Random Loads with Data Analysis
— Computer Exercises —

Pär Johannesson
Fraunhofer-Chalmers Centre

Igor Rychlik
Mathematical Statistics
Chalmers University of Technology

Version 2005-04-14

This short course is based on the course "Load and Fatigue Analysis" developed by Pär Johannesson, Igor Rychlik, Georg Lindgren, and Jesper Rydén at Mathematical Statistics, Lund Institute of Technology, where it was given in January 2000. The goal of the the course is to demonstrate the use of the Matlab toolbox WAFO, Wave Analysis for Fatigue and Oceanography (`www.maths.lth.se/matstat/wafo/`), in the areas of fatigue analysis and modelling of random loads.

The course took place at Fraunhofer-Chalmers Centre at Chalmers Science Park with lectures on Wednesdays 15-17 and computer exercises on Thursdays 10-12 March 2 and 3, April 6, 7, 13 and 14, 2005.

0. **General Introduction.** Introducing some basic concepts used in the following analysis.
1. **Analysis of Load Data.** Analyse a load signal by means of turning points, rainflow filter, level crossings, irregularity, rainflow cycles, load spectrum, Palmgren-Miner damage, upper and lower bounds for rainflow damage. Fatigue life evaluation for constant and variable amplitude loads.
2. **Markov modelling of loads.** A Markov chain of turning points is used for the modelling. The limiting rainflow matrix can be computed, and sample paths simulated. The concept of switching loads is combined with the Markov model, and decomposition of a mixed rainflow matrix is treated.
3. **Spectral modelling of loads.** A load is specified by its power spectrum. Its limiting rainflow matrix is computed through Markov approximation. Further, the expected damage can be computed using the narrow band approximation, or accurate numerical approximations.
4. **Analysis of Measured Loads.** Analysis of a measured switching load. Estimation from measured time signal and from observed rainflow matrix. Decomposition of the mixed rainflow matrix into stationary blocks.

Computers

The students are supposed to bring their own laptops equipped with Matlab and WAFO (Download: `www.maths.lth.se/matstat/wafo/`).

Starting Matlab with WAFO

In order to access the routines needed for the exercises and start Matlab, and add the path to the WAFO-root directory. At the Matlab prompt, type

```
>> initwafo('full')  % Initiate WAFO toolbox
>> itmlab            % Initiate Course files
```

Use the command `help` whenever you need, e.g.

```
>> help
>> help wafo       % Contents of WAFO toolbox
>> help initwafo   % Help about initwafo
>> help fatigue    % Help about the fatigue part of WAFO
>> help dat2tp     % Help about routine dat2tp
>> help tp2rfc     % Help about routine tp2rfc
```

General Introduction

This course is intended to present some tools for analysis of (random) loads in order to assess the fatigue damage. Throughout the course we will use the Matlab toolbox WAFO (Wave Analysis for Fatigue and Oceanography). 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 $x(t)$, $0 \le t \le T$, where t is time and T is the duration of the measurements.
2. In the frequency domain (that is important in system analysis) as a power spectrum. This means that the signal is represented by a Fourier series
$$x(t) \approx m + \sum_{i=1}^{N} a_i \cos(\omega_i\, t) + b_i \sin(\omega_i\, t)$$
where $\omega_i = i \cdot 2\pi/T$ are angular frequencies, m is the mean of the signal and a_i, b_i are Fourier coefficients.
3. In the rainflow domain, i.e. the measured load is given in the form of a rainflow matrix.

We shall now review some simple means of characterizing and analysing loads that are given in forms 1)–3), and how to derive some characteristics, important for fatigue evaluation and testing. More details will also be given in exercises.

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.

Parameters for Measured Load Histories

Some general properties of measured loads can be summarized by using a few simple characteristics. Those are *the mean* m, defined as the average of all values, which is approximately equal to $m = 1/T \int_0^T x(t)\,dt$, and *the variance* σ^2 that measures the variability around the mean and is defined as $\sigma^2 = 1/T \int_0^T (x(t) - m)^2\,dt$, *the mean frequency* f_0 defined as the number of times $x(t)$ crosses upwards (upcrosses) the mean m normalized by the length of the observation interval T, and *the irregularity factor* α, defined as the intensity of mean upcrossings f_0 divided by the intensity of local maxima (intensity of cycles) in $x(t)$. (Note, a small α means an irregular process, $(0 < \alpha \le 1)$.) Another important property is the crossing spectrum $\mu(u)$ defined as the intensity of upcrossings of a level u by $x(t)$ as a function of u. Obviously $f_0 = \mu(m)$.

The process of damage accumulation depends only on the values and the order of the local extremes in the load. The sequence of local extremes is called the *sequence of turning points*. The irregularity factor α is measuring how dense the local extremes are relatively to the mean frequency f_0. For a regular function it 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. A particularly useful filter is the so-called rainflow filter that removes all local extremes that builds rainflow cycles with amplitude smaller than a given threshold. We shall always assume that the signals are rainflow filtered.

Fatigue Life Prediction

Obviously when the signal is given, the rainflow cycles can be extracted and fatigue damage analysis performed. However, often the observed function is too short to contain all possible cycles that a structure can experience and there is a need to model the damage when $x(t)$ is modelled as a possible outcome of a "random" measurement or, more precisely, as a random process, denoted in the following by $X(t)$. The main objective is then to predict the fatigue life from the specification of a random load $X(t)$. This problem is resolved on several levels of complexity.

First we shall use the fact that the crossing intensity can be used to give a conservative estimate (overestimation) of the accumulated damage caused by $X(t)$, see Rychlik [9] for algorithm and more detailed discussion. Now the crossing intensity can be computed using the so-called Rice's formula. Another possibility is to include the intensity in the model specification as is done for the so-called transformed Gaussian loads.

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

- the Markov models constitute a broad class of processes that can accurately model many real loads
- for Markov models, the fatigue damage prediction using rainflow method is particularly simple, Rychlik [8] and Johannesson [4]

In the simplest case, the necessary information is the intensity of pairs of local maxima and the following minima (the so-called Markov matrix or min-max matrix). The dependence between other extremes is modelled using Markov chains, see Frendahl & Rychlik [1].

Frequency Modelling of Load Histories

The important characteristic of signals in frequency domain is their power spectrum $\hat{s}_i = (a_i^2 + b_i^2)/(2\Delta\omega)$, where $\Delta\omega$ is the sampling interval in frequency domain, i.e. $\omega_i = i \cdot \Delta\omega$. The two-column matrix $\hat{s}(\omega_i) = (\omega_i, \hat{s}_i)$ will be called the power spectrum of $x(t)$.

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

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

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

As we have written before, 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 λ_0. Similarly higher-order spectral moments are defined by

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

Random Functions in Spectral Domain

Assume that we get new measurements of a signal that one is willing to consider as equivalent, but it is seldom identical to the first one. Obviously it will have a different spectrum $\hat{s}(\omega)$ and the phases will be changed. A useful mathematical model for such a situation are the so-called random functions (stochastic processes) which will be denoted by $X(t)$. Here $x(t)$ is seen as particular randomly chosen function. The simplest case that models stationary signals with a fixed spectrum $\hat{s}(\omega)$ is

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

where Θ_i are independent uniformly distributed phases. However, it is not a very realistic model, since in practice we often observe variability in spectrum $\hat{s}(\omega)$ between measured functions and hence $\hat{s}_i$ should be modelled as random variables too. Here we assume that there is a deterministic function $S(\omega)$ such that the average value of $\hat{s}(\omega_i)\Delta\omega$ can be approximated by $S(\omega_i)\Delta\omega$ and in many cases one can model $\hat{s}_i = R_i^2 \cdot S(\omega_i)/2$ where R_i are independent random factors, all Rayleigh distributed. (Observe that the average value of R_i^2 is 2.) This gives the following random function

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

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

If $Y(t)$ is an output of a linear filter with $X(t)$ on the input, then $Y(t)$ is also normally distributed and we need to derive the spectrum of $Y(t)$ to be able to analyse its properties. This is a simple task, since if the transfer function of the filter $H(\omega)$ is given, then the spectrum of $Y(t)$, denoted by S_Y, is given by $S_Y(\omega) = |H(\omega)|^2 S(\omega)$. For example, the derivative $X'(t)$ is a Gaussian process with mean zero and spectrum $S_Y(\omega) = \omega^2 S(\omega)$. The variance of the derivative is $\sigma^2_{X'} = \int S_Y(\omega)\, d\omega = \lambda_2$.

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

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

Using spectral moments we have that $\sigma^2 = \lambda_0$ while $f_0 = \frac{1}{2\pi}\sqrt{\frac{\lambda_2}{\lambda_0}}$.

Another approach is to model the turning points of a Gaussian process by a Markov chain, where the so-called Markov matrix is computed from the specified spectrum $S(\omega)$. Then calculation of rainflow matrices and fatigue damages are possible. This approach requires a considerable amount of computation, but often renders accurate results, see Rychlik [8] and Rychlik et al. [10] for extension to transformed Gaussian processes.

Fatigue Life Prediction – Rainflow Method

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 constants, and counts the number of cycles (periods) until it breaks. The number of load cycles $N(s)$ as well as the amplitudes s are recorded. Note that for small amplitudes, $s < s_\infty$, $N(s) \approx \infty$, i.e. no damage is observed. The amplitude s_∞ is called *the fatigue limit* or *the endurance limit*. In practice, one often uses a simple model for $N(s)$,

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

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

For irregular loads, also called variable amplitude loads, one is often combining the S-N curve with a cycle counting method by means of the Palmgren-Miner linear damage accumulation theory, to predict fatigue failure time. The cycle counting forms equivalent load cycles. The now commonly used cycle counting method is rainflow counting and was introduced by Endo [6] in 1968. It was designed to catch both slow and rapid variations of the load by forming cycles by pairing high maxima with low minima even if they are separated by intermediate extremes. More precisely, each local maximum is a top of a hysteresis loop with an amplitude that is computed using rainflow algorithm. The definition of rainflow cycles as illustrated in Figure 1 is due to Rychlik [7].

