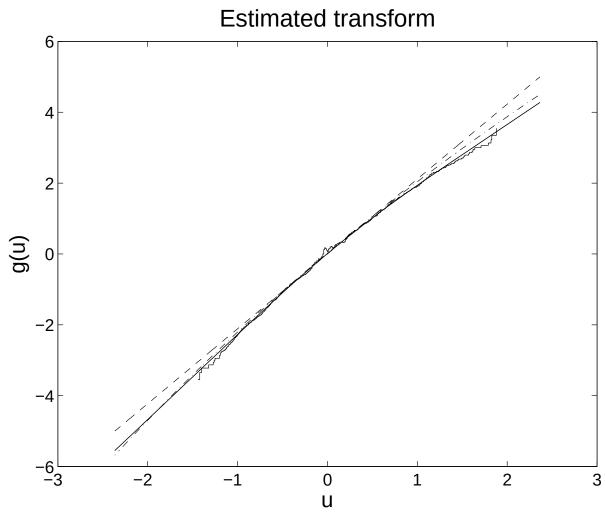


Figure 2.6: *Comparisons of the three transformations g, straight line is the Gaussian model, dash dotted line the Hermite transformation* `gh` *and solid line the Rychlik method* `glc`*.*

In Figure 2.6 we compare the three transformations, the straight line is the Gaussian linear model, the dash 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

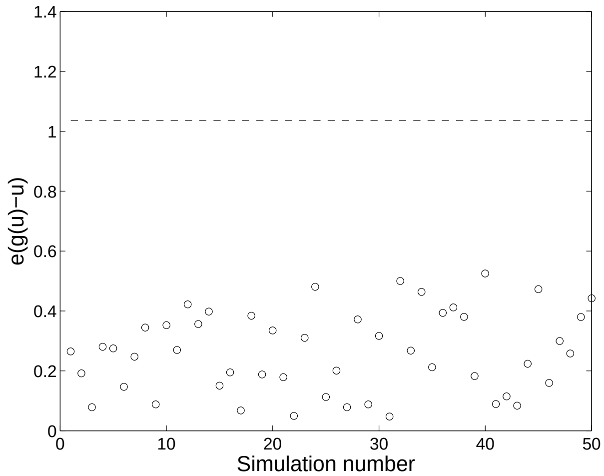


Figure 2.7: *The simulated 50 values of the* `test` *variable for the Gaussian process with spectrum* `S1` *compared with the observed value (dashed line).*

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` 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 `testgaussian`. The output from the program is a plot of the ratio `test1` between the simulated (Gaussian) `test0`-values and the sample `test0`:

```
N = length(xx);
test1 = testgaussian(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 [57] 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 observations of identically distributed Gaussian variables form 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 a straight line as N tends to infinity.

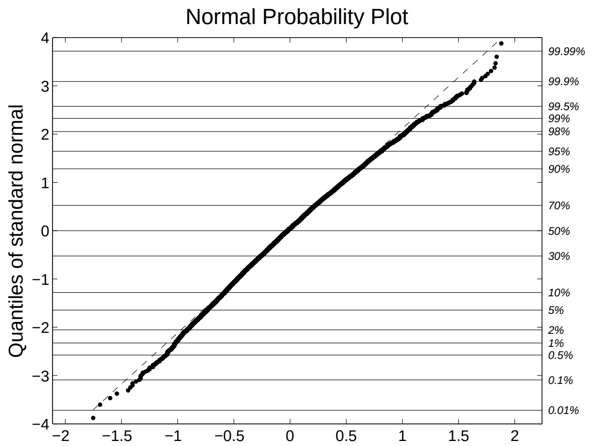


Figure 2.8: *The data* `sea.dat` *on normal probability plot.*

The command

```
plotnorm(xx(:,2))
```

produces Figure 2.8. As we can see the normal probability plot is slightly curved indicating that the underlying distribution has a heavy upper tail and a light lower tail. □

2.2.5 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 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
ochihubble    - Bimodal Ochi-Hubble spectral density
torsethaugen  - Bimodal (swell + wind) spectral density
bretschneider - Bretschneider (Pierson-Moskowitz)
                spectral density
mccormick     - McCormick spectral density
tmaspec       - JONSWAP spectral density
                for finite water depth
```

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

The spectrum of the sea can be given in many different formats, that are interconnected by the dispersion relation[4]. The spectrum can be given using frequencies, angular frequencies or wave numbers, and it can also be directional.

[4]The dispersion relation between frequency ω and wave number $\varkappa$ on finite water depth h, reads $\omega^2 = g\varkappa \tanh h\varkappa$, where g is the acceleration of gravity.

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, which is a quantity in space domain, from an input that is the directional frequency spectrum, which is related to the time domain.)

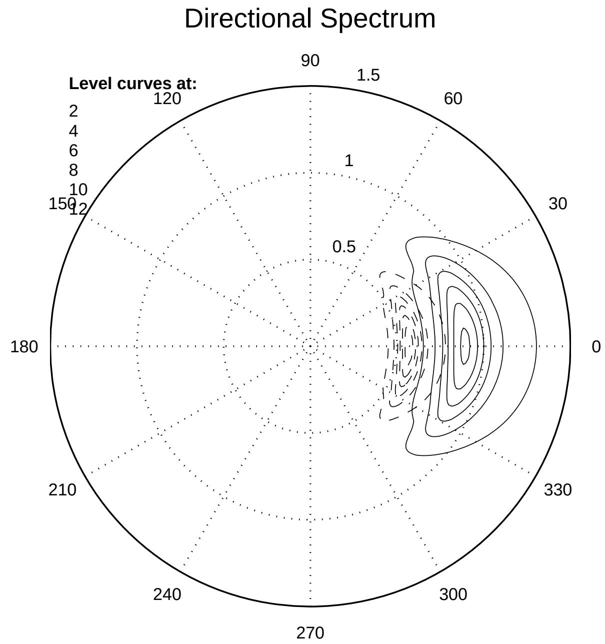


Figure 2.9: *Directional spectrum of* JONSWAP *sea (dashed line) compared with the encountered directional spectrum for heading sea, speed 10 [m/s] (solid line).*

Example 2. *(Different forms 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 level 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 consider 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);
plotspec(Se), hold on
plotspec(Sd,1,'--'), hold off
```

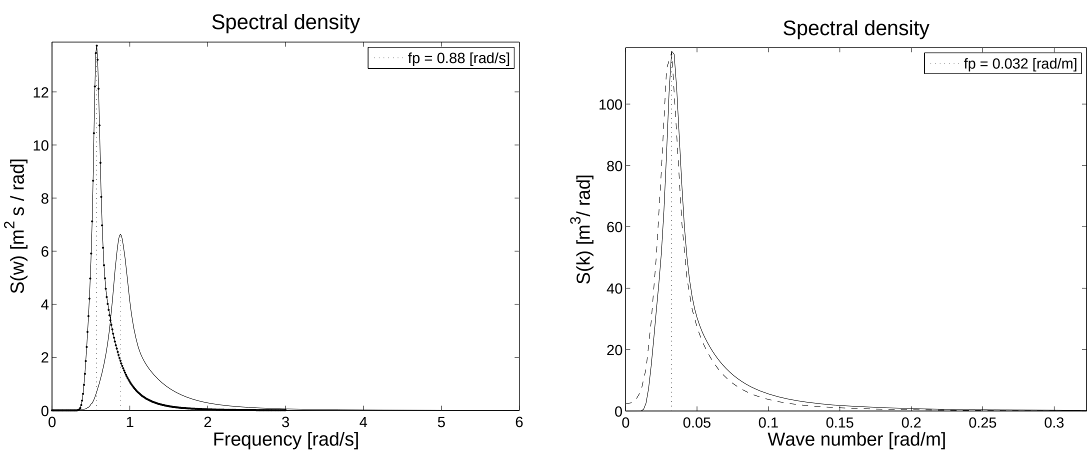
Obviously, the periods of waves in the directional sea are defined by the JONSWAP spectrum (spreading is not affecting the sea level at a fixed point), but the encountered periods will be shorter with heading seas. This can be seen by comparing the JONSWAP spectrum `spec` with the following two spectra:

```
Sd1 = spec2spec(Sd,'freq');
Sd2 = spec2spec(Se,'enc');
plotspec(spec), hold on
plotspec(Sd1,1,'.'),
plotspec(Sd2), hold off
```

We can see in Figure 2.10(a) that the spectra `spec` and `Sd1` are identical (in numerical sense), while spectrum `Sd2` 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')
plotspec(Sk), hold on
plotspec(Skd,1,'--'), hold off
```



(a) (b)

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.*

Finally, we shall show how the JONSWAP spectrum can be corrected for a finite depth, see [12]. The WAFO `phi1` computes the spectrum for water of finite depth, here 20 [m].

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

The resulting spectra are shown in Figure 2.11. □

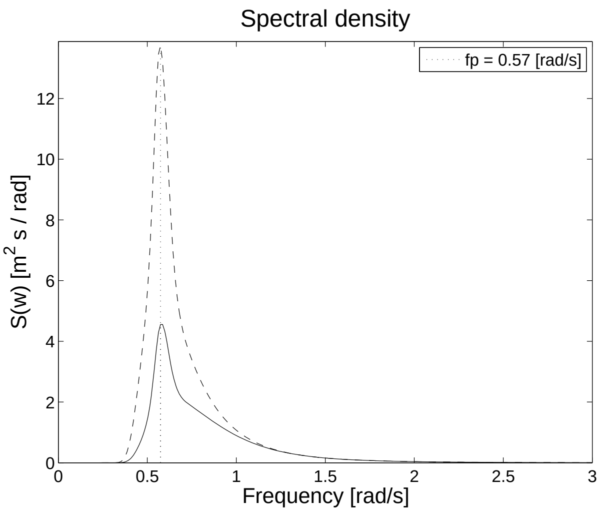


Figure 2.11: *Standard* JONSWAP *spectrum (dashed line) compared with the spectrum on finite depth of 20 [m] (solid line)*

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 $X(t) = G(\widetilde{X}(t))$.

The first important case is when we wish to reproduce random versions of the measured signal $x(t)$. Using `dat2tr` one 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 `dat2spec`. 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 simulation 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 possibly 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 modifying 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: When the simulation routine `spec2sdat` is called with a spectrum argument that contains a scale changing transformation `spectrum.tr`, then it is assumed that the input spectrum is standardized 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 that 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 on page 19 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 [8, 9] 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 from the simulated data. First transform data to obtain a sample $\tilde{x}(t)$:

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

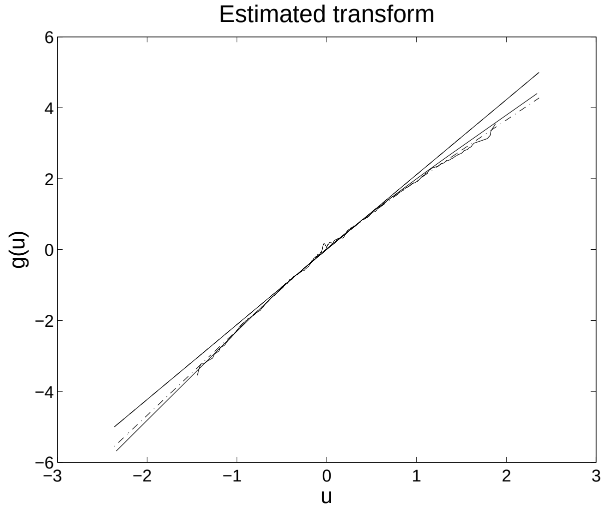


Figure 2.12: *The transformation computed from the reconstructed signal* `y` *(solid line) compared with the transformation computed from the original signal* `xx` *(dashed dotted line).*

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`, and `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 ($2 \times 60 \times 4$ points) of the signal, to obtain a synthetic wave equivalent to the reconstructed non-Gaussian sea data.

```
Ny = fix(2*60/dt)  % = 2 minutes
Sx.tr = grec;
ysim = spec2sdat(Sx,Ny);
waveplot(ysim,'-')
```

The result is shown in Figure 2.13. □

In the next example we consider a signal with a given 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 [57] 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 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.

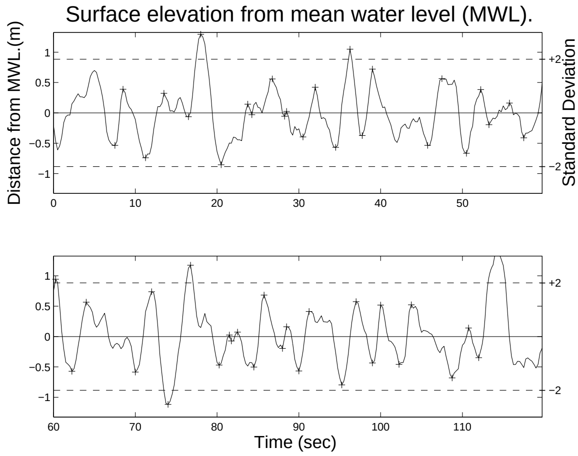


Figure 2.13: *Two minutes of simulated sea data, equivalent to the reconstructed data.*

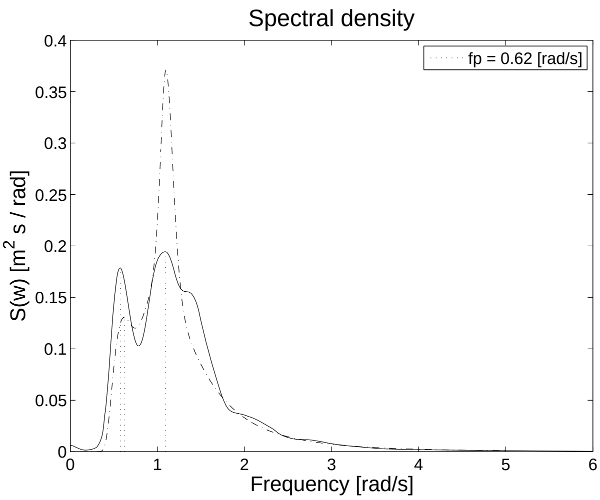


Figure 2.14: *Comparison between the estimated spectrum in the signal* `sea.dat` *(solid line) and the theoretical spectrum of the Torsethaugen type (dash-dotted line).*

Example 3. (contd.) Since the spectrum `S1` in Figure 2.4 is clearly two-peaked with peak frequency $T_p = 1.1$ [Hz] we choose to use the Torsethaugen spectrum. (This spectrum is derived for a specific location and we should not expect that it will work well 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]);
plotspec(S1), hold on
plotspec(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 the example.

We shall now create the spectrum $\widetilde{S}(\omega)$ (= `Snorm`), i.e. the spectrum for the standardized Gaussian 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] (= $\pi/5$), 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;
```

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, are quite rare and for moderate waves the difference is negligible.

In order to illustrate the difference in distribution for extreme waves we will simulate a sample of 4 minutes of $X(t)$ with sampling frequency 2 Hz. The result is put into ysim_t. In order to obtain the corresponding sample path of the process $\widetilde{X}$ we use the transformation gh, stored in Snorm.tr, 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 $\tilde{X}(t)$ always has variance one, in order to compare the Gaussian and non-Gaussian models we scale 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) have troughs deeper and crests 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 empirical transformation glc instead of the Hermite transformation gh would give errors of ca 11%, which for waves with higher significant wave height would give considerable underestimation of the crest height 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.

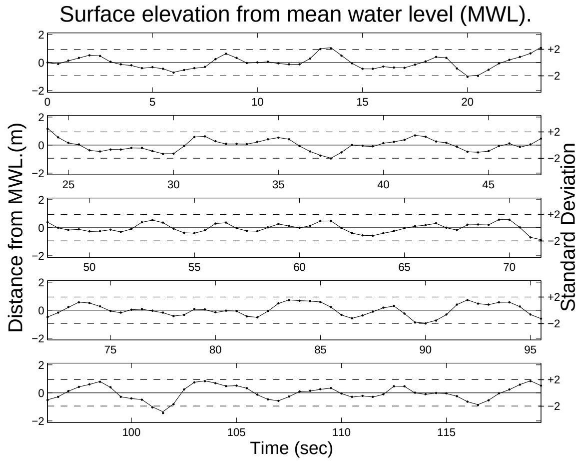


Figure 2.15: *Simulated $X(t) = G(\widetilde{X}(t))$ (solid line) compared with $\widetilde{X}(t)$ (dots) scaled to have the same H_s as $X(t)$ for a theoretical spectrum given by Torsethaugen spectrum* `St`.

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

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

CHAPTER 3

Empirical wave characteristics

One of the unique capabilities of WAFO is the treatment of the statistical properties of wave characteristic. This, and the next chapter, describe how to extract information on distributions of observables like wave period, wave length, crest height, etc, either directly from data, or from empirically fitted approximative models, or, in the next chapter, by means of exact statistical distributions, numerically computed from a spectral model.

We first define the different wave characteristics commonly used in oceanographic engineering and science, and present the WAFO routines for handling them. Then we compare the empirical findings with some approximative representations of the statistical distributions, based on empirical parameters from observed sea states. The code for the examples are found in the m-file `Chapter3.m`, and it takes a few seconds to run.

3.1 Introduction

3.1.1 The Gaussian paradigm - linear wave theory

In the previous chapter we discussed modeling of random functions 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 the 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 when they disappear.

The wave action on marine structures is often modeled using linear filters. Then the sea spectrum, together with the filter frequency function, gives a complete characterization of the response of the structure. 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 higher than some threshold, water may fill a structure and change its dynamical properties. The combined effect of apparent waves, often described by their height and wave period, is therefore important in ocean engineering. These aspects are discussed in more detail in the textbook [46].

The apparent waves will be described by some geometric properties, called wave characteristics, while frequencies of occurrences of waves with specified characteristics will be 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.

3.1.2 Wave characteristics in time and space

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 the subject of ongoing research, for example, [4, 43], for some general studies of Gaussian random surfaces, and [2, 5, 6, 64], for new wave related results. Related results for Lagrange models, are found in [29, 1, 30, 31, 32, 33].

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 they 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 meter. The *encountered sea*, another important one-dimensional record, can be collected by means of a ship-borne wave recorder moving across the random sea.

To analyze collected wave data we need natural and operational definitions of an individual wave, its period, height, steepness, and possibly some other meaningful characteristics. 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 characteristics are given in Figure 3.1.

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 help commands for `wavedef`,

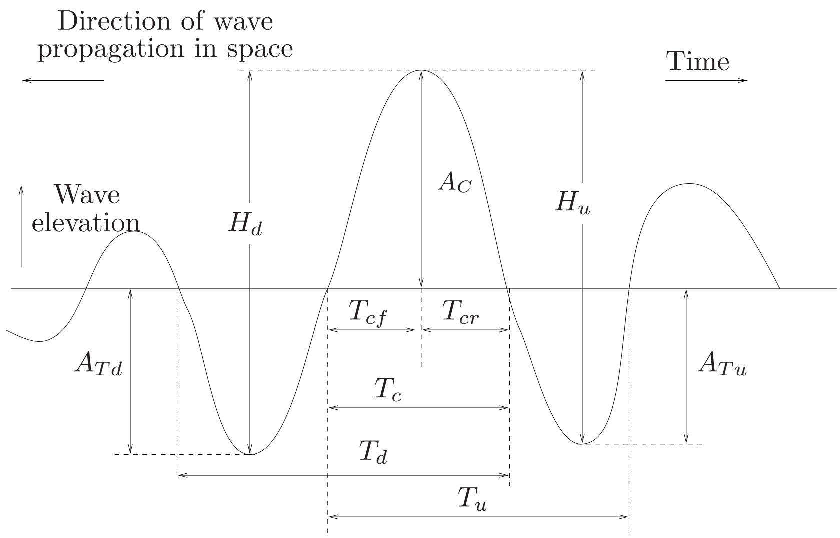


Figure 3.1: *Definition of wave parameters. The notation for the parameters used in our examples are given in Table 3.1.*

`perioddef`, `ampdef`, and `crossdef`, producing the output in Section 3.4.

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 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 will be insignificant for sufficiently large records. 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 a sea we seldom have a so long period 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 last from 20 minutes to only a 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 then extrapolate beyond the observed types of waves. What we shall be concerned with next is how to compute such distributional properties.

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

- To fit an 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

upcrossing wave		wave between two successive mean level upcrossings
downcrossing wave		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 back (rear) wave period	$T_{cb}(T_{cr})$	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 mean level upcrossings
downcrossing period	T_d	time between mean level downcrossings
crest-to-crest wave period	T_{cc}	time between successive wave crests
crest amplitude	A_c	crest height above mean level
trough depth	A_t	through depth below mean level ($A_t > 0$)
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 period		time from local minimum to next local maximum
min-to-max amplitude		height between local minimum and the next local maximum
max-to-min period/amplitude ...		similar to min-to-max definitions

Table 3.1: *Wave characteristic definitions*

to extract waves, estimate the densities and compute some simple statistics will be presented here in Chapter 3

- To simplify the model for the sea surface to such a degree that explicit computation of wave characteristic densities (in the simplified model) is possible. Some examples of proposed models from the literature will also be given here in this chapter.

- To exactly compute the densities from the mathematical form of a random seaway. This requires computation of infinite dimensional integrals and expectations that have to be computed numerically. WAFO contains efficient numerical algorithms to compute these integrals, algorithms which do not require any particular form of the sea surface spectrum. The method are illustrated in Chapter 4 on period, wavelength and amplitude distributions, for 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.)

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 that affect the mean level. De-trending can be done using the WAFO functions `detrend` or `detrendma`.

3.2.1 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 the possible definitions of period is the time 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 the 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 code will return crest2crest wave periods 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 density 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);
kopt = kdeoptset('L2',0);
ftc1 = kde(Tc,kopt,t);
pdfplot(ftc1), hold on
histgrm(Tc,[],[],1)
axis([0 8 0 0.5])
```

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

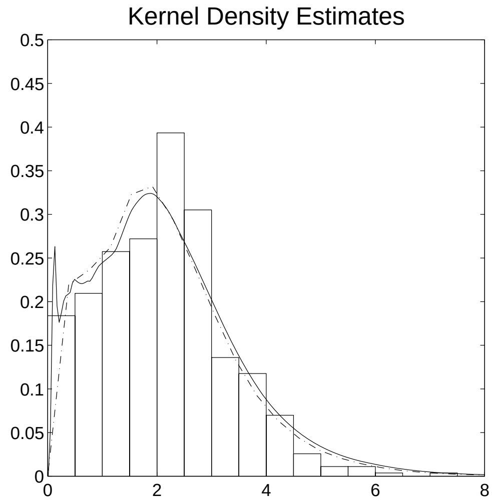


Figure 3.2: *Kernel estimate of the crest period density observed in* `sea.dat`*; solid line: full KDE, dash dotted line: binned KDE, compared with the histogram of the data.*

In Figure 3.2 we can see 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 Example 7a in Chapter 4, page 71. □

Remark 3.1 . Note that the program `kde` can be quite slow for large data sets. If a faster estimate of the density 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.)

```
kopt.inc = 128;
ftc2 = kdebin(Tc,kopt); pdfplot(ftc2,'-.')
title('Kernel Density Estimates'), hold off
```

The result is in Figure 3.2 □

3.2.2 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 `dat2steep` identifies waves and for each wave gives several wave characteristics (use the help function on `dat2steep` 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 code finds a sequence of waves and their characteristics:

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

The first preliminary analysis of the data is to find the individual waves which are extreme by some specified criterion, e.g. the steepest or the highest waves, etc. To do such an analysis one can use the function `spwaveplot(xx,ind)`, which plots waves in `xx` that 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 also be 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-')
axis([1482 1498 -1 3]), hold off
```

(Note that we have used the function `tranproc` instead of `gaus2dat`, since the last function requires 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 the interpolated signal `yn` has a few additional small waves that are not in `xx`.)

In Figure 3.3(b) the crosses are the removed values from the wave. The reconstructed wave, plotted by a solid line, is close 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, pointwise. From the figure we can deduce that the 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 than 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 a transformed Gaussian model.

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

We shall not investigate this question further in this tutorial. □

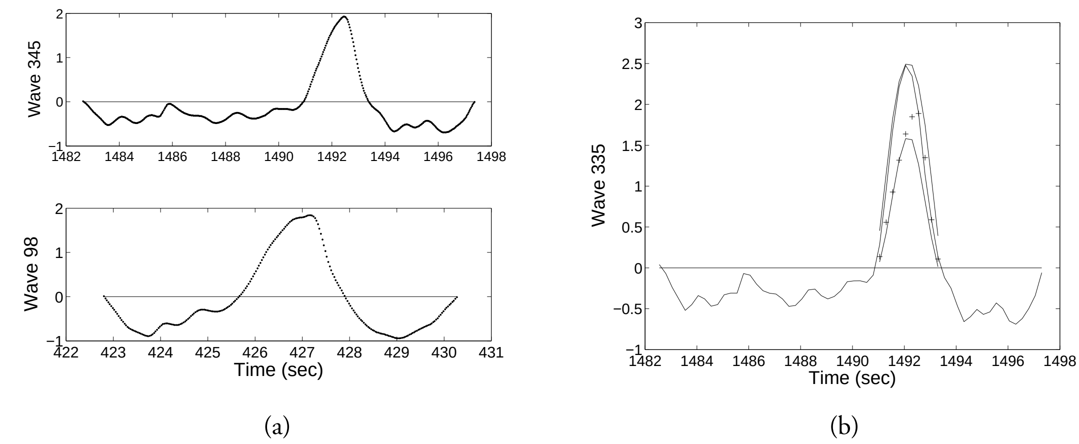


Figure 3.3: *(a): Two waves, the highest and the steepest, observed in* `sea.dat`*. (b): Crosses are observations removed from the highest wave. The reconstructed wave, using transformed Gaussian model, is given by the middle solid line. Upper and lower curves give the confidence band defined as the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

3.2.3 Crest height

We turn now to the kernel estimators of the crest height density. 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 the 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;
plotnorm(Ac.^L2)
fac = kde(Ac,{'L2',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 uncertainty of the computed probability is at least 0.03. We suspect that this is due to the estimated density being non-zero for negative values. In order to check this we compute the cumulative distribution using the formula,

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

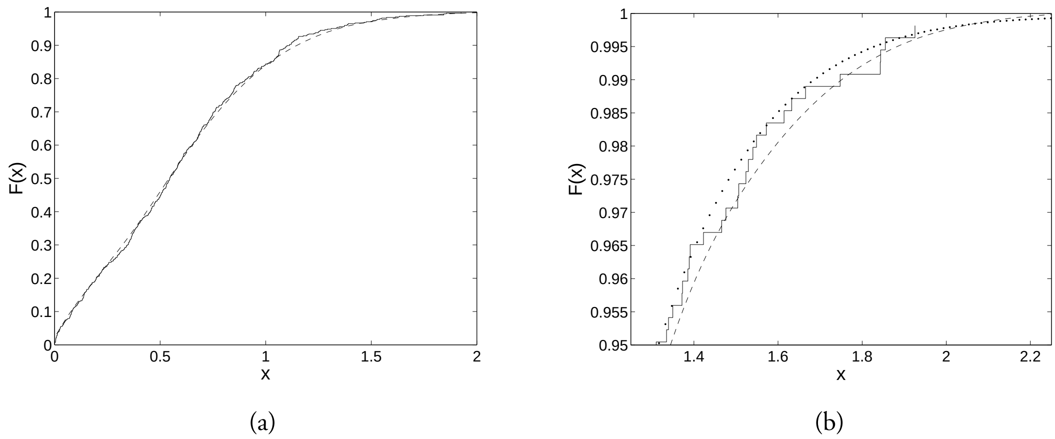


Figure 3.4: *(a) Comparison of the empirical distribution of the crest height with the cumulative distribution computed from the KDE estimator. (b) Zooming in on the tails of distributions in (a) together with the tail of the transformed Rayleigh approximation (dots) to the crest height distribution.*

where $f_{Ac}(x)$ is the estimated probability density of Ac. For the pdf saved in `fac` the following 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 `edf` or `plotedf`

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

Since a kernel density estimator KDE in principal is a smoothed histogram it is not very well suited for extrapolation of the density to the region where no data are available, e.g. for the high crests. In such a case a parametric model should be used. In WAFO there is a function `trraylpdf` that combines the non-parametric 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 Rayleigh variable should be used for the crests of the transformed Gaussian waves. The method has several nice properties and will be described more in Section 3.3.3. Here we just use it in order to compare with the non-parametric KDE method.

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

Figure 3.4(a) shows that our hypothesis that the pdf `fac` is slightly too low for small crests seems to be correct. Next from Figure 3.4(b) we can see that also the tail is reasonably modeled even if it is lighter than, i.e. gives smaller probabilities of high waves than, the one derived from the transformed Gaussian model. □

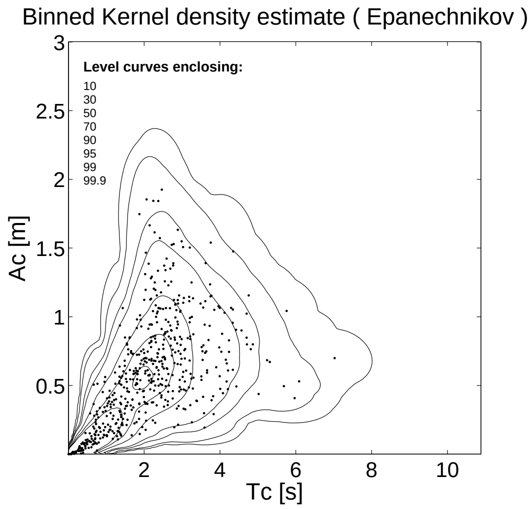


Figure 3.5: *Kernel estimate of joint density of crest period* `Tc` *and crest height* `Ac` *in* `sea.dat` *compared with the observed data (dots). The contour lines are drawn in such a way that they contain specified (estimated) proportions of data.*

3.2.4 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 estimators give poor estimates of the tail of the distribution, unless large amounts of data is available. However, a KDE gives qualitatively good estimates in regions with 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 `Tc = Tcf+Tcb` and crest height `Ac` in `sea.dat`.

