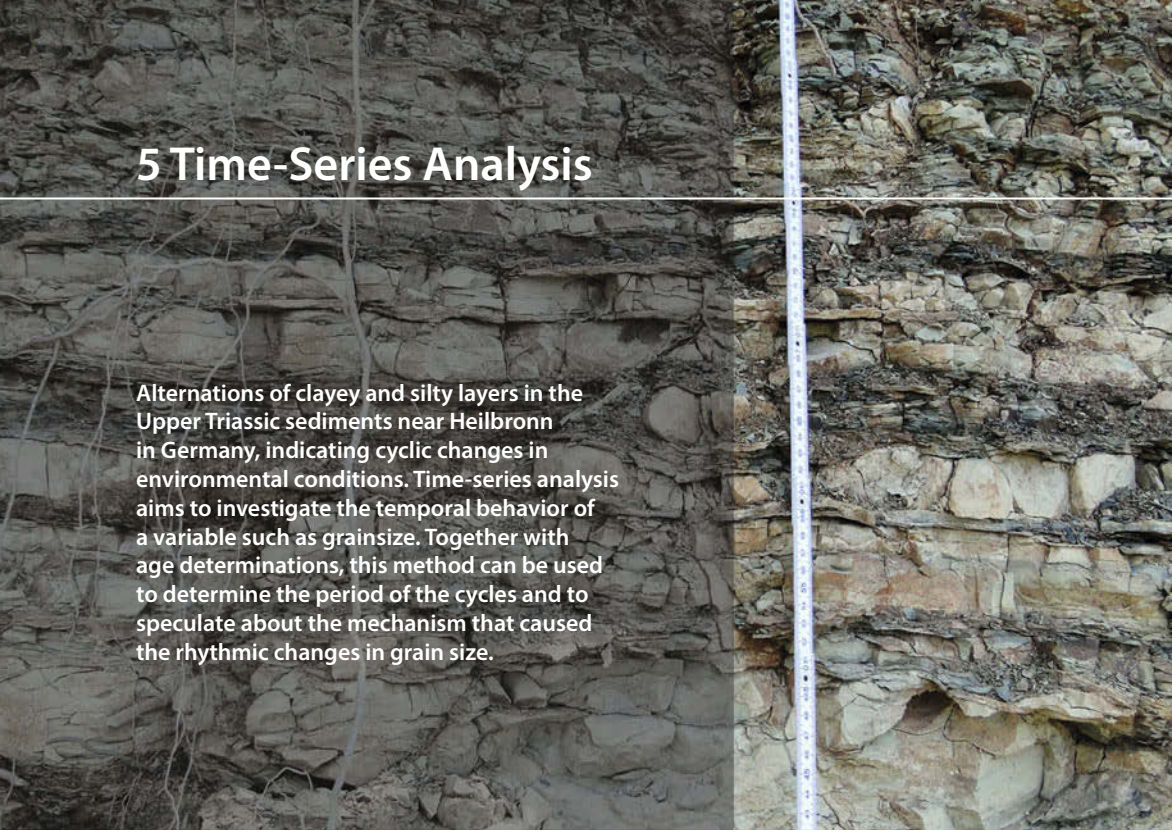
5 Time-Series Analysis



Alternations of clayey and silty layers in the Upper Triassic sediments near Heilbronn in Germany, indicating cyclic changes in environmental conditions. Time-series analysis aims to investigate the temporal behavior of a variable such as grainsize. Together with age determinations, this method can be used to determine the period of the cycles and to speculate about the mechanism that caused the rhythmic changes in grain size.

5.1 Introduction

Time-series analysis aims to investigate the temporal behavior of a variable $x(t)$. Examples include the investigation of long-term records of mountain uplift, sea-level fluctuations, orbitally-induced insolation variations and their influence on the ice-age cycles, millennium-scale variations in the atmosphere-ocean system, the effect of the El Niño/Southern Oscillation on tropical rainfall and sedimentation (Fig. 5.1), and tidal influences on noble gas emissions from bore holes. The temporal pattern of a sequence of events can be random, clustered, cyclic, or chaotic. Time-series analysis provides various tools with which to detect these temporal patterns. Understanding the underlying processes that produced the observed data allows us to predict future values of the variable. We use the Signal Processing and Wavelet Toolboxes, which contain all the necessary routines for time-series analysis (MathWorks 2014a and b).