Let t_k be the time of the k:th local maximum and s_k the amplitude of the attached hysteresis loop. Define the total damage by

$$D(t) = \sum_{t_k \le t} \frac{1}{N(s_k)} = K \sum_{t_k \le t} s_k^\beta = K D_\beta(t) \tag{2}$$

where the sum contains all cycles up to time t. The fatigue life time T^f, say, is shorter than t if $D(t) > 1$. In other words, T^f is defined as the time when $D(t)$ crosses level 1. A very

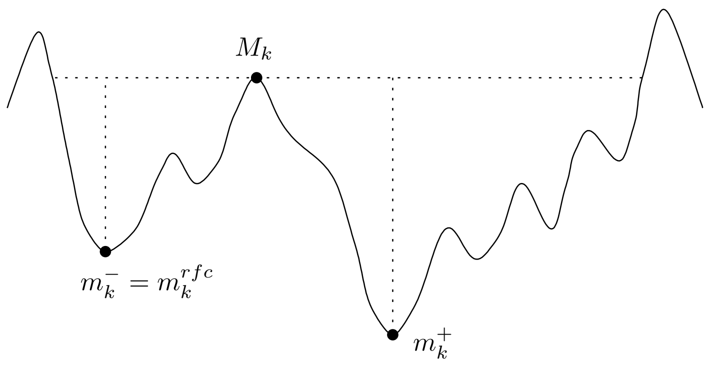

Figure 1: *Definition of the rainflow cycle, as given by Rychlik [7]. From each local maximum M_k one shall try to reach above the same level, in the backward(left) and forward(right) directions, with an as small downward excursion as possible. The minimum, of m_k^- and m_k^+, which represents the smallest deviation from the maximum M_k is defined as the corresponding rainflow minimum m_k^{rfc}. The k:th rainflow cycle is defined as $(m_k^{\mathrm{rfc}}, M_k)$.*

simple predictor of T^f is obtained by replacing K in Eq. (2) by a constant, for example the median value of K equal to γ. For high cycle fatigue, the time to failure is long (more than $10^5/f_0$). Then for stationary (and ergodic and some other mild assumptions) loads, the damage $D_\beta(t)$ can be approximated by its mean $E[D_\beta(t)] = d_\beta \cdot t$. Here d_β is the damage intensity, i.e. how much damage is accumulated per time unit. This leads to a very simple predictor of fatigue life time

$$\hat{T}^f = \frac{1}{\gamma d_\beta}. \tag{3}$$

Switching Loads – Rainflow Matrices

Often the real measurements are gathered in the forms of rainflow matrices. In the same time there is a need of modelling real loads that leads to the observed rainflow matrix. In particular the load can be built up by blocks of stationary load conditions that switch between each other. The rainflow matrix is then a nonlinear mixture of the rainflow matrices for the stationary models and for switching between them. The objective is to model the real loads, see Johannesson [4] for detailed presentation.

When studying switching loads one has to model both the switching between the subloads and the characteristics of the different subloads. We will use a hidden Markov model (HMM) to describe the switching load. This means that the switching is controlled by a Markov chain, called the regime process, which can not be observed and therefore is called hidden. Only the switching load process can be observed, see e.g. Figure 2.1. The regime process is defined by the conditional probabilities of switching between the different regime states. This determines the mean length of each subload and the proportion of the different subloads. The length of a subload is geometrically distributed ($\sim$ exponential). The subloads are modelled by min-Max (and Max-min) matrices, see Figure 2. This means that the sequence of local extremes (also called turning points) are discretized to fixed levels (often 64 or 128 levels in practice). The transitions from a local extreme to the next local extreme are approximated by a Markov chain. This is a 1-step Markov approximation, as the distribution of the next turning point only depends on the current turning point and not on the whole history of turning points. For a thorough description of the models and the algorithms see Johannesson [4, 3]. A summary without any mathematical details is found in Johannesson et al. [5].

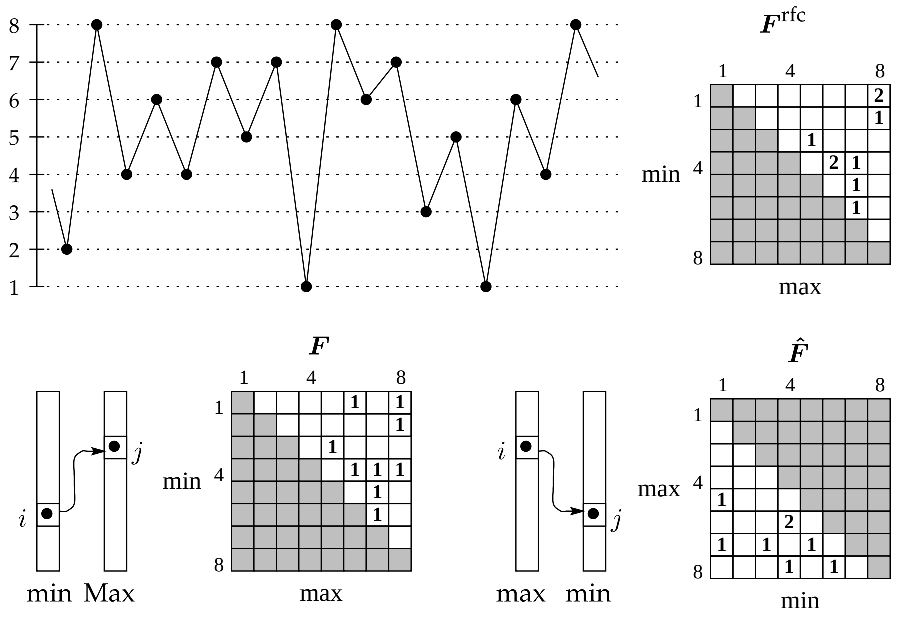

Figure 2: *Part of a discrete load process where the turning points are marked with •. The scale to the left is the discrete levels. The transitions from minimum to maximum and the transitions from maximum to minimum are collected in the min-max matrix, $\boldsymbol{F}$ and max-min matrix, $\hat{\boldsymbol{F}}$, respectively. The rainflow cycles are collected in the rainflow matrix, $\boldsymbol{F}^{\text{rfc}}$. The figures are the number of observed cycles and the grey areas are by definition always zero.*

Computer Exercise 1

Analysis of Load Data

1.1 Measured data

Here we will consider a measured wave load from deep water. The load signal $x(t)$ is the level of the sea-surface (measured in meters) at a fixed point. A measurement of $x(t)$ is saved in the file `deep.dat`. The first column contains the time and the second values of the load. Plot the whole load and zoom in the first 1000 values

```
>> load deep.dat
>> x = deep;
>> plot(x(:,1),x(:,2))
>> plot(x(1:1000,1),x(1:1000,2))
```

The duration of the measurements in seconds, T, is computed by

```
>> T=x(end,1)-x(1,1);
```

You can view the variables in the workspace by typing

```
>> whos
```

Estimate the mean and the standard deviation of $X(t)$. (Hint: Use `mean` and `std`. Start with `help mean`.)

$$m = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, \quad \sigma = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.$$

When analyzing load data only the sequence of turning points (i.e. the sequence of local extremes) is of interest and not the exact path between the local extremes. Obtain the turning points for `deep` and compare with the original time signal:

```
>> tp = dat2tp(x);
>> plot(x(:,1),x(:,2),tp(:,1),tp(:,2),'.-')
>> axis([0 100 -20 20])
```

To store the turning points instead of the original time signal is a good way to compress load data. When analyzing the power spectrum of the load, one needs the whole time signal, but when analyzing the level crossings and the rainflow cycles, then the turning points yield sufficient information.

It is also possible to apply a rainflow filter (also called hysteresis filter), which removes small oscillations from the signal. All rainflow cycles with amplitudes below the threshold `h` are removed.

```
>> tp1 = dat2tp(x,1);
>> plot(x(:,1),x(:,2),tp(:,1),tp(:,2),tp1(:,1),tp1(:,2))
>> axis([0 100 -20 20])
```

Next we shall compute the mean frequency f_0 and the irregularity factor α, see Introduction for definitions.

First we compute a two column matrix with levels and number of upcrossings of these levels. Then the crossings will be divided by the time duration T in order to get the intensity of crossings; "how many per time unit (second)".

```
>> lc = tp2lc(tp);
>> lc(:,2)=lc(:,2)/T;
>> plot(lc(:,1),lc(:,2))
>> semilogx(lc(:,2),lc(:,1))
```

In order to obtain the mean frequency f_0 we will use the Matlab function `interp1`. Type `help interp1` to read about the routine. (You are recommended to use `help` whenever a new function or routine is introduced.)

```
>> m=mean(x(:,2));
>> f0 = interp1(lc(:,1),lc(:,2),m,'linear');
>> f0
```

Finally we compute the irregularity factor α. The intensity of local maxima is equal to the number of local extremes in the sequence of turning points divided by $2T$, so the parameter α can be computed by

```
>> extr0=length(tp)/2/T;
>> alfa=f0/extr0
```

1.2 Gaussian process as a model for the deep water data

Wave data for deep water is often modelled as a Gaussian process, see Introduction for definitions and simple properties. The most important notion is the pdf function[1] for the

[1] **p**robability **d**ensity **f**unction

normal distribution with mean m and standard deviation σ computed in the previous section.

Using normal probability paper, we can check the agreement between data and the assumed, normal model.

```
>> wnormplot(x(:,2))
```

Although the pdf function is important in fatigue analysis it is more important that the crossing intensity derived from the model is in agreement with the one observed from the signals, see Computer Exercise 3 for more detailed discussion.

We shall use Rice's formula, given in the Introduction, to compute the theoretical crossing intensity for Gaussian processes. It contains the two spectral moments λ_0 and λ_2 and in order to compute them we need to estimate the spectrum of the load x. Estimate the spectral density of the deep water data

```
>> S = dat2spec(deep);
>> wspecplot(S);
```

The spectral density S is saved as a Matlab structure containing some additional information; to observe the structure and plot the estimated density, just execute

```
>> S
>> plot(S.w,S.S)
```

The spectral moments can be computed from the estimated spectral density by means of numerical integration by using `spec2mom`.

```
>> lam = spec2mom(S,4); L0=lam(1); L2=lam(2); L4=lam(3);
```

The variables `L0`, `L2`, and `L4` contain the spectral moments λ_0, λ_2, and λ_4, respectively. Now we can compare the intensity of level crossings from Rice's formula with the observed number of level crossings. First we compute the mean frequency f_0, then the crossing intensity function $\mu(u)$. (Note that we assume that the mean of signal m is zero.)

```
>> f0=1/(2*pi)*sqrt(L2/L0)
>> ux = -20:0.1:20;
>> ricex = f0*exp(-ux.*ux./(2*L0));
>> plot(lc(:,1),lc(:,2),'-',ux,ricex,'--')
>> semilogx(lc(:,2),lc(:,1),'-',ricex,ux,'--')
```

1.3 Rainflow Cycles

Recall the definition of rainflow and min-max cycle counts. The demo program `democc` illustrates these definitions. Use it to identify the first few rainflow and min-max cycles in `x`.

```
>> proc = x(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 (with the buttons RFC,TP). The cycles are visualized in the other window.

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

```
>> RFC = tp2rfc(tp);
>> mM = tp2mm(tp);
```

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

```
>> subplot(1,2,1), ccplot(RFC)
>> subplot(1,2,2), ccplot(mM)
```

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

```
>> ampRFC = cc2amp(RFC);
>> ampmM = cc2amp(mM);
>> subplot(1,2,1), hist(ampRFC)
>> subplot(1,2,2), hist(ampmM)
```

1.3.1 Turning Points & Rainflow Filter

Which threshold ranges are appropriate for our signal? A rule of thumb is about 10% of the total range. Try some thresholds, and compare the results. How large reduction do we obtain? How much damage is kept in the signal?

```
>> h1=2; h2=5; h3=10;            % Threshold ranges for the rainflow filter
>> tp_0 = dat2tp(x);               % No rainflow filter
>> tp_1 = dat2tp(x,h1);            % Rainflow filter, h1
>> tp_2 = dat2tp(x,h2);            % Rainflow filter, h2
>> tp_3 = dat2tp(x,h3);            % Rainflow filter, h3

>> whos   % How large reduction in number of cycles?