```
kopt2 = kdeoptset('L2',0.5,'inc',256);
Tc = Tcf+Tcb;
fTcAc = kdebin([Tc Ac],kopt2);
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 are plotted 544 pairs of crest period and height. 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 chapter we shall compute the theoretical joint density of crest period and height from the transformed Gaussian model and compare with the KDE estimate. □

3.3 Explicit results - parametric wave models

In this section we shall consider the Gaussian sea. We assume that the reference level is zero and that the spectrum is known. We will present some explicit results that are known and studied in the literature about wave characteristics. Some of them are exact, others 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 behavior of many wave characteristics. The WAFO routines `spec2char` and `spec2bw` compute a long list of wave characteristic parameters.

Example 4. *(Simple wave characteristics obtained from 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 spectral 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.

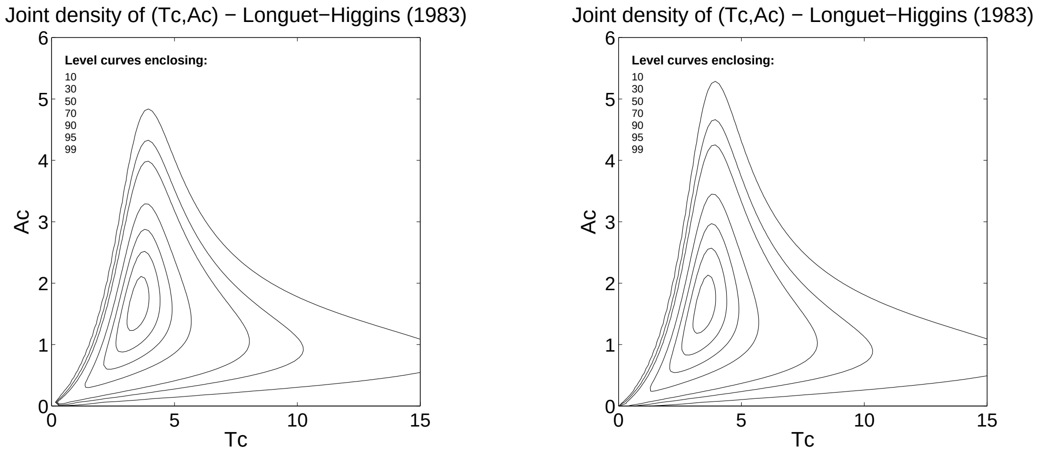
```
spec2bw(S)
[ch Sa2] = spec2char(S,[1  3])
```

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 [28], [71] and [74] 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 approximations of wave distributions

In the module `wavemodels`, we have implemented some of the approximative models that have been suggested in the literature. To get an overview of the routines in the module, use the help function on `wavemodels`.

We will investigate two suggested approximations for the joint pdf of `(Tc,Ac)` (for the nomenclature, see the routines `perioddef` and `ampdef` in the module `docs`). Both functions need spectral moments as inputs. One should bear in mind that the models only depend on a few spectral moments and not on the full wave spectrum.



(a) (b)

Figure 3.6: *Longuet-Higgins model 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.*

Model by Longuet-Higgins

Longuet-Higgins, [37, 38], derived his approximative 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 results for big waves, e.g. for waves with significant amplitudes.

The Longuet-Higgins density depends, besides the significant wave height H_s and peak period T_p, on the spectral width parameter $\nu = \frac{m_0 m_2}{m_1^2} - 1$, which can be calculated by the command `spec2bw(S,'eps2')`, (for a narrow-band process, $\nu \approx 0$). The explicit density is 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)]);
```

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.

```
[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 contour lines are drawn in such a way that they contain predefined proportions of the total probability mass inside the contours. 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 proposed by Cavanié et al., [13]. Here any positive local maximum is considered as a crest of a wave, and then the second derivative (curvature) at the local maximum defines the wave period by means of a cosine function with the same height and the same crest curvature.

The model uses the parameter ν and a higher order bandwidth parameter[1] ε, defined by

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

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

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

where

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

The density is computed by

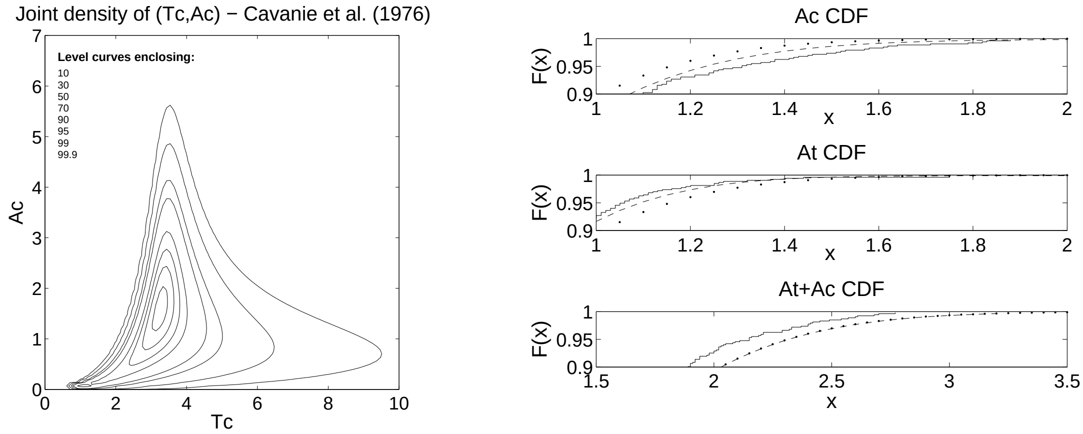
```
t = linspace(0,10,100);
h = linspace(0,7,100);
fcav = cav76pdf(t,h,[m(1) m(2) m(3) m(5)],[]);
```

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

3.3.3 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 simple and most frequently used model is the Rayleigh

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



(a) (b)

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.*

density. The standardized Rayleigh variable R has the density given by $f(r) = r\exp(-r^2/2)$, $x > 0$. It is well known that for Gaussian sea the Rayleigh approximation works very well for high waves, and actually it is a conservative approximation since we have

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

see [58]. In that paper it is also shown that for any sea wave model with crossing intensity $\mu(u)$, one has $\mathsf{P}(A_c > h) \leq \mu(u)/\mu(0)$. 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 $X'(t)$ is known, see Section 2.2.3,

$$\mu(u) = \int_0^{+\infty} z f_{X(t),X'(t)}(u,z)\,dz.$$

For a Gaussian sea it can be computed explicitly

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

For non-linear wave models with random Stokes waves the crossing intensity has to be computed using numerical integration; see the work by Machado and Rychlik, [39].

Knowing the crossing intensity $\mu(u)$ one can compute the transformation g, by using the routine `lc2tr`, such that the transformed Gaussian model has crossing intensity equal to $\mu(u)$. Consequently, we have that $\mathsf{P}(A_c > h) \leq \mathsf{P}(R > g(h)) = 1 - \mathsf{P}(G(R) \leq h)$. 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 high 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 [58].

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 sea spectrum. 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);
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 = plotedf(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 = plotedf(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 = plotedf(AcAt,Facat_h); hold on
r2 = (0:05:2.1*Hs)';
plot(r2,1-exp(-2*r2.^2/Hs^2),'.')
axis([1.5 3.5 0.9 1]), hold off
```

In Figure 3.7(b) we can see some differences between the observed crest and trough distributions and the one obtained from the transformation gh. However, it still gives a much better approximation than 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 too conservative approximations to the distribution of wave amplitude. □

3.4 WAFO wave characteristics

3.4.1 spec2char

```
help spec2char

 SPEC2CHAR Evaluates spectral characteristics and their variance

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

       ch = vector of spectral characteristics
       r  = vector of the corresponding variances given T
   chtext = a cellvector of strings describing the elements of ch
       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    = 2*pi/{w | max(S(w))} 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    = abs(int S(w) exp(i*w*Tm02) dw) / m0
                                    Groupiness parameter
   10 Rs    = se help spec2char     Quality control parameter
   11 Tp    = 2*pi*int S(w)^4 dw    Peak Period
              ------------------    (more robust estimate)
              int w*S(w)^4 dw

   12 alpha = m2/sqrt(m0*m4)        Irregularity factor
   13 eps2  = sqrt(m0*m2/m1^2-1)    Narrowness factor
   14 eps4  = sqrt(1-m2^2/(m0*m4)) = sqrt(1-alpha^2)  Broadness factor
   15 Qp    = (2/m0^2)int_0^inf w*S(w)^2 dw           Peakedness factor

 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.
```

3.4.2 spec2bw

```
help spec2bw}

 SPEC2BW Evaluates some spectral bandwidth and irregularity factors

  CALL:  bw = spec2bw(S,factors)

         bw = vector of factors
         S  = spectrum struct
    factors = vector with integers, see below. (default [1])

  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 alpha=m2/sqrt(m0*m4)                        (irregularity factor)
     2 eps2 = sqrt(m0*m2/m1^2-1)                   (narrowness factor)
     3 eps4 = sqrt(1-m2^2/(m0*m4))=sqrt(1-alpha^2) (broadness factor)
     4 Qp=(2/m0^2)int_0^inf f*S(f)^2 df            (peakedness factor)
  Order of output is the same as order in 'factors'

  Example:
    S=demospec;
    bw=spec2bw(S,[1 2 3 4]);
```

3.4.3 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
 an up-crossing to the next up-crossing.

 Definition of trough and crest waves:
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 A trough to trough 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.
 A Max2Max wave (Mw) is 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/    \    :    /: \          /:\   level v
------d--------u----------d-------u----d--------u---d--------
       \      /:           \  :  /: :  :\_    _/  : :\_   L
        \_   / :            \_t_/ : :  :  \t_/    : :  \m/
          \t/  <-------uw---------> :  <-----dw----->
           :                  :     :             :
           <--------tw-------->     <------cw----->
 (F= first value and L=last value).
See also: tpdef, crossdef, dat2tc, dat2wa, dat2crossind
```

3.4.4 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/    \:      :/  \          / \   level v
 ------d--------u----------d------u----d--------u---d--------
        \      /            \    /     :\_    _/:   :\_   L
         \_   /              \t_/      :  \t_/  :   :  \m/
           \t/                :        :        :   :
            :<-------Ttt----->:        <---Tt--->   :
                                       :<----Td---->:
   Tu   = Wave up-crossing period
   Td   = Wave 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/    \           /  \___:     :/ \ level v
------d---------u----------d---------u-------d-----u---d-------
      :\_      /            \     __/:        \   /     \_   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 rules apply 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)

             <----- Direction of wave propagation
                      <--TMm-->
           <-TmM->    :       :
   M       :     :    M       :
  / \      :     M   /:\_     :     M_            M
 F   \     :    / \m/ :  \    :    /: \          / \
      \    :   /      :   \   :   / :  \        /   \
       \   :  /       :    \  :  /  :   \_    _/     \_   L
        \_ : /        :     \_m_/   :     \m_/         \m/
          \m/         :             :      :            :
                      <-----TMM----->      <----Tmm----->

  TmM = Period between minimum and the following Maximum
  TMm = Period between Maximum and the following minimum
  TMM = Period between Maximum and the following Maximum
  Tmm = Period between minimum and the following minimum
See also: wavedef, ampdef, crossdef, tpdef
```

3.4.5 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/     |      \    |    /  \  level v
------d----|---u------------------d---|---u----d------
       \   |  /|                   \  |  /      \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
```

3.4.6 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 down-crossing.
  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 up-crossing.

  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   level v
  ----------------------u-------d-------o---------------------
               .     .           .   .   u
                 .                 m
                  m

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

CHAPTER 4

Exact wave characteristics

The wave characteristic distributions in Chapter 3 have been empirical, either constructed directly from data, or from a specific model fitted by means of data, via a few spectral moments. In this chapter we will use the Gaussian paradigm, described in Section 3.1.1, to produce exact wave characteristic distributions directly from an assumed spectral density, without any further assumptions than Gaussianity and a following transformation. This is a unique facility in WAFO, not available in any other wave analysis software. The routines are collected in the module `trgauss` and they are listed in Section 4.4.

The functions are the results of long time research at Lund University, see [48] and [34], where a review of the historical development and the mathematical tools behind the algorithms are given.

The MATLAB code for the examples in this chapter are found in `Chapter4.m` and is takes 20 minutes to run on a 2.93 GHz 64-bit PC, in fast mode.

4.1 Exact wave distributions routines

By means of a number of examples, we shall demonstrate the most important functions for computation of exact wave probability distributions. The variables are the crest and wave periods, `Tc`, `Tu = Tc+Tt`, the corresponding crest length and wave length variables, `Lc`, `Lu = Lc+Lt`, and crest and trough height `Ac, At`, and we compute both marginal densities and joint densities for combination of variables. The same functions compute densities for trough period, length, and height, as for the corresponding crest variables. The common form of the routines is `spec2yyxxx` .

In WAFO there are also functions computing exact densities for other wave characteristics, which will not be presented here. The WAFO routines are collected in the module `trgauss`. Use the help function on `trgauss` to see the following list of all the routines.

4.2 Marginal distributions of wave characteristics

In this section we analysis the marginal distributions of crest and wave period/length variables, and how they depend on the crest height. We also discuss the numerical accuracy of the WAFO routines, and how to obtain a reasonable compromise between accuracy and computational speed. More example on this matter will follow in subsequent sections. We start will some introductory examples.

4.2.1 Crest period, crest length and crest height

One of the most useful functions in WAFO is the routine `spec2tpdf`, which computes the density function for crest and trough period, as well as for the corresponding length variables. The function also computes the density of 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 (length) of such high waves.

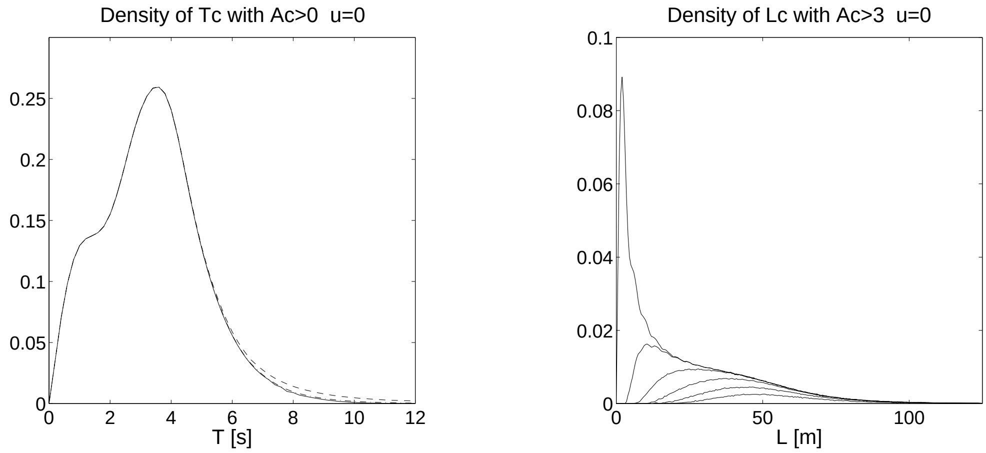
The function `spec2tpdf` performs all necessary transformations, scalings, etc, making it very flexible. It handles different spectra as inputs. Which kind of density is computed (output) is defined by the variable `def` that takes values `'Tc'` for crest period, `'Lc'` for crest length, `'Tt'` for trough period, and `'Lt'` for trough length. The transformation is only affecting the value of the still water level `u` and the threshold `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. *(Torsethaugen waves)* We start by defining the same frequency spectrum, $S(\omega)$, as we used in Chapter 1; 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; see Figure 1.2. The energy is divided between two peaks, corresponding to contributions from wind and swell. We shall also use the two directional spectra from Chapter 1 with frequency dependent, `SD1`, and frequency independent, `SD12`, spreading.

```
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);
```

Example 6a. Crest period: 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 a theoretically derived formula, which is described, e.g., in [35]. The algorithm gives an upper bound (and if requested lower bound too) 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_4 = spec2tpdf(S1,[],'Tc',[0 12 61],[],4);
f_tc_1 = spec2tpdf(S1,[],'Tc',[0 12 61],[],-1);
```



(a) (b)

Figure 4.1: *(a) Densities* `f_tc_1` *(solid),* `f_tc_2` *(dashed), and* `f_tc_4` *(dash dotted) of crest period* `Tc` *for Torsethaugen spectrum. (b) Densities of crest length* `Lc`*, (most peaked curve) compared to the density when restricted to waves with crest height* `Ac` *more than 10%, 20%, 30%, 40%, 50% of the significant wave height above the still water level, for Gaussian sea with the Torsethaugen spectrum with* `Hs = 6[m]`*. Lowest curve corresponds to* `Ac > 3[m]`.

```
pdfplot(f_tc_4,'-.'), hold on
pdfplot(f_tc_1), hold off
simpson(f_tc_4.x{1},f_tc_4.f)
simpson(f_tc_1.x{1},f_tc_1.f)
```

The crest period density is shown in Figure 4.1(a). The integral of the density `f_tc_4` computed using the function `simpson` is 1.005, showing the high accuracy of the approximation. The density `f_tc_1` uses another algorithm, which is faster, and it has the integral 0.9993. The computation time is 1.4 and 0.5 seconds, respectively, on a PC, Pentium 2.9 GHz. The computation time depends on the required accuracy and how broad banded the spectrum is. For example, the same accuracy is achieved for the JONSWAP spectrum in about half the time. The computation time increases if there is a considerable probability for long waves with low crests.

The last argument in the calls above to `spec2tpdf` is worth special attention, and we will later study its effect in detail. It controls the numerical algorithm that computes the density. Here, we only note that a positive choice, here 4, gives an upper bound to the density, more accurate and more time consuming the higher the value, while a negative value, here `-1`, given an almost unbiased value in much shorter time.

Example 6b. Crest length: 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'`.

```
f_Lc = spec2tpdf(S1,[],'Lc',[0 125 251],[],-1);
```

```
pdfplot(f_Lc,'-.'), hold on
```

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; see [36] for a general comparison of wave period and wave length. 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 have crest higher than a certain proportion of the significant wave height, e.g. 25%, i.e. one standard deviation, `Hs/4 = 1.5[m]`, and give the density of the crest length for these waves. This can be done by specifying an extra argument in the call to `spec2tpdf`.

```
f_Lc_1 = spec2tpdf(S1,[],'Lc',[0 125 251],1.5,-1);
pdfplot(f_Lc_1)
```

Figure 4.1(b) presents the results when the crest height is restricted to more than 10%, 20%, 30%, 40%, 50% of the significant wave height. and we can see that all short waves in fact were small. (The algorithm produces some very small negative density values. These have been removed before the plotting; see the following section on numerical accuracy, Section 4.2.2.)

The proportion of waves with crests above 1.5 [m] (one standard deviation) is computed by the following commands.

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

Taking the ratio, we can see that more than half of the waves are small, about 37% of the waves have crests above 1.5 [m]. Similar calculations for the curves in Figure 4.1(b), give the proportions of crests above the levels in Table 4.1.

level [m]	0.6	1.2	1.5	1.8	2.4	3.0
proportion above level	0.607	0.435	0.367	0.305	0.191	0.102
CDF of crest height `Ac`	0.391	0.563	0.630	0.693	0.806	0.895

Table 4.1: *Second row: proportion of crest heights above a level, computed by* `spec2tpdf`*; Third row: CDF of crest height computed by* `spec2acdf`.

Example 6c. Crest height The table of the proportion of high crest waves is related to the cumulative distribution function (cdf) of the crest height `Ac` in a natural way. The WAFO routine `spec2acdf` computes the cdf directly, both for the crest height over a crest period in time and for the crest height over a crest length in space. Figure 4.2 shows the empirical distribution of `Ac` in a long simulation in time, and the theoretical distribution functions for crest height in time and in space, as well as the Rayleigh approximation from Section 3.3.3. The simulations contain 9255 space wave crests and 5823 time wave crests. The agreement between the empirical and theoretical distribution function are very good. The Rayleigh distribution gives a good approximation of the time crest height for large crest values but overestimates the smallest crests.

The third row in Table 4.1 shows the cdf values for crest height in space computed by `spec2acdf`. The sum of the second and third row should be one; all sums in the table are greater than `0.997`.

```
clf; Hs = 6;
r = (0:0.12:1.1*Hs)';
F_Ac_s1_T = spec2acdf(S1,[],'Tc',[0 12 61],r,-1); hold on
T = spec2sdat(S1,[40000,100],0.01);
[SteepT,HeightT,AcT] = dat2steep(T);
plotedf(AcT,'-.')
F_Ac_s1_L = spec2acdf(S1,[],'Lc',[0 125 251],r,-1);
L = spec2sdat(spec2spec(S1,'k1d'),[40000 100],0.1);
[SteepL,HeightL,AcL] = dat2steep(L);
plotedf(AcL,'-.')
plot(r,1-exp(-8*r.^2/Hs^2)), hold off
```

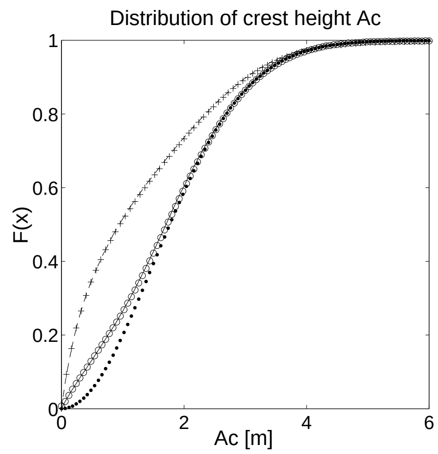


Figure 4.2: *Cumulative distribution (cdf) for crest height* `Ac` *with Torsethaugen spectrum. Curves most to the left = theoretical (solid) and empirical (dash dotted) cdf for* `Ac` *over a crest length. Middle curves show theoretical and empirical cdf for* `Ac` *over a crest period. Curve most to the right is the cdf for the Rayleigh approximation.*

Example 6d. Directional spreading: We finish this example by considering the Torsethaugen spectrum with the two different spreading functions `SD1` and `SD12`. In Figure 1.5 we presented simulations of the sea surfaces with these spectra. From the figures 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` have different main wave directions, 90^o and 0^o degrees, respectively, and hence we shall choose different azimuths for the two spectra. More precisely, for both cases we shall consider heading waves; this is achieved using the function `spec2spec`.

```
f_Lc_d1 = spec2tpdf(spec2spec(SD1,'rotdir',pi/2),[],...
```

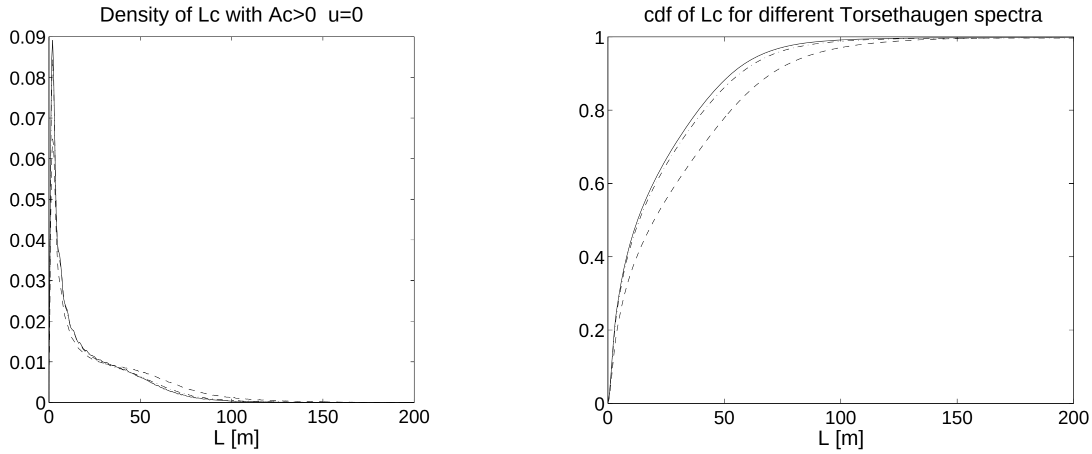


Figure 4.3: *Computed pdf (left) and cdf (right) for* `Lc` *in Gaussian sea with Torsethaugen spectrum with different spreading: unidirectional spectrum* `S1` *(solid line) ; frequency independent spreading* `SD1` *(dash-dotted line); frequency dependent spreading* `SD12` *(dashed line).*

```
          'Lc',[0 200 401],[],-1);
pdfplot(f_Lc_d1,'-.'), hold on
f_Lc_d12 = spec2tpdf(SD12,[],'Lc',[0 200 401],[],-1);
pdfplot(f_Lc_d12), hold off

figure(2)
dx = f_Lc.x{1}(2)-f_Lc.x{1}(1);
dx1 = f_Lc_d1.x{1}(2)-f_Lc_d1.x{1}(1);
dx12 = f_Lc_d12.x{1}(2)-f_Lc_d12.x{1}(1);
plot(f_Lc.x{1},cumsum(f_Lc.f)*dx), hold on
plot(f_Lc_d1.x{1},cumsum(f_Lc_d1.f)*dx1,'-.')
plot(f_Lc_d12.x{1},cumsum(f_Lc_d12.f)*dx12,'--'), hold off
```

As expected, after examination of the simulated sea surfaces in Figure 1.5, the crest length for the two directional spectra are different. The sea with frequency dependent spreading seems to be more irregular. We can see in Figure 4.3 that waves with frequency independent spreading are only slightly longer than the waves in unidirectional sea, while the crest length of both seas are much shorter than for frequency dependent spreading. From Figure 1.4 it is clear that the spectrum with frequency independent spreading function is more similar to the unidirectional spectrum than that with the frequency dependent spreading. □

4.2.2 Numerical accuracy and computational speed

The basic algorithm in the routine `spec2tpdf` computes a finite-dimensional approximation to an "infinite-dimensional" normal probability. The last input in all the previous calls to the routine is a parameter called `nit`, and it determines both the integration method and the dimensionality of the computed integral. Important references on how to compute normal probabilities are [3, 10, 11, 18, 19, 55].

The nit parameter can be positive, negative, and zero. Positive nit values use numerical, deterministic, integration algorithms, while negative values use a simulation technique based on importance sampling; see [10, 11] for a review of different methods.

The methods with positive nit are very reliable and have been tested on different wave problems since the first version was used already in 1987; see [53]. As default, they give an upper bound to the densities. The integration methods corresponding to negative nit values are still under tests and modifications. However, they are often much faster and also very accurate in cases when the deterministic method has troubles with too long execution times.

Although the method with negative nit is based on simulation the accuracy is still controlled. If the number of simulations is too small to achieve the required accuracy the program gives an error statement with an estimate of the possible error in the computed density.

One should be aware that both positive and negative nit values can produce negative density values with spec2tpdf. This is the result of the way the densities are computed, namely as differences between "cumulative distribution type" functions. Then, small numerical variations may cause negative density estimates, mostly for very small density values.

The routine spec2tpdf is the MATLAB interface to a FORTRAN 95 program. All programs computing exact 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; see [35] for more details. 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.

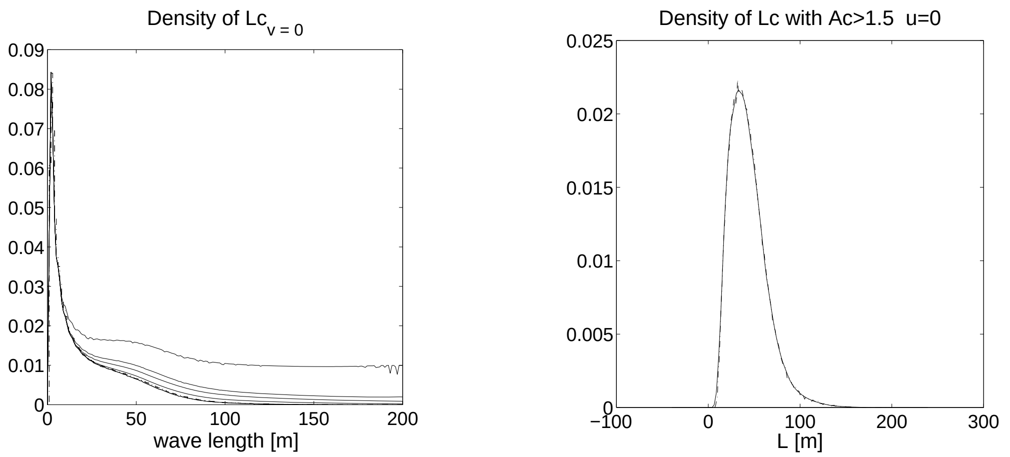
An example is a function spec2tpdf2 which uses the program rind. The program is slower than spec2tpdf, and it does not have an option that allows to choose waves with crest above some level, but on the other hand it is easier to use for experimentation, and it can also be used to learn how to create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

Besides the parameter nit, the input parameter speed will also control the accuracy of the computations in module trgauss; see the help text to the routines for information.

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.

