WAFO Chapter 1

November 26, 2014

1 CHAPTER 1 demonstrates some applications of WAFO

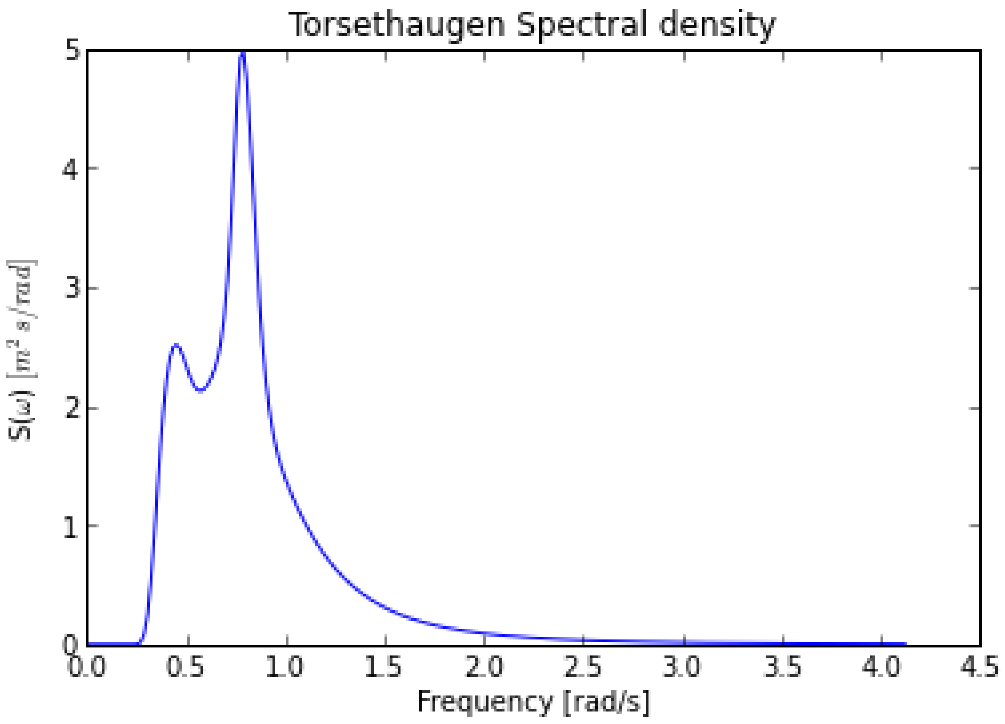
CHAPTER1 gives an overview through examples some of the capabilities of WAFO. WAFO is a toolbox of Matlab routines for statistical analysis and simulation of random waves and loads. The commands are edited for fast computation.

1.1 Section 1.4 Some applications of WAFO

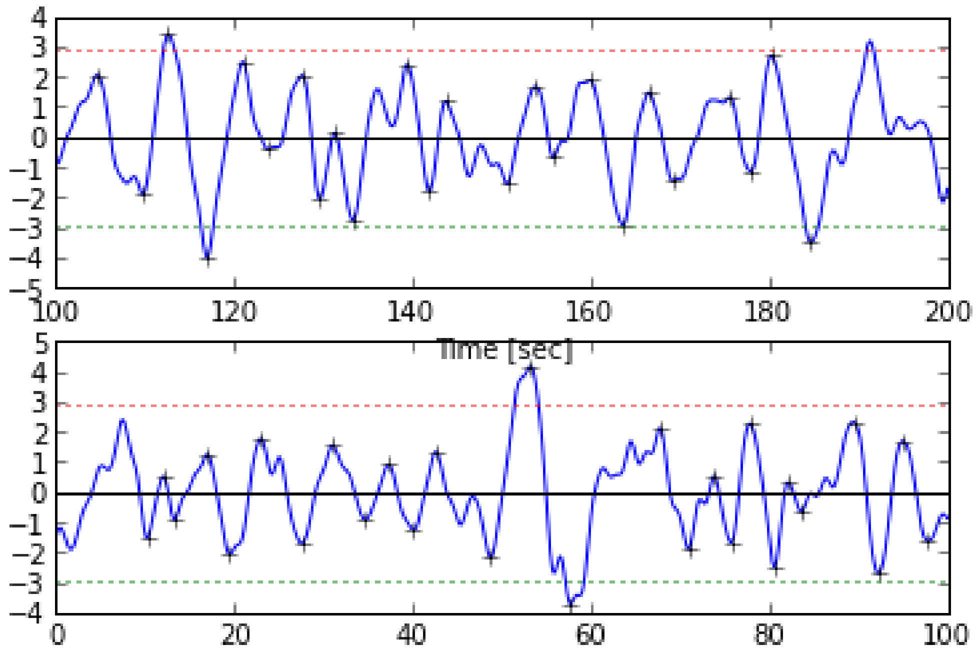
1.1.1 Section 1.4.1 Simulation from spectrum, estimation of spectrum