% How much damage do we loose?
>> beta = 5; % Define a damage exponent
>> dam_0 = cc2dam(tp2rfc(tp_0,'CS'),beta); dam_1 = cc2dam(tp2rfc(tp_1,'CS'),beta);
>> cc2dam(tp2rfc(tp_1,'CS'),beta); dam_2 = cc2dam(tp2rfc(tp_2,'CS'),beta);
>> cc2dam(tp2rfc(tp_2,'CS'),beta); dam_3 = cc2dam(tp2rfc(tp_3,'CS'),beta);
>> [dam_1 dam_2 dam_3]              % Damage
>> [dam_1 dam_2 dam_3]/dam_0        % Relative damage
```

Questions:

- How large reduction in number of cycles did you obtain?
- How much of the damage was kept?
- Which threshold would you like to chose?

Choose a threshold value.

```
>> h = ... your choice ...

>> tp = dat2tp(x,h);                % Rainflow filter
>> rfc = tp2rfc(tp,'CS');           % Rainflow cycles
>> dam = cc2dam(rfc,beta);          % Damage

>> dam/dam_0                        % Relative damage
>> length(tp_0)                     % Number of turning points
>> length(tp)                       % Number of TP after rainflow filter
>> length(tp_0)/length(tp)          % Relative length

>> plot(x(:,1),x(:,2),tp(:,1),tp(:,2)) % Compare signals before/after rainflow
```

1.3.2 Rainflow Matrix

There are different ways of plotting the rainflow matrix.

```
>> n = 64;                          % Number of discrete levels
>> [RFM,u,param] = dat2rfm(tp,h,n);% Rainflow matrisx

% Draw the rainflow matrix in Min-Max-format
>> cmatplot(u,u,RFM,3), colorbar

% Draw the rainflow matrix in Mean-Amplitude-format
>> [RFMrm,paramM,paramR,paramA] = cmat2rmcmat(RFM,param);
>> ua=levels(paramA); um=levels(paramM); cmatplot(um,ua,RFMrm',3), colorbar
>> xlabel('Mean'), ylabel('Amplitude')

% It is also possible to define the discretization levels directly
>> param = [-150 150 100]; % Define discretization
>> [RFM,u,param] = dat2rfm(tp,h,param);

% Draw the rainflow matrix in Min-Max-format
>> cmatplot(u,u,RFM,3), colorbar
```

From the rainflow matrix the level crossings can be obtained. How is the level crossings calculated from the rainflow matrix?

```
>> lc = cmat2lc(param,RFM);        % Calculate the level crossing spectrum
% Plot the load spectrum in different ways
>> figure(2),
>> plot(lc(:,1),lc(:,2))           % Frequency function
>> semilogy(lc(:,1),lc(:,2))       % Frequency function (log-scale)
>> semilogx(lc(:,2),lc(:,1))       % The fatigue way of plotting
```

The rainflow amplitude histogram can obtained from the rainflow matrix. How?

```
>> amp = cmat2amp(param,RFM);      % Calculate the amplitude histogram
>> figure(3)
>> plot(amp(:,1),amp(:,2));        % Plot the frequency function
>> semilogy(amp(:,1),amp(:,2),'*');% Frequency function in log-scale
```

The load spectrum is the most common way to present the rainflow amplitudes, where the cumulative number of cycles above a certain amplitude is plotted versus the amplitude, i.e. the load spectrum is the survival function.

```
>> lsplot(amp);                    % Cumulative number of cycles
>> lsplot(amp,0,0);                % Histogram of the number of cycles
>> lsplot(amp,0,0,beta);           % Damage histogram
```

1.3.3 Calculation of damage intensity

In the section with optional exercises one can estimate parameters in the S-N curve. The estimated parameters are: $\gamma = 5.5 \cdot 10^{-10}$, $\beta = 3.2$, and $\sigma_K^2 = 0.06$. These numerical values will be used in the examples below. For our load x the intensity is estimated as follows

```
>> beta=3.2; gam=5.5E-10;
>> d_beta=cc2dam(RFC,beta)/T;
>> time_fail=1/gam/d_beta/3600 %in hours of the specific storm
```

1.4 Additional exercises, Optional

In the following exercises we shall use a slightly different parameterization of the S-N curve than given in Introduction, viz.

$$N(s) = \begin{cases} K^{-1}\epsilon^{-1}s^{-\beta} & s > s_\infty, \\ \infty & s \leq s_\infty, \end{cases} \tag{1.1}$$

where K is a lognormally distributed random factor, i.e. $\ln(K) \in \mathrm{N}(0, \sigma_K^2)$, and ϵ, β are fixed material dependent constants.

1.4.1 Estimation of S-N curve

Taking the logarithm of Eq. (1.1) and assuming that $\ln(K) \in \mathrm{N}(0, \sigma_K^2)$ we obtain

$$\ln(N(s)) = -\ln(K) - \ln(\epsilon) - \beta \ln(s) \in \mathrm{N}(-\ln(\epsilon) - \beta \ln(s), \sigma_K^2), \tag{1.2}$$

for every fixed $s > s_\infty$, ϵ and β.

Let T be the fatigue life time. Since the frequency of the load oscillation ω is constant we have

$$\mathbf{P}[\,T \le t\,] = \mathbf{P}[\,N(s) \le \tfrac{\omega}{2\pi} t\,] = \mathbf{P}[\,K \le \epsilon s^\beta \tfrac{\omega}{2\pi} t\,],$$

where $\frac{\omega}{2\pi} t$ is the number of cycles in the interval $[0, t]$.

In the following exercises we shall estimate the parameters in the model (1.1).

Load the SN-data by typing

```
>> load SN
```

There are two variables `s` and `N` representing s and $N(s)$. Plot $N(s)$ against s by

```
>> plot(N,s,'o')
>> axis([0 14e5 5 35 ])
>> loglog(N,s,'o')
```

In the following we assume that $s_\infty = 0$.

The plotted data consist of 5 groups at s = 10, 15, 20, 25 and 30 MPa. Each group has 8 observations of $N(s)$ making a total of 40 observations. Assume that the observations are independent, that the model (1.1) holds and K is a lognormal variable.

1. Propose an estimation procedure for ϵ, β and σ_K^2.
2. Check the applicability of (1.1) by using normal probability paper,

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

3. Use `snplot` to get estimates of ϵ, β and σ_K^2. (Try `help snplot`.)

   $$\epsilon \approx \underline{\hspace{2cm}}, \quad \beta \approx \underline{\hspace{2cm}}, \quad \sigma_K^2 \approx \underline{\hspace{2cm}}.$$

   Solution:

```
>> [e0,beta0,s20] = snplot(s,N,12)
>> [e0,beta0,s20] = snplot(s,N,14)
e0    = 5.5361e-10
beta0 = 3.2286
s20   = 0.0604
```

1.4.2 Calculation of the 95% quantile for the fatigue life time

Estimate $t_{0.95}$ defined by

$$\mathbf{P}[T > t_{0.95}] = 0.95,$$

for $s = 22$ MPa and $\omega = 10 \cdot 2\pi$.

Solution: We want to solve the equation

$$\alpha = \mathbf{P}[T > t_\alpha] = 1 - \mathbf{P}[T \le t_\alpha].$$

Since

$$\{T \le t_\alpha\} \quad \Leftrightarrow \quad \{N(s) \le \tfrac{\omega}{2\pi} t_\alpha\}$$

we have

$$\begin{aligned}\alpha &= 1 - \mathbf{P}[T \le t_\alpha] = 1 - \mathbf{P}[N(s) \le \tfrac{\omega}{2\pi} t_\alpha] = 1 - \mathbf{P}[\tfrac{K}{\epsilon s^\beta} \le \tfrac{\omega}{2\pi} t_\alpha] \\ &= 1 - \mathbf{P}[K \le \epsilon s^\beta \tfrac{\omega}{2\pi} t_\alpha] = 1 - \Phi\left(\frac{\ln(\epsilon s^\beta \omega t_\alpha/(2\pi))}{\sigma_K}\right)\end{aligned}$$

which gives

$$\frac{\ln(\frac{\epsilon s^\beta \omega t_\alpha}{2\pi})}{\sigma_K} = \lambda_\alpha \quad \Leftrightarrow \quad t_\alpha = \frac{2\pi \exp(\lambda_\alpha \sigma_K)}{\epsilon s^\beta \omega}$$

where λ_α is the α-quantile of N(0,1), i.e. $\mathbf{P}(X > \lambda_\alpha) = \alpha$ where $X \in$ N(0,1).

1.4.3 Fatigue life distribution under variable random load

Compare the total damage caused by rainflow cycles for loads `L1` and `L2`.

```
>> D0 = e0*cumsum((RFC(:,2)-RFC(:,1)).^beta0);
>> plot(D0)
```

Let T^f be the fatigue failure time. The failure time distribution is computed as follows

$$\mathbf{P}[T^f \le t] = \mathbf{P}[D(t) \ge 1] = \mathbf{P}[K \le \epsilon D_\beta(t)],$$

where $D_\beta(t)$ is defined by (2). For loads with short memory the damage $D_\beta(t)$ is asymptotically Gaussian, i.e.

$$D_\beta(t) \approx \mathrm{N}(d_\beta t, \sigma_\beta^2 t), \qquad \text{for large values of } t.$$

where d_β is called *the damage intensity*

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

Since ϵ is small and $\ln K \in \mathrm{N}(0,\sigma^2)$

$$F_{T^f}(t) = \mathbf{P}[\,T^f \le t\,] = \mathbf{P}[\,D(t) \ge 1\,] = \mathbf{P}[\,K \le \epsilon D_\beta(t)\,] \approx$$

$$\approx \mathbf{P}\left[\,K \le \epsilon\left(d_\beta t + \sigma_\beta \sqrt{t} Z\right)\,\right] = \int_{-\infty}^{\infty} \mathbf{P}[\,K \le \epsilon(d_\beta t + \sigma_\beta\sqrt{t} z)\,]\,\phi(z)\,\mathrm{d}z =$$

$$= \int_{-\infty}^{\infty} \Phi\left(\frac{\ln\epsilon + \ln d_\beta t + \ln(1 + \frac{\sigma_\beta}{d_\beta}\frac{1}{\sqrt{t}} z)}{\sigma}\right)\phi(z)\,\mathrm{d}z, \qquad (1.3)$$

where we used that $D_\beta(t) \approx d_\beta t + \sigma_\beta\sqrt{t}Z$, $Z \in \mathrm{N}(0,1)$. If the cycle count $\{(x,y)_{t_i}\}$ is given, then the damage intensity d_β is estimated by using the function `cc2dam`.

Estimate the damage intensity, d_β, (as a function of parameter β) due to the rainflow count in load `x`.

```
>> beta = 3:0.1:8;
>> DRFC = cc2dam(RFC,beta);
>> dRFC = DRFC/T
>> plot(beta,dRFC)
```

Recall that for the S-N data we have $\epsilon = 5.5 \cdot 10^{-10}$, $\beta = 3.2$ and $\sigma^2 = 0.06$. Further we have estimated $\sigma_\beta^2 = 0.5$. Estimate the failure distribution, using formula (1.3) implemented in the function `ftf`

```
>> help ftf
>> [t0,F0] = ftf(e0,cc2dam(RFC,beta0)/T,s20,0.5,1);
```

Check the influence of the parameter σ_β^2 on the T^f-distribution by putting $\sigma_\beta^2 = 0$ and $\sigma_\beta^2 = 5$, respectively.