```
opt1 = rindoptset('speed',5,'method',3);
SD1r = rotspec(SD1,pi/2);
f_Lc_d1_5 = spec2tpdf(SD1r,[],'Lc',[0 200 201],[],5);
f_Lc_d1_3 = spec2tpdf(SD1r,[],'Lc',[0 200 201],[],3);
f_Lc_d1_2 = spec2tpdf(SD1r,[],'Lc',[0 200 201],[],2);
f_Lc_d1_0 = spec2tpdf(SD1r,[],'Lc',[0 200 201],[],0);
f_Lc_d1_neg = spec2tpdf(SD1r,[],'Lc',[0 200 201],[],-1);
f_Lc_d1_n4 = spec2tpdf2(SD1r,[],'Lc',[0 200 201],opt1);

pdfplot(f_Lc_d1_5), hold on
pdfplot(f_Lc_d1_2), pdfplot(f_Lc_d1_3)
pdfplot(f_Lc_d1_0), pdfplot(f_Lc_d1_neg)
pdfplot(f_Lc_d1_n4,'LineWidth',2,'-.')
simpson(f_Lc_d1_n4.x{1},f_Lc_d1_n4.f)
```



(a) (b)

Figure 4.4: *(a) Approximations by different methods and accuracy of the crest length for the directional spectrum with frequency independent spreading. The top solid curves are computed with positive* `nit = 0 (top), 2, 3, 5`*, and negative* `nit = -1 (bottom)`*, in routine* `spec2tpdf`*, while the dash-dotted curve has negative* `nit` *with routine* `spec2tpdf2`*. (b) Solid curve = the empirical density of crest length* `Lc` *with crest height* `Ac > 1.5[m]`*, dash-dotted curve is the normalized computed density* `f_Lc_1`*.*

The execution times for the densities were 270 seconds, 11 seconds, 2.4 seconds, 0.3 seconds, 7.7 seconds, and 5.9 seconds, respectively. In Figure 4.4(a) the different approximations are presented and we can see how the density decreases with increasing positive `nit`. 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` often suffices, but here clearly the negative `nit` is preferable.

In Figure 4.4(b) we compare an empirical density of crest length `Lc`, conditioned on crest height `Ac > 1.5[m]`, based on almost 500 000 observed waves, with the normalized density `f_Lc_1` from page 65, computed with `nit = -1`. The agreement is almost perfect. □

4.2.3 Wave period and wave length

In the previous sections we described routines for the marginal distributions of crest and trough periods, and height, `Tc`, `Tt`, `Ac`, and the corresponding crest and trough lengths length, `Lc`, `Lt`. We also showed how to limit the population to waves for which the crest (trough) amplitudes are above some predetermined threshold.

We now turn to the wave period, `Tu = Tc+Tt`, which is the time between two successive upcrossings of the still water level `u`. It is related to, but not equal to, the crest-to-crest wave period `Tcc`, which is the time span between two successive crests. The density of `Tu` can be computed using the function `spec2tccpdf`. The wave length `Lu` or encountered wave period can also be computed by `spec2tccpdf`, with just a few inputs to be modified; see the help text. Hence, these variables shall not be discussed here any more.

The computations using `spec2tccpdf` are slower than those using `spec2tpdf`, since one needs to compute the joint density of `Tc` and `Tt` and then change variable to integrate the convolution to get `Tu = Tc+Tt`. See also the discussion in the remark in the previous section about speed of programs. It should be mentioned that, in addition to the methods to reduce computation time, one of the best methods to speed up computation 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.

Example 7. *(Sea data wave distributions)* In order to be able to make comparisons with the wave characteristic distributions in `sea.dat` we shall use the estimated spectrum `SS`, see Example 1 on pages 18 and 24.

We first re-compute the spectrum estimate and the transformation to Gaussianness, and extract some characteristics. The estimated spectrum is plotted in Figure 4.5, together with pointwise 95% confidence intervals.

```
xx = load('sea.dat');
x = xx;
x(:,2) = detrend(x(:,2));
SS = dat2spec(x);
si = sqrt(spec2mom(SS,1));
SS.tr = dat2tr(x);
Hs = 4*si
```

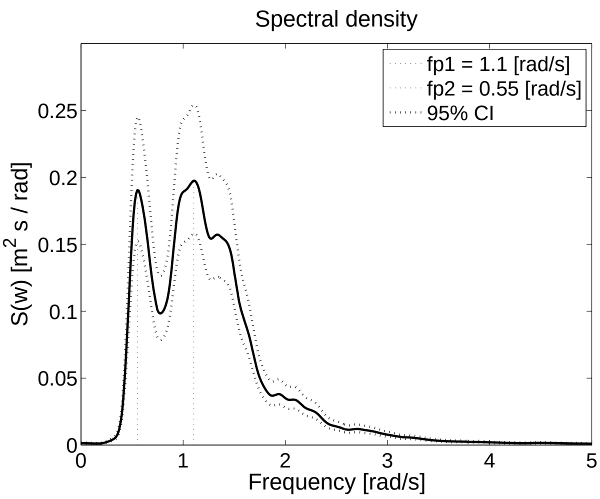


Figure 4.5: *Estimated spectrum for data* `sea.dat` *with confidence bands.*

Example 7a. Crest period: We first consider the crest period, as we did in Example 6a and also the proportion of crests with significant crest height, i.e. `Tc` when `Ac > Hs/2`, in the same way as we did for crest length in Example 6b. After that we will do the same for wave period and consider `Tu` when `Ac > Hs/2`. The proportion of crests periods with significant crest height should be the same as the proportion of wave periods with significant crest height, i.e. `Tu` when `Ac > Hs/2`. The difference between the two proportions gives an indication of the accuracy in the computation of the convolution `Tu = Tc + Tt`.

We can also compare the calculated proportion of significant crests with the proportion observed in data and with the approximative Rayleigh model. Finally, we estimate the density using KDE from data and compare to the theoretically computed one, based on the transformed Gaussian model.

For completeness we again estimate the transformation and find wave characteristics in the signal. The crest period, Tc, distribution, estimated from data, and the computed density are almost identical, except for very short waves; see Figure 4.6(a), obtained by the following commands. Note the last output variable yn, which is an interpolated series, to be used later.

```
method = 0; rate = 2;
[S,H,Ac,At,Tcf,Tcb,z_ind,yn] = dat2steep(x,rate,method);
Tc = Tcf+Tcb;
t = linspace(0.01,8,200);
f_tc1emp = kde(Tc,{'L2',0},t);
pdfplot(f_tc1emp), hold on
f_tc1 = spec2tpdf(SS,[],'Tc',[0 8 81],0,4);
simpson(f_tc1.x{1},f_tc1.f)
pdfplot(f_tc1,'-.'), hold off
```

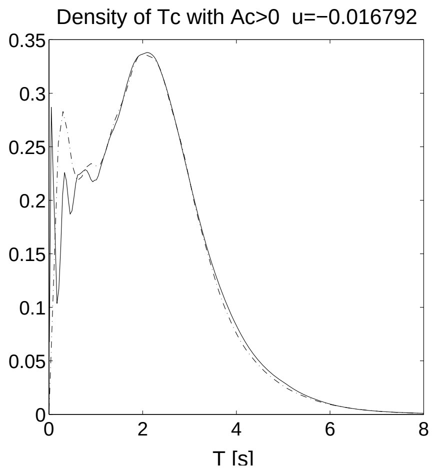
We next consider computation of the density of crest period, but now for waves with significant crest height, i.e. waves for which `Ac > Hs/2`. In the following call to `spec2tpdf` the restriction to `Ac > Hs/2` is indicated by the argument `[Hs/2]`.

```
nit = 4;
f_tc2 = spec2tpdf(SS,[],'Tc',[0 8 81],[Hs/2],nit);
Pemp = sum(Ac>Hs/2)/sum(Ac>0)
simpson(f_t2.x{1},f_t2.f)
index = find(Ac>Hs/2);
f_tc2emp = kde(Tc(index),{'L2',0},t);
f_tc2emp.f = Pemp*f_tc2emp.f;
pdfplot(f_tc2emp), hold on
pdfplot(f_tc2,'-.'), hold off
```

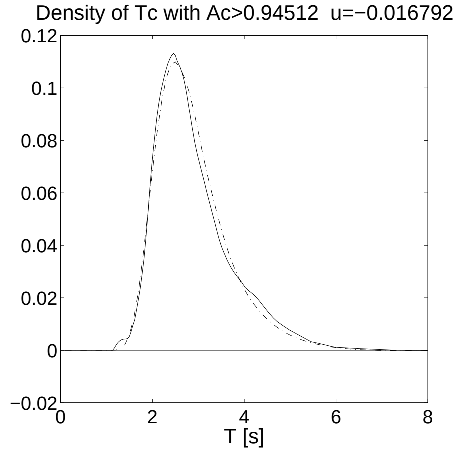
The observed frequency of significant crests, Pemp, is 0.1778 which is remarkably close to the theoretically computed value 0.1789, obtained with a computation time of 21 seconds. (Observe that the Rayleigh approximation would give a probability equal to 0.1353. This is not surprising since crests in non-Gaussian sea tend to be higher than those in Gaussian sea.) Clearly, by changing the input Hs/2 to any other fixed level h, and integrating the resulting density we obtain approximations to the probability `P(Ac > h)`.

If h is a vector then it is more efficient to use the program `spec2Acdf` to compute `P(Ac > h)`, as in Example 6c. However, before using the program it is important to first use `spec2tpdf` 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 very different.



(a)



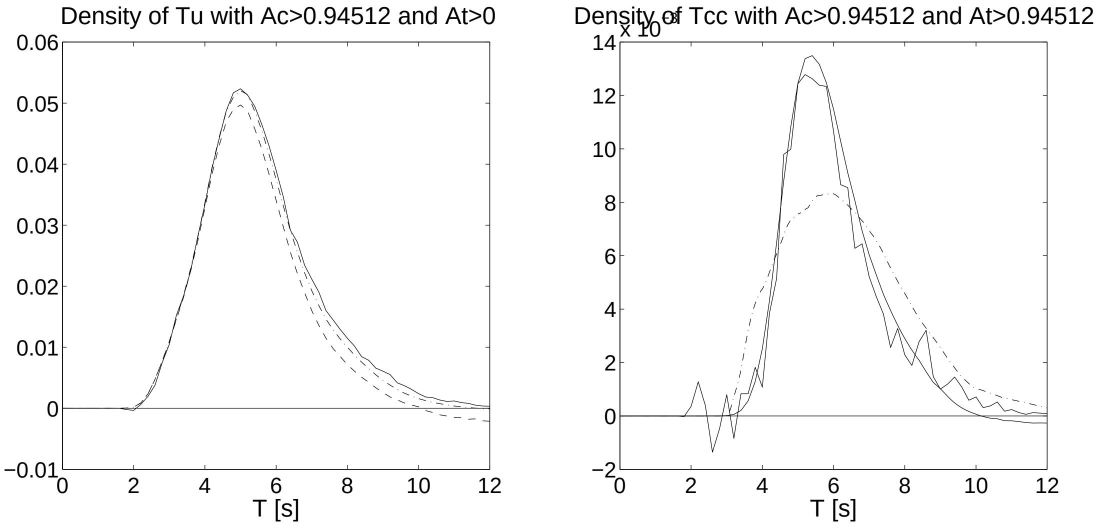
(b)

Figure 4.6: *(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`.

Example 7b. Wave period for high-crest waves: We turn now to the more difficult problem of wave period density for waves with significant crest height, `Ac > Hs/2` and with `At > 0`. As mentioned, this differs from Example 7a in that it involves the distribution of the sum `Tc + Tt` 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 unconditional wave period and for the period of waves with crest below a given threshold), we need to use a high positive `nit` value, so that the total sum of the density is close to 0.1789, or use a negative `nit`. We begin with negative `nit`, which gives faster results very close to the true density, and then take `nit = 3`.

```
f_tun = spec2tccpdf(SS,[],'t>',[0 12 61],[Hs/2],[0],-1);
simpson(f_tun.x{1},f_tun.f)
f_tu3 = spec2tccpdf(SS,[],'t>',[0 12 61],[Hs/2],[0],3,5);
simpson(f_tu3.x{1},f_tu3.f)
pdfplot(f_tun), hold on
pdfplot(f_tu3,'--'), hold off
```

The integral of the density `f_tccn` is 0.1778, which is close to the previously computed value 0.1789. However, the execution time was 66 seconds, compared to 21 seconds for `f_t2`. The choice `nit=3` takes 3 minutes and give the integral 0.15. We have checked the program with `nit=5` (execution times 66 minutes), and the integrals was 0.17. The densities are shown in Figure 4.7(a). We can see that the density computed using `nit=-1` (dash-dotted line) is quite accurate, even if it slightly wiggly, being 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.



(a) (b)

Figure 4.7: *(a) Densities of upcrossing period* Tu *for waves with significant crest in the transformed Gaussian model of the sea data in* sea.dat *computed with different degree of accuracy; (solid line)* nit=-1*; the dashed line is computed with* nit=5 *and the dash dotted line with* nit=3*. (b) Densities of period* Tu *for waves with significant crest and trough in the same model; solid wiggled line* nit=-1*; solid smooth line* nit=4*; the dash dotted line is estimated from the data with KDE.*

Example 7c. Wave period for high-crest, deep-trough waves: We finish the example with an even more interesting case, the density of wave period of waves with both significant crest and significant trough, i.e. really big waves. We first estimate the probability of such waves in the data; then we use the interpolated series yn from Example 7a.

```
[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))
Tu = yn(v_ind(1+2*ind),1)-yn(v_ind(1+2*(ind-1)),1);
t = linspace(0.01,14,200);
f_tu2_emp = kde(Tcc,{'L2',0},t);
f_tu2_emp.f = Pemp*f_tu2_emp.f;
pdfplot(f_tu2_emp,'-.')
```

The probability is estimated to be Pemp = 0.0370, which is slightly higher than what we could expect if high crests and low troughs occur independently of each other (the probability would then be less 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_tu2_n = spec2tccpdf(SS,[],'t>',[0 12 61],[Hs/2],[Hs/2],-1);
simpson(f_tu2_n.x{1},f_tu2_n.f), hold on
pdfplot(f_tu2_n), hold off
```

The execution time is less than 2 minutes, and the computed probability with `nit = -1` is 0.0358, which is well in agreement with the estimated number. The more time-demanding `nit = 4` gives almost the same result, with an execution time of 50 minutes.

In Figure 4.7(b) we see the computed densities of wave period for these big waves. Those are well concentrated around the mean value. It is also compared to the KDE estimator. We have not tried to tune up the estimator that is based on only 20 values and hardly can be considered as accurate. □

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 – the process is simply not stationary.

The data come from the Gullfaks C platform, see Figure 4.8(a). See the help text of `gfaksr89` for a detailed description of the data and `northsea` for the instructions how the map showing location of the platform was drawn.

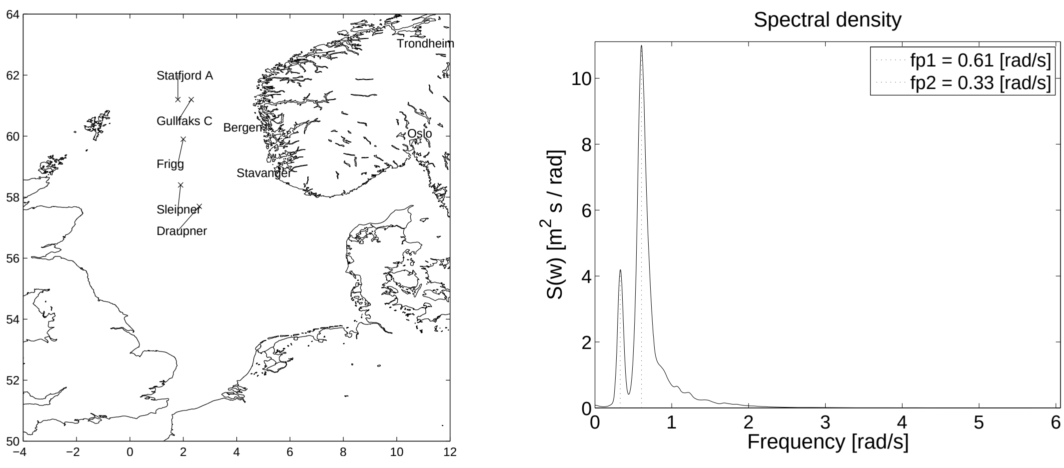


Figure 4.8: *Location of Gullfaks C platform (a). The estimated spectrum (b).*

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 = -1, 1`, (default is 2). These choices will produce fast but still useful approximations.

4.3.1 Preliminary analysis of data

Example 8. *(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 4.8(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     = ecross(yy(:,1),yy(:,2),v_ind_d,v);
T_u     = ecross(yy(:,1),yy(:,2),v_ind_u,v);
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 compute the crest period density and compare it with that observed in data.

```
t = linspace(0.01,15,200);
kopt3 = kdeoptset('hs',0.25,'L2',0);
ftc1 = kde(Tc,kopt3,t);
```

```
ftt1 = kde(Tt,kopt3,t);
pdfplot(ftt1,'k'), hold on
pdfplot(ftc1,'k-.')
f_tc4 = spec2tpdf(SS,[],'Tc',[0 12 81],0,4,5);
f_tcn = spec2tpdf(SS,[],'Tc',[0 12 81],0,-1);
pdfplot(f_tcn,'b'), hold off
```

We do not present the graphical result for this computations but simply comment that the agreement between theory and data is very good for both densities, 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. □

4.3.2 Joint distribution of crest period and height

We turn now to the joint density for the wave crest variables `Tc,Tcf,Ac`. We shall compute the empirical densities from the observations and compute the theoretical ones from the transformed Gaussian process with estimated spectrum and the transformation using 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 triple `Tc,Tcf,Ac`. (Execute the help function on `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 9. *(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.

```
ind = find(4.4<Tc & Tc<4.6);
f_AcTcf = kde([Tcf(ind) Ac(ind)],{'L2',[1 .5]});
plot(Tcf(ind), Ac(ind),'.'), hold on
pdfplot(f_AcTcf), hold off
```

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 the distribution of `Tc` can be computed using the program `spec2tpdf`.

```
opt1 = rindoptset('speed',5,'method',3);
opt2 = rindoptset('speed',5,'nit',2,'method',0);
f_tcfac1 = ...
   spec2thpdf(SS,[],'TcfAc',[4.5 4.5 46],[0:0.25:8],opt1);
```

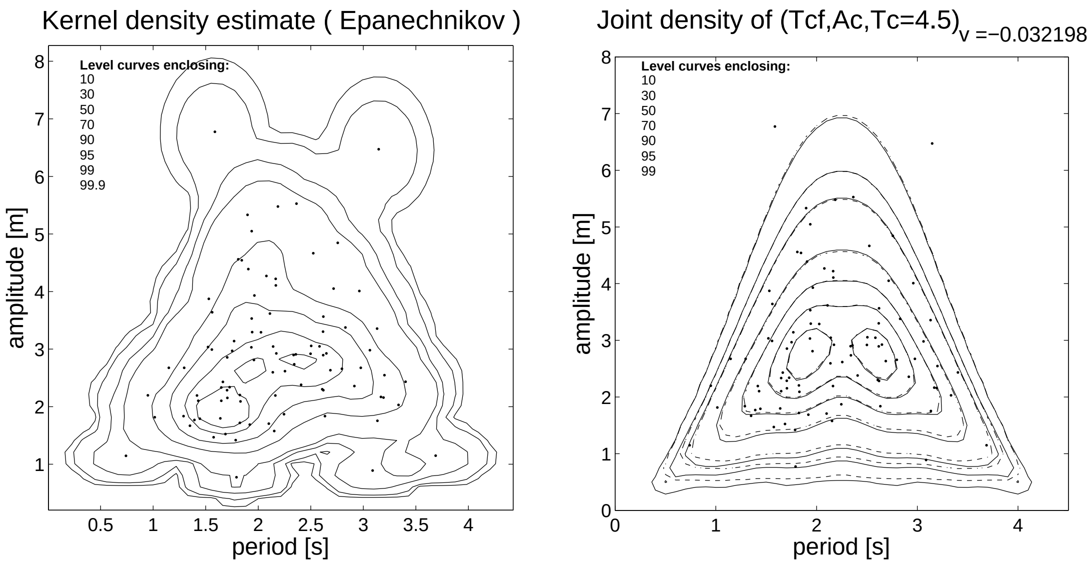
```
f_tcfac2 = ...
   spec2thpdf(SS,[],'TcfAc',[4.5 4.5 46],[0:0.25:8],opt2);

pdfplot(f_tcfac1,'-.'), hold on
pdfplot(f_tcfac2)
plot(Tcf(ind), Ac(ind),'.'), hold off

simpson(f_tcfac1.x{1},simpson(f_tcfac1.x{2},f_tcfac1.f,1))
simpson(f_tcfac2.x{1},simpson(f_tcfac2.x{2},f_tcfac2.f,1))

f_tcf6=spec2tpdf(SS,[],'Tc',[4.5 4.5 46],[0:0.25:8],6);
f_tc6.f(46)
```

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 densities `f_tcfac1` and `f_tcfac2`.



(a) (b)

Figure 4.9: *Distribution of crest position and crest height for waves with crest period* `Tc = 4.5[s]`*. (a) The estimated (KDE) density of crest position and height together with observations (dots). (b) The theoretically computed density with* `nit = -1, 2` *and the data.*

In Figure 4.9(a) the estimated (KDE) joint density is given and it should be compared with Figure 4.9(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`, and this will take some time. However, the most time is spent on the computation of the density of long and small waves, and these are not interesting. Hence we can start to compute the joint density of `Tc,Ac` for significant waves.

Example 9. (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. The following call takes substantial time (45 minutes), and gives the "exact" distribution. It is not included in the "fast" version of the command file `Chapter4.m`.

```
f_tcac_s = spec2thpdf(SS,[],'TcAc',[0 12 81],[Hs/2:0.1:2*Hs],opt1);
```

Next, we find the modified Longuet-Higgins (L-H)-density, i.e. the density with transformed crest heights. The original (L-H)-density underestimes the high crests with up to one meter. We can see that for significant waves and the present spectrum the modified 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);
ind = find(Ac>Hs/2);
plot(Tc(ind), Ac(ind),'.'); hold on
pdfplot(flh_g,'k-.');  pdfplot(f_tcac_s); hold off
```

In Figure 4.10(a) the theoretical density is plotted with solid lines and the transformed L-H density with dash dotted lines. 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.

```
f_tcac = spec2thpdf(SS,[],'TcAc',[0 12 81],[0:0.2:8],opt1);
pdfplot(f_tcac)
```

In Figure 4.10(b) the theoretical density is compared with the data, and as we see, the agreement is again quite good. This routine take about 25 minutes to run, and it is not excecuted in the default version of the command file `Chapter4.m`. □

4.3.3 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 crest and trough periods `Tc, Tt`, and then the joint density of `Tc,Tt` was derived in order to get the wave period `Tu`. 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

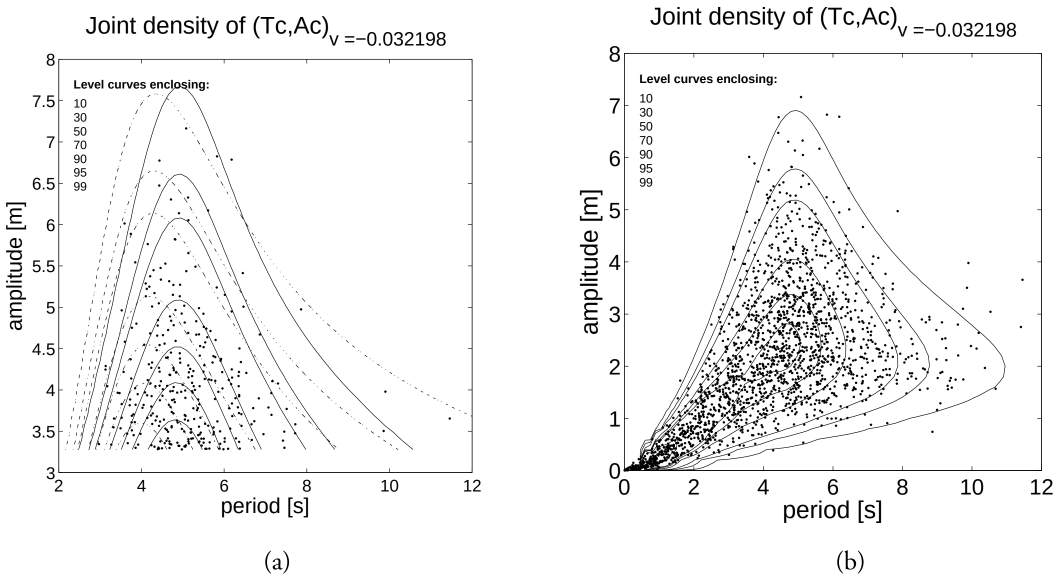


Figure 4.10: *Joint density of* `Tc` *and* `Ac` *for the transformed Gaussian model of the sea measurements from Gullfaks C platform (solid line) compared with the transformed Longuet-Higgins density (dash dotted line) and the data (dots) for waves with significant crest.*

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. Observe that, by numerical derivation, one can compute the joint density of `Tc,Ac,Tt,At` using `spec2tccpdf` (or `spec2AcAt`) but it would take many hours 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 height of the wave crest. Consequently, the steepness of a wave is mainly determined by the height 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 minima and maxima, denoted `TmM` can be used to compute steepness of a wave, see [10, 11] 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 joint density of all pairs of local minima, maxima and the distance in between; see [34] for examples.

4.3.4 Min-to-max distributions – Markov method

We shall now investigate another 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 the help text to 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 a transformed Gaussian model by a Markov chain; see [57] for detailed description of the algorithm. The approximation has been checked on many different sea data giving very accurate results, and it is also relatively fast. There is another program spec2cmat which is a function adapted from WAT. It is somewhat less accurate but even faster. It is used to compute Markov matrices and rainflow matrices used in fatigue.

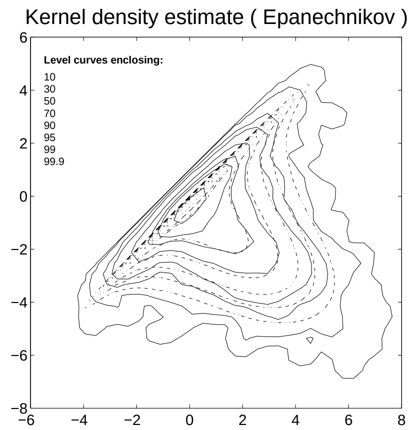


Figure 4.11: *The joint density of maximum and the following minimum for the transformed Gaussian model of the sea measurements from Gullfaks C platform (dash dotted lines) compared with the estimated (KDE) density from data (solid lines).*

Example 10. *(min-max problems with Gullfaks data)* In this example we continue the analysis of the Gullfaks C platform data. First we shall retrieve the sequence of turning points, i.e. the minima and maxima, in yy and calculate the theoretical distribution.