Section 5.2 discusses signals in general and contains a technical description of how to generate synthetic signals for time-series analysis. The use of spectral analysis to detect cyclicities in a single time series (auto-spectral analysis) and to determine the relationship between two time series

M.H. Trauth, *MATLAB® Recipes for Earth Sciences*,
DOI 10.1007/978-3-662-46244-7_5

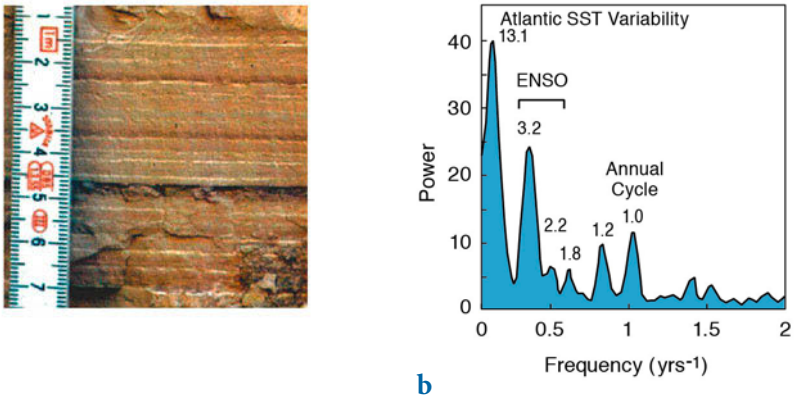


Fig. 5.1 a Photograph of ca. 30 kyr-old varved sediments from a lake in the Andes of Northwest Argentina. The distribution of the source rocks and the interannual precipitation pattern in the area suggest that the reddish-brown layers reflect cyclic recurrences of enhanced precipitation, erosion, and sediment input into the lake. **b** The power spectrum of a red-color intensity transect across 70 varves is dominated by significant peaks at frequencies of ca. 0.076, 0.313, 0.455 and 1.0 yrs^{-1}. These cyclicities suggest a strong influence of the tropical Atlantic sea-surface temperature (SST) variability, the El Niño/Southern Oscillation (ENSO), and the annual cycle that occurred 30 kyrs ago, similar to today's cyclicities (Trauth et al. 2003).

as a function of frequency (cross-spectral analysis) is then demonstrated in Sections 5.3 and 5.4. Since most time series in earth sciences have uneven time intervals, various interpolation techniques and subsequent methods of spectral analysis are required, and these are introduced in Section 5.5. Evolutionary power spectra to map changes in cyclicity through time are demonstrated in Section 5.6. An alternative technique for analyzing unevenly-spaced data is explained in Section 5.7. Section 5.8 introduces the very popular wavelet power spectrum, which is able to map temporal variations in the spectra in a similar way to the method demonstrated in Section 5.6. Section 5.9 then introduces a non-parametric method to detect abrupt transitions in central tendency and dispersion within time series. This chapter closes with an overview of nonlinear techniques, in particular the method of recurrence plots (Section 5.10).

5.2 Generating Signals

A time series is an ordered sequence of values of a variable $x(t)$ at certain times t_k.

$$x(t_k) = x(t_1), x(t_2), \ldots, x(t_N)$$

If the time interval between any two successive observations $x(t_k)$ and $x(t_{k+1})$ is constant, the time series is said to be equally spaced and the sampling interval is

$$\Delta t = t_{k+1} - t_k$$

The sampling frequency f_s is the inverse of the sampling interval Δt. We generally try to sample at regular time intervals or constant sampling frequencies, but in many earth science examples this is not possible. As an example, imagine deep-sea sediments sampled at five-centimeter intervals along a sediment core. Radiometric age determinations at certain levels in the sediment core revealed significant fluctuations in the sedimentation rates. Despite the samples being evenly spaced along the sediment core they are not equally spaced on the time axis. Here, the quantity

$$\Delta t = T / N$$

where T is the full length of the time series and N is the number of data points, represents only an average sampling interval. In general, a time series $x(t_k)$ can be represented as the linear sum of a periodic component $x_p(t_k)$, a long-term component or trend $x_{tr}(t_k)$, and random noise $x_n(t_k)$.

$$x(t_k) = x_p(t_k) + x_{tr}(t_k) + x_n(t_k)$$

The long-term component is a linear or higher-degree trend that can be extracted by fitting a polynomial of a certain degree and subtracting the values of this polynomial from the data (see Chapter 4). Noise removal will be described in Chapter 6. The periodic – or cyclic in a mathematically less rigorous sense – component can be approximated by a linear combination of sine (or cosine) waves that have different amplitudes A_i, frequencies f_i, and phase angles ψ_i.

$$x_p(t_k) = \sum_i A_i \cdot \sin(2\pi f_i t_k - \psi_i)$$

The phase angle ψ helps to detect temporal shifts between signals of the same frequency. Two signals x and y with the same period are out of phase unless the difference between ψ_x and ψ_y is equal to zero (Fig. 5.2).