```
>> [t1,F1] = ftf(e0,cc2dam(RFC,beta0)/T,s20,0,1);
>> [t2,F2] = ftf(e0,cc2dam(RFC,beta0)/T,s20,5,1);
>> plot(t0,F0,t1,F1,t2,F2)
```

Since ϵ is small, σ_β^2 has little influence on the T^f-distribution and can be omitted, i.e. $\sigma_\beta^2 = 0$.

Under the assumption that $\sigma_\beta^2 = 0$ compute the t_α quantile, i.e.

$$\mathbf{P}[\,T^f > t_\alpha\,] = \alpha.$$

Solution: $t_\alpha = \epsilon^{-1} d_\beta^{-1} \mathrm{e}^{\lambda_\alpha \sigma}$ where λ_α is the α-quantile of $\mathrm{N}(0,1)$-distribution.

Plot $t_{0.99}$ for $3 \le \beta \le 8$ and rainflow count `RFC`, and min-max count `mM`.

```
>> taRFC = exp(-1.96*sqrt(0.06))/e0./dRFC;
>> DmM = cc2dam(mM,beta);
>> dmM = DmM/T
>> tamM = exp(-1.96*sqrt(0.06))/e0./dmM;
>> plot(beta,taRFC,beta,tamM,'r')
```

1.4.4 Crack growth data

In some applications the degradation of material is defined as the length of a crack. The strength of a material is assumed to be zero when the length of the crack reaches a critical level a_{crt}. In laboratory experiments one is subjecting a specimen to a constant amplitude load. The length of a crack as a function of the number of periods is recorded. A set of crack length data with very high accuracy of measurement was presented by Virkler et al. [11] in 1979. We shall briefly analyse this data set.

Load Virkler data by

```
>> clear
>> load virkler
```

The material is saved as a 164 $\times$ 69 matrix with the crack length in the first column and the number of the cycles for the 68 specimens in the following columns. Plot the first column against the second by

```
>> plot(v(:,2),v(:,1))
```

The figure shows typical non-linear character of crack growth phenomena. Plot all 68 data series on one plot by

```
>> plot(v(:,2:69),v(:,1),'b-')
```

Define the life time T_a as the number of cycles needed to get a crack with length a. For each specimen one obtains an independent observation of T_a defined as the number of cycles when the crack growth curve crosses the level a.

Use the function `alevel` to get the life time T_a for $a = 15$ by

```
>> N = alevel(v,15);
```

and view the material graphically by

```
>> plot(N,ones(1,length(N)),'o')
```

1. For a fixed level $a = 20$, choose an appropriate model for the life time distribution T_a. Check extreme value, lognormal, etc. Use the commands `wnormplot`, `gumbelplot`, and `weibplot`. Which distribution gives a good fit?

2. Let a_N denote the crack length after N cycles. From each specimen we can get an observation of a_N. The function `nlevel` returns the crack length of the specimens after a specified number of cycles, here with $N = 2 \cdot 10^5$.

   ```
   >> a = nlevel(v,2e5);
   >> plot(a,ones(1,length(a)),'o')
   ```

   Find a model for the distribution of the crack length—check the proposed model in Example 2, Chapter 2 (lognormal).

Computer Exercise 2

Switching Markov Loads and Rainflow Analysis

2.1 Introduction

In the PhD thesis "Rainflow Analysis of Switching Markov Loads", Johannesson [4] algorithms were developed for calculating the theoretical rainflow matrix for switching processes, and for decomposing a mixed rainflow matrix. (Some of the material is also found in the Licentiate thesis "Rainflow Cycles for Random Loads with Markov Regime", Johannesson [2]) These algorithms were implemented in Matlab, and is included in the WAFO toolbox.

2.2 Markov Chains of Turning Points

First we will examine a non-switching load, modelled as a Markov chain of turning points.

2.2.1 Model Definition

The model is defined by the min-max matrix, `G`. The matrix has dimension $n \times n$, where n is the number of discrete levels (e.g. 32 or 64). Here the discrete levels are chosen in the range from -1 to 1.

```
>> n = 32; param = [-1 1 n];                % Define discretization
>> u = levels(param);                       % Discrete levels
>> G = mktestmat(param,[-0.2 0.2],0.15,1);  % min-max matrix
```

The command `mktestmat` (try `help mktestmat`) creates a test matrix according to the model in [4, p. 127, Eqs. (8.15,8.16)].

2.2.2 Simulation

The model is easy to simulate and this is performed by the routine `mctpsim`

```
>> T = 5000;                 % Length of simulation (number of TP)
>> xD = mctpsim({G []},T);  % Simulate
>> x = u(xD)';               % Change scale to levels -1,..,1
```

and returns the discrete load xD, which takes values $1, \dots, n$. By changing the scale of xD the load x takes values between -1 and 1. The first 200 samples of the simulation is plotted by

```
>> t=1:200; plot(t,x(t))
```

2.2.3 Computation of the Limiting Rainflow Matrix

The limiting rainflow matrix Grfc for the Markov model is calculated by the routine mctp2rfm.

```
>> Grfc=mctp2rfm({G,[]});
```

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(1,2,1),cmatplot(u,u,G),axis('square')
>> subplot(1,2,2),cmatplot(u,u,Grfc),axis('square')
```

Both 2D- and 3D-plots are possible draw, see help cmatplot. It is also possible to plot many matrices in one call.

```
>> cmatplot(u,u,{G Grfc},3)
```

Try also to plot with method=1, which gives a 3D-plot, and with method=4, which gives contour lines. 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.

Calculate the observed rainflow matrix obtained from the simulation (using dtp2rfm). Compare it with the theoretical one.

```
>> FrfcObs = dtp2rfm(xD,n);
>> cmatplot(u,u,{FrfcObs Grfc*T/2})
```

In order to compare the observation FrfcObs with the theoretical rainflow matrix Grfc we have to multiply it by the number of cycles in the simulation which is equal to T/2.

2.3 Switching Markov Chains of Turning Points

Here we will examine an example of a switching load modelled by a switching Markov chain of turning points. This is the same model as used in [4, p. 57, Example 4.1]. The load will be a mixture of two subloads specified via its min-max matrices. First we will define the model and make a simulation. Then we will calculate the limiting rainflow matrix for the model and compare it with a simple superposition of the two subloads and also with the observed rainflow matrix from the simulation. Finally, we will examine level crossings and damage. See also the demo `rfcdemo2` which considers the examples shown below of switching Markov chains of turning points.

2.3.1 Model Definition

The model is defined by the min-max matrices for each subload together with the transition matrix `P` for the regime process. Here we consider a mixture of two subloads, which are defined according to model A in Table 2.1. (We will examine model B later on.) By using

	Model A				Model B			
subload z	x_{1z}	x_{2z}	s_z	λ_z	x_{1z}	x_{2z}	s_z	λ_z
1	-0.4	-0.3	0.15	1.0	-0.1	0.1	0.28	0.5
2	0.3	0.4	0.15	1.0	0.0	0.0	0.12	2.0

Table 2.1: *Parameters for the subloads of models A and B. The min-max matrix $\boldsymbol{G}$ for each subload is given by the templates describes in [4, p. 127, Eqs. (8.15,8.16)]. The subloads are assumed to be time-reversible, and hence the max-min matrix is $\hat{\boldsymbol{G}} = \boldsymbol{G}^T$.*

the routine `mktestmat`, we specify the min-max matrices, `G1` and `G2`. They both have dimension $n \times n$, where $n = 32$ is the number of discrete levels, ranging from -1 to 1.

```
>> n=32; param = [-1 1 n];                      % Define discretization
>> u=levels(param);                             % Discrete levels
>> G1 = mktestmat(param,[-0.4 -0.3],0.15,1); % regime 1
>> G2 = mktestmat(param,[0.3 0.4],0.15,1);   % regime 2
```

Plot the matrices `G1` and `G2` by using `cmatplot`.

Next we specify a transition matrix for the regime process. (You may choose a different one if you like.)

$$\boldsymbol{P} = \begin{pmatrix} 0.90 & 0.10 \\ 0.05 & 0.95 \end{pmatrix} \tag{2.1}$$

```
>> p1=0.10; p2=0.05;
>> P=[1-p1 p1; p2 1-p2]                  % Transition matrix
>> statP=mc2stat(P)                      % Stationary distribution
```

which has stationary distribution `statP`, equal to $\boldsymbol{\rho} = (1/3, 2/3)$. This means that (in mean) $1/3$ of the time is spent in regime 1 and $2/3$ of the time in regime 2.

2.3.2 Simulation

The model is simulated by the routine `smctpsim`

```
>> T=5000;                       % Length of simulation (number of TP)
>> [xD,z] = smctpsim(P,{G1 []; G2 []},T); % Simulate
>> x=u(xD)';                     % Change scale to levels -1,..,1
```

and returns the switching load xD, which takes values $1, \ldots, n$, and the regime process z. By changing the scale of xD the switching load x takes values between -1 and 1. The first 400 samples of the simulation is plotted by

```
>> t=1:400;
>> hmmplot(x(t),z(t),t,[1 2])         % Same colour
>> hmmplot(x(t),z(t),t,[1 2],'','',1) % Different colours
```

and a simulated load is shown in Figure 2.1. The regime process z controls the current characteristics of the switching load x, which is clearly seen as changes of the mean level.

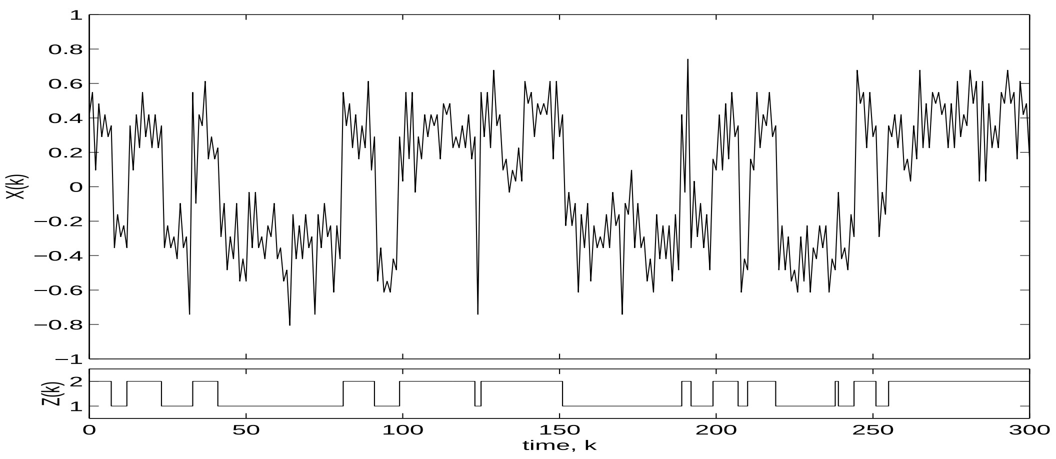

Figure 2.1: *The first 400 samples from a simulation of model A. The upper graph shows the turning points* x *and the lower the regime process* z.