```
opt2 = rindoptset('speed',5,'nit',2,'method',0);
tp = dat2tp(yy);
Mm = fliplr(tp2mm(tp));
fmm = kde(Mm);
f_mM = spec2mmtpdf(SS,[],'mm',[],[-7 7 51],opt2);
pdfplot(f_mM,'-.'), hold on
pdfplot(fmm,'k-'), hold off
```

In Figure 4.11 we can see that the theoretically computed density agrees very well with the estimated one, even with an as low a nit as 2. □

Example 11. *(crest-trough distribution from min-max transitions)* We turn now to the joint density of crest and trough. We first compute the exact distribution with the help of `spec2mmtpdf`, and then compare the result with that obtained by means of the Markov approximation for the min-max sequence; see Section 5.2.3. As mentioned, 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],opt2);
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]);
f_acat = spec2mmtpdf(SS,[],'AcAt',[],[-7 7 51],opt2);
pdfplot(f_acat,'-.'), hold on
pdfplot(facat,'k-'), hold off
```

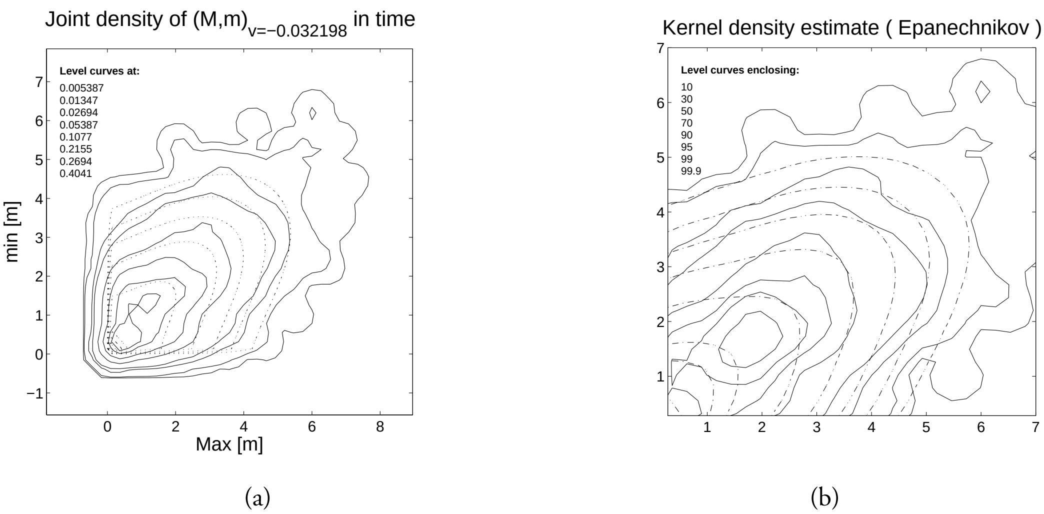


Figure 4.12: *Estimated joint density (KDE) of "still water separated" min-to-max values for the measurements from Gullfaks C (solid line) compared with: (a) the transformed Gaussian model for the measurements (dash-dotted line). (b) Markov approximation for the joint density of crest and trough height* `Ac,At` *(dashdotted line).*

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 replacement of the true min-to-max transition probabilities by the transition probabilities that are valid for the "still water separated" min-to-max values. The results are presented in Figure 4.12. 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. □

4.4 WAFO wave characteristics routines

```
help trgauss

 Module TRGAUSS in WAFO Toolbox.
 Version 2.5.2   07-Feb-2011

 Readme          - New features, bug fixes, and changes in TRGAUSS.

 Misc
   createpdf     - PDF struct constructor.
   pdfplot       - Plot contents of pdf structures.
   trplot        - Plots transformation, g, eg. estimated with dat2tr.

 Transforms and non-linearities
   dat2gaus      - Transforms  x  using the transformation  g.
   gaus2dat      - Transforms  xx  using the inverse of  g.
   testgaussian  - Test if a stochastic process is Gaussian.
   spec2skew     - Estimates the moments of 2'nd order non-linear waves.
   trangood      - Makes a transformation that is suitable for
                   efficient transforms.
   tranproc      - Transforms process X and up to four derivatives.
   trmak         - Put together a transformation object.
   troptset      - Create or alter TRANSFORM OPTIONS structure.
   trunmak       - Split a transformation object into its pieces.

 Transformed Gaussian model estimation
   cdf2tr        - Estimate transformation, g, from observed CDF.
   dat2tr        - Estimate transformation, g, from data.
   hermitetr     - Estimate transformation, g, from the first 4 moments.
   ochitr        - Estimate transformation, g, from the first 3 moments.
   lc2tr         - Estimate transformation, g, from observed
                   crossing intensity.
   lc2tr2        - Estimate transformation, g, from observed
                   crossing intensity, version 2.

 Gaussian probabilities and expectations
   cdfnorm2d     - Bivariate normal cumulative distribution function.
   prbnorm2d     - Bivariate normal probability.
   prbnormnd     - Multivariate normal probability by Genz' algorithm.
   prbnormndpc   - Multivariate normal probabilities with
                   product correlation.
   prbnormtnd    - Multivariate normal or T probability by
                   Genz' algorithm.
   prbnormtndpc  - Multivariate normal or T probability with
                   product correlation structure.
```

```
  rind          - Computes multivariate normal expectations.
  rindoptset    - Create or alter RIND OPTIONS structure.

Probability density functions (pdf) or intensity matrices
  chitwo2lc_sorm - SORM-approximation of crossing intensity,
                   noncentral Chi^2 process.
  chitwo2lc_sp  - Saddlepoint approximation of crossing
                  intensity, noncentral Chi^2 process.
  dirsp2chitwo  - Parameters in non-central CHI-TWO process for
                  directional Stokes waves.
  iter          - Calculates a Markov matrix given a rainflow matrix.
  iter_mc       - Calculates a kernel of a MC given a rainflow matrix.
  mc2rfc        - Calculates a rainflow matrix given a
                  Markov chain with kernel f_xy.
  mctp2rfc      - Rainflow matrix given a Markov matrix of a
                  Markov chain of turning points.
  mctp2tc       - Calculates frequencies for the upcrossing
                  troughs and crests.
  nt2fr         - Calculates the frequency matrix given the
                  counting distribution matrix.
  spec2cmat     - Joint intensity matrix for cycles (max,min)-,
                  rainflow- and (crest,trough).
  spec2mmtpdf   - Joint density of Maximum, minimum and period.
  spec2tccpdf   - Density of crest-to-crest
                  wave-period or -length.
  spec2thpdf    - Joint density of amplitude and
                  period/wave-length characteristics.
  spec2tpdf     - Density of crest/trough- period or length.
  spec2tpdf2    - Density of crest/trough- period or length,
                  version 2.
  specq2lc      - Saddlepoint approximation of crossing
                  intensity for quadratic sea.
  th2vhpdf      - Transform joint T-H density to V-H density.

Cumulative distribution functions (cdf)
  cdflomax      - CDF for local maxima for a zero-mean
                  Gaussian process.
  spec2AcAt     - Survival function for crests and troughs,
                  R(h1,h2)=P(Ac>h1,At>h2).
  spec2Acdf     - CDF for crests P(Ac<=h) or troughs P(At<=h).
```

CHAPTER 5

Fatigue load analysis and rain-flow cycles

This chapter contains some elementary facts about random fatigue and how to compute expected fatigue damage from a stochastic, stationary load process. The commands can be found in `Chapter5.m`, taking about 7 seconds to run on a 2.93 GHz 64 bit PC.

5.1 Random fatigue

5.1.1 Random load models

This chapter presents some tools from WAFO for analysis of random loads in order to assess random fatigue damage. A complete list of fatigue routines can be obtained from the help function on `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

$$x(t) \approx m + \sum_{i=1}^{[T/2]} a_i \cos(\omega_i\, t) + b_i \sin(\omega_i\, t)$$

   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. The properties are summarized in a spectral density as in described in Section 2.2.

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 $\mu(u)$, introduced in Section 2.1, 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 (maxima and minima), 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 build rainflow cycles with amplitude smaller than a given threshold. We shall always assume that the signals are rainflow filtered; see Section 5.2.1.

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, [54] and [26].

In the simplest case, the necessary information is the intensity of pairs of local maxima and the following minima, summarized in the so-called Markov matrix or min-max matrix. The dependence between other extremes is modeled using Markov chains, see [61] and [17].

5.1.2 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 $N(s)$ until failure, 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 the S-N curve, also called the Wöhler curve, i.e. the relation between the amplitude s and $N(s)$,

$$N(s) = \begin{cases} K^{-1}s^{-\beta}, & s > s_\infty, \\ \infty, & s \le s_\infty, \end{cases} \tag{5.1}$$

where K and β are material dependent parameters. Often K is considered as a random variable, usually lognormally distributed, i.e. with $K^{-1} = E\varepsilon^{-1}$ 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.

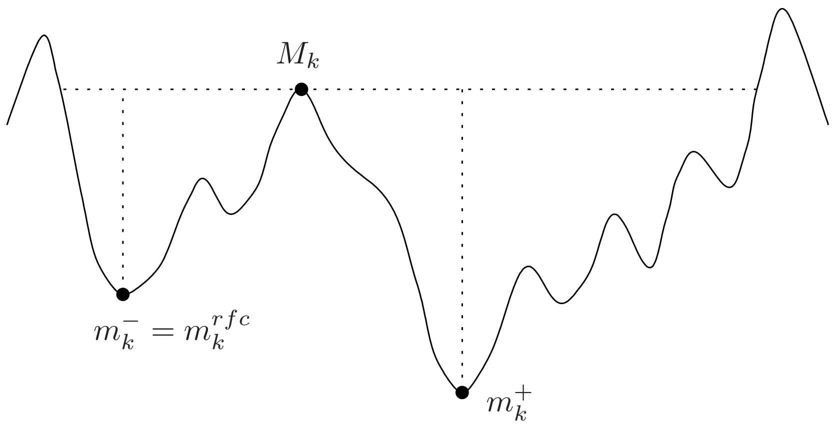


Figure 5.1: *Definition of the rainflow cycle as given by [52].*

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

$$D(t) = \sum_{t_k \le t} \frac{1}{N(s_k)} = K \sum_{t_k \le t} s_k^\beta = K D_\beta(t), \tag{5.2}$$

where the sum contains all cycles that 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 $K = E^{-1}\varepsilon$ in Eq. (5.2) by a constant, for example the median value of K, which is equal to ε, under the lognormal assumption. 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 unit time. This leads to a very simple predictor of fatigue life time,

$$\widehat{T}^f = \frac{1}{\varepsilon d_\beta}. \tag{5.3}$$

5.1.3 Rainflow cycles and hysteresis loops

The now commonly used cycle counting method is the rainflow counting, which was introduced 1968 by Matsuishi and Endo in [42]. 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 1987 by Rychlik, [52]. The formal definition is also illustrated in Figure 5.1.

Definition 5.1 (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. The minimum that 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 $(m_k^{\mathrm{RFC}}, M_k)$.*

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

$$D(t) = \sum_{t_k \le t} \frac{1}{N(s_k^{\mathrm{RFC}})} = K \sum_{t_k \le t} (s_k^{\mathrm{RFC}})^\beta = K D_\beta(t), \tag{5.4}$$

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

To use Eq. (5.3) 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, after rainflow filtering, and equally many rainflow cycles, and each rainflow cycle causes an expected damage $\varepsilon E(1/N_{S^{\mathrm{RFC}}})$ 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 rainflow amplitudes raised to the material dependent power parameter β. WAFO contains a number of routines for handling the rainflow cycles in observed load data and in theoretical load models.

5.2 Load cycle characteristics

5.2.1 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 wave forces or to compute wave characteristics for specified random wave models, e.g. those defined by their power spectrum.

Measured wave or load 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`, as described in Section 2.2.4, 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 or cycles. 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 4 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 cycle concept is a nonlinear means to remove small irregularities from a load series. Another nonlinear method to remove small cycles from data is the rainflow filtering, introduced in [56], 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 that 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

[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 average number of rainflow cycles per time unit.

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 [56] 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 5.7 for an example.

5.2.2 Oscillation count and the 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 *oscillation count*, $N^{\mathrm{OSC}}(u, v)$ that counts the number of times a signal upcrosses an interval $[u, v]$. The oscillation count is thus a function of 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, for discrete set of levels, $u_1, \leq u_2 \leq \ldots \leq u_n$, 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. 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 similarly, the expected oscillation count, also called an *oscillation intensity matrix*, is an important characteristic of the random processes used as a model for the data. 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 way to handle random processes 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.

5.2.3 Markov chain of turning points, Markov matrix

An upcrossing of an interval $[u, v]$ occurs if the process, after an upcrossing of the level u, passes the higher level v before it returns below u. Therefore, the oscillation intensity is closely related to a special first passage problem, and 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

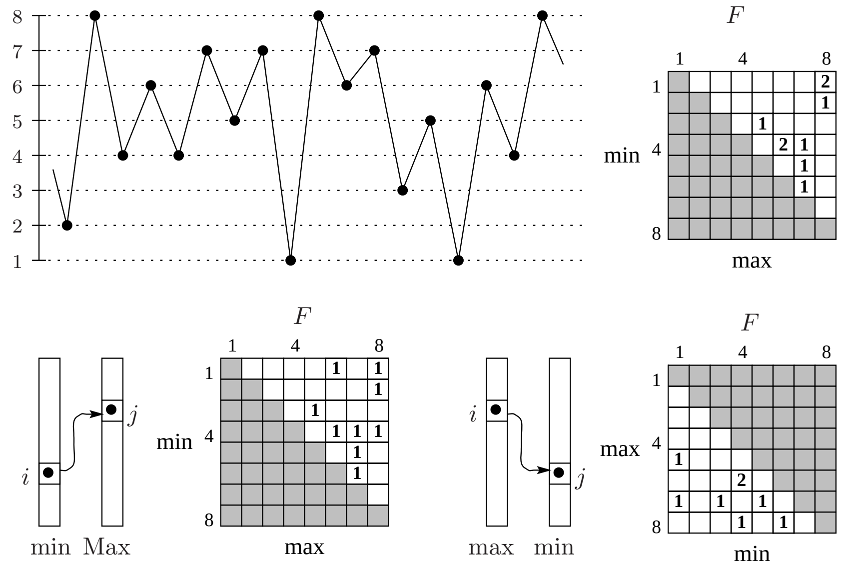


Figure 5.2: *Part of a discrete load process where the turning points are marked with •. The scale to the left is the discrete levels. The transitions from minimum to maximum and the transitions from maximum to minimum are collected in the min-max matrix,* $\mathbf{F}$ *and max-min matrix,* $\widehat{\mathbf{F}}$*, respectively. The rainflow cycles are collected in the rainflow matrix,* $\mathbf{F}^{\mathrm{RFC}}$*. The numbers in the squares are the number of observed cycles and the grey areas are by definition always zero.*

of a local extremum, given all previous extrema, depends only on the value and type (minimum or maximum) of the most recent 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 a transformed Gaussian process, a Markov matrix for min-to-max cycles was computed in Section 4.3.4 by means of the WAFO function `spec2mmtpdf`. In WAFO there is also an older version of that program, called `spec2cmat`, which we shall use in this chapter. The max-to-min matrix is obtained by symmetry.

Next, the function `mctp2tc` (= Markov Chain of Turning Points to Trough Crests), computes the trough2crest intensity, using a Markov matrix to approximate the sequence of turning points by a Markov chain. This approximation method is called the *Markov method*. Be aware that the Markov matrix is not the transition matrix of the Markov chain of turning points, but the intensity of different pairs of turning points.

Figure 5.2 shows the general principle of a Markov transition count between turning points of local maxima and minima. The values have been discretized to levels labeled `1, ..., n`, from smallest to largest.

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`, and these are connected by the following nonlinear equation

$$\texttt{Frfc} = \texttt{FmM} + \mathcal{F}(\texttt{FmM}), \tag{5.5}$$

where $\mathcal{F}$ is a matrix valued function, defined in [56], where also an algorithm to compute the inverse $(\mathcal{I} + \mathcal{F})^{-1}$ is given. The WAFO functions for computing `Frfc` from `FmM` are `mctp2rfm` and `mctp2rfc`, while the inverse, i.e. `FmM` as a function of `Frfc`, is computed by `arfm2mctp`. It might be a good idea to check the modules `cycles` and `trgauss` in WAFO for different routines for handling these matrices.

5.3 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 the corresponding 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 a special Markov structure.

5.3.1 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.

```
xx_sea = load('sea.dat');
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 routines `dat2tp` and `tp2lc` take a load sequence and extracts the turning points, and from this calculates the number of up-crossings as a function of level. The plots produced, Figure 5.3, show the crossing intensity plotted in two common modes, lin-lin of $(u, \mu(u))$ and log-lin of $(\log \mu(u), u)$.

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 factor, 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`, (make help `interp1`), to find the crossing intensity of the mean level.

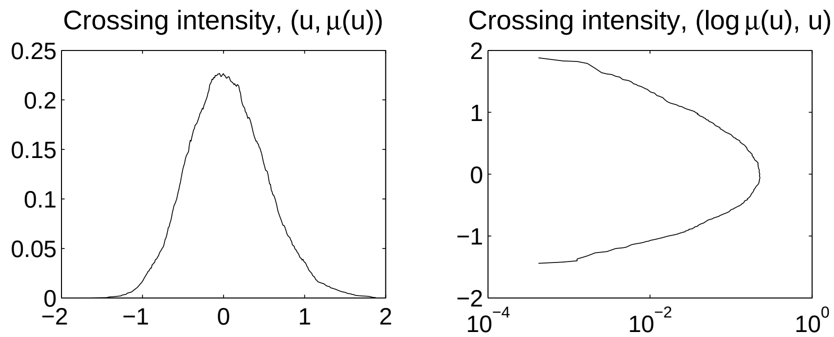


Figure 5.3: *Level crossing intensity for sea data*

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

5.3.2 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 $\mathbb{R}^2$-plane. This is done by the routine `ccplot`. Compare the min-max and rainflow counts in the load in Figure 5.4 obtained by the following commands.

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

Observe that `RFC` contains more cycles with high amplitudes, compared to `mM`. This becomes more evident in an amplitude histogram as seen in Figure 5.5.

```
ampmM_sea = cc2amp(mM_sea);
ampRFC_sea = cc2amp(RFC_sea);
subplot(221), hist(ampmM_sea,25);
```

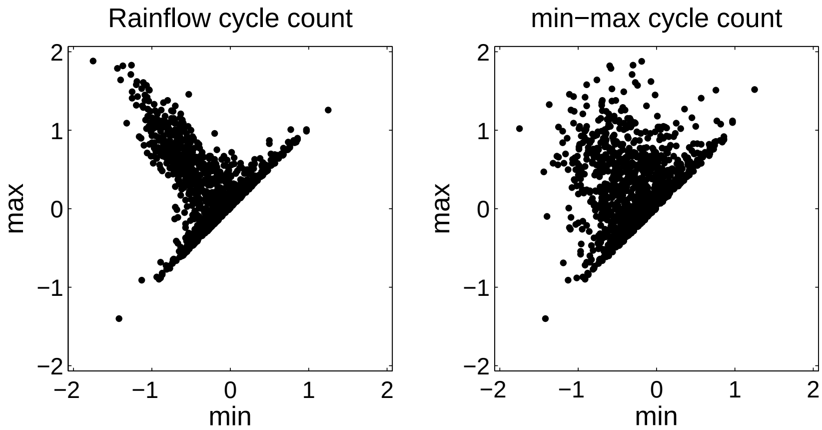


Figure 5.4: *min-max and rainflow cycle plots for sea data.*

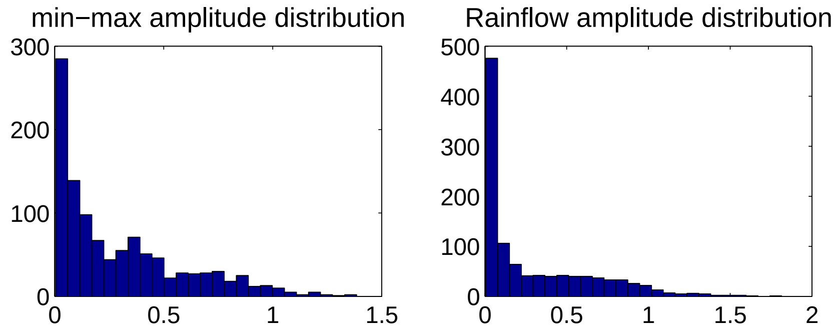


Figure 5.5: *min-max and rainflow cycle distributions for sea data.*

```
title('min-max amplitude distribution')
subplot(222), hist(ampRFC_sea,25);
title('Rainflow amplitude distribution')
```

5.3.3 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 completely 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; see the help function for mktestmat for the generation of G.

```
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, as in the third command line, we get the load xx_markov, which takes values between -1 and 1. The first 50 samples of the simulation is plotted in Figure 5.6 by

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

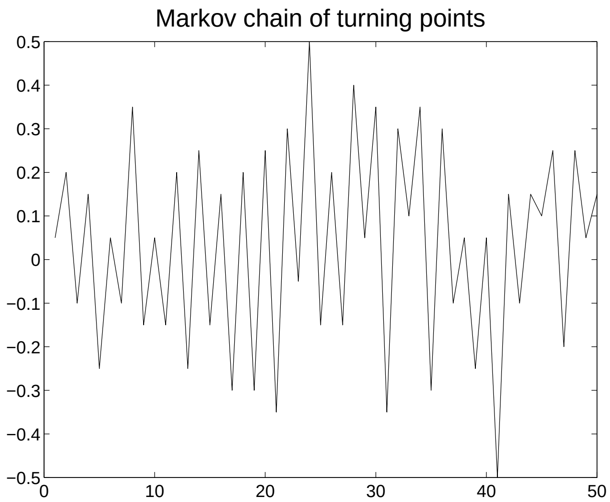


Figure 5.6: *Simulated Markov sequence of turning points.*

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 that 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 transformed 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 spectrum structure including a transformation, the input spectrum must be normalized to have standard deviation 1, i.e. one must divide the spectral values by the variance sa^2.

```
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/sa^2;
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);
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 5.7 the dashed curve connects the remaining turning points after filtration.

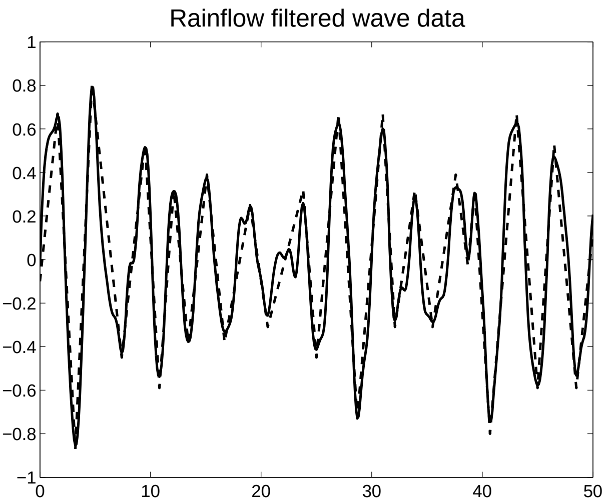


Figure 5.7: *Hermite transformed wave data and rainflow filtered turning points,* `h = 0.2`.

Try different degree of filtering on the Ochi transformed sequence and see how it affects the min-max cycle distribution. You can use the following sequence of commands, with different `h` -values; see Figure 5.8 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);
subplot(121), ccplot(RFC_herm)
title('h=0')
subplot(122), ccplot(RFC_herm_1)
title('h=1')
```

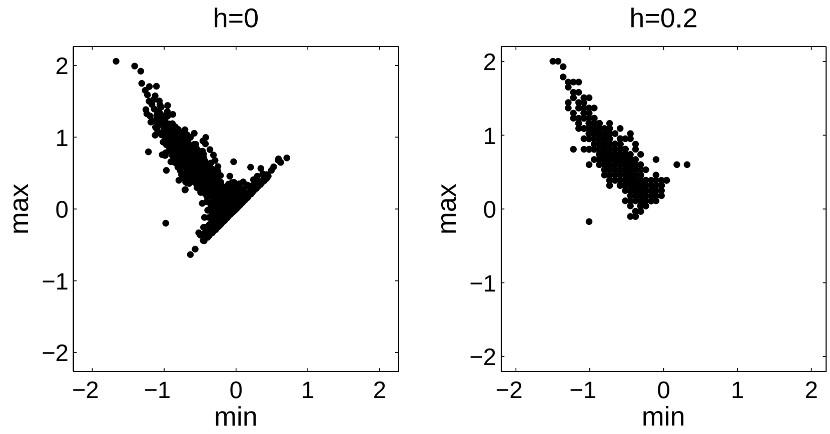


Figure 5.8: *Rainflow cycles and rainflow filtered rainflow cycles in the transformed Gaussian process.*

5.3.4 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 with some spectral density. 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 by the mehods described in Section 4.3.4.

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 5.3.3 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.

```
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 the help on `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 5.9. 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.

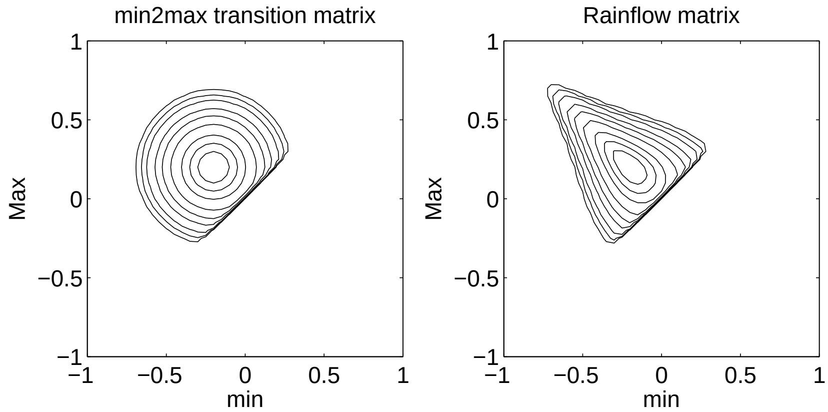


Figure 5.9: *min-max-matrix and theoretical rainflow matrix for test Markov sequence.*

```
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')
```

We now compare the theoretical rainflow matrix with an observed rainflow matrix obtained in the simulation. In this case we have simulated a 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 5.10 between the observed rainflow matrix and the theoretical one is produced as follows.

```
n = length(u_markov);
Frfc_markov = dtp2rfm(xxD_markov,n);
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`.