Simulation of the sea surface from spectrum. The following code generates 200 seconds of data sampled with 10Hz from the Torsethaugen spectrum.

```
In [5]: import wafo.spectrum.models as wsm
        S = wsm.Torsethaugen(Hm0=6, Tp=8);
        S1 = S.tospecdata()
        S1.plot()
        show()
```

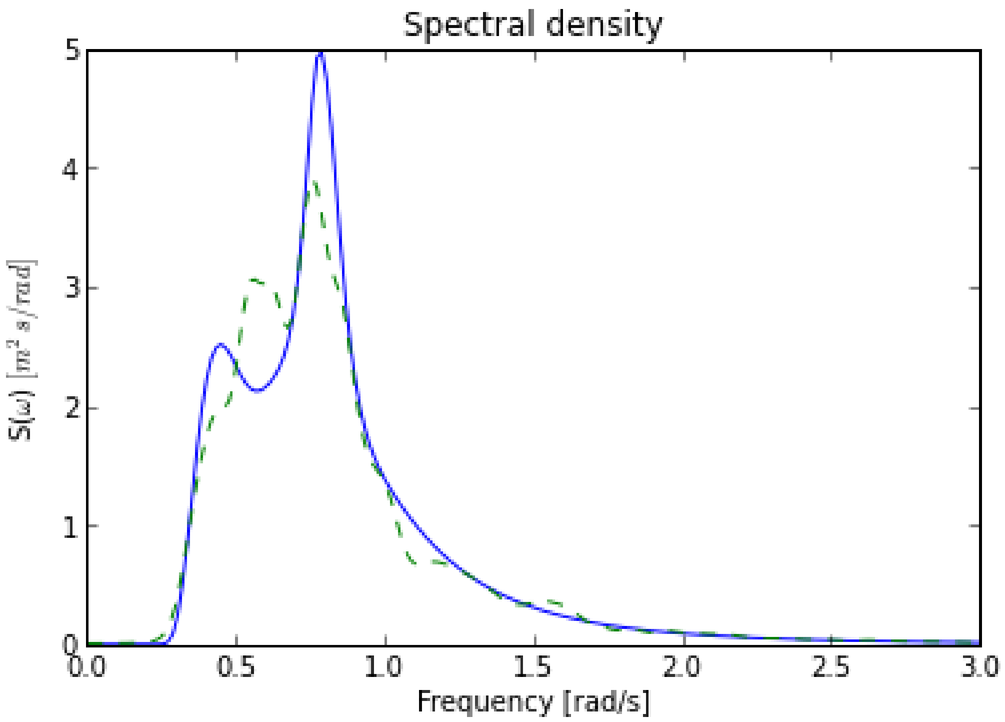


```
In [6]: import wafo.objects as wo
        xs = S1.sim(ns=2000, dt=0.1)
        ts = wo.mat2timeseries(xs)
        ts.plot_wave('-')
        show()
```



Estimation of spectrum A common situation is that one wants to estimate the spectrum for wave measurements. The following code simulate 20 minutes signal sampled at 4Hz and compare the spectral estimate with the original Torsethaugen spectum.

```
In [7]: clf()
        Fs = 4;
        xs = S1.sim(ns=fix(20 * 60 * Fs), dt=1. / Fs)
        ts = wo.mat2timeseries(xs)
        Sest = ts.tospecdata(L=400)
        S1.plot()
        Sest.plot('--')
        axis([0, 3, 0, 5]) # This may depend on the simulation
        show()
```