2.3.3 Computing the Limiting Rainflow Matrix

The two subloads are described by G1 and G2, respectively, and their limiting rainflow matrices can be calculated by the routine mctp2rfm. The rainflow matrix for the switching load is calculated by the routine smctp2rfm which takes the transition matrix P together with G1 and G2 as input.

```
>> Grfc=smctp2rfm(P,{G1,[];G2,[]});
>> Grfc1=mctp2rfm({G1,[]});
>> Grfc2=mctp2rfm({G2,[]});
>> GrfcSum=statP(1)*Grfc1+statP(2)*Grfc2;
```

The matrix GrfcSum is a superposition of the rainflow matrices for subloads 1 and 2, respectively. This is the rainflow matrix we obtain if we don't take the switching into account. The rainflow matrices can be plotted by cmatplot.

```
>> cmatplot(u,u,{Grfc1 Grfc2; GrfcSum Grfc})
```

Note that, if the rainflow matrices for the two regimes are superimposed like `GrfcSum`, then the largest cycles (low minima and large maxima) are lost, compare the two lower plots.

2.3.4 Model B (Optional)

Skip this section if you don't have time, and come back to it later.

Here we shall examine model B in Table 2.1, the same model as in [4, p. 61, Example 4.2]. For this model the subloads have the same mean level, equal to zero, but different standard deviations. The transition matrix `P` for the regime process will be the same as before. By using the routine `mktestmat`, we specify the min-max matrices, `G1B` and `G2B`.

```
>> G1B = mktestmat(param,[-0.1 -0.1],0.28,0.5);  % regime 1
>> G2B = mktestmat(param,[0 0],0.12,2);          % regime 2
```

Plot the matrices `G1B` and `G2B` by using `cmatplot`. Simulate a load and plot it:

```
>> [xDB,zB] = smctpsim(P,{G1B []; G2B []},T); % Simulate
>> xB=u(xDB)';                               % Change scale to levels -1,..,1
>> hmmplot(xB(t),zB(t),t,[1 2],'','',1) % Different colours
```

A simulated signal is shown in Figure 2.2.

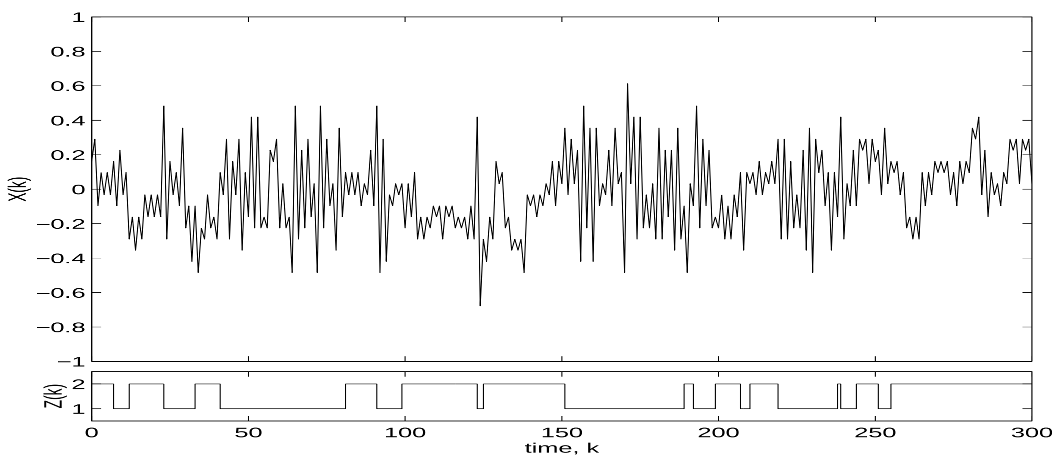

Figure 2.2: *The first 400 samples from a simulation of model B. The upper graph shows the turning points* `x` *and the lower the regime process* `z`*.*

Next we calculate the rainflow matrices for the switching load, for the subloads, and the superposition of the two subloads.

```
>> GrfcB=smctp2rfm(P,{G1B,[];G2B,[]});
>> Grfc1B=mctp2rfm({G1B,[]});
>> Grfc2B=mctp2rfm({G2B,[]});
>> GrfcSumB=statP(1)*Grfc1B+statP(2)*Grfc2B;
```

Plot the matrices and compare `GrfcB` with `GrfcSumB`. What is your observations and conclusions? Can you see the switching when looking at the rainflow matrix?

2.3.5 Observed Rainflow Matrix and Smoothing

Now we consider model A again. From the simulated load we can find the rainflow cycles and then compute the observed rainflow matrix `FrfcObs0`.

```
>> TP = dat2tp([(1:T)' xD]);            % Turning points
>> RFC = tp2rfc(TP);                    % Rainflow cycles
>> paramD = [1 n n];
>> FrfcObs0 = cc2cmat(paramD,RFC);      % Observed rainflow matrix
```

For a discrete sequence of turning points (which `xD` is) one can perform the above operation by one command

```
>> FrfcObs = dtp2rfm(xD,n);             % Observed rainflow matrix
```

Compare the observed rainflow matrix `FrfcObs` with the theoretical one `Grfc`.

```
>> cmatplot(u,u,{FrfcObs/(T/2) Grfc},1)
```

If the observed rainflow matrix is too wiggly it can be smoothed using `smoothcmat` to obtain a better estimate of the limiting rainflow matrix. The smoothing procedure can also be used to extrapolate (and interpolate) the observed rainflow matrix to cycles that were not observed.

```
>> h=0.8; FrfcSmooth = smoothcmat(FrfcObs,1,h);
>> cmatplot(u,u,{FrfcObs/(T/2) FrfcSmooth/(T/2) Grfc},1)
```

The so called bandwidth `h` is a smoothing parameter. Try different values of the smoothing parameter `h` and see the difference. A small `h` gives little smoothing, while a large `h` gives much smoothing.

2.3.6 Level Crossings

Information about level crossings is contained in the rainflow matrix.

```
>> mu = cmat2lc(param,Grfc);
>> muSum = cmat2lc(param,GrfcSum);
>> muObs = cmat2lc(param,FrfcObs/(T/2));
>> subplot(2,1,1), plot(mu(:,1),mu(:,2),muSum(:,1),muSum(:,2),'--')
>> subplot(2,1,2), plot(mu(:,1),mu(:,2),muObs(:,1),muObs(:,2),'--')
```

The first plot compares the upcrossing intensity `mu` of the switching load with the upcrossing intensity `muSum` of the superimposed rainflow matrix. Observe that, as expected, the switching gives rise to more zero upcrossings than in the weighted sum of the individual upcrossing intensities. In the second plot the observed upcrossing intensity `muObs` is shown together with the theoretical one `mu`.

2.3.7 Damage

The toolbox contains routines for calculating the damage of a load. The Palmgren-Miner damage hypothesis together with the Wöhler curve is used. This leads to the damage

$$\mathcal{D}_\beta = \sum \frac{1}{N_{s_i}} = \sum K \left(S_i^{\text{rfc}} \right)^\beta, \qquad S_i^{\text{rfc}} = \left(M_i - m_i^{\text{rfc}} \right)/2 \tag{2.2}$$

where K and β are material parameters. (The routines use $K = 1$.) The routines are `cc2dam` for a cycle count (e.g. rainflow cycles) and `cmat2dam` for a cycle matrix (e.g. rainflow matrix).

```
>> beta = 4;
>> Dam = cmat2dam(param,Grfc,beta)
>> DamSum = cmat2dam(param,GrfcSum,beta)
>> DamObs = cc2dam(u(RFC),beta)/(T/2)
```

The calculated damages are scaled to damage per cycle, giving the results `Dam=0.0098`, `DamSum=0.0017` and for the simulation `DamObs=0.0097`. The damage from the simulation can also be computed from `FrfcObs` by using `cmat2dam`.

The damage matrix is calculated by `cmat2dmat` and shows how the damage is distributed among the different cycles. The sum of all the elements in the damage matrix gives the total damage.

```
>> Dmat = cmat2dmat(param,Grfc,beta);
>> DmatSum = cmat2dmat(param,GrfcSum,beta);
>> subplot(1,2,1), cmatplot(u,u,Dmat), axis('square'), v=axis;
>> subplot(1,2,2), cmatplot(u,u,DmatSum), axis('square'), axis(v)
```

Note that the rainflow matrix for the switching process gives much more damage than that of the simple superposition.

2.4 Decomposition of a Mixed Rainflow Matrix

When collecting a load history, in the form of a rainflow matrix, all the cycles of the switching load are stored in one mixed rainflow matrix. Hence, there is a need to interpret the mixed rainflow matrix, and e.g. tell how much of the different load types a vehicle has experienced.

Here we will see that one can use the methods in [4, Chapter 8] for decomposition of a mixed rainflow matrix, i.e. methods for estimation of a SMCTP model given a measurement of a mixed rainflow matrix. Hence, the task is to estimate the characteristics of the different subloads as well as the characteristics of the switching between the different subloads. The fact that we are able to calculate the mixed expected rainflow matrix for a SMCTP model makes this estimation possible. The principle of the decomposition is to find the best fit between the measured rainflow matrix and the theoretically computed one, i.e. the expected rainflow matrix. One problem is to decide what to mean by “best fit”. The maximum likelihood (ML) method is often the best method and yields the model that is "the most probable one". We will propose an approximate ML estimator. There is also the possibility to minimize the distance between the measured and the theoretical rainflow matrices. We will propose three such distances, namely the chi-square, the Hellinger, and the Kullback-Leibler distances.

2.5 Model and Estimation

We want to estimate a SMCTP model, where the number of regime states r is fixed. The model is parametrized by the min-max and the max-min transition matrices

$$\boldsymbol{Q}^{(1)}, \ldots, \boldsymbol{Q}^{(r)} \qquad \text{and} \qquad \hat{\boldsymbol{Q}}^{(1)}, \ldots, \hat{\boldsymbol{Q}}^{(r)}$$

respectively, and the transition matrix $\boldsymbol{P}$ for the regime process.

Obviously, one would like to estimate the whole model, i.e. the $\boldsymbol{P}$-matrix, and the subloads. However, this is not possible, since the number of parameters in the SMCTP model is larger than the number of observations, i.e. the number of elements in the rainflow matrix. Therefore, one needs to impose some additional structure on the model in order to get fewer parameters to estimate.

Sometimes the min-max matrices of the subprocesses are known (or can be considered known) and thus the parameters of the subloads are given. Hence, in this case, only the transition matrix $\boldsymbol{P}$ needs to be estimated. For a SMCTP with two regime states we have a model with the parameters

$$\boldsymbol{P} = \begin{pmatrix} 1-p_1 & p_1 \\ p_2 & 1-p_2 \end{pmatrix}, \quad \boldsymbol{Q}^{(1)}, \hat{\boldsymbol{Q}}^{(1)}, \boldsymbol{Q}^{(2)}, \hat{\boldsymbol{Q}}^{(2)}$$

and when the models of the subloads are known, then only the parameters $\boldsymbol{\theta} = (p_1, p_2)$ need to be estimated. One such case is when the measured rainflow matrix is believed to be a mixture of known standard rainflow matrices, which could e.g. reflect different parts of a testing track. The goal is then to find the proportion and the switching frequency of the different subprocesses. All this information is contained in the $\boldsymbol{P}$ matrix.