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`. 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);
cmatplot(u_markov,u_markov,...
         {Frfc_markov_smooth Grfc_markov*T/2},4)
```

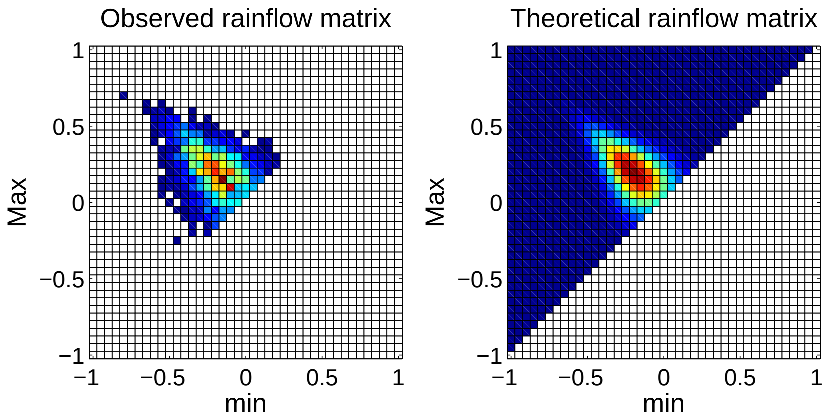


Figure 5.10: *Observed and theoretical rainflow matrix for test Markov sequence.*

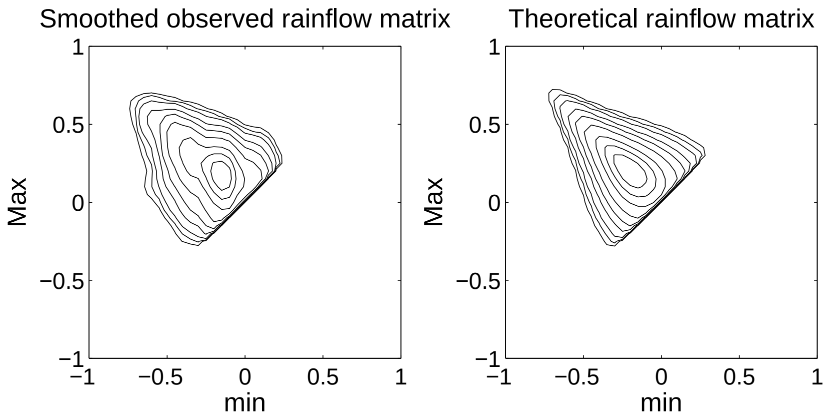


Figure 5.11: *Smoothed observed and calculated rainflow matrix for test Markov sequence.*

```
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 5.11.

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 transformed Gaussian model with the Hermite transform `xx_herm` which was studied in Section 5.3.3. 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 4.3.4. We start by computing an approximation, `GmM3_herm`, of the min-max density by means of the cycle routine `spec2cmat` (as an alternative one can use `spec2mmtpdf`). The type of cycle is specified by a cycle parameter, in this case `'Mm'`.

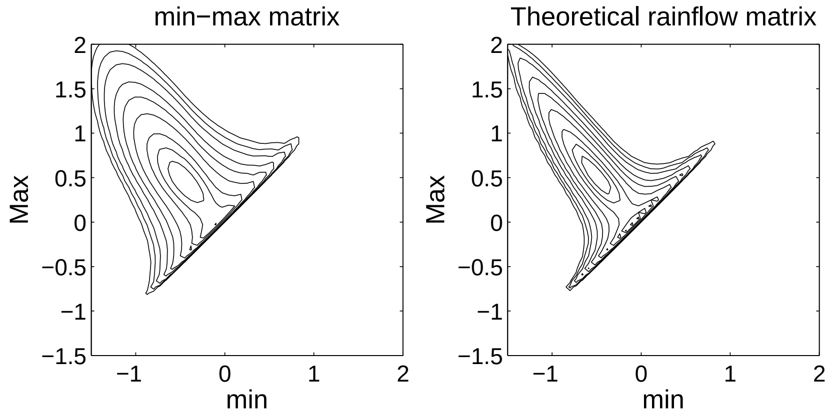


Figure 5.12: *min-max matrix and theoretical rainflow matrix for Hermite-transformed Gaussian waves.*

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

The result is seen in Figure 5.12.

Then, we approximate the distribution of the turning points by a Markov chain with transitions between extrema calculated from `GmM3_herm`, and compute the rainflow matrix by Eq. (5.5).

```
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 5.12, obtained using the following commands.

```
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')
```

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 a 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);
```

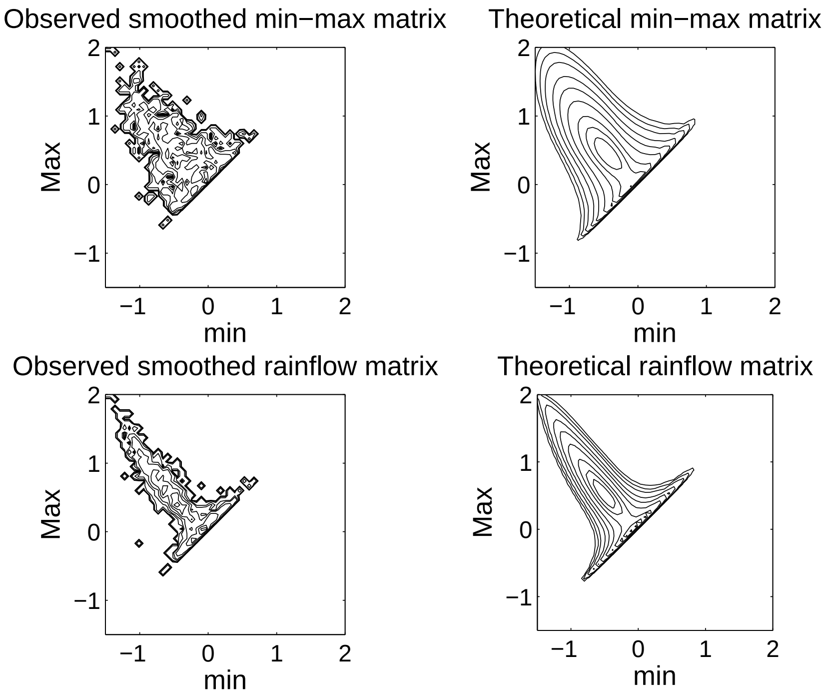


Figure 5.13: *Observed smoothed and theoretical min-max matrix, and observed smoothed and theoretical rainflow matrix for Hermite-transformed Gaussian waves.*

```
T_herm=xx_herm(end,1)-xx_herm(1,1);
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')
```

5.3.5 Simulation from crossings structure

In fatigue experiments it is important to generate load sequences with a prescribed rainflow or other crossing property. Besides the previously used simulation routines for Markov loads and spectrum loads, WAFO contains algorithms for generation of random load sequences that have a specified average rainflow distribution or a specified irregularity and crossing spectrum. We illustrate the crossing structure simulation by means of the routine `lc2sdat`. Simulation from a rainflow distribution can be achieved by first calculating the corresponding Markov matrix and then simulate by means of `mctpsim`.

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 5.14. The figures were generated by the following commands.

```
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')
```

5.4 Fatigue damage and fatigue life distribution

5.4.1 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 Eq. (5.1) and the Palmgren-Miner damage accumulation rule Eq. (5.2),

$$N(s) = \begin{cases} K^{-1}s^{-\beta}, & s > s_\infty, \\ \infty, & s \le s_\infty, \end{cases} \tag{5.6}$$

$$D(t) = \sum_{t_k \le t} \frac{1}{N(s_k)} = K \sum_{t_k \le t} s_k^\beta = K D_\beta(t). \tag{5.7}$$

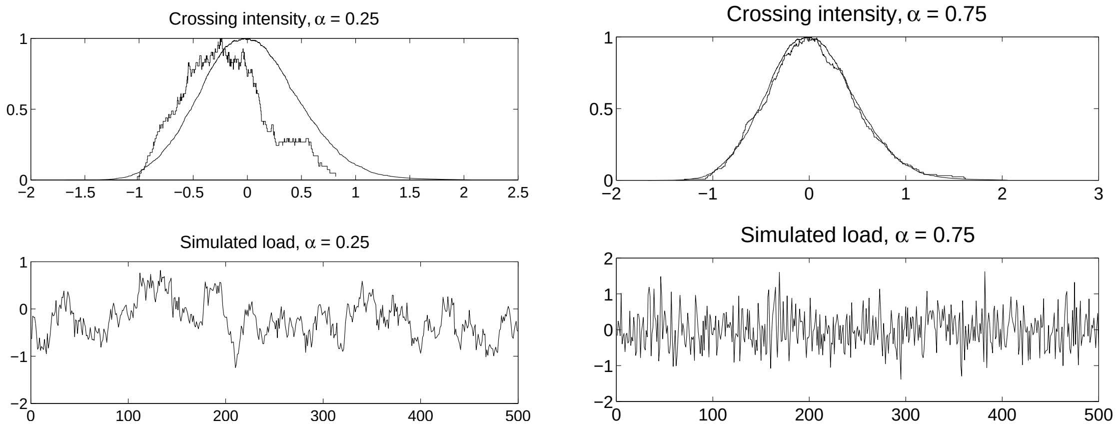


Figure 5.14: *Upper figures show target crossing spectrum (smooth curve) and obtained spectrum (wiggled curve) for simulated process shown in lower figures. Irregularity factor: left $\alpha = 0.25$, right $\alpha = 0.75$.*

Here $N(s)$ is the expected fatigue life from constant amplitude test with amplitude s, and $D(t)$ is the total damage at time t caused by variable amplitude cycles s_k, 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 (5.7) 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 be directly used for prediction of expected fatigue life. The expression Eq. (5.3) gives the expected time to fatigue failure in terms of the material constant ε and the expected damage d_β per time unit. The parameters ε and β can be estimated from an S-N curve. In the examples here we will use $\varepsilon = 5.5 \cdot 10^{-10}$, $\beta = 3.2$; see Section 5.4.4. 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 – of course, the sea wave data is not a fatigue load sequence, so the example is meaningless from a fatigue point of view.

5.4.2 Level Crossings

We have in Section 5.3.5 seen how the crossing 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 5.3.3 for the sequence of turning points as an example. First we go from the rainflow matrix to the crossing intensity.

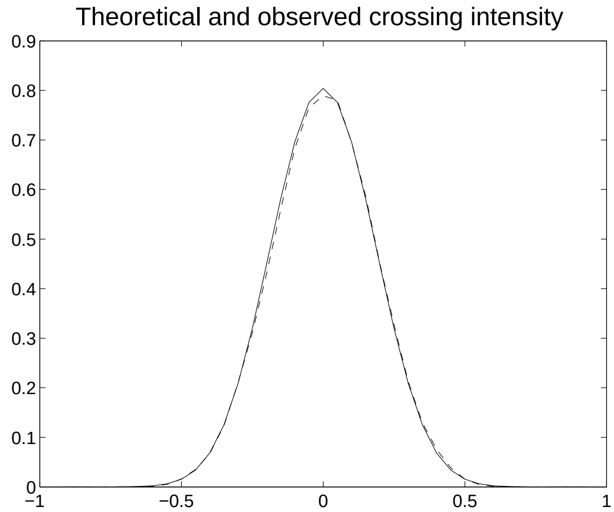


Figure 5.15: *Crossing intensity as calculated from the Markov observed rainflow matrix (solid curve) and from the observed rainflow matrix (dashed curve).*

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

The plot in Figure 5.15 compares the theoretical upcrossing intensity `mu_markov` with the observed upcrossing intensity `muObs_markov`, as calculated from the theoretical and observed rainflow matrices.

5.4.3 Damage

The WAFO toolbox contains a number of routines to compute and bound the damage, as defined by (5.7), 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

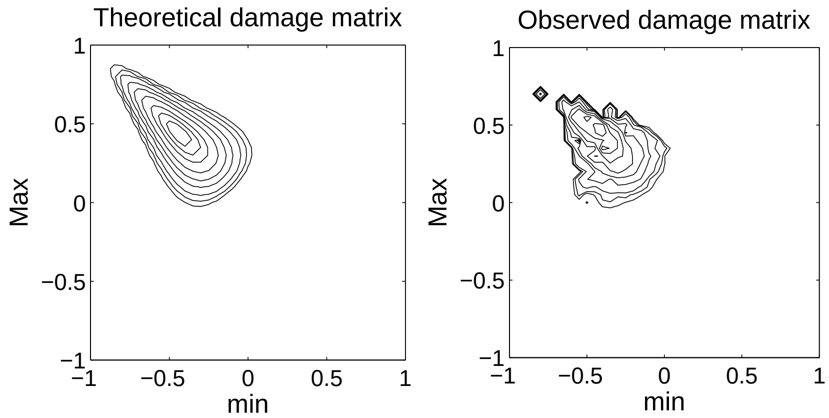


Figure 5.16: *Distribution of damage from different RFC cycles, from calculated theoretical and from observed rainflow matrix.*

count and from the rainflow matrix, respectively. For this model the result should be `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 5.16. 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);}
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 crossing intensity only, without using the rainflow cycles. This is done by the routine `lc2dplus`, which works on any theoretical or observed crossing intensity function.

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

5.4.4 Estimation of S-N curve

WAFO contains routines for computation of parameters in the basic S-N curve (5.1), for the relation between the load cycle amplitude s and the fatigue life $N(s)$ in fixed amplitude tests, defined by (5.6). The variation of the material dependent variable K is often taken to be random with a lognormal distribution,

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

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 parameters, ε, β, σ_E, to be estimated from an S-N

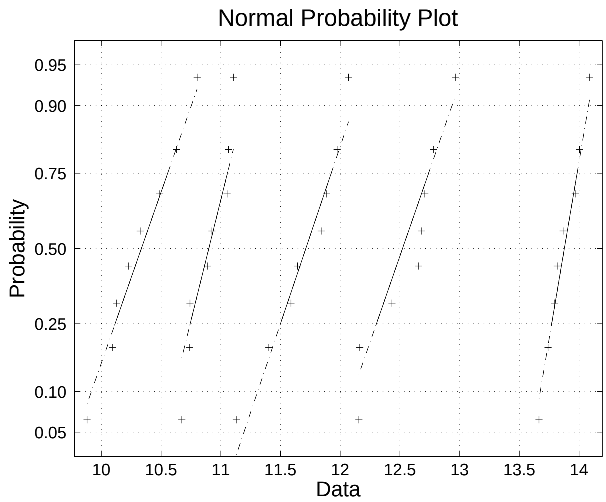


Figure 5.17: *Check of S-N-model on normal probability paper.*

experiment. Taking logarithms in (5.1) the problem turns into a standard regression problem,

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

in which the parameters can easily be estimated.

The WAFO toolbox contains a data set `SN` with fatigue lives from 40 experiments with s = 10, 15, 20, 25, and 30 MPa, stored in a variable `N`, 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
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 5.17 the assumptions seem acceptable since the data fall on almost parallel straight lines.

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

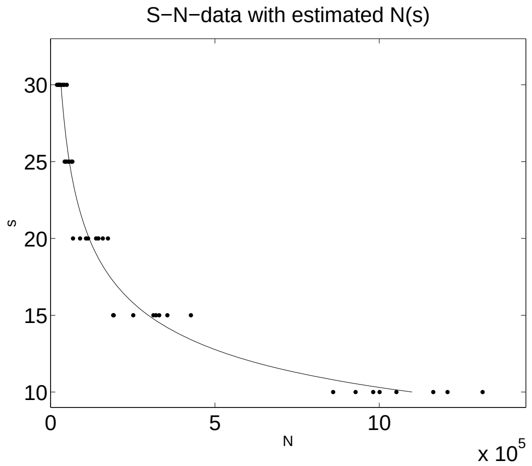
The estimation is performed and fitted lines plotted in Figure 5.18, with linear and log-log plotting scales:

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

gives linear scale and

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

gives log-log scales.



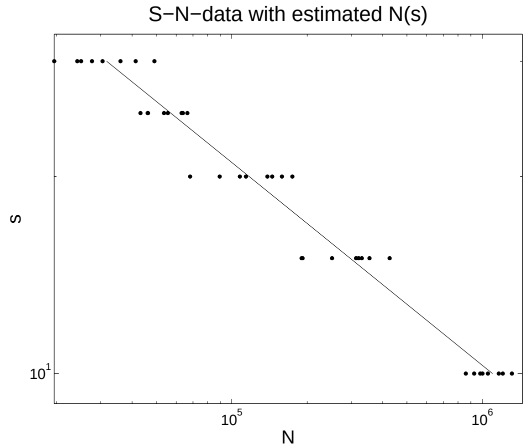


Figure 5.18: *Estimation of S-N-model on linear and log-log scale.*

5.4.5 From S-N-curve to fatigue life distribution

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

$$P(T_f \le t) = P(D(t) \ge 1) = P(K \le \varepsilon D_\beta(t)).$$

Here $K = E^{-1}\varepsilon$ 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 $\ln K = \ln\varepsilon - \ln E$ is normal with mean $\ln\varepsilon$ 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 $\varphi(x)$ and $\Phi(x)$, respectively, an approximate explicit expression for the failure probability within time t is

$$P(T^f \le t) \approx \int_{-\infty}^{\infty} \Phi\left(\frac{\ln\varepsilon + \ln d_\beta t + \ln(1+\frac{\sigma_\beta}{d_\beta\sqrt{t}}z)}{\sigma_E}\right)\varphi(z)\,dz. \tag{5.8}$$

We have already estimated the material dependent 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')
```

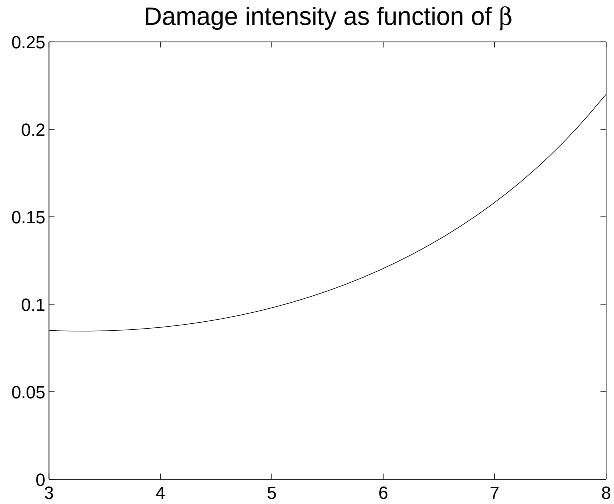


Figure 5.19: *Increasing damage intensity from sea-load with increasing β.*

The plot in Figure 5.19 shows the increase in damage with increasing β.

Next, we shall see how the load 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 5.4.4, 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)
```

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

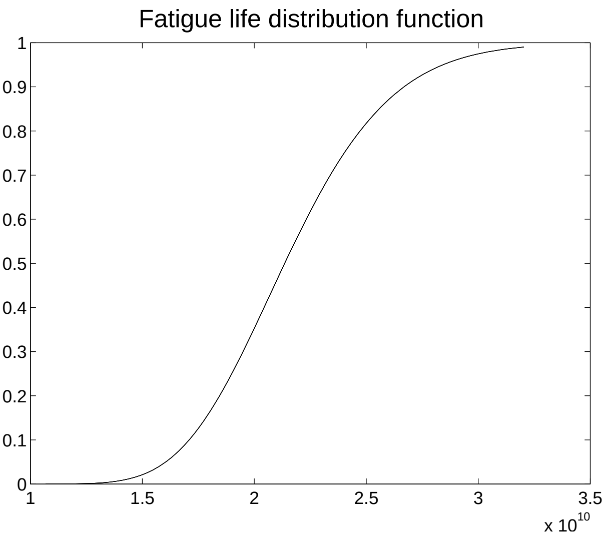


Figure 5.20: *Fatigue life distribution with sea load.*

The resulting fatigue life distribution function is shown in Figure 5.20. 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.

5.4.6 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 of 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 WAFO toolbox can be used to analyze also loads of complex structure and we shall illustrate some of these capabilities in this section. To be eligible for WAFO-analysis, the loads have 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 [25, 26].

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 5.21 the observed load is shown in the upper part. The alternating curve in the lower part shows the switches between the two states.

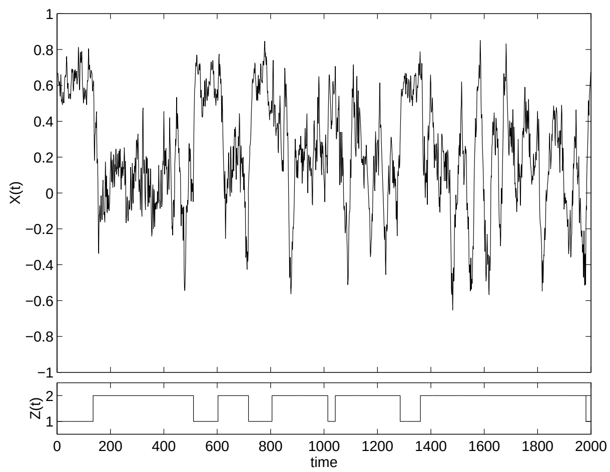
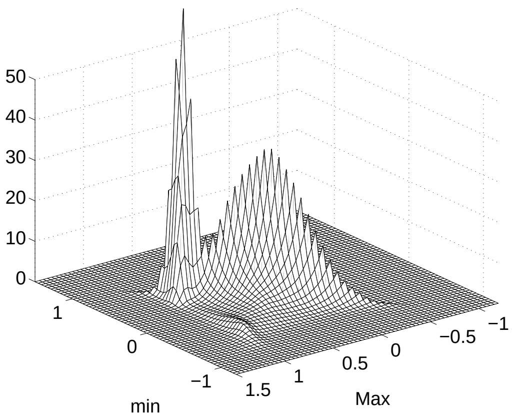


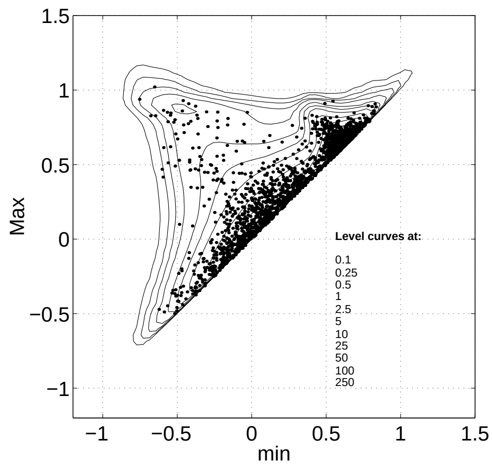
Figure 5.21: *Simulated switching load with two states. Upper graph shows the load, and the states are indicated in the lower graph.*

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 cycles with larger amplitudes. These extra cycles can be seen in the total rainflow matrix, shown in Figure 5.22. 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 [24].



(a) Rainflow matrix



(b) Rainflow matrix

Figure 5.22: *3D-plot (left) and isolines (right) of calculated rainflow matrix for switching load in Figure 5.21. The dots in the right figure are the observed rainflow cycles.*

CHAPTER 6

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. Both these distributions are part of a general family of extreme value distributions, known as the *Generalized Extreme Value Distribution*, (GEV). The *Generalized Pareto Distribution* (GPD) is another distribution family, particularly adapted for *Peaks Over Threshold* (POT), analysis. WAFO contains routines for fitting of such distributions, both for the Weibull and Gumbel distributions, and for the two more general classes of distributions. For a general introduction to statistical extreme value analysis, the reader is referred to [14].

This chapter illustrates how WAFO can be used for elementary extreme value analysis in the direct GEV method and in the POT method. The example commands in `Chapter6.m` take less than 35 seconds to run on a 2.93 GHz 64 bit PC. We start with a simple application of the classical Weibull and Gumbel analysis before we turn to the general techniques.

6.1 Weibull and Gumbel papers

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

$$\text{Weibull:} \qquad F_W(x;a,c) = 1 - e^{-(x/a)^c}, \quad x > 0, \tag{6.1}$$

$$\text{Gumbel:} \qquad F_G(x;a,b) = \exp\left(-e^{-(x-b)/a}\right), \quad -\infty < x < \infty. \tag{6.2}$$

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. In practice it is 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 6.2.

6.1.1 Estimation and plotting

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 Eqs. (6.1) and (6.2). The file `atlantic.dat` is included in WAFO, and it contains significant wave-height data recorded approximately 14 times a month in the Atlantic Ocean in December to February during seven years and at two locations. The data are stored in the vector `Hs`. We try to fit a Weibull distribution to this data set, by the WAFO-routine `plotweib`, which performs both the estimation and the plotting.

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

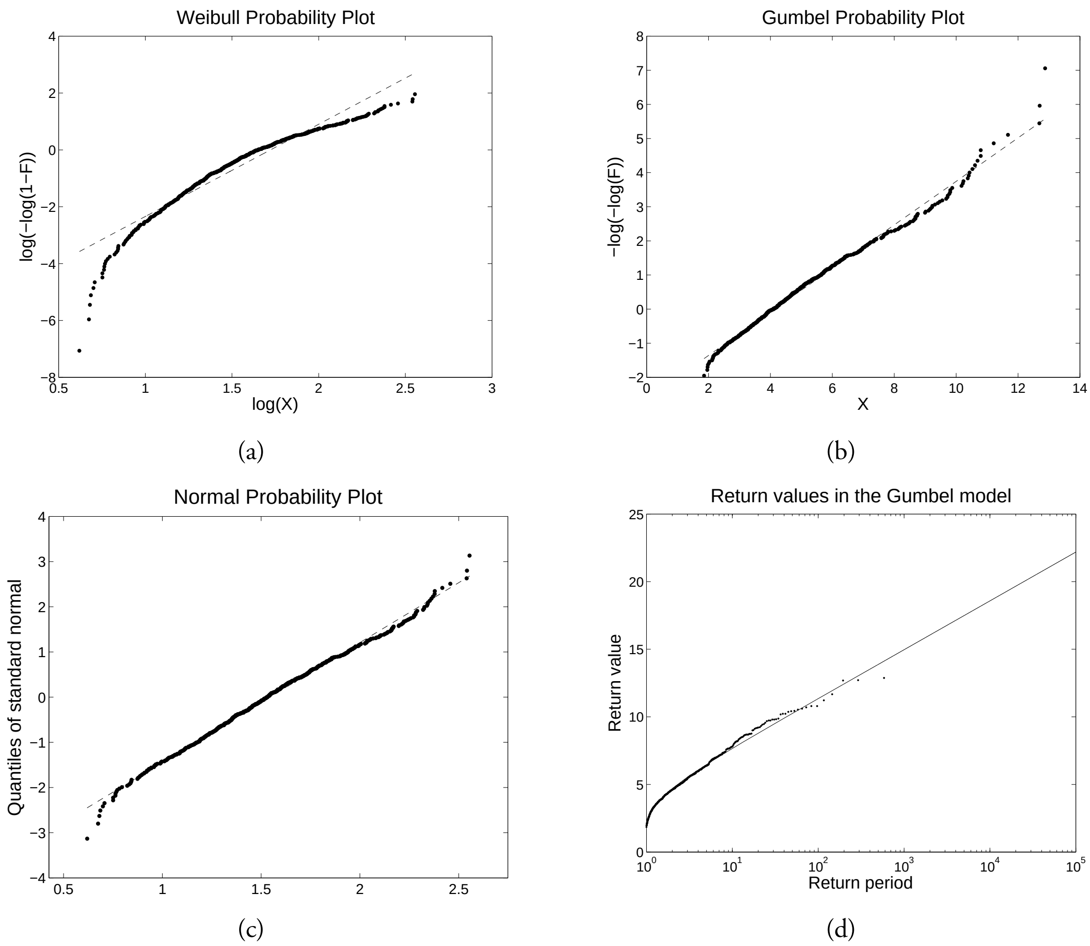


Figure 6.1: *Significant wave-height data: (a) on Weibull paper, (b) on Gumbel paper, (c) logarithm of data on Normal probability paper, and (d) return values calculated in the Gumbel model with observed data.*

This will result in a two element vector `wei = [ahat chat]` with estimated values of the parameters (a, c) in (6.1). 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 6.1(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 (6.2) for the data in `Hs`. The command

```
gum = plotgumb(Hs)
```

results in a vector `gum` with estimated values `[ahat bhat]` and the plot in Figure 6.1(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 much better fit to the Gumbel than to a Weibull distribution.

A distribution that 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 6.1(c) also shows a good fit.

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

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

$$\log\{-\log[1-F_W(x;a,c)]\} = c\log(x) - c\log(a), \tag{6.3}$$

and

$$-\log\{-\log[F_G(x;a,b)]\} = x/a - b/a, \tag{6.4}$$

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.