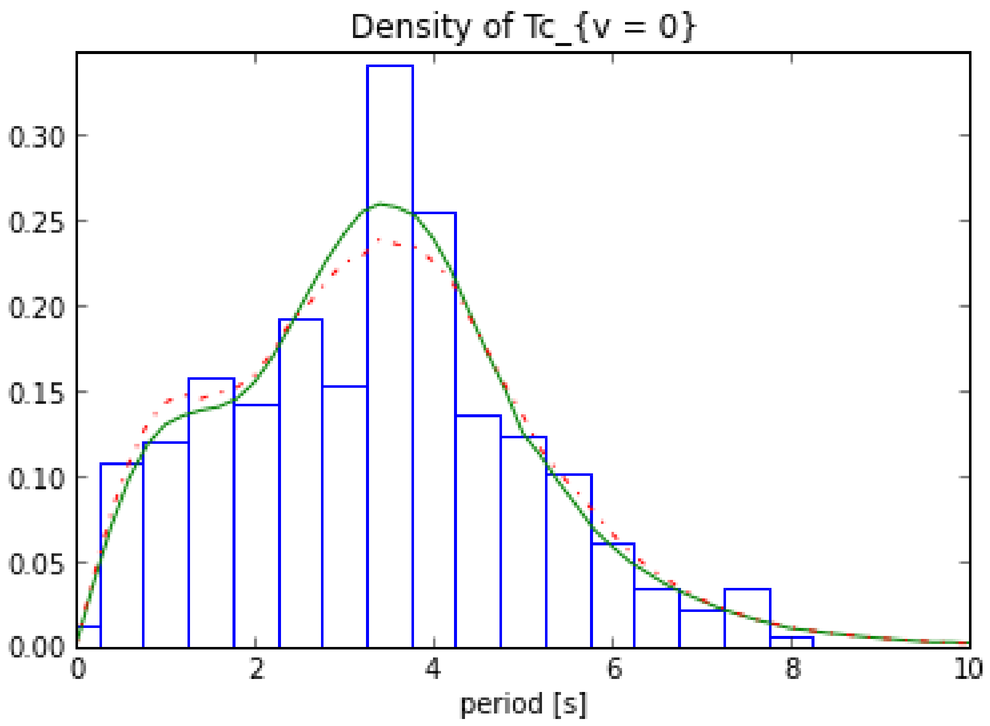
1.1.2 Section 1.4.2 Probability distributions of wave characteristics.

Probability distribution of wave trough period: WAFO gives the possibility of computing the exact probability distributions for a number of characteristics given a spectral density. In the following example we study the trough period extracted from the time series and compared with the theoretical density computed with exact spectrum, S1, and the estimated spectrum, Sest.

```
In [8]: clf()
        import wafo.misc as wm
        dtyex = S1.to_t_pdf(pdef='Tt', paramt=(0, 10, 51), nit=3)
        dtyest = Sest.to_t_pdf(pdef='Tt', paramt=(0, 10, 51), nit=3)

        T, index = ts.wave_periods(vh=0, pdef='d2u')
        bins = wm.good_bins(T, num_bins=25, odd=True)
        wm.plot_histgrm(T, bins=bins, normed=True)

        dtyex.plot()
        dtyest.plot('-.')
        axis([0, 10, 0, 0.35])
        show()
```