Define a SMCTP model and simulate it in order to obtain an observed mixed rainflow matrix.

```
>> n = 8; param = [-1 1 n];
>> M1.x0=[-0.4 -0.3]; M1.s=0.15; M1.lam=1;
>> M2.x0=[0.3 0.4]; M2.s=0.15; M2.lam=1;
>> G1 = mktestmat(param,M1.x0,M1.s,M1.lam);
>> G2 = mktestmat(param,M2.x0,M2.s,M2.lam);
>> P=[1-0.1 0.1; 0.05 1-0.05]                % Transition matrix
>> [xD,z] = smctpsim(P,{G1 []; G2 []},5000); % Simulate
>> Fobs = dtp2rfm(xD,n);                     % observed mixed rainflow matrix
```

In order to get reasonably short computing times in Matlab when doing the decomposition, we have chosen only 8 discrete levels. However, the programs for doing the decomposition don't have any limitations in the size of the model.

Three Scenarios

We will consider three cases of specifying the parameter vector $\boldsymbol{\theta}$, that we want to estimate, depending on how much knowledge we have on the min-max matrices. The approximate ML method will be used.

1. The min-max matrices of the subloads can be considered known, so that only the regime process is unknown, and only the parameters p_1 and p_2 need to be estimated. $\boldsymbol{\theta} = (p_1,\ p_2)$

```
>> known1.F = {G1 []; G2 []}; % known min-max and max-min matrices
>> init1.P = P;               % initial guess of P-matrix
>> [Fest1,Est1] = estsmctp(Fobs,'P','ML',known1,[],init1);
>> Est1.P                     % Estimated P-matrix
```

2. Here the min-max matrix for each subload z is specified, but we allow a translation $\tilde{m}_z$ and a scaling $\tilde{s}_z$, which are used as parameters for the subloads. The translation $\tilde{m}_z$ corresponds to a change of the mean level and the scaling $\tilde{s}_z$ corresponds to a change in the variance for the subload, i.e. a transformation of the form $X^{(z)}(t) = \tilde{s}_z\tilde{X}^{(z)}(t) + \tilde{m}_z$, where $\tilde{X}^{(z)}(t)$ is the process according to the specified min-max matrix. Now we have six parameters to estimate,
$\boldsymbol{\theta} = (p_1,\ p_2,\ \tilde{m}_1,\ \tilde{s}_1,\ \tilde{m}_2,\ \tilde{s}_2)$.

```
... Estimate ...
```

3. Here we will use, for each subload, the parametric model described by Eqs. (8.15,8.16) with four parameters. In total this gives ten parameters to estimate,
$\boldsymbol{\theta} = (p_1,\ p_2,\ x_{11}\ , x_{21}\ , s_1\ , \lambda_1,\ x_{12}\ , x_{22}\ , s_2\ , \lambda_2)$.

```
>> known3.Ffun = 'f_funm';   % Function for calculating a submodel
>> known3.trModel2X = 'tr_m2x'; % transform from Model to X-vector
>> known3.trX2Model = 'tr_x2m'; % transform from X-vector to model
>> known3.param =param;
>> init3.P = P;              % initial guess of P-matrix
>> init3.M = {M1 M2};        % initial guess of Models for min-max mat
>> [Fest3,Est3] = estsmctp(Fobs,'P,CalcF','ML',known3,[],init3);
>> Est3.P                    % Estimated P-matrix
>> Est3.M{:}                 % Estimated parameters in models
```

Estimation from a Measurement of the Regime Process

If the regime process $\{Z_k\}$ itself is observed, then one has as much information about the switching that one can possibly get. The ML estimates of p_1 and p_2 can easily be obtained from the observed regime process, $\boldsymbol{z} = \{z_k\}_{k=0}^{T}$, see [4, Appendix A].

```
>> Pest_z = estmc(z,2)
```

Compare the estimated stationary distributions and the true one:

```
>> mc2stat(Est1.P)
>> mc2stat(Est3.P)
>> mc2stat(Pest_z)
>> mc2stat(P)
```

Methods for Estimation (Optional)

The four methods for estimation, described in [4, Section 8.2], will now be examined. The first method is the approximate maximum likelihood (AML) estimator, where we assume that the sample of the rainflow matrix is a sample from a multinomial distribution. The next three methods are based on minimizing the distance between the measured rainflow matrix and the expected rainflow matrix. The distances that will be used are, the chi-square distance (χ^2), the Hellinger distance (HD), and the Kullback-Leibler distance (KL). The estimates are obtained by numerical optimization by using the routine `fmins`, see [4, Section 8.6] for further details.

```
>> known.F = {G1 []; G2 []};   % known min-max and max-min matrices
>> init.P = P;                 % initial guess of P-matrix
>> [Fest,Est] = estsmctp(Fobs,'P','ML',known,[],init); Est.P
>> [Fest,Est] = estsmctp(Fobs,'P','chi2',known,[],init); Est.P
>> [Fest,Est] = estsmctp(Fobs,'P','HD',known,[],init); Est.P
>> [Fest,Est] = estsmctp(Fobs,'P','KL',known,[],init); Est.P
```

Computer Exercise 3

Power Spectrum and Rainflow Analysis

In this exercise we present tools for computation of rainflow matrices (and consequently fatigue failure predictors) for Gaussian random loads. As was mentioned in Introduction, a Gaussian load is uniquely defined by its mean value m and spectrum $S(\omega)$. For simplicity only, the mean is assumed to be zero; hence, the spectrum is the single input argument into the model. The spectrum can be estimated from a measured record (as we did in Computer Exercise 1) or theoretically derived, as it is often the case in fatigue damage analysis of sea vessels.

3.1 Power Spectrum

Choose any spectrum you wish to analyse. It can be an estimated spectrum from Lab 1 or some standard spectra given in WAFO. Four examples are given below:

```
>> S=jonswap;
>> S=oscspec;
>> S=torsethaugen;
>> load deep.dat, S=dat2spec(deep);
```

Compute mean frequency f_0, standard deviation σ, and irregularity factor α for the Gaussian process with computed spectrum S. Solution:

```
>> L=spec2mom(S,4);
>> f0=sqrt(L(2)/L(1))/2/pi
>> s=sqrt(L(1))
>> alfa=f0/(sqrt(L(3)/L(2))/2/pi)
```

The four spectra are unimodal and only the estimated spectrum can be considered as a broad band spectrum. For narrow band spectra, with one mode, the mean frequency f_0 is close to the the mode of the spectrum. Check it for your choice of spectrum.

```
>> wspecplot(S);
>> 2*pi*f0
```

Remark. The parameters f_0 and σ are important to perform a crude analysis of the damage intensity of the load. The mean frequency indicates a rate of bigger cycles while σ is the scaling factor for amplitudes. The so-called moment method postulates that amplitudes of these cycles are $\sigma \cdot R$, Rayleigh distributed (the pdf for a Rayleigh variable is $r\exp(-0.5r^2)$). This gives a very simple predictor of fatigue life time T^f

$$\hat{T}^f = \frac{1}{\gamma\, f_0 E[(\sigma R)^\beta]} = \frac{1}{\gamma\, f_0 \sigma^\beta\, 2^{\beta/2}\Gamma(\frac{\beta}{2}+1)}.$$

As a matter of fact, one can prove that this predictor is always conservative (gives too short fatigue life than the one obtained by means of rainflow method). Summarizing, knowing f_0 and σ, the upper bound for the damage intensity can be calculated. This is what the function `spec2dplus` is computing. If the spectrum is unimodal and particularly if it is narrow banded, the method gives very accurate results. Observe that in some special cases, when $\beta = 1$ and as β goes to infinity, the method is exact. In addition, this approach can be extended to non-Gaussian loads; then the crossing spectrum $lc(u)$ has to be given. When the crossing spectrum $lc(u)$ is known, one can compute conservative bounds for the damage intensity by means of `lc2dplus`. (Obviously for Gaussian loads, `lc2dplus` gives the same results as `spec2dplus`).

For the chosen spectrum, compute the conservative bound for the predictor of T^f, time to fatigue failure. Use $\beta = 3.0, 3.1, 3.2, \ldots, 5.0$ and $\gamma = 5 \cdot 10^{-9}$. (Obviously, our examples are somewhat artificial since the signals are not stresses and the parameters β and γ are not based on real tests.) Computation of $\hat{T}^f$

```
>> beta=3:0.1:5;
>> gam=5E-9;
>> dpl=gam*spec2dplus(S,beta);
>> Tfpl=(1./dpl)/3600/24  %in days
>> clf, plot(beta,Tfpl)
```

3.2 Simulation of X and the damage intensity

In this section we shall simulate $X(t)$, extract rainflow cycles, estimate damage intensity and derive an estimate of time to fatigue failure T^f. If your spectrum is narrow banded, you should obtain results close to the conservative bounds from the previous subsection. For broad banded spectra the difference can be more relevant and one may need to use other methods to derive estimates or T^f.

Suppose that one wishes to simulate $X(t)$ with approximatively 1000 non-negligible cycles. The duration time T of the signal is then $T = 1000/f_0$. Compute (approximatively) the proportion of fatigue lifetime consumed by $X(t)$, $0 \le t \le T$, in % ($100\%\ T/T^f$), for $\beta = 4.22$ and $\gamma = 5 \cdot 10^{-9}$. Solution:

```
>> T=1000/f0 % in seconds
>> gam=5E-9;
>> 100*T*gam*spec2dplus(S,4.22) % in percent
```

Remark. When one wishes to simulate the process $X(t)$ for a given spectrum S, it may not be obvious how to choose an appropriate sampling frequency. One possibility is to use the highest angular frequency in the spectrum ω_{max}, say. Then a natural sampling frequency would be $f = \omega_{max}/2\pi$. However, this is often a too low frequency to get correct values of rainflow amplitudes. Simply, one would miss exact positions of local maxima and minima in X, leading to underestimation of rainflow amplitudes. In many cases the choice of between 50 to 100 times higher frequency than the mean frequency is giving good results. In the following example we will use $f = 60f_0$, that gives 60000 observations for the chosen T value.