6.1.2 Return value and return period

The results of an extreme value analysis is often expressed in terms of *return values* or *return levels*, which are simply related to the quantiles of the distribution. A return value is always coupled to a *return period*, expressed in terms of the length of an observation period, or the number of (independent) observations of a random variable.

Suppose we are interested in the return levels for the largest significant wave height that is observed during one year at a measuring site. Denote by $M_{H_s}^k$ the maximum during year number k and let its distribution function be $F(x)$. Then the N-year return level, s_N, is defined by

$$F(s_N) = 1 - 1/N. \tag{6.5}$$

For example, $P(H_s > s_{100}) = 1 - F(s_{100}) = 1/100$, which means that,

- the probability that the level s_{100} is exceeded during one particular year is 0.01,
- on the average, the yearly maximum significant wave height exceeds s_{100} one year in 100 years, (note that there may several exceedances during that particular year),
- the probability that s_{100} is exceeded *at least* one time during a time span of 100 years is $1 - (1-0.01)^{100} \approx 1 - 1/e = 0.6321$, provided years are independent.

To make it simple, we consider the Gumbel distribution, and get, from (6.5), the T-year return value for the yearly maximum in the Gumbel distribution (6.2):

$$s_T = b - a\log(-\log(1 - 1/T)) \approx b + a\log T, \tag{6.6}$$

where the last approximation is valid for large T-values.

As an example we show a return value plot for the Atlantic data, as if they represented a sequence of yearly maxima. Figure 6.1(d) gives the return values as a function of the return period for the Atlantic data. The WAFO-commands are:

```
T = 1:100000;
sT = gum(2) - gum(1)*log(-log(1-1./T));
semilogx(T,sT), hold on
N = 1:length(Hs); Nmax = max(N);
plot(Nmax./N,sort(Hs,'descend'),'.')
title('Return values in the Gumbel model')
xlabel('Return priod')
ylabel('Return value'), hold off
```

In the next section we shall see a more realistic example of return value analysis. The Atlantic data did not represent yearly maxima and the example was included only as an alternative way to present the result of a Gumbel analysis.

6.2 The GPD and GEV families

The Generalized Pareto Distribution (GPD) has the distribution function

$$\text{GPD:} \qquad F(x;k,\sigma) = \begin{cases} 1 - \left(1 - kx/\sigma\right)^{1/k}, & \text{if } k \neq 0, \\ 1 - \exp\{-x/\sigma\}, & \text{if } k = 0, \end{cases} \tag{6.7}$$

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

$$\text{GEV:} \qquad F(x;k,\mu,\sigma) = \begin{cases} \exp\left\{-(1 - k(x-\mu)/\sigma)^{1/k}\right\}, & \text{if } k \neq 0, \\ \exp\left\{-\exp\{-(x-\mu)/\sigma\}\right\}, & \text{if } k = 0, \end{cases} \tag{6.8}$$

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.

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.

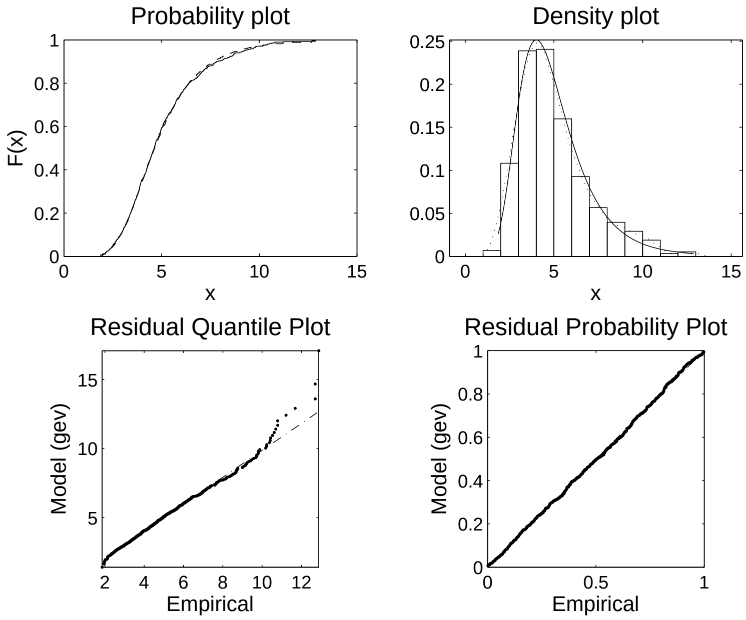


Figure 6.2: *Empirical distribution (solid), cdf and pdf, of significant wave-height in* `atlantic` *data, with estimated (dashed) Generalized Extreme Value distribution, and two diagnostic plots og goodness of fit.*

6.2.1 Generalized Extreme Value distribution

For the Generalized Extreme Value (GEV) distribution the estimation methods used in the WAFO toolbox are the Maximum Likelihood (ML) method and the method with Probability Weighted Moments (PWM), described in [50] and [23]. The programs have been adapted to MATLAB from a package of S-Plus routines described in [7].

We start with the significant wave-height data for the `Atlantic` data, stored in `Hs`. The command

```
gev = fitgev(Hs,'plotflag',2)
```

will give estimates `gev.params = [khat sigmahat muhat]` of the parameters (k, σ, μ) in the GEV distribution (6.8). The output matrix field `gev.covariance` 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 and two diagnostic plots that give indications of the goodness of fit; see Figure 6.2.

The routine `plotkde`, which is a simplified version of the kernel density estimation routines in `kdetools`, is used to compare the GEV density given estimated parameters with a non-parametric estimate (note that `plotkde` can be slow for large data sets like `Hs`). The commands

```
clf
x = linspace(0,14,200);
plotkde(Hs,[x;pdfgev(x,gev)]')
```

will give the upper right diagram Figure 6.2.

The default estimation algorithm for the GEV distribution is the method with Probability Weighted Moments (PWM). An optional second argument, `fitgev(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.

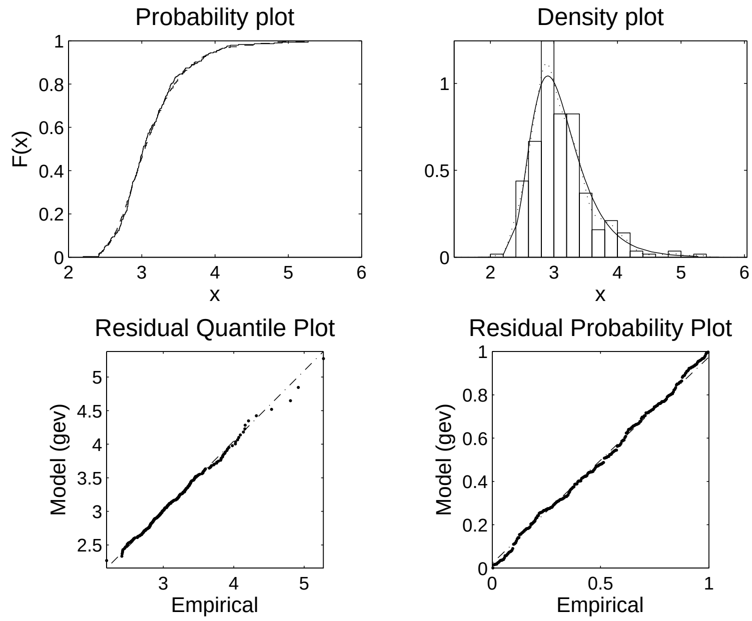


Figure 6.3: *GEV analysis of 285 maxima over 5 minute intervals of sea level data Yura87.*

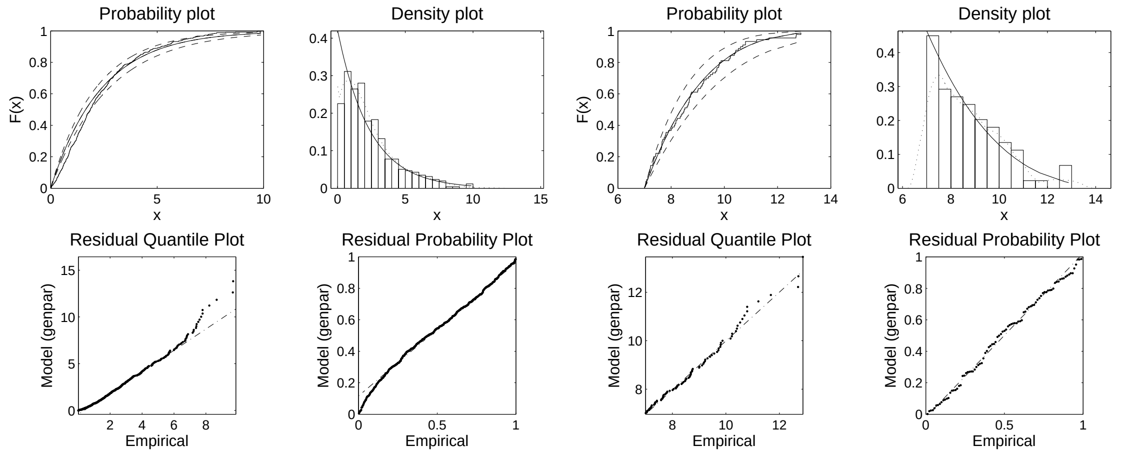
Example 12. *(Wave data from the Yura station)* The WAFO toolbox contains a data set `yura87` of more than 23 hours of water level registrations at the Poseidon platform in the Japan Sea; see `help yura87`. Sampling rate is 1 Hz and to smooth data we interpolate to 4 Hz, and then group the data into a matrix with 5 minutes of data in each column, leaving out the last, unfinished period.

```
xn = load('yura87.dat');
XI = 0:0.25:length(xn);
N  = length(XI); N = N-mod(N,4*60*5);
YI = interp1(xn(:,1),xn(:,2),XI(1:N),'spline');
YI = reshape(YI,4*60*5,N/(4*60*5)); % Each column holds
                                    %  5 minutes of interpolated data.
```

It turns out that the mean value and standard deviation change slowly during the measuring period, and we therefore standardize each column to zero mean and unit variance, before we take the maximum over each 5 minute interval and perform the GEV analysis; compare the results with those in the simpler analysis in Section 1.4.5.

```
Y5 = (YI-ones(1200,1)*mean(YI))./(ones(1200,1)*std(YI));
Y5M = max(Y5);
Y5gev = fitgev(Y5M,'plotflag',2)
```

The estimated parameters in `Y5gev.params` are $k = -0.314$ with a 95% confidence interval of $(-0.12, 0.06)$, indicating that a Gumbel distribution might be an acceptable choice. Location and



(a) (b)

Figure 6.4: *(a) Exceedances of significant wave-height data over level 3, (b) Significant wave-heigh over level 7, in* `atlantic` *data*

scale are estimated to $\mu = 2.91$ and $\sigma = 0.34$. Figure 6.3 shows a good fit to the GEV model for the series of 5 minute maxima in the (standardized) Yura series, except for the few largest values, which are underestimated by the model. This is posssibly due to a few short periods with very large variability in the data. □

6.2.2 Generalized Pareto distribution

For the Generalized Pareto distribution (GPD) the WAFO uses the method with Probability Weighted Moments (PWM), described in [22], and the standard Method of Moments (MOM), as well as a general method suggested by Pickands, in [47]. S-Plus routines for these methods are described in [7].

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 in the `atlantic` H_s series over of thresholds 3 and 7, one uses the commands

```
gpd3 = fitgenpar(Hs(Hs>3)-3,'plotflag',1);
figure
gpd7 = fitgenpar(Hs(Hs>7),'fixpar',...
              [nan,nan,7],'plotflag',1);
```

This will give estimates `gpd.params = [khat sigmahat]` of the parameters (k, σ) in the Generalized Pareto distribution (6.7) based on exceedance data `Hs(Hs>u)-u`. The optional output matrix `gpd.covariance` 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 6.4. The fit is better for exceedances over level 7 than over 3, but there are less data available, and the confidence bounds are wider.

The choice of estimation method is rather dependent on the actual parameter values. The default estimation algorithm in WAFO for estimation in the Generalized Pareto distribution is the Maximum Product of Spacings (MPS) estimator since it works for all values of the shape parameter and have the same asymptotic properties as the Maximum Likelihood (ML) method (when it is valid). The Pickands' (PKD) and Least Squares (LS) estimator also work for any value of the shape parameter k in Eq. (6.7). The ML method is only useful when $k \leq 1$, the PWM when $k > -0.5$, the MOM when $k > -0.25$. The variances of the ML estimates are usually smaller than those of the other estimators. However, for small sample sizes it is recommended to use the PWM, MOM or MPS if they are valid.

It is possible to simulate independent GEV and GPD observations in WAFO. The command series

```
Rgev = rndgev(0.3,1,2,1,100);
gp = fitgev(Rgev,'method','pwm');
gm = fitgev(Rgev,'method','ml','start',gp.params,...
            'plotflag',0);
x=sort(Rgev);
plotedf(Rgev,gp,{'-','r-'}); hold on
plot(x,cdfgev(x,gm),'--'); 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 = rndgenpar(0.4,1,0,1,100);
plotedf(Rgpd); hold on
gp = fitgenpar(Rgpd,'method','pkd','plotflag',0);
x=sort(Rgpd);
plot(x,cdfgenpar(x,gp))
gw = fitgenpar(Rgpd,'method','pwm','plotflag',0);
plot(x,cdfgenpar(x,gw),'g:')
gml = fitgenpar(Rgpd,'method','ml','plotflag',0);
plot(x,cdfgenpar(x,gml),'--')
gmps = fitgenpar(Rgpd,'method','mps','plotflag',0);
plot(x,cdfgenpar(x,gmps),'r-.'); hold off
```

with the four different methods of parameter estimation. The results are shown in Figure 6.5(a) and (b).

6.2.3 Return value analysis

As in the Gumbel model, one can calculate the return levels in the GEV by inverting (6.5) with the GEV distribution function (6.8). The return level corresponding to return period N satisfies $1 - F(s_N) = 1/N$, so when F is a GEV distribution function with shape parameter $k \neq 0$,

$$s_N = \mu + \frac{\sigma}{k}\left(1 - (-\log(1 - 1/N))^k\right) \approx \mu + \frac{\sigma}{k}\left(1 - N^{-k}\right), \tag{6.9}$$

where the last expression holds for N large, so one can use $-\log(1 - 1/N) \approx 1/N$. As always in practice, the parameters in the return level have to be replaced by their estimated values, which introduces uncertainties in the computed level.

Example 12. (contd.) Applied to the `Yura87` data and the estimated GEV-model, we perform the return level extrapolation by the commands,

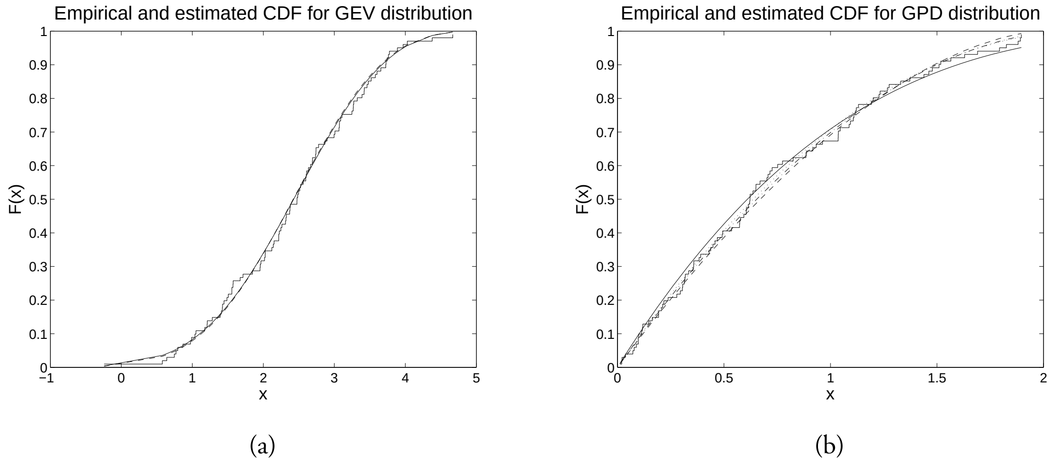


Figure 6.5: *Empirical (solid) distributions and estimated (dashed) distribution functions for 100 observations of GEV (a) and GPD (b) variables.*

```
T = 1:100000;
k = Y5gev.params(1); mu=Y5gev.params(3);
sigma = Y5gev.params(2);
sT = mu + sigma/k*(1-(log(1-1./T))^k);
semilogx(T,sT), hold
N = 1:length(Y5M); Nmax=max(N);
plot(Nmax./N,sort(Y5M,'descend'),'.')
title('Return values in the GEV model')
xlabel('Return priod')
ylabel('Return value')
grid on; hold off
```

The result is shown in Figure 6.6, which is consistent with the quantile plot in Figure 6.3. □

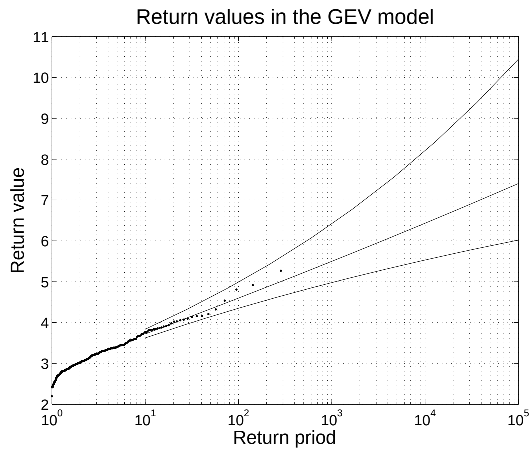


Figure 6.6: *Return level extrapolation in the* Yura87 *data depends on the good fit in the main part of the distribution. A few deviating large observations are disturbing.*

6.3 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 that 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.

6.3.1 Expected exceedance

The simplest distribution to fit to the exceedances over a level u is the Generalized Pareto distribution, GPD, with distribution function (6.7). 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 but with different (if $k \neq 0$) scale parameter $\sigma - ku$,

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

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:

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

Plotting the mean exceedance as a function of u can help on decide on a proper threshold value. The resulting plot is called *Mean residual life plot*, also referred to as mean excess plots in statistical literature. The following command illustrate this for the significant wave height `atlantic` data:

```
plotreslife(Hs,'umin',2,'umax',10,'Nu',200);
```

The result is plotted in Figure 6.7, and it seems to exhibit an almost linear relationship for $u \geq 7$.

6.3.2 Poisson + GPD = GEV

If one is successful in fitting a Generalized Pareto distribution to the tail 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.l, 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 size of 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.

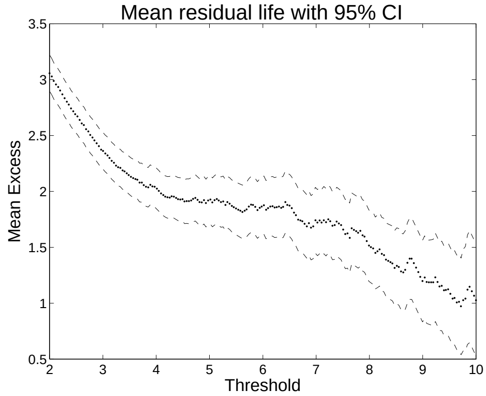


Figure 6.7: *Estimated expected exceedance over level u of* `atlantic` *data as function of u.*

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,

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

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) months 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 = fitgenpar(Hs(Hs>7)-7,'method','pwm','plotflag',0);
khat = gpd7.params(1);
sigmahat = gpd7.params(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,cdfgev(x,khat,bhat,ahat))
```

We have here used the threshold $u = 7$ since the exceedances over this level seem to fit well to a GPD distribution in Figures 6.4(b) and 6.7. 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 approximately 14 data points for 41 complete months, we can compute the monthly maxima `mm` and fit a GEV distribution directly:

```
mm = zeros(1,41);
```

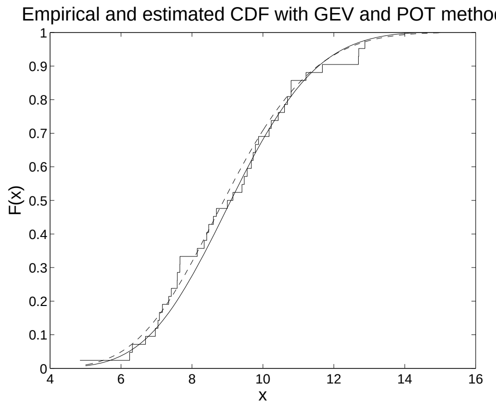


Figure 6.8: *Estimated distribution functions of monthly maxima with the POT method (solid), fitting a GEV (dashed) and the empirical distribution.*

```
for i=1:41                    % Last month is not complete
  mm(i) = max(Hs(((i-1)*14+1):i*14));
end
gev = fitgev(mm);
plotedf(mm), hold on
plot(x,cdfgev(x,gev),'--'), hold off
```

The results of the two methods agree very well in this case as can be seen in Figure 6.8, where the estimated distributions are plotted together with the empirical distribution of `mm`.

6.3.3 Declustering

The POT method relies on two properties of peaks over the selected threshold: they should occur randomly in time according to an approximate Poisson process, and the exceedances should have an approximate GPD distribution and be approximately independent. 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. The selected peaks should be sufficiently far apart for the exceedances to be independent. The WAFO toolbox contains the routine `decluster` to perform the declustering.

To select the clusters and check the Poisson character one can use the *dispersion index*, which is the ratio between the variance and the expectation of the number of peaks. For a Poisson distribution this ratio is equal to one. An acceptable peak separation should give a dispersion index near one.

Example 13. *(Declustering sea data)* We will extract peaks over threshold in the `sea.dat`, which is a recording of almost 40 minutes of sea level data, sampled at a rate of 4 [Hz].

We first define some parameters, `Nmin,Tmin,Tb`, to control the declustering, and to identify the peaks that exceed 90% of the median peak size and are separated by at least `Tmin`.

```
Nmin = 7;                          % minimum number of extremes
Tmin = 5;                          % minimum distance between extremes
Tb = 15;                           % block period
```

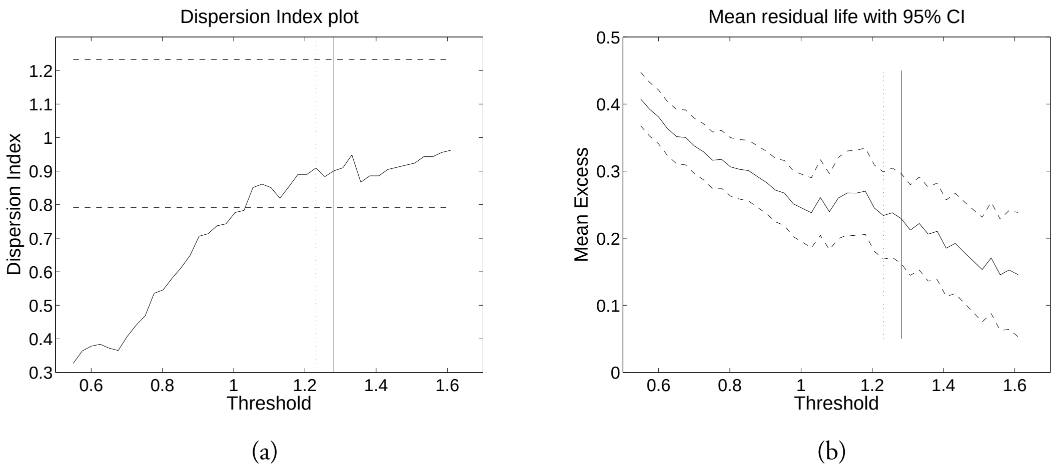


Figure 6.9: *Threshold selection in POT analysis. Dashed vertical line indicates threshold selected by the dispersion index, solid line by the residual life analysis.*

```
xx = load('sea.dat');
timeSpan = (xx(end,1)-xx(1,1))/60;    % in minutes
dt = xx(2,1)-xx(1,1);                 % in seconds
tc = dat2tc(xx);
umin = median(tc(tc(:,2)>0,2));
Ie0 = findpot(tc, 0.9*umin, Tmin);
Ev = sort(tc(Ie0,2));
Ne = numel(Ev)
if Ne>Nmin && Ev(Ne-Nmin)>umin, umax = Ev(Ne-Nmin);
else umax = umin;
end
```

Next, we calculate the expected residual life and the dispersion index for thresholds between `umin` and `umax` and select an interval which is compatible with the Poisson distribution for the number of peaks.

```
Nu = floor((umax-umin)/0.025)+1;
u = linspace(umin,umax,Nu);
mrl = reslife(Ev, 'u',u);
umin0 = umin;
for io = numel(mrl.data):-1:1,
  CI = mrl.dataCI(io:end,:);
  if ~(max(CI(:,1))<=mrl.data(io) & mrl.data(io)<=min(CI(:,2))),
      umin0 = mrl.args(io); break;
  end
end
[di, threshold, ok_u] = ...
      disprsnidx(tc(Ie0,:), 'Tb', Tb, 'alpha',0.05, 'u',u);
```

The plots from the following commands are shown in Figure 6.9. It seems as if `threshold = 1.23[m]` is a suitable threshold.

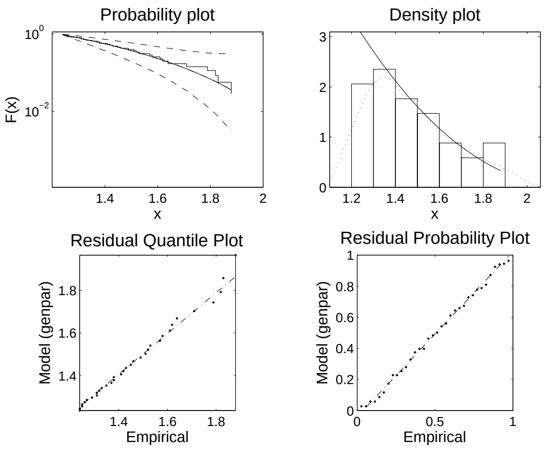


Figure 6.10: *Diagnostic GPD plot for sea data return levels.*

```
figure(1); plot(di)
vline(threshold)       % Threshold from dispersion index
vline(umin0,'g')       % Threshold from mean residual life plot
figure(2); plot(mrl)
vline(threshold)       % Threshold from dispersion index
vline(umin0,'g')       % Threshold from mean residual life plot
```

A GPD fit for peaks above `1.23[m]` with diagnostic plot is obtained by the commands

```
Ie = findpot(tc, threshold, Tmin);
lambda = numel(Ie)/timeSpan; % # Y>threshold per minute
varLambda = lambda*(1-(dt/60)*lambda)/timeSpan;
stdLambd = sqrt(varLambda)
Ev = tc(Ie,2);
phat = fitgenpar(Ev, 'fixpar',[nan,nan,threshold], 'method','mps');
figure(3); phat.plotfitsumry() % check fit to data
```

The diagnostic plots are found in Figure 6.10. The last step is to calculate the numerical value and some confidence intervals for a return level, and we do so for a three hour period, `180 min`.

```
Tr = 3*60              % Return period in minutes
[xr,xrlo,xrup] = invgenpar(1./(lambda*Tr),phat,...
  'lowertail',false,'alpha', 0.05) % return level + 95%CI
[xr,xrlo5,xrup5] = invgenpar(1./(lambda*Tr),phat,...
  'lowertail',false,'alpha', 0.5)  % return level + 50%CI
```

The three hour return level is thus estimated to `xr + threshold = 2.02 + 1.23 = 3.25[m]` with a 95% confidence interval `(2.53, 11.31)`. The 50% confidence bounds are `(2.81, 4.28)`; as expected, a high confidence leads to a very high upper limit. □

6.4 Summary of statistical procedures in WAFO

The extreme value analysis presented in this chapter is part of a comprehensive library of statistical routines for random number generation, probability distributions, and parameter and density estimation and likelihood analysis, etc.

```
help statistics

  Module STATISTICS in WAFO Toolbox.
  Version 2.5.2  07-Feb-2011

  What's new
    Readme            - New features, bug fixes, and changes
                         in STATISTICS.