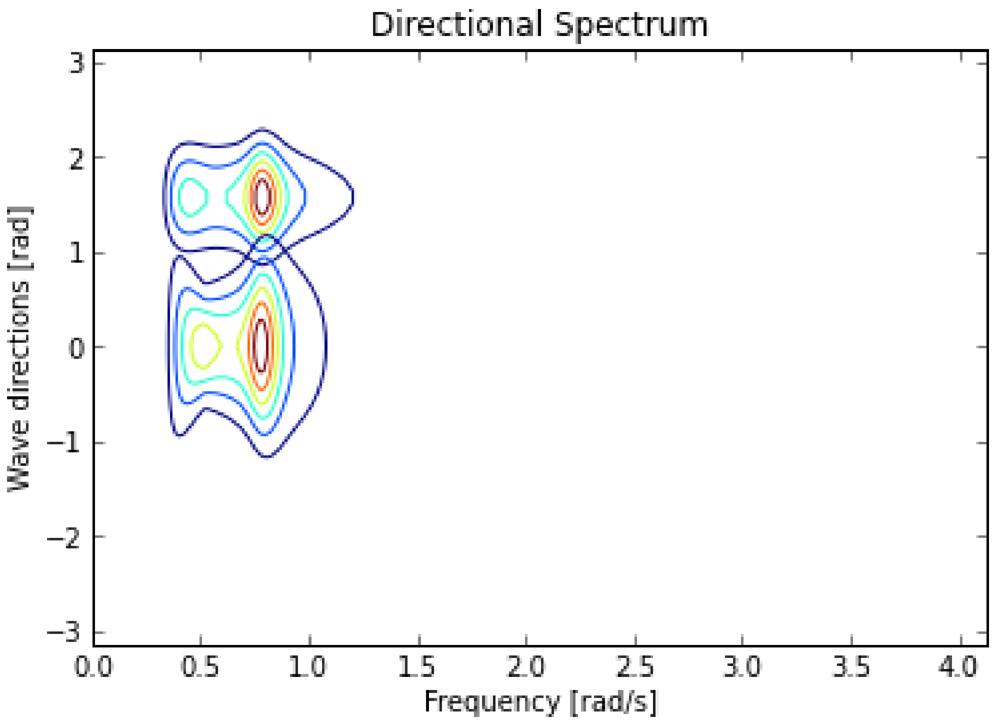
1.1.3 Section 1.4.3 Directional spectra

Here are a few lines of code, which produce directional spectra with frequency independent and frequency dependent spreading.

```
In [9]: clf()
        plotflag = 1
        Nt = 101;    # number of angles
        th0 = pi / 2; # primary direction of waves
        Sp = 15;    # spreading parameter

        D1 = wsm.Spreading(type='cos', theta0=th0, method=None) # frequency independent
        D12 = wsm.Spreading(type='cos', theta0=0, method='mitsuyasu') # frequency dependent

        SD1 = D1.tospecdata2d(S1)
        SD12 = D12.tospecdata2d(S1)
        SD1.plot()
        SD12.plot()#linestyle='dashdot')
        show()
```



3D Simulation of the sea surface The simulations show that frequency dependent spreading leads to much more irregular surface so the orientation of waves is less transparent compared to the frequency independent case.

Frequency independent spreading

```
In []: #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);
       #wafostamp('', '(ER)')
       #axis('fill')
       #disp('Block = 6'), pause(pstate)
```

Frequency dependent spreading

```
In []: #randn('state', iseed)
       #Y12 = seasim(SD12, Nx, Ny, Nt, dx, dy, dt, fftdim, plotflag);
       #wafostamp('', '(ER)')
       #axis('fill')
```

1.1.4 Estimation of directional spectrum

The figure is not shown in the Tutorial

```
In []: # Nx = 3; Ny = 2; Nt = 2 ^ 12; dx = 10; dy = 10;dt = 0.5;
       # F = seasim(SD12, Nx, Ny, Nt, dx, dy, dt, 1, 0);
```

```
        # Z = permute(F.Z, [3 1 2]);
        # [X, Y] = meshgrid(F.x, F.y);
        # N = Nx * Ny;
        # types = repmat(sensortypeid('n'), N, 1);
        # bfs = ones(N, 1);
        # pos = [X(:), Y(:), zeros(N, 1)];
        # h = inf;
        # nfft = 128;
        # nt = 101;
        # SDe = dat2dspec([F.t Z(:, :)], [pos types, bfs], h, nfft, nt);
        #plotspec(SDe), hold on
        #plotspec(SD12, '--'), hold off
        #disp('Block = 8'), pause(pstate)
```

1.1.5 Section 1.4.4 Fatigue, Load cycles and Markov models

Switching Markow chain of turningpoints. In fatigue applications the exact sample path is not important, but only the tops and bottoms of the load, called the sequence of turning points (TP). From the turning points one can extract load cycles, from which damage calculations and fatigue life predictions can be performed.