We will simulate five sample paths of loads and compute the damage intensity. The sampling interval is chosen to $dt = 1/(60f_0)$. For fast and accurate simulation, the spectrum is recomputed so that the highest frequency $\omega_{max} = 2\pi/dt$. Then $X(t)$ is simulated. From the sample path, turning points are extracted and the rainflow cycles are calculated. The damage intensity can then be computed, and a prediction of the fatigue failure time T^f obtained.

```
>> max(S.w)/pi
>> dt=1./f0/60
>> S1=specinterp(S,dt);
>> [max(S1.w) pi/dt]
>> clf, wspecplot(S), hold on, wspecplot(S1,1,'r.')
>> clf, plot(beta,Tfpl)
>> hold on
>> for  i=1:5
>>   X=spec2sdat(S1,60000);
>>   tp=dat2tp(X);
>>   rfc=tp2rfc(tp);
>>   db=cc2dam(rfc,beta);
>>   plot(beta,(1000/f0)*(1./db)*(1/gam)/3600/24,'r.') % Tf in days
>> end
```

3.3 Theoretical computation of damage intensity

For more complicated spectra that are not unimodal, the moment method can be too conservative. In order to illustrate this we choose the following spectrum

$$S(\omega) = 9\exp(-8(\omega - 0.5)^2) + 2\exp(-2|w - 2|^{1.4}).$$

It can be created as follows

```
>> S=createspec;
>> w=levels([0 4 253]);
>> S.S=2*exp(-2*abs(w-2).^1.4);
>> S.S=S.S+9*exp(-8*(w-0.5).^2);
>> S.w=w;
>> S.note=['S(w)=9*exp(-8(w-0.5)^2)+2*exp(-2|w-2|.^1.4)']
>> wspecplot(S)
```

Check the accuracy of the conservative bound for the damage intensity by simulating 10 sample paths of the load.

```
>> L=spec2mom(S,4);
>> f0=sqrt(L(2)/L(1))/2/pi
>> dpl=gam*spec2dplus(S,beta);
>> Tfpl=(1./dpl)/3600/24  %in days
>> dt=1./f0/60
>> S1=specinterp(S,dt);
>> figure(1)
>> clf, plot(beta,Tfpl)
>> hold on
>> for  i=1:10
>>   X=spec2sdat(S1,60000);
>>   tp=dat2tp(X);
>>   rfc=tp2rfc(tp);
>>   db=cc2dam(rfc,beta);
>>   plot(beta,(1000/f0)*(1./db)*(1/gam)/3600/24,'r.') % Tf in days
>> end
```

We turn now to alternative method of computation of the rainflow matrix and damage intensity. The method consists of two steps. First, the transition matrix from maximum to the following minimum `fMm`, say, in X, is computed using a generalized Rice's formula. (Since numerical integrations are needed, this step is relatively slow. The accuracy parameter is called `nit`$= 0, \ldots, 5$. Higher `nit` gives better approximation). Next, the sequence of turning points is approximated by a MCTP. The methods presented in Computer Exercise 2 then give the rainflow matrix.

```
>> a=sqrt(L(3)/L(2))/2/pi
>> s=sqrt(L(1))
>> help spec2cmat
>> paramu=[-4.5*s 4.5*s 46];
>> nit=2;
>> figure(2), clf
>> [frfc fMm]=spec2cmat(S,[],'rfc',[],paramu,nit);
>> hold on, plot(rfc(:,2),rfc(:,1),'.')
>> clf, pdfplot(fMm)
>> dg=a*cmat2dam(paramu,frfc.f,beta);
>> figure(1), plot(beta,(1./dg)*(1/gam)/3600/24,'g') % Tf in days
```

For this broad banded spectrum the parameter `nit`$= 2$ is too low; we propose to use `nit`$= 3$, but it can take 4 minutes to get the results.

```
>> figure(2), clf
>> [frfc1 fMm1]=spec2cmat(S,[],'rfc',[],paramu,3);
>> dg1=a*cmat2dam(paramu,frfc1.f,beta);
>> figure(1)
>> plot(beta,(1./dg1)*(1/gam)/3600/24,'k') % Tf in days
```

We turn now to some additional applications of the computed matrix `fMm`. (Those exercises are optional.)

First, the matrix can be used to simulate much longer and faster sequences of turning points of X, than using `spec2sdt`.

```
>> help mctpsim
>> F{1,1}=fMm.f;
>> F{1,2}=fMm.f';
>> mctp=mctpsim(F,1000);
>> clf,plot(mctp(1:40))
```

Next, if the simulated sequence contains too many small amplitudes one can derive the Markov matrix for the rainflow filtered sequence. It can be done as follows: replace a few sub-diagonals in the `frfc` matrix by zeros, then invert it to get the new transition matrix. Now you can use it to simulate the filtered sequence of turning points.

```
>> frfc1=zeros(size(frfc.f));
>> frfc1=triu(frfc.f,5);
>> fMm1=iter(frfc1,fMm.f,20,0.001);
>> clf,contour(fMm.f,20)
>> hold on,  contour(fMm1,20)
>> F{1,1}=fMm1;
>> F{1,2}=fMm1';
>> mctp1=mctpsim(F,1000);
>> clf, subplot(2,1,1)
>> plot(mctp(1:40))
>> subplot(2,1,2)
>> plot(mctp1(1:40))
```

Computer Exercise 4

Modelling of Measured Loads

We want to model a measured load signal that changes characteristics over time. Suppose that a switching process would be an appropriate model for the load. Our strategy is then to fit a switching Markov chain of turning points (SMCTP) to the measurements. This model can then be used to calculate the expected rainflow matrix for the load, or to simulate load processes. The procedure is illustrated in the form of a flow chart in Figure 4.1, which is explained and commented below.

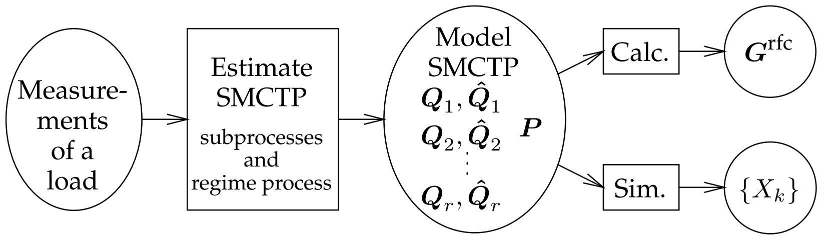

Figure 4.1: *Flow chart for the modelling of measurements of a load.*

4.1 Estimation from Time Signal

In this example we have a measurement of a truck load, with the truck driving on a road that consists of curves and straights with some small bends, see [4, Example 6.3]. The same piece of road has been measured twice. The time signal is stored in the variable `x0` with time (seconds) in the first column and the load values in the second column. Load and plot the data.

```
>> load switchingload
>> plot(x0(:,1),x0(:,2))
```

The load changes between two duty cycles which can be observed as changes in mean and standard deviation.

Does it seem appropriate to model the load by a SMCTP model with two regime states? The two regime states represent straights (regime 1, low mean, large variation), and curves (regime 2, high mean, low variation).

First, we will remove small oscillations in the load, which either come from measurement noise, or are irrelevant to the fatigue damage. This is done by a rainflow filter, which deletes all rainflow cycles with ranges smaller than a given threshold, which is often chosen as the discretization step. Define the discretization and apply a rainflow filter:

```
>> n = 32; param = [-1 1.2 n]; % Define discretization
>> u = levels(param);     % Discrete levels
>> delta = u(2)-u(1)      % Discretization step
>> TP = dat2tp(x0,delta); % Get turning points and rainflow filter
```

Compare the turning points of the original time signal with the rainflow filtered turning points. How much does the rainflow filter reduce the amount of data?

```
>> TP0 = dat2tp(x0);
>> plot(x0(:,1),x0(:,2),TP0(:,1),TP0(:,2),TP(:,1),TP(:,2))
>> length(x0), length(TP0), length(TP)
```

Check the rainflow cycles and the damage before and after rainflow filtering. Is there any significant difference? Try different values of the damage exponent `beta`.

```
>> RFC0 = tp2rfc(TP0); subplot(1,2,1), ccplot(RFC0)
>> RFC = tp2rfc(TP);   subplot(1,2,2), ccplot(RFC)
>> beta = 6;                 % Damage exponent
>> Dam0 = cc2dam(RFC0,beta) % Damage
>> Dam = cc2dam(RFC,beta)   % Damage, after rainflow filter
>> Dam/Dam0
```

Examine the level crossing spectrum of the load by using `tp2lc`, both for the original sequence `TP0` and for the rainflow filtered sequence `TP`. Is there any difference between the two crossing spectra? Is it possible, from the level crossings, to see that the load consists of two subloads?

Examine if the load can be modelled as a Gaussian process by using e.g. `wnormplot`. Also compare the level crossing spectrum with the one obtained from Rice's formula, see Computer Exercise 1.

Now we will discretize the load by using `dat2dtp`, which makes discretization to the nearest discrete level.

```
>> dtp = dat2dtp(param,TP,delta); % Discretized turning points
>> tp = [dtp(:,1) u(dtp(:,2))'];  % Discretized turning points
>> T = length(dtp);               % Number of turning points
>> clf, plot(x0(:,1),x0(:,2),TP(:,1),TP(:,2),tp(:,1),tp(:,2))
>> v=axis; hold on,
>> plot([v(1:2)],[u(2:end)'-delta/2 u(2:end)'-delta/2],'k:')
>> hold off, axis([v(1:2) param(1:2)])
```

What is the error in damage due to the discretization?

```
>> rfc = tp2rfc(tp);        % Get rainflow cycles
>> dam = cc2dam(rfc,beta)   % Damage, after discretization & rainflow filter
>> dam/Dam0
```

Next we will (by hand) identify the two duty cycles in the load signal. Find the times when load switches regime state, and split up the load by using the routine `splitload`. The first column of `tz` contains times (in seconds) and the second column contains the regime state that the load switches to.

```
>> tz = [2 2; 46.40 1; 114.5 2; 161 1; 225.1 2; 270 1; 337.5 2; ...
>> 384.8 1; 433.2 2; 600 1];
>> [xxd,xd,z] = splitload(dtp,tz);
>> plot(xxd{1}(:,1),xxd{1}(:,2))
>> plot(xxd{2}(:,1),xxd{2}(:,2))
>> hmmplot(xd(:,2),z,xd(:,1),[1 2],'','',1)
```

Here `xxd{1}` and `xxd{2}` contain subload 1 and 2, respectively, `xd` the switching load, and `z` the regime process.

Estimation of the Subloads

Now assume that the subloads are time-reversible, which implies that their expected max-min matrices can be obtained from their respective expected min-max matrix through $\hat{\boldsymbol{G}} = \boldsymbol{G}^T$. For each subload we need to calculate the min-max and max-min transition matrices $\boldsymbol{Q}$ and $\hat{\boldsymbol{Q}}$, respectively. This can be done in the following three steps:

1. *Calculation of min-max matrix $\boldsymbol{F}$.* This is no problem if we have a measurement of $\boldsymbol{F}$. (If instead we have measured the rainflow matrix $\boldsymbol{F}^{\text{rfc}}$, it can be inverted to find the min-max matrix $\boldsymbol{F}$, see [4, Chapter 7], and see also the routines `rfm2mctp` and `arfm2mctp`.) In our case the subloads have been measured as time series and then it is straightforward to calculate the min-max matrix $\boldsymbol{F}$ (and max-min matrix $\hat{\boldsymbol{F}}$).

   ```
   >> dtp1 = dat2tp(xxd{1});
   >> [mM1,Mm1] = tp2mm(dtp1);
   >> F1 = dcc2cmat(mM1,n);
   >> dtp2 = dat2tp(xxd{2});
   >> [mM2,Mm2] = tp2mm(dtp2);
   >> F2 = dcc2cmat(mM2,n);
   >> cmatplot(u,u,{F1 F2})
   ```

2. *Estimate $\boldsymbol{G}$ through smoothing.* To obtain an estimate of the expected min-max matrix $\boldsymbol{G}$, the min-max matrix $\boldsymbol{F}$ is smoothed using a 2-dimensional kernel smoother, see [4, Appendix D].