  Parameter estimation
    fitbeta           - Parameter estimates for Beta data
    fitchi2           - Parameter estimates for
                                  Chi squared data
    fitexp            - Parameter estimates for
                                  Exponential data
    fitgam            - Parameter estimates for Gamma data
    fitgengam         - Parameter estimates for
                                  Generalized Gamma data
    fitgenpar         - Parameter estimates for
                                  Generalized Pareto data
    fitgenparml       - Internal routine for fitgenpar
                        (ML estimates for GPD data)
    fitgenparrange    - Parameter estimates for GPD model
                                  over a range of thresholds
    fitgev            - Parameter estimates for GEV data
    fitgumb           - Parameter estimates for Gumbel data
    fitinvnorm        - Parameter estimates for
                                  Inverse Gaussian data
    fitlognorm        - Parameter estimates for Lognormal data
    fitmarg2d         - Parameter estimates for MARG2D data
    fitmargcnd2d      - Parameter estimates for DIST2D data
    fitnorm           - Parameter estimates for Normal data
    fitray            - Parameter estimates for Rayleigh data
    fitraymod         - Parameter estimates for
                                  Truncated Rayleigh data
    fitt              - Parameter estimates for
                                  Student's T data
    fitweib           - Parameter estimates for Weibull data
    fitweib2d         - Parameter estimates for 2D Weibull data
    fitweibmod        - Parameter estimates for
                                  truncated Weibull data
```

```
  likgenpar        - Log likelihood function for GPD
  likweib2d        - 2D Weibull log-likelihood function

  loglike          - Negative Log-likelihood function.
  logps            - Moran's negative log Product
                               Spacings statistic
  mlest            - Maximum Likelihood or Maximum Product
                     Spacing estimator

Probability density functions (pdf)
  pdfbeta          - Beta PDF
  pdfbin           - Binomial PDF
  pdfcauchy        - Cauchy's PDF
  pdfchi2          - Chi squared PDF
  pdfdiscrete      - Discrete PDF
  pdfempirical     - Empirical PDF
  pdfexp           - Exponential PDF
  pdff             - Snedecor's F PDF
  pdffrech         - Frechet PDF
  pdfgam           - Gamma PDF
  pdfgengam        - Generalized Gamma PDF
  pdfgengammod     - Modified Generalized Gamma PDF (stable)
  pdfgenpar        - Generalized Pareto PDF
  pdfgev           - Generalized Extreme Value PDF
  pdfgumb          - Gumbel PDF.
  pdfhyge          - Hypergeometric probability mass function
  pdfinvnorm       - Inverse Gaussian PDF
  pdflognorm       - Lognormal PDF
  pdfmarg2d        - Joint 2D PDF due to Plackett given as
                     f{x1}*f{x2}*G(x1,x2;Psi).
  pdfmargcnd2d     - Joint 2D PDF computed as
                     f(x1|X2=x2)*f(x2)
  pdfnorm          - Normal PDF
  pdfnorm2d        - Bivariate Gaussian distribution
  pdfnormnd        - Multivariate Normal PDF
  pdfray           - Rayleigh PDF
  pdfraymod        - Truncated Rayleigh PDF
  pdft             - Student's T PDF
  pdfpois          - Poisson probability mass function
  pdfweib          - Weibull PDF
  pdfweib2d        - 2D Weibull PDF
  pdfweibmod       - Truncated Weibull PDF

Cumulative distribution functions (cdf)
  cdfcauchy        - Cauchy CDF
```

```
cdfdiscrete       - Discrete CDF
cdfempirical      - Empirical CDF
cdfmarg2d         - Joint 2D CDF due to Plackett
cdfmargcnd2d      - Joint 2D CDF computed as
                    int F(X1<v|X2=x2).*f(x2)dx2
cdfmargcnd2dfun   - is an internal function to cdfmargcnd2d
                    and prbmargcnd2d.
cdfnormnd         - Multivariate normal CDF
cdfweib2d         - Joint 2D Weibull CDF
cdfbeta           - Beta CDF
cdfbin            - Binomial CDF
cdfchi2           - Chi squared CDF
cdfexp            - Exponential CDF
cdff              - Snedecor's F CDF
cdffrech          - Frechet CDF
cdfgam            - Gamma CDF
cdfgengam         - Generalized Gamma CDF
cdfgengammod      - Modified Generalized Gamma CDF
cdfgenpar         - Generalized Pareto CDF
cdfgev            - Generalized Extreme Value CDF
cdfgumb           - Gumbel CDF
cdfhyge           - The hypergeometric CDF
cdfinvnorm        - Inverse Gaussian CDF
cdflognorm        - Lognormal CDF
cdfmargcnd2d      - Joint 2D CDF computed as
                    int F(X1<v|X2=x2).*f(x2)dx2
cdfnorm           - Normal CDF
cdfray            - Rayleigh CDF
cdfraymod         - Modified Rayleigh CDF
cdft              - Student's T CDF
cdfpois           - Poisson CDF
cdfweib           - Weibull CDF
cdfweibmod        - Truncated Weibull CDF

edf               - Empirical Distribution Function
edfcnd            - Empirical Distribution Function
                    conditioned on X>=c

prbmargcnd2d      - returns the probability for rectangular
                    regions
prbweib2d         - returns the probability for rectangular
                    regions
margcnd2dsmfun    - Smooths the MARGCND2D distribution
                    parameters
margcnd2dsmfun2   - Smooths the MARGCND2D distribution
                    parameters
```

```
Inverse cumulative distribution functions
  invbeta           - Inverse of the Beta CDF
  invbin            - Inverse of the Binomial CDF
  invcauchy         - Inverse of the Cauchy CDF
  invchi2           - Inverse of the Chi squared CDF
  invcmarg2d        - Inverse of the conditional CDF of
                      X2 given X1
  invcweib2d        - Inverse of the conditional 2D weibull
                      CDF of X2 given X1
  invdiscrete       - Disrete quantile
  invempirical      - Empirical quantile
  invexp            - Inverse of the Exponential CDF
  invf              - Inverse of the Snedecor's F CDF
  invfrech          - Inverse of the Frechet CDF
  invgam            - Inverse of the Gamma CDF
  invgengam         - Inverse of the Generalized Gamma CDF
  invgengammod      - Inverse of the Generalized Gamma CDF
  invgenpar         - Inverse of the Generalized Pareto CDF
  invgev            - Inverse of the Generalized
                      Extreme Value CDF
  invgumb           - Inverse of the Gumbel CDF
  invhyge           - Inverse of the Hypergeometric CDF
  invinvnorm        - Inverse of the Inverse Ga(ussian CDF
  invlognorm        - Inverse of the Lognormal CDF
  invnorm           - Inverse of the Normal CDF
  invray            - Inverse of the Rayleigh CDF
  invt              - Inverse of the Student's T CDF
  invweib           - Inverse of the Weibull CDF
  invpois           - Inverse of the Poisson CDF
  invraymod         - Inverse of the modified Rayleigh CDF
  invweibmod        - Inverse of the modified Weibull CDF

Random number generators
  rndalpha          - Random matrices from a symmetric
                      alpha-stable distribution
  rndbeta           - Random matrices from a Beta distribution
  rndbin            - Random numbers from the binomial
                      distribution
  rndboot           - Simulate a bootstrap resample from a
                      sample
  rndcauchy         - Random matrices a the Cauchy
                      distribution
  rndchi2           - Random matrices from a Chi squared
                      distribution
  rnddiscrete       - Random sample
```

```
rndempirical     - Bootstrap sample
rndexp           - Random matrices from an Exponential
                   distribution
rndf             - Random matrices from the Snedecor's F
                   distribution
rndfrech         - Random matrices from a Frechet
                   distribution
rndgam           - Random matrices from a Gamma distribution
rndgengam        - Random matrices from a Generalized Gamma
                   distribution.
rndgengammod     - Random matrices from a Generalized
                   Modified Gamma distribution.
rndgenpar        - Random matrices from a Generalized Pareto
                   Distribution
rndgev           - Random matrices from a Generalized
                   Extreme Value distribution
rndgumb          - Random matrices from a Gumbel
                   distribution
rndhyge          - Random numbers from the Hypergeometric
                   distribution
rndinvnorm       - Random matrices from a Inverse Gaussian
                   distribution
rndlognorm       - Random matrices from a Lognormal
                   distribution.
rndmarg2d        - Random points from a MARG2D
                   distribution
rndmargcnd2d     - Random points from a MARGCND2D
                   distribution
rndnorm          - Random matrices from a Normal
                   distribution
rndnormnd        - Random vectors from a multivariate
                   Normal distribution
rndpois          - Random matrices from a Poisson
                   distribution
rndray           - Random matrices from a Rayleigh
                   distribution
rndraymod        - Random matrices from modified Rayleigh
                   distribution
rndt             - Random matrices from a Student's T
                   distribution
rndweib          - Random matrices a the Weibull
                   distribution
rndweibmod       - Random matrices from the modified Weibull
                   distribution
rndweib2d        - Random numbers from the 2D Weibull
                   distribution
```

```
Moments
  mombeta          - Mean and variance for the Beta
                     distribution
  mombin           - Mean and variance for the Binomial
                     distribution
  momchi2          - Mean and variance for the Chi squared
                     distribution
  momexp           - Mean and variance for the Exponential
                     distribution
  momf             - Mean and variance for the Snedecor's F
                     distribution
  momfrech         - Mean and variance for the Frechet
                     distribution
  momgam           - Mean and variance for the Gamma
                     distribution
  momgengam        - Mean and variance for the Generalized
                     Gamma distribution
  momgenpar        - Mean and variance for the Generalized
                     Pareto distribution
  momgev           - Mean and variance for the GEV
                     distribution
  momgumb          - Mean and variance for the Gumbel
                     distribution
  momhyge          - Mean and variance for the Hypergeometric
                     distribution
  mominvnorm       - Mean and variance for the Inverse
                     Gaussian distribution
  momlognorm       - Mean and variance for the Lognormal
                     distribution
  mommarg2d        - Mean and variance for the MARG2D
                     distribution
  mommargcnd2d     - Mean and variance for the MARGCND2D
                     distribution
  momnorm          - Mean and variance for the Normal
                     distribution
  mompois          - Mean and variance for the Poisson
                     distribution
  momray           - Mean and variance for the Rayleigh
                     distribution
  momt             - Mean and variance for the Student's T
                     distribution
  momweib          - Mean and variance for the Weibull
                     distribution
  momweib2d        - Mean and variance for the 2D Weibull
                     distribution
```

```
Profile log likelihood functions
  lnkexp           - Link for x,F and parameters of
                     Exponential distribution
  lnkgenpar        - Link for x,F and parameters of
                     Generalized Pareto distribution
  lnkgev           - Link for x,F and parameters of
                     Generalized Extreme value distribution
  lnkgumb          - Link for x,F and parameters of Gumbel
                     distribution
  lnkgumbtrnc      - Link for x,F and parameters of truncated
                     Gumbel distribution
  lnkray           - Link for x,F and parameters of Rayleigh
                     distribution
  lnkweib          - Link for x,F and parameters of Weibull
                     distribution
  loglike          - Negative Log-likelihood function
  logps            - Moran's negative log Product Spacings
                     statistic
  ciproflog        - Confidence Interval using Profile Log-
                     likelihood or Product Spacing- function
  proflog          - Profile Log- likelihood or
                     Product Spacing-function
  findciproflog    - Find Confidence Interval from proflog
                     function

Extremes
  decluster        - Decluster peaks over threshold values
  extremalidx      - Extremal Index measuring the dependence
                     of data
  findpot          - Find indices to Peaks over threshold
                     values
  fitgev           - Parameter estimates for GEV data
  fitgenpar        - Parameter estimates for Generalized
                     Pareto data
  prb2retper       - Return period from Probability of
                     exceedance
  retper2prb       - Probability of exceedance from return
                     period

Threshold selection
  fitgenparrange   - Parameter estimates for GPD model vs
                     thresholds
  disprsnidx       - Dispersion Index vs threshold
  reslife          - Mean Residual Life, i.e., mean excesses
                     vs thresholds
```

```
  plotdisprsnidx   - Plot Dispersion Index vs thresholds
  plotreslife      - Plot Mean Residual Life
                     (mean excess vs thresholds)

Regression models
  logit            - Logit function.
  logitinv         - Inverse logit function.
  regglm           - Generalized Linear Model regression
  reglm            - Fit multiple Linear Regression Model.
  reglogit         - Fit ordinal logistic regression model.
  regnonlm         - Non-Linear Model Regression
  regsteplm        - Stepwise predictor subset selection for
                     Linear Model regression

Factor analysis
  princomp         -  Compute principal components of X

Descriptive Statistics
  ranktrf          - Rank transformation of data material.
  spearman         - Spearman's rank correlation coefficient
  mean             - Computes sample mean (Matlab)
  median           - Computes sample median value (Matlab)
  std              - Computes standard deviation (Matlab)
  var              - Computes sample variance (Matlab)
  var2corr         - Variance matrix to correlation matrix
                     conversion
  cov              - Computes sample covariance matrix
                     (Matlab)
  corrcoef         - Computes sample correlation coefficients
                     (Matlab toolbox)
  skew             - Computes sample skewness
  kurt             - Computes sample kurtosis
  lmoment          - L-moment based on order statistics
  percentile       - Empirical quantile (percentile)
  iqrange          - Computes the Inter Quartile Range
  range            - Computes the range between the maximum
                     and minimum values

Statistical plotting
  clickslct        - Select points in a plot by clicking
                     with the mouse
  histgrm          - Plot histogram
  plotbox          - Plot box-and-whisker diagram
  plotdensity      - Plot density.
  plotexp          - Plot data on Exponential distribution
                     paper
```

```
  plotedf          - Plot Empirical Distribution Function
  plotedfcnd       - Plot Empirical Distribution Function
                     CoNDitioned that X>=c
  plotfitsumry     - Plot diagnostic of fit to data
  plotgumb         - Plot data on Gumbel distribution paper
  plotkde          - Plot kernel density estimate of PDF
  plotmarg2dcdf    - Plot conditional CDF of X1 given X2=x2
  plotmarg2dmom    - Plot conditional mean and standard
                     deviation
  plotmargcnd2dcdf - Plot conditional empirical CDF of X1
                     given X2=x2
  plotmargcnd2dfit - Plot parameters of the conditional
                     distribution
  plotmargcnd2dmom - Plot conditional mean and
                     standard deviation
  plotnorm         - Plot data on a Normal distribution paper
  plotqq           - Plot empirical quantile of X vs empirical
                     quantile of Y
  plotray          - Plot data on a Rayleigh distribution
                     paper
  plotresprb       - Plot Residual Probability
  plotresq         - Plot Residual Quantile
  plotscatr        - Pairwise scatter plots
  plotweib         - Plot data on a Weibull distribution paper
  plotweib2dcdf    - Plot conditional empirical CDF of X1
                     given X2=x2
  plotweib2dmom    - Plot conditional mean and standard
                     deviation

Hypothesis Tests
  anovan           - multi-way analysis of variance (ANOVA)
  testgumb         - Tests if shape parameter in a GEV is
                     equal to zero
  testmean1boot    - Bootstrap t-test for the mean equal to 0
  testmean1n       - Test for mean equals 0 using
                     one-sample T-test
  testmean2n       - Two-sample t-test for mean(x) equals
                     mean(y)
  testmean1r       - Wilcoxon signed rank test for
                     H0: mean(x) equals 0
  testmean2r       - Wilcoxon rank-sum test for
                     H0: mean(x) equals mean(y)

Confidence interval estimation
  ciboot           - Bootstrap confidence interval.
  ciquant          - Nonparametric confidence interval for quantile
```

```
  momci1b          - Moment confidence intervals using
                     Bootstrap

Bootstrap & jacknife estimates
  covboot          - Bootstrap estimate of the variance of
                     a parameter estimate.
  covjack          - Jackknife estimate of the variance of
                     a parameter estimate.
  stdboot          - Bootstrap estimate of the
                     standard deviation of a parameter
  stdjack          - Jackknife estimate of the
                     standard deviation of a parameter

Design of Experiments
  yates            - Calculates main and interaction effects
                     using Yates' algorithm.
  ryates           - Reverse Yates' algorithm to give
                     estimated responses
  fitmodel         - Fits response by polynomial
  alias            - Alias structure of a fractional design
  cdr              - Complete Defining Relation
  cl2cnr           - Column Label to Column Number
  cnr2cl           - Column Number to Column Label
  ffd              - Two-level Fractional Factorial Design
  getmodel         - Return the model parameters
  sudg             - Some Useful Design Generators
  plotresponse     - Cubic plot of responses
  nplot            - Normal probability plot of effects

Misc
  comnsize         - Calculates common size of all non-scalar
                     arguments
  dgammainc        - Incomplete gamma function with derivatives
  gammaincln       - Logarithm of incomplete gamma function.
  parsestatsinput  - Parses inputs to pdfxx, prbxx, invxx and
                     rndxx functions
  createfdata      - Distribution parameter struct constructor
  getdistname      - Return the distribution name

  stdize           - Standardize columns to have mean 0 and
                     standard deviation 1
  center           - Recenter columns to have mean 0