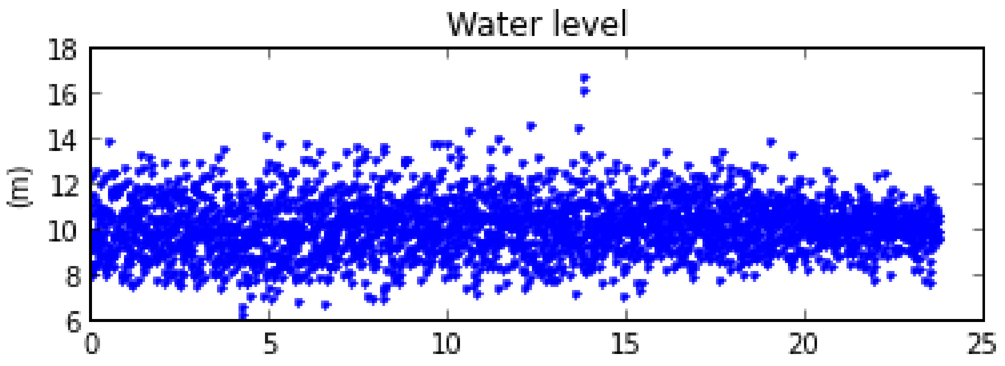
The commands below computes the intensity of rainflowcycles for the Gaussian model with spectrum S1 using the Markov approximation. The rainflow cycles found in the simulated load signal are shown in the figure.

```
In []: #clf()
       #paramu = [-6 6 61];
       #frfc = spec2cmat(S1, [], 'rfc', [], paramu);
       #pdfplot(frfc);
       #hold on
       #tp = dat2tp(xs);
       #rfc = tp2rfc(tp);
       #plot(rfc(:, 2), rfc(:, 1), '.')
       #wafostamp('', '(ER)')
       #hold off
       #disp('Block = 9'), pause(pstate)
```

1.1.6 Section 1.4.5 Extreme value statistics

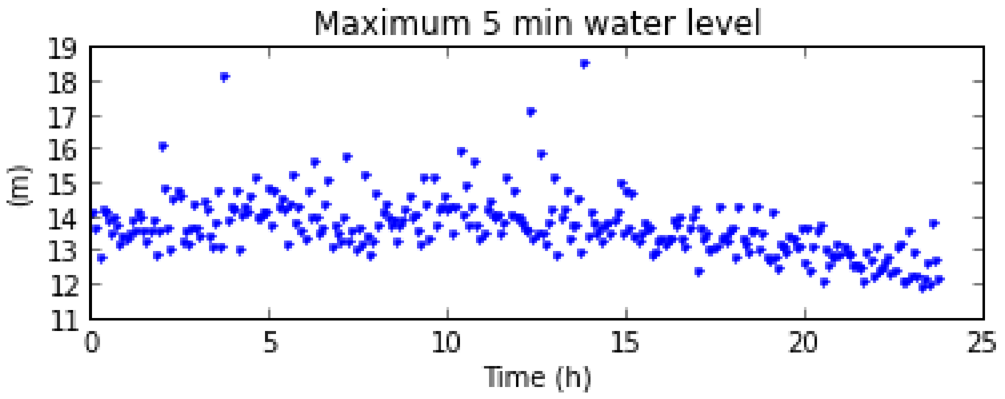
```
In [10]: clf()
         import wafo.data as wd
         xn = wd.yura87()
         #xn = load('yura87.dat');
         subplot(211)
         plot(xn[::30, 0] / 3600, xn[::30, 1], '.')
         title('Water level')
         ylabel('(m)')

Out[10]: <matplotlib.text.Text at 0x730d070>
```