   ```
   >> [G1s,h1] = smoothcmat(F1);
   >> G1 = smoothcmat(F1,1,1.0,0);
   >> [G2s,h2] = smoothcmat(F2);
   >> G2 = smoothcmat(F2,1,0.8,0);
   ```

```
>> cmatplot(u,u,{G1s G2s; G1 G2})
>> cmatplot(u,u,{F1 F2; G1 G2})
```

Choose appropriate values of the smoothing parameter `h`. (Here we can improve the estimates, if the max-min matrix $\hat{\boldsymbol{F}}$ is also used, by smoothing the sum $\boldsymbol{F} + \hat{\boldsymbol{F}}^T$, instead of smoothing only $\boldsymbol{F}$.)

3. *Normalizing.* The min-max and max-min transition matrices $\boldsymbol{Q}$ and $\hat{\boldsymbol{Q}}$, respectively, are obtained from the expected min-max and max-min matrices $\boldsymbol{G}$ and $\hat{\boldsymbol{G}} = \boldsymbol{G}^T$, respectively, by normalizing each row sum to 1. This is done automatically by the programs (`mctp2rfm` and `smctp2rfm`).

Estimation of the Regime Process

The $\boldsymbol{P}$-matrix for the regime process is obtained through ML-estimation

$$\boldsymbol{P}^* = \begin{pmatrix} 1-p_{12}^* & p_{12}^* \\ p_{21}^* & 1-p_{21}^* \end{pmatrix} \tag{4.1}$$

where

$$p_{12}^* = \frac{N_{12}}{N_1} = \frac{\#\{\text{Jumps from 1 to 2}\}}{N_1} = \frac{4}{N_1} = 0.0134, \tag{4.2}$$

$$p_{21}^* = \frac{N_{21}}{N_2} = \frac{\#\{\text{Jumps from 2 to 1}\}}{N_2} = \frac{4}{N_2} = 0.0081. \tag{4.3}$$

where N_{ij} is the number of switches from regime state i to state j, and N_i is the total number of turning points in regime state i. Now we estimate `P` from the rainflow filtered and discretized load signal.

```
>> N1 = length(dtp1), N2 = length(dtp2)
>> N12 = 4; N21 = 4;
>> p1=N12/N1; p2=N21/N2;
>> P = [1-p1 p1; p2 1-p2]  % P-matrix
>> statP = mc2stat(P)      % Stationary distribution
```

From the estimated SMCTP model (`P`, `G1`, and `G2`), we can calculated the expected rainflow matrix

```
>> GG = {G1 []; G2 []};
>> [Grfc,mu_rfc] = smctp2rfm(P,GG);
>> cocc(param,RFC,Grfc)
>> Frfc = dtp2rfm(dtp(:,2),n);
>> cmatplot(u,u,{Frfc Grfc*T/2})
```

Calculate the damage and the damage matrix.

```
>> beta = 6;
>> Dam0, Dam, dam
>> damG = cmat2dam(param,Grfc,beta)*T/2
>> damG/Dam0
>> beta = 4;
>> Dmat = cmat2dmat(param,Frfc,beta);
>> DmatG = cmat2dmat(param,Grfc,beta)*T/2;
>> cmatplot(u,u,{Dmat,DmatG},3)
```

Simulate the estimated SMCTP model and compare it with the measured signal.

```
>> [xsim,zsim] = smctpsim(P,GG,T);
>> figure(1), hmmplot(u(xd(:,2))’,z,1:length(xd),[1 2],’’,’’,1)
>> figure(2), hmmplot(u(xsim)’,zsim,1:T,[1 2],’’,’’,1)
```

Make some more simulations and compare the simulated load signals.

4.2 Decomposition of a Mixed Rainflow Matrix

The goal of this example is to make a decomposition of the measured mixed rainflow matrix, see [4, Example 8.1]. The measurement is is the same truck load as before. We will make the decomposition with different assumptions on the parametrization of the subloads, according to scenarios 1 and 3, see Section 2.4.

In this example we will discretize the measured load by $n = 16$ discrete levels, ranging from $u_1 = -1.0$ to $u_n = 1.2$. The pre-processing of the time signal is performed in the same way as previously.

```
>> n = 16; param = [-1 1.2 n]; % Define discretization
>> u = levels(param);     % Discrete levels
>> delta = u(2)-u(1)      % Discretization step
>> TP = dat2tp(x0,delta); % Get turning points and rainflow filter
```

Check the difference in damage between the original load history and the rainflow filtered one.

Next we will discretize the load and compute its damage.

```
>> dtp = dat2dtp(param,TP,delta); % Discretized turning points
>> tp = [dtp(:,1) u(dtp(:,2))’];  % Discretized turning points
>> T = length(dtp);               % Number of turning points
>> rfc = tp2rfc(tp);        % Get rainflow cycles
>> beta = 6;
>> dam = cc2dam(rfc,beta)   % Damage, after discretization & rainflow filter
>> dam/Dam0
```

From the discretized load we compute the observed rainflow matrix, which will be the input to the decomposition.

```
>> Frfc = dtp2rfm(dtp(:,2),n);  % Observed rainflow matrix
```

For scenario 1, the min-max matrices `G1` and `G2` for the subloads will be treated as a priori information, and will hence be considered known. Therefore, for scenario 1, we only need to estimate the $\boldsymbol{P}$-matrix with two parameters p_1 and p_2. To obtain `G1` and `G2` we split the time signal, and estimate them (and then forget about the time signal).

```
>> tz = [2 2; 46.40 1; 114.5 2; 161 1; 225.1 2; 270 1; 337.5 2; ...
>> 384.8 1; 433.2 2; 600 1];
>> [xxd,xd,z] = splitload(dtp,tz);
>> hmmplot(xd(:,2),z,xd(:,1),[1 2],'','',1)
>> dtp1 = dat2tp(xxd{1});
>> [mM1,Mm1] = tp2mm(dtp1);
>> F1 = dcc2cmat(mM1,n);
>> G1 = smoothcmat(F1,1,1.0,0);
>> dtp2 = dat2tp(xxd{2});
>> [mM2,Mm2] = tp2mm(dtp2);
>> F2 = dcc2cmat(mM2,n);
>> G2 = smoothcmat(F2,1,0.8,0);
```

Plot the estimated matrices `G1` and `G2`.

By using the observed rainflow matrix `Frfc` (and `G1` and `G2`) we do the decomposition.

```
>> known1.F = {G1 []; G2 []};   % known min-max and max-min matrices
>> init1.P = P;                 % initial guess of P-matrix
>> warning off                  % Don't display warnings
>> [Fest1,Est1] = estsmctp(Frfc,'P','ML',known1,[],init1);
>> Est1.P           % Estimated P-matrix
>> mc2stat(Est1.P) % Estimated stationary distribution
```

For scenario 3, we will for each subload use the simple parametric model, which we used in Computer Exercise 2. Hence, for scenario 3, we have 10 parameters to estimate: p_1, p_2, and 4 parameters for each subload,
$\boldsymbol{\theta} = (p_1,\ p_2,\ x_{11}\ , x_{21}\ , s_1\ , \lambda_1,\ x_{12}\ , x_{22}\ , s_2\ , \lambda_2)$.

```
>> known3.Ffun = 'f_funm';      % Function for calculating a submodel
>> known3.trModel2X = 'tr_m2x'; % transform from Model to X-vector
>> known3.trX2Model = 'tr_x2m'; % transform from X-vector to model
>> known3.param = param;
>> init3.P = P;       % initial guess of P-matrix
>> M1.x0=[0.1 0.1]; M1.s=0.15; M1.lam=2; % submodel 1
>> M2.x0=[0.5 0.7]; M2.s=0.1; M2.lam=1;   % submodel 2
>> init3.M = {M1 M2}; % initial guess of Models for min-max matrices
```

To shorten the computation time you can lower the accuracy by setting the input `OPTIONS` which is used by `fmins`. See help `options` for more details.

```
>> OPTIONS(2)  = 1e-1;    % the termination tolerance for x;
>> OPTIONS(3)  = 1e-1;    % the termination tolerance for F(x);
>> [Fest3,Est3] = estsmctp(Frfc,'P,CalcF','ML',known3,[],init3);
>> Est3.P           % Estimated P-matrix
>> mc2stat(Est3.P) % Estimated stationary distribution
>> Est3.M{:}       % Estimated parameters in models
```

Compare the results (`Fest1`, and `Fest2`) in terms of estimated $\boldsymbol{P}$-matrices and there stationary distributions. Remember that the model estimated from the time signal has transition matrix `P` and stationary distribution `statP`. Hopefully, you can observe that even though the estimates of the parameters p_1 and p_2 differs from `P`, the estimates of the stationary distribution are quite accurate.

For each estimated SMCTP model we can compute its expected damage as a function of the damage exponent β. We can also compute the damage from the measured time signal.

```
>> beta = 3:0.2:8;
>> Dam0 = cmat2dam(param,Frfc,beta)/(T/2); % Damage from load signal
>> FrfcEst1 = smctp2rfm(Fest1.P,Fest1.F);
>> Dam1 = cmat2dam(param,FrfcEst1,beta);   % Damage, scenario 1
>> FrfcEst3 = smctp2rfm(Fest3.P,Fest3.F);
>> Dam3 = cmat2dam(param,FrfcEst3,beta);   % Damage, scenario 3
>> plot(beta,Dam0,'b',beta,Dam1,'r',beta,Dam3,'g')
>> plot(beta,Dam1./Dam0,'r',beta,Dam3./Dam0,'g')
```

Are the results, in terms of damage, acceptable? Do they agree better for small or for large values of the damage exponent β.

Bibliography

[1] M. Frendahl and I. Rychlik. Rainflow analysis: Markov method. *International Journal of Fatigue*, 15:265–272, 1993.

[2] P. Johannesson. *Rainflow Cycles for Random Loads with Markov Regime*. Licentiate thesis TFMS–2003, Department of Mathematical Statistics, Lund Institute of Technology, 1997.

[3] P. Johannesson. Rainflow cycles for switching processes with Markov structure. *Probability in the Engineering and Informational Sciences*, 12:143–175, 1998.

[4] P. Johannesson. *Rainflow Analysis of Switching Markov Loads*. PhD thesis, Mathematical Statistics, Centre for Mathematical Sciences, Lund Institute of Technology, 1999.

[5] P. Johannesson, G. Lindgren, I. Rychlik, J. Rydén, and S. Krenk. Load and fatigue analysis – final report phase I. ITM-report 1997:2, Oct. 1997.

[6] M. Matsuishi and T. Endo. Fatigue of metals subjected to varying stress, paper presented to Japan Society of Mechanical Engineers, Jukvoka, Japan. 1968.

[7] I. Rychlik. A new definition of the rainflow cycle counting method. *International Journal of Fatigue*, 9:119–121, 1987.

[8] I. Rychlik. Rain-Flow-Cycle distribution for ergodic load processes. *SIAM Journal on Applied Mathematics*, 48:662–679, 1988.

[9] I. Rychlik. On the 'narrow-band' approximation for expected fatigue damage. *Probabilistic Engineering Mechanics*, 8:1–4, 1992.

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

[11] D. A. Virkler, B. M. Hillberry, and P. K. Goel. The statistical nature of fatigue crack propagation. *Journal of Engineering Materials and Technology, ASME*, 101:148–153, 1979.