 Demo
  demofitgenpar    - Script to check the variance of estimated
                     parameters
```

Part III

Appendices

APPENDIX A

Kernel density estimation

Histograms are among the most popular ways to visually present data. They are particular examples of density estimates and their appearance depends on both the choice of origin and the width of the intervals (bins) used. In order for the histogram to give useful information about the true underlying distribution, a sufficient amount of data is needed. This is even more important for histograms in two dimensions or higher. Also the discontinuity of the histograms may cause problems, e.g., if derivatives of the estimate are required.

An effective alternative to the histogram is the kernel density estimate (KDE), which may be considered as a "smoothed histogram", only depending on the bin-width and not depending on the origin, see [63].

A.1 The univariate kernel density estimator

The univariate KDE is defined by

$$\hat{f}_X(x; h_s) = \frac{1}{n\,h_s} \sum_{j=1}^{n} K_d\left(\frac{x - X_j}{h_s}\right), \tag{A.1}$$

where n is the number of datapoints, $X_1, X_2, \ldots, X_n$, is the data set, and h_s is the smoothing parameter or window width. The kernel function K_d is usually a unimodal, symmetric probability density function. This ensures that the KDE itself is also a density. However, kernels that are not densities are also sometimes used [see 72], but these are not implemented in the WAFO toolbox.

To illustrate the method, consider the kernel estimator as a sum of "bumps" placed at the observations. The shape of the bumps are given by the kernel function while the width is given by the smoothing parameter, h_s. Fig. A.1 shows a KDE constructed using 7 observations from a standard Gaussian distribution with a Gaussian kernel function. One should note that the 7 points used here, is purely for clarity in illustrating how the kernel method works. Practical density estimation usually involves much higher number of observations.

Fig. A.1 also demonstrates the effect of varying the smoothing parameter, h_s. A too small value for h_s may introduce spurious bumps in the resulting KDE (top), while a too large value may obscure the details of the underlying distribution (bottom). Thus the choice of value for the smoothing parameter, h_s, is very important. How to select one will be elaborated further in the next section.

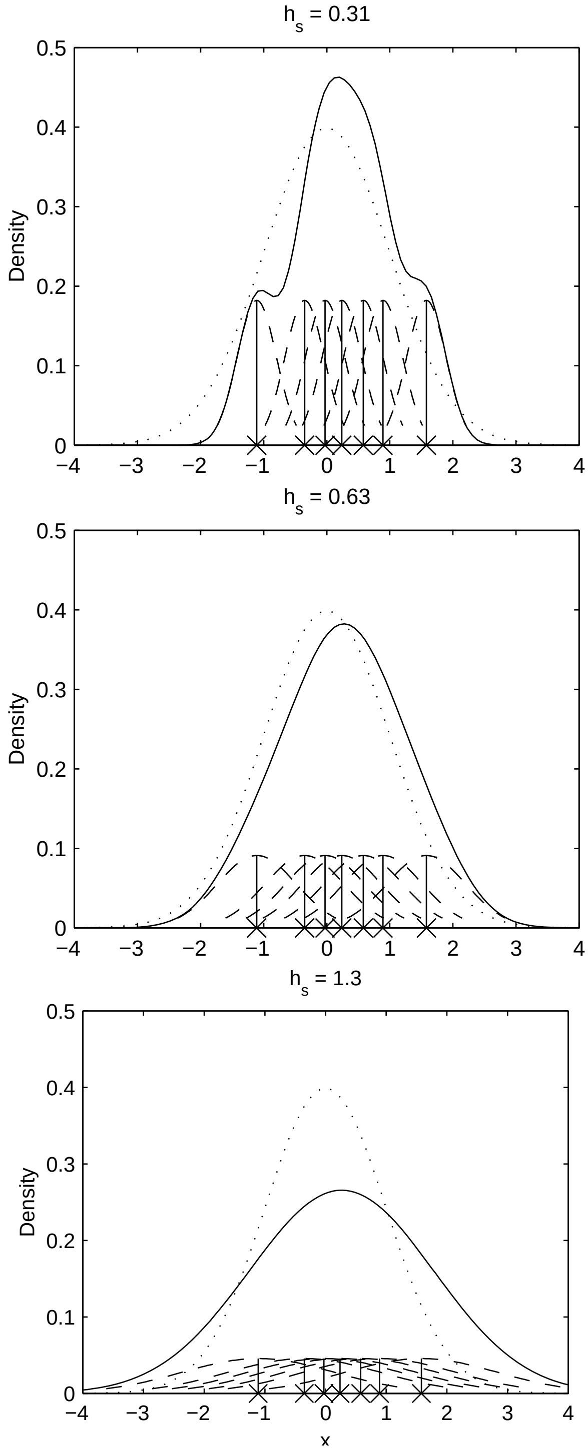


Figure A.1: *Smoothing parameter, h_s, impact on KDE: True density (dotted) compared to KDE based on 7 observations (solid) and their individual kernels (dashed).*

The particular choice of kernel function, on the other hand, is not very important since suboptimal kernels are not suboptimal by very much, [see 72, pp. 31]. However, the kernel that minimizes the mean integrated square error is the Epanechnikov kernel, and is thus chosen as the default kernel in the software, see Eq. (A.8). For a discussion of other kernel functions and their properties, see [72].

A.1.1 Smoothing parameter selection

The choice of smoothing parameter, h_s, is very important, as exemplified in Fig.A.1. In many situations it is satisfactory to select the smoothing parameter subjectively by eye, i.e., look at several density estimates over a range of bandwidths and selecting the density that is the most "pleasing" in some sense. However, there are also many circumstances where it is beneficial to use an automatic bandwidth selection from the data. One reason is that it is very time consuming to select the bandwidth by eye. Another reason, is that, in many cases, the user has no prior knowledge of the structure of the data, and does not have any feeling for which bandwidth gives a good estimate. One simple, quick and commonly used automatic bandwidth selector, is the bandwidth that minimizes the mean integrated square error (MISE) asymptotically. As shown in [72, Section 2.5 and 3.2.1], the one dimensional AMISE[1]-optimal normal scale rule assuming that the underlying density is Gaussian, is given by

$$h_{AMISE} = \left[\frac{4}{3\,n}\right]^{1/5} \widehat{\sigma}, \tag{A.2}$$

where $\widehat{\sigma}$ is some estimate of the standard deviation of the underlying distribution. Common choices of $\widehat{\sigma}$ are the sample standard deviation, $\widehat{\sigma}_s$, and the standardized interquartile range (denoted IQR):

$$\widehat{\sigma}_{IQR} = (\text{sample IQR})/(\Phi^{-1}(3/4) - \Phi^{-1}(1/4)) \approx (\text{sample IQR})/1.349, \tag{A.3}$$

where Φ^{-1} is the standard normal quantile function. The use of $\widehat{\sigma}_{IQR}$ guards against outliers if the distribution has heavy tails. A reasonable approach is to use the smaller of $\widehat{\sigma}_s$ and $\widehat{\sigma}_{IQR}$ in order to lessen the chance of oversmoothing, [see 63, pp. 47].

Various other automatic methods for selecting h_s are available and are discussed in [63] and in more detail in [72].

A.1.2 Transformation kernel denstity estimator

Densities close to normality appear to be the easiest for the kernel estimator to estimate. The estimation difficulty increases with skewness, kurtosis and multimodality [72, Chap. 2.9].

Thus, in the cases where the random sample $X_1, X_2, \ldots, X_n$, has a density, f, which is difficult to estimate, a transformation, t, might give a good KDE, i.e., applying a transformation to the data to obtain a new sample $Y_1, Y_2, \ldots, Y_n$, with a density g that more easily can be estimated using the basic KDE. One would then backtransform the estimate of g to obtain the estimate for f.

Suppose that $Y_i = t(X_i)$, where t is an increasing differentiable function defined on the support of f. Then a standard result from statistical distribution theory is that

$$f(x) = g(t(x))\, t'(x), \tag{A.4}$$

[1]AMISE = asymptotic mean integrated square error

where $t'(x)$ is the derivative. Backtransformation of the KDE of g based on $Y_1, Y_2, \ldots, Y_n$, leads to the explicit formula

$$\hat{f}_X(x; h_s, t) = \frac{1}{n\, h_s} \sum_{j=1}^{n} K_d \left(\frac{t(x) - t(X_j)}{h_s} \right) t'(x) \tag{A.5}$$

A simple illustrative example comes from the problem of estimating the Rayleigh density. This density is very difficult to estimate by direct kernel methods. However, if we apply the transformation $Y_i = \sqrt{X_i}$ to the data, then the normal plot of the transformed data, Y_i, becomes approximately linear. Fig. A.2 shows that the transformation KDE is a better estimate around 0 than the ordinary KDE.

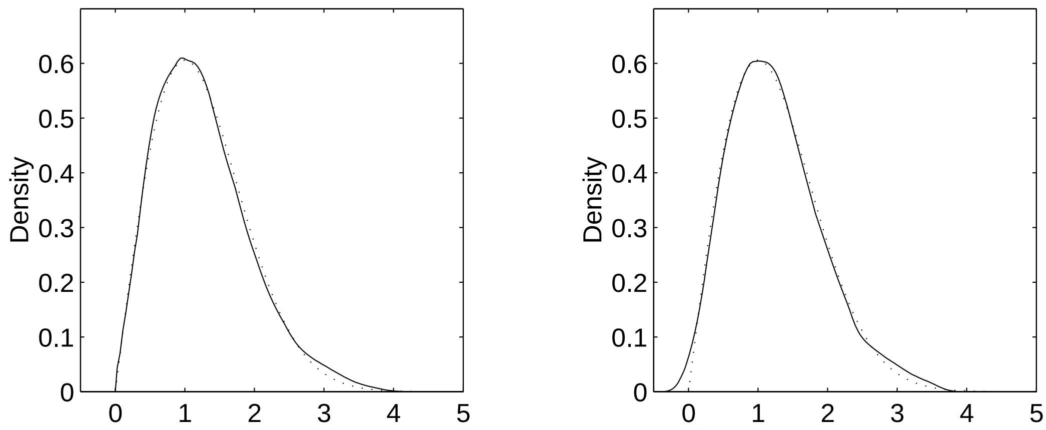


Figure A.2: *True Rayleigh density (dotted) compared to transformation KDE (solid,left) and ordinary KDE (solid, right) based on 1000 observations.*

A.2 The multivariate kernel density estimator

The multivariate kernel density estimator is defined in its most general form by

$$\hat{f}_{\mathbf{X}}(\mathbf{x}; \mathbf{H}) = \frac{|\mathbf{H}|^{-1/2}}{n} \sum_{j=1}^{n} K_d \left(\mathbf{H}^{-1/2} (\mathbf{x} - \mathbf{X}_j) \right), \tag{A.6}$$

where $\mathbf{H}$ is a symmetric positive definite $d \times d$ matrix called the *bandwidth matrix*. A simplification of Eq. (A.6) can be obtained by imposing the restriction $\mathbf{H} = \mathrm{diag}(h_1^2, h_2^2, \ldots, h_d^2)$. Then Eq. (A.6) reduces to

$$\hat{f}_{\mathbf{X}}(\mathbf{x}; \mathbf{h}) = \frac{1}{n \prod_{i=1}^{n} h_i} \sum_{j=1}^{n} K_d \left(\frac{x - X_{j\,1}}{h_1}, \frac{x - X_{j\,2}}{h_2}, \ldots, \frac{x - X_{j\,d}}{h_d} \right), \tag{A.7}$$

and is, in combination with a transformation, a reasonable solution to visualize multivariate densities.

The multivariate Epanechnikov kernel also forms the basis for the optimal spherically symmetric multivariate kernel and is given by

$$K_d(\mathbf{x}) = \frac{d+2}{2\, v_d} \left(1 - \mathbf{x}^T \mathbf{x} \right) \mathbf{1}_{\mathbf{x}^T \mathbf{x} \leq 1}, \tag{A.8}$$

where $v_d = 2\, \pi^{d/2} / (\Gamma(d/2)\, d)$ is the volume of the unit d-dimensional sphere.

In this tutorial we use the KDE to find a good estimator of the central part of the joint densities of wave parameters extracted from time series. Clearly, such data are dependent, so it is assumed that the time series are ergodic and short range dependent to justify the use of KDE:s, [see 72, Chap. 6]. Usually, KDE gives poor estimates of the tail of the distribution, unless large amounts of data is available. However, a KDE gives qualitatively good estimates in the regions of sufficient data, i.e., in the main parts of the distribution. This is good for visualization, e.g. detecting modes, symmetries of distributions.

The kernel density estimation software is based on `KDETOOL`, which is a MATLAB toolbox produced by Christian Beardah.[2] However, over the past 10 years the toolbox is totally rewritten and extended to include the transformation kernel estimator and generalized to cover any dimension for the data. The computational speed has also been improved.

[2] See http://science.ntu.ac.uk/msor/ccb/densest.html

Appendix B

Standardized wave spectra

Knowledge of which kind of spectral density is suitable to describe sea state data are well established from experimental studies. Qualitative considerations of wave measurements indicate that the spectra may be divided into 3 parts, (see Fig. B.1):

1. Sea states dominated by wind sea but significantly influenced by swell components.
2. More or less pure wind seas or, possibly, swell component located well inside the wind frequency band.
3. Sea states more or less dominated by swell but significantly influenced by wind sea.

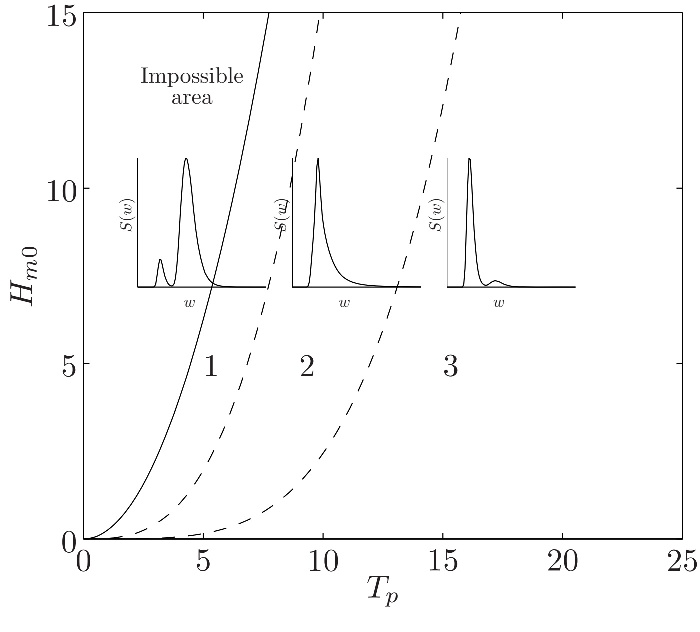


Figure B.1: *Qualitative indication of spectral variability.*

One often uses some parametric form of the spectral density. The three most important parametric spectral densities implemented in WAFO will be described in the following sections.

B.1 JONSWAP spectrum

The JONSWAP (JOint North Sea WAve Project) spectrum of [20] is a result of a multinational project to characterize standardized wave spectra for the Southeast part of the North Sea. The spectrum is valid for not fully developed sea states. However, it is also used to represent fully developed sea states. It is particularly well suited to characterize wind sea when $3.6\sqrt{H_{m0}} < T_p < 5\sqrt{H_{m0}}$. The JONSWAP spectrum is given in the form:

$$S^{+}(\omega) = \frac{\alpha g^2}{\omega^M} \exp\Big(-\frac{M}{N}\big(\frac{\omega_p}{\omega}\big)^N \Big)\, \gamma^{\exp\Big(\frac{-(\omega/\omega_p-1)^2}{2\,\sigma^2}\Big)}, \tag{B.1}$$

where

$$\begin{aligned} \sigma &= \begin{cases} 0.07 & \text{if } \omega < \omega_p, \\ 0.09 & \text{if } \omega \geq \omega_p, \end{cases} \\ M &= 5, \quad N = 4, \\ \alpha &\approx 5.061 \frac{H_{m0}^2}{T_p^4} \Big\{ 1 - 0.287\ \ln(\gamma) \Big\}. \end{aligned}$$

A standard value for the peakedness parameter, γ, is 3.3. However, a more correct approach is to relate γ to H_{m0} and T_p, and use

$$\gamma = \exp\Big\{ 3.484\,\big(1 - 0.1975\,(0.036 - 0.0056\,T_p/\sqrt{H_{m0}})\ T_p^4/H_{m0}^2\big) \Big\}. \tag{B.2}$$

Here γ is limited by $1 \leq \gamma \leq 7$. This parameterization is based on qualitative considerations of deep water wave data from the North Sea; see [70] and [21].

The relation between the peak period and mean zero-upcrossing period may be approximated by

$$T_{m02} \approx T_p/\,\big(1.30301 - 0.01698\,\gamma + 0.12102/\gamma\big) \tag{B.3}$$

The JONSWAP spectrum is identical with the two-parameter Pierson-Moskowitz, Bretschneider, ITTC (International Towing Tank Conference) or ISSC (International Ship and Offshore Structures Congress) wave spectrum, given H_{m0} and T_p, when $\gamma = 1$. (For more properties of this spectrum, see the WAFO function `jonswap.m`.

B.2 Torsethaugen spectrum

Torsethaugen, [67, 68, 69], proposed to describe bimodal spectra by

$$S^{+}(\omega) = \sum_{i=1}^{2} S_J^{+}(\omega; H_{m0,i}, \omega_{p,i}, \gamma_i, N_i, M_i, \alpha_i) \tag{B.4}$$

where S_J^{+} is the JONSWAP spectrum defined by Eq. (B.1). The parameters $H_{m0,\,i}$, $\omega_{p,\,i}$, N_i, M_i, and α_i for $i = 1, 2$, are the significant wave height, angular peak frequency, spectral shape and normalization parameters for the primary and secondary peak, respectively.

These parameters are fitted to 20 000 spectra divided into 146 different classes of H_{m0} and T_p obtained at the Statfjord field in the North Sea in the period from 1980 to 1989. The measured H_{m0} and T_p values for the data range from 0.5 to 11 meters and from 3.5 to 19 seconds, respectively.

Given H_{m0} and T_p these parameters are found by the following steps. The borderline between wind dominated and swell dominated sea states is defined by the fully developed sea, for which

$$T_p = T_f = 6.6\, H_{m0}^{1/3}, \tag{B.5}$$

while for $T_p < T_f$, the local wind sea dominates the spectral peak, and if $T_p > T_f$, the swell peak is dominating.

For each of the three types a non-dimensional period scale is introduced by

$$\varepsilon_{l\,u} = \frac{T_f - T_p}{T_f - T_{lu}},$$

where

$$T_{lu} = \begin{cases} 2\sqrt{H_{m0}} & \text{if } T_p \le T_f \quad \text{(Lower limit)}, \\ 25 & \text{if } T_p > T_f \quad \text{(Upper limit)}, \end{cases}$$

defines the lower or upper value for T_p. The significant wave height for each peak is given as

$$H_{m0,1} = R_{pp}\, H_{m0} \quad H_{m0,2} = \sqrt{1 - R_{pp}^2}\, H_{m0},$$

where

$$R_{pp} = \big(1 - A_{10}\big)\, \exp\Big(-\big(\tfrac{\varepsilon_{l\,u}}{A_1}\big)^2 \Big) + A_{10},$$

$$A_1 = \begin{cases} 0.5 & \text{if } T_p \le T_f, \\ 0.3 & \text{if } T_p > T_f, \end{cases} \qquad A_{10} = \begin{cases} 0.7 & \text{if } T_p \le T_f, \\ 0.6 & \text{if } T_p > T_f. \end{cases}$$

The primary and secondary peak periods are defined as

$$T_{p,1} = T_p,$$
$$T_{p,2} = \begin{cases} T_f + 2 & \text{if } T_p \le T_f, \\ \left(\frac{M_2\, (N_2/M_2)^{(N_2-1)/M_2} / \Gamma((N_2-1)/M_2)}{1.28\, (0.4)^{N_2} \{1 - \exp(-H_{m0,\,2}/3)\}} \right)^{1/(N_2-1)} & \text{if } T_p > T_f, \end{cases}$$

where the spectral shape parameters are given as

$$N_1 = N_2 = 0.5\sqrt{H_{m0}} + 3.2,$$
$$M_i = \begin{cases} 4\left(1 - 0.7\exp\left(\frac{-H_{m0}}{3}\right)\right) & \text{if } T_p > T_f \text{ and } i = 2, \\ 4 & \text{otherwise}. \end{cases}$$

The peakedness parameters are defined as

$$\gamma_1 = 35\left(1 + 3.5\, \exp\big(-H_{m0}\big)\right)\gamma_T, \qquad \gamma_2 = 1,$$

where

$$\gamma_T = \begin{cases} \left(\frac{2\,\pi\, H_{m0,\,1}}{g\, T_p^2}\right)^{0.857} & \text{if } T_p \le T_f, \\ \big(1 + 6\,\varepsilon_{lu}\big)\left(\frac{2\,\pi\, H_{m0}}{g\, T_f^2}\right)^{0.857} & \text{if } T_p > T_f. \end{cases}$$

Finally the normalization parameters α_i $(i = 1, 2)$ are found by numerical integration so that

$$\int_0^\infty S_J^+(\omega; H_{m0,i}, \omega_{p,i}, \gamma_i, N_i, M_i, \alpha_i)\, d\omega = H_{m0,\,i}^2/16.$$

Preliminary comparisons with spectra from other areas indicate that the empirical parameters in the Torsethaugen spectrum can be dependent on geographical location. This spectrum is implemented as a matlab function `torsethaugen.m` in the WAFO toolbox.

B.3 Ochi-Hubble spectrum

Ochi and Hubble [45], suggested to describe bimodal spectra by a superposition of two modified Bretschneider (Pierson-Moskovitz) spectra:

$$S^+(\omega) = \frac{1}{4}\sum_{i=1}^{2} \frac{\left(\left(\lambda_i + 1/4\right)\, \omega_{p,\,i}^4\right)^{\lambda_i}}{\Gamma(\lambda_i)} \frac{H_{m0,\,i}^2}{\omega^{4\,\lambda_i} + 1} \exp\left(\frac{-\left(\lambda_i + 1/4\right)\, \omega_{p,\,i}^4}{\omega^4}\right),$$

where $H_{m0,\,i}$, $\omega_{p,\,i}$, and λ_i for $i = 1, 2$, are significant wave height, angular peak frequency, and spectral shape parameter for the low and high frequency components, respectively.

The values of these parameters are determined from an analysis of data obtained in the North Atlantic. The source of the data is the same as that for the development of the Pierson-Moskowitz spectrum, but analysis is carried out on over 800 spectra including those in partially developed seas and those having a bimodal shape. In contrast to the JONSWAP and Torsethaugen spectra, which are parameterized as function of H_{m0} and T_p, Ochi and Hubble, [45] gave, from a statistical analysis of the data, a family of wave spectra consisting of 11 members generated for a desired sea severity (H_{m0}) with the coefficient of 0.95.

The values of the six parameters as functions of H_{m0} are given as:

$$\begin{aligned}
H_{m0,1} &= R_{p,1}\, H_{m0}, \\
H_{m0,2} &= \sqrt{1 - R_{p,1}^2}\, H_{m0}, \\
\omega_{p,i} &= a_i\, \exp\left(-\, b_i\, H_{m0}\right), \\
\lambda_i &= c_i\, \exp\left(-\, d_i\, H_{m0}\right),
\end{aligned}$$

where $d_1 = 0$ and the remaining empirical constants a_i, b_i $(i = 1, 2)$, and d_2, are given in Table B.1. (See also the function `ochihubble.m` in the WAFO toolbox.)

Member no. 1 given in Table B.1 defines the most probable spectrum, while member no. 2 to 11 define the 0.95 percent confidence spectra.

A significant advantage of using a family of spectra for design of marine systems is that one of the family members yields the largest response such as motions or wave induced forces for a specified sea severity, while anothers yield the smallest response with confidence coefficient of 0.95.

Rodrigues and Soares [51], used the Ochi-Hubble spectrum with 9 different parameterizations representing 3 types of sea state categories: swell dominated (a), wind sea dominated (b) and mixed wind sea and swell system with comparable energy (c). Each category is represented by 3 different inter-modal distances between the swell and the wind sea spectral components. These three subgroups are denoted by I, II and III, respectively. The exact values for the six parameters are given in Table B.2. (See the function `ohspec3.m` in the WAFO toolbox.)

Member no.	$R_{p,1}$	a_1	a_2	b_1	b_2	c_1	c_2	d_2
1	0.84	0.70	1.15	0.046	0.039	3.00	1.54	0.062
2	0.84	0.93	1.50	0.056	0.046	3.00	2.77	0.112
3	0.84	0.41	0.88	0.016	0.026	2.55	1.82	0.089
4	0.84	0.74	1.30	0.052	0.039	2.65	3.90	0.085
5	0.84	0.62	1.03	0.039	0.030	2.60	0.53	0.069
6	0.95	0.70	1.50	0.046	0.046	1.35	2.48	0.102
7	0.65	0.61	0.94	0.039	0.036	4.95	2.48	0.102
8	0.90	0.81	1.60	0.052	0.033	1.80	2.95	0.105
9	0.77	0.54	0.61	0.039	0.000	4.50	1.95	0.082
10	0.73	0.70	0.99	0.046	0.039	6.40	1.78	0.069
11	0.92	0.70	1.37	0.046	0.039	0.70	1.78	0.069

Table B.1: *Empirical parameter values for the Ochi-Hubble spectral model.*

Sea state type	Sea state group	$H_{m0,1}$	$H_{m0,2}$	$\omega_{p,1}$	$\omega_{p,2}$	λ_1	λ_2
a	I	5.5	3.5	0.440	0.691	3.0	6.5
	II	6.5	2.0	0.440	0.942	3.5	4.0
	III	5.5	3.5	0.283	0.974	3.0	6.0
b	I	2.0	6.5	0.440	0.691	3.0	6.5
	II	2.0	6.5	0.440	0.942	4.0	3.5
	III	2.0	6.5	0.283	0.974	2.0	7.0
c	I	4.1	5.0	0.440	0.691	2.1	2.5
	II	4.1	5.0	0.440	0.942	2.1	2.5
	III	4.1	5.0	0.283	0.974	2.1	2.5

Table B.2: *Target spectra parameters for mixed sea states.*

Bibliography

[1] S. Åberg and G. Lindgren. Height distribution of stochastic Lagrange ocean waves. *Probabilistic Engineering Mechanics*, 23(4):359–363, 2008.

[2] S. Åberg. *Applications of Rice's formula in oceanographic and environmental problems*. PhD thesis, Math. Stat., Center for Math. Sci., Lund Univ., Sweden, 2007.

[3] A. Ambartzumian, A. Der Kiureghian, V. Ohanian, and H. Sukiasian. Multinormal probability by sequential conditioned importance sampling: Theory and application. *Prob. Eng. Mech.*, 13(4):299–308, 1998.

[4] J.-M. Azaïs and M. Wschebor. *Level sets and extrema of random processes and fields*. John Wiler and Sons, Hoboken, 2009.

[5] A. Baxevani. *Modelling Sea Surface Dynamics Using Crossing Distributions*. PhD thesis, Math. Stat., Center for Math. Sci., Lund Univ., Sweden, 2004.

[6] A. Baxevani and I. Rychlik. Maxima for Gaussian seas. *Ocean Eng.*, 33:895–911, 2006.

[7] S. Borg. Xs - a statistical program package in splus for extreme-value analysis. Dept. of Mathematical Statistics, Lund University, 1992.

[8] P. Brodtkorb, D. Myrhaug, and H. Rue. Joint distribution of wave height and wave crest velocity from reconstructed data. In *Proc. 9'th Int. Offshore and Polar Eng. Conf., ISOPE, Brest, France*, volume III, pages 66–73, June 1999.

[9] P. Brodtkorb, D. Myrhaug, and H. Rue. Joint distribution of wave height and wave crest velocity from reconstructed data with application to ringing. *Int. J. Offshore and Polar Eng.*, 11(1):23–32, March 2001.

[10] P. A. Brodtkorb. *The Probability of Occurrence of Dangerous Wave Situations at Sea.* PhD thesis, Norwegian Univ. of Sci. and Technology, NTNU, Trondheim, Norway, 2004. Dr. Ing. thesis.

[11] P. A. Brodtkorb. Evaluating nearly singular multinormal expectations with application to wave distributions. *Methodology and Computing in Applied Probability*, 8:65–91, 2006.

[12] E. Buows, H. Gunther, W. Rosenthal, and C. Vincent. Similarity of the wind wave spectrum in finite depth water: 1 spectral form. *J. Geophys. Res.*, 90(C1):975–986, 1985.

[13] A. Cavanié, M. Arhan, and R. Ezraty. A statistical relationship between individual heights and periods of storm waves. In *Proc. 1'st Int. Conf. on Behaviour of Offshore Structures, BOSS, Trondheim, Norway*, pages 354–360, 1976.

[14] S. Coles. *An Introduction to Statistical Modeling of Extreme Values*. Springer-Verlag, London, 2001.

[15] C. R. Dietrich and G. N. Newsam. Fast and exact simulation of stationary Gaussian processes through circulant embedding of the covariance matrix. *SIAM J. Sci. Comp.*, 18(4):1088–1107, July 1997.

[16] M. Frendahl, G. Lindgren, and I. Rychlik. Fatigue analysis toolbox, 1993. Dept. of Math. Stat., Lund Univ., Sweden.

[17] M. Frendahl and I. Rychlik. Rainflow analysis: Markov method. *Int. J. Fatigue*, 15:265–272, 1993.

[18] A. Genz. Numerical computation of multivariate normal probabilities. *J. Comp. Graph. Stat.*, 1:141–149, 1992.

[19] A. Genz and K.-S. Kwong. Numerical evaluation of singular multivariate normal distributions. *J. Stat. Comp. Simul.*, 68:1–21, 2000.

[20] K. Hasselmann, T. Barnett, E. Buows, H. Carlson, D. Carthwright, K. Enke, J. Ewing, H. Gienapp, D. Hasselmann, P. Kruseman, A. Meerburg, A. Müller, D. Olbers, K. Richter, W. Sell, and H. Walden. Measurements of wind-wave growth and swell decay during the Joint North Sea Wave Project. *Deutschen Hydrografischen Zeitschrift*, 12:9–95, 1973.

[21] S. Haver and K. A. Nyhus. A wave climate description for long term response calculations. In *Proc. 5'th Int. Symp. on Offshore Mech. and Arctic Eng., OMAE, Tokyo, Japan*, volume 4, pages 27–34, 1986.

[22] J. Hosking and J. Wallis. Parameter and quantile estimation for the generalized pareto distribution. *Technometrics*, 1987.

[23] J. Hosking, J. Wallis, and E. Wood. Estimation of the generalized extreme-value distribution by the method of probability-weighted moments. *Technometrics*, 1985.

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

[25] P. Johannesson. Rainflow cycles for switching processes with Markov structure. *Prob. Eng. Inform. Sci.*, 12(2):143–175, 1998.

[26] P. Johannesson. *Rainflow Analysis of Switching Markov Loads*. PhD thesis, Math. Stat., Center for Math. Sci., Lund Univ., Sweden, 1999.

[27] H. Krogstad and S. Barstow. Directional distributions in ocean wave spectra. In *Proc. 9'th Int. Offshore and Polar Eng. Conf., ISOPE, Brest, France*, volume III, pages 76–89, June 1999.

[28] H. Krogstad, J. Wolf, S. Thompson, and L. Wyatt. Methods for intercomparison of wave measurements. *Coastal Eng.*, 37:235–257, 1999.

[29] G. Lindgren. Slepian models for the stochastic shape of individual Lagrange sea waves. *Advances in Applied Probability*, 38(2):430–450, 2006.

[30] G. Lindgren. Exact asymmetric slope distributions in stochastic Gauss–Lagrange ocean waves. *Applied Ocean Research*, 31(1):65–73, 2009.

[31] G. Lindgren. Slope distribution in front-back asymmetric stochastic Lagrange time waves. *Advances in Applied Probability*, 42(2):489–508, 2010.

[32] G. Lindgren and S. Åberg. First Order Stochastic Lagrange Model for Asymmetric Ocean Waves. *Journal of Offshore Mechanics and Arctic Engineering*, 131(3):031602–1–031602–8, 2009.

[33] G. Lindgren, D. Bolin, and F. Lindgren. Non-traditional stochastic models for ocean waves. *The European Physical Journal Special Topics*, 185:209–224, 2010.

[34] G. Lindgren and K. Broberg. Cycle range distributions for Gaussian processes exact and approximative results. *Extremes*, 7(1):69, 2004.

[35] G. Lindgren and I. Rychlik. Slepian models and regression approximations in crossing and extreme value theory. *Int. Stat. Rev.*, 59:195–225, 1991.

[36] G. Lindgren, I. Rychlik, and M. Prevosto. The relation between wave length and wave period distributions in random Gaussian waves. *Int. J. Offshore and Polar Eng.*, 8(4):258–264, December 1998.

[37] M. Longuet-Higgins. On the joint distribution wave periods and amplitudes of sea waves. *J. Geophys. Res.*, 80:2688–2694, 1975.

[38] M. Longuet-Higgins. On the joint distribution wave periods and amplitudes in a random wave field. In *Proc. R. Soc.*, volume A389, pages 24–258., 1983.

[39] U. Machado and I. Rychlik. Wave statistics in nonlinear random sea. *Extremes*, 6(6):125–146, 2003.

[40] T. Marthinsen and S. Winterstein. On the skewness of random surface waves. In *Proc. 2'nd Int. Offshore and Polar Eng. Conf., ISOPE, San Francisco, USA*, volume III, pages 472–478, 1992.

[41] S. R. Massel. *Ocean Surface Waves: Their Physics and Prediction*, volume 11 of *Advances on ocean engineering*. World Scientific, 1996.

[42] M. Matsuishi and T. Endo. Fatigue of metals subject to varying stress. Paper presented to Japan Soc. Mech. Engrs, Jukvoka, Japan, 1968.

[43] C. Mercardier. Numerical bounds for the distribution of maxima of some one- and two-parameter gaussian processes. *Adv. Appl. Probab.*, 38:149–170, 2006.

[44] M. Ochi and K. Ahn. Probability distribution applicable to non-Gaussian random processes. *Prob. Eng. Mech.*, 9:255–264, 1994.

[45] M. Ochi and E. Hubble. On six parameter wave spectra. In *Proc. 15'th Int. Conf. on Coastal Eng., ICCE*, volume 1, pages 301–328, 1976.

[46] M. K. Ochi. *Ocean Waves, The Stochastic Approach*. Ocean Tech. Series 6. Campridge University Press, 1998.

[47] J. Pickands. Statistical inference using extreme order statistics. *Annals of Statistics*, 3:119–131, 1975.

[48] K. Podgórski, I. Rychlik, and U. Machado. Exact distributions for apparent waves in irregular seas. *Ocean Eng.*, 27(1):979–1016, 2000.

[49] K. Podgórski, I. Rychlik, J. Rydén, and E. Sjö. How big are the big waves? *Int. J. Offshore and Polar Eng.*, 10:161–169, 2000.

[50] P. Prescott and A. Walden. Maximum likelihood estimation of the parameters of the generalized extreme-value distribution. *Biometrika*, 67:723–724, 1980.

[51] G. Rodriguez and C. Guedes Soares. Wave period distribution in mixed sea states. In *Proc. ETCE/OMAE2000 Joint Conf. Energy for the New millenium, New Orleans, LA*, 2000.

[52] I. Rychlik. A new definition of the rainflow cycle counting method. *Int. J. Fatigue*, 9:119–121, 1987.

[53] I. Rychlik. Regression approximations of wavelength and amplitude distributions. *Adv. Appl. Prob.*, 19:396–430, 1987.

[54] I. Rychlik. Rain-Flow-Cycle distribution for ergodic load processes. *SIAM J. Appl. Math.*, 48:662–679, 1988.

[55] I. Rychlik. Confidence bands for linear regressions. *Comm. Stat. Simul.*, 21(2):333–352, 1992.

[56] I. Rychlik. Simulation of load sequences from rainflow matrices: Markov method. Stat. Research Report 29, Dept. of Mathematical Statistics, Lund, Sweden, 1995.

[57] I. Rychlik, P. Johannesson, and M. R. Leadbetter. Modelling and statistical analysis of ocean-wave data using transformed Gaussian process. *Marine Structures, Design, Construction and Safety*, 10:13–47, 1997.

[58] I. Rychlik and M. R. Leadbetter. Analysis of ocean waves by crossing- and oscillation-intensities. In *Proc. 7'th Int. Offshore and Polar Eng. Conf., ISOPE, Honolulu, USA*, 1997.

[59] I. Rychlik and G. Lindgren. CROSSREG – a technique for first passage and wave density analysis. *Prob. Eng. Inform. Sci.*, 7:125–148, 1993.

[60] I. Rychlik and G. Lindgren. Wave analysis toolbox - a tutorial. http://www.maths.lth.se/matstat/staff/georg/wat/watinfo.html, 1995. Dept. of Math. Stat., Lund Univ., Sweden.

[61] I. Rychlik, G. Lindgren, and Y. Lin. Markov based correlations of damage cycles in Gaussian and non-Gaussian loads. *Probabilistic Engineering Mechanics*, 10(2):103–115, 1995.

[62] A. Schuster. On the investigation of hidden periodicities with application to a supposed 26 day period of meteorological phenomena. *Terrestrial Magnetism and Atmospheric Electricity*, 3:13–41, 1898.

[63] B. W. Silverman. *Density Estimation for Statistics and Data Analysis.* Number 26 in Monographs on Stat. and Appl. Prob. Chapman and Hall, 1986.

[64] E. Sjö. *Crossings and Maxima in Gaussian Fields and Seas.* PhD thesis, Math. Stat., Center for Math. Sci., Lund Univ., Sweden, 2000. ISBN91-628-4299-4.

[65] M. A. Tayfun. Breaking limited wave heights. *J. Waterway, Port, Coastal and Ocean Div., ASCE*, 107(2):59–69, May 1981.

[66] M. A. Tayfun. Distribution of large wave heights. *J. Waterway, Port, Coastal and Ocean Eng., ASCE*, 116(6):686–707, November/December 1990.

[67] K. Torsethaugen. A two peak wave spectral model. In *Proc. 12'th Int. Conf. on Offshore Mech. and Arctic Eng., OMAE, Glasgow, Scotland*, 1993.

[68] K. Torsethaugen. Model for doubly peaked wave spectrum. Lifetime and fatigue strength estimation implications. Int. Workshop on Floating structures in Coastal zone, Hiroshima, November 1994.

[69] K. Torsethaugen. Model for doubly peaked wave spectrum. Technical Report No. STF22 A96204, SINTEF Civ. and Env. Eng., Trondheim, Norway, 1996.

[70] K. Torsethaugen et al. Characteristica for extreme sea states on the Norwegian continental shelf. Technical Report No. STF60 A84123, Norwegian Hydrodyn. Lab., Trondheim, 1984.

[71] M. Tucker. Recommended standard for wave data sampling and near-real-time processing. *Ocean Eng.*, 20(5):459–474, 1993.

[72] M. Wand and M. Jones. *Kernel Smoothing.* Number 60 in Monographs on Stat. and Appl. Prob. Chapman and Hall, 1995.

[73] S. Winterstein. Nonlinear vibration models for extremes and fatigue. *J. Eng. Mech., ASCE*, 114(10):1772–1790, 1988.

[74] I. Young. Wind generated ocean waves. *Elsevier Ocean Engineering Book Series*, 2:239, 1999.

Index of m-files

Index