Formation of 5 min maxima

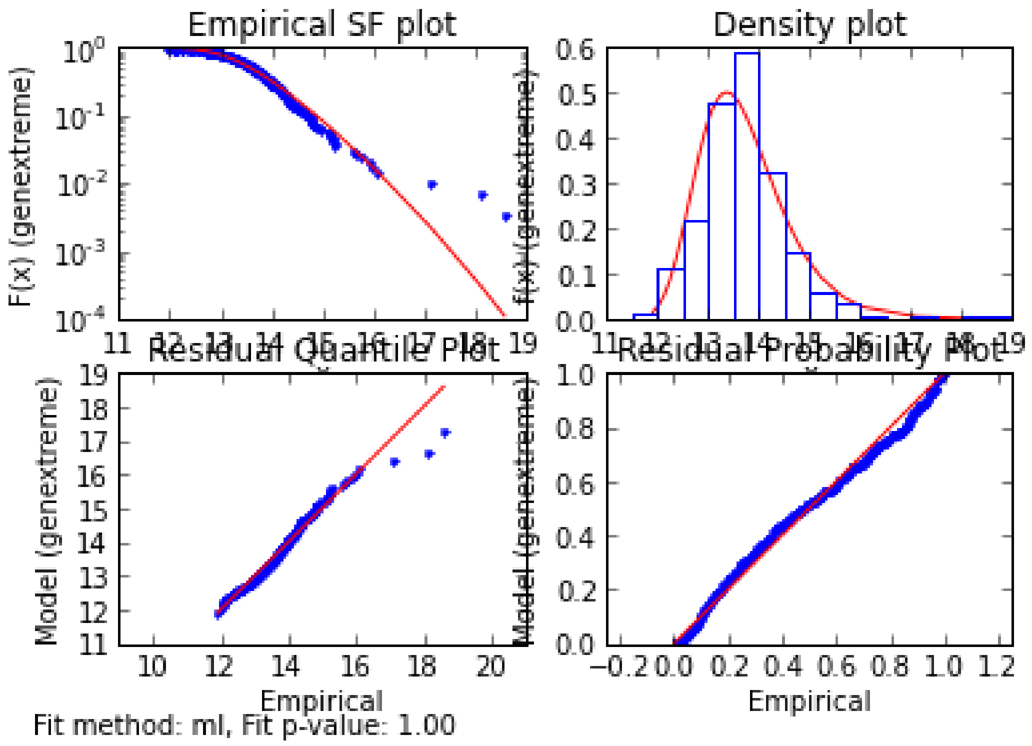
```
In [11]: yura = xn[:85500, 1]
         yura = np.reshape(yura, (285, 300)).T
         maxyura = yura.max(axis=0)
         subplot(212)
         plot(xn[299:85500:300, 0] / 3600, maxyura, '.')
         xlabel('Time (h)')
         ylabel('(m)')
         title('Maximum 5 min water level')
         show()
```



Estimation of GEV for yuramax

```
In [12]: clf()
         import wafo.stats as ws
         phat = ws.genextreme.fit2(maxyura, method='ml')
         phat.plotfitsummary()
         show()
         #disp('Block = 11, Last block')
```

```
c:\pab\workspace\pywafo_svn\pywafo\src\wafo\stats\estimation.py:1080: UserWarning: P-value is on the con
  warnings.warn('P-value is on the conservative side (i.e. too large) due to ties in the data!')
```


Empirical SF plot
F(x) (genextreme)
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
11 12 13 14 15 16 17 18 19
Density plot
f(x) (genextreme)
0.0 0.1 0.2 0.3 0.4 0.5 0.6
11 12 13 14 15 16 17 18 19
Residual Quantile Plot
Model (genextreme)
11 12 13 14 15 16 17 18 19
10 12 14 16 18 20
Empirical
Residual Probability Plot
Model (genextreme)
0.0 0.2 0.4 0.6 0.8 1.0
−0.2 0.0 0.2 0.4 0.6 0.8 1.0 1.2
Empirical
Fit method: ml, Fit p-value: 1.00