The frequency f of a periodic signal is the inverse of the period τ. The *Nyquist frequency* f_{nyq} is half the sampling frequency f_s and represents the maximum frequency the data can produce. As an example audio compact

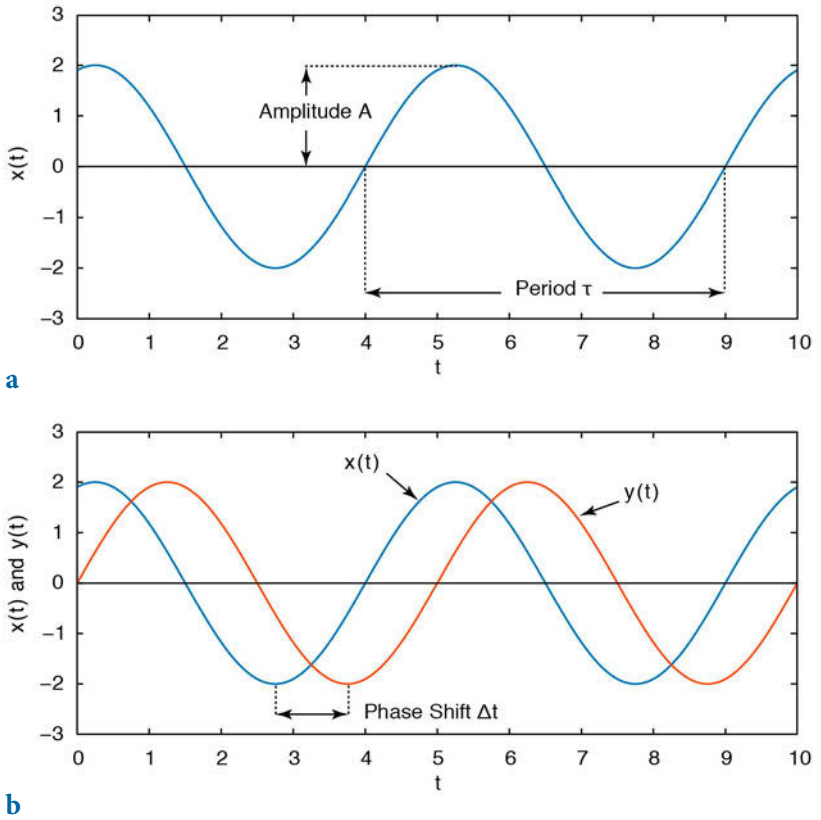


Fig. 5.2 a Periodic signal x a function of time t defined by the amplitude A, and the period τ which is the inverse of the frequency f. **b** Two signals x and y of the same period are out of phase if the difference between ψ_x and ψ_y is not equal to zero.

disks (CDs) are sampled at frequencies of 44,100 Hz (Hertz, where 1 Hz=1 cycle per second), but the corresponding Nyquist frequency is 22,050 Hz, which is the highest frequency a CD player can theoretically produce. The performance limitations of anti-alias filters used by CD players further reduce the frequency band and result in a cutoff frequency of around 20,050 Hz, which is the true upper frequency limit of a CD player.

We can now generate synthetic signals to illustrate the use of time-series analysis tools. When using synthetic data we know in advance which features the time series contains, such as periodic or random components, and we can introduce a linear trend or gaps in the time series. The user will encounter plenty of examples of the possible effects of varying the parameter settings, as well as potential artifacts and errors that can result from the application of spectral analysis tools. We will start with simple data and then apply the

methods to more complex time series. The first example illustrates how to generate a basic synthetic data series that is characteristic of earth science data. First, we create a time axis `t` running from 1 to 1000 in steps of one unit, i.e., the sampling frequency is also one. We then generate a simple periodic signal `y`: a sine wave with a period of five and an amplitude of two by typing

```
clear

t = 1 : 1000;
x = 2*sin(2*pi*t/5);

plot(t,x), axis([0 200 -4 4])
```

The period of τ=5 corresponds to a frequency of f=1/5=0.2. Natural data series, however, are more complex than a simple periodic signal. The slightly more complicated signal can be generated by superimposing several periodic components with different periods. As an example we compute such a signal by adding three sine waves with the periods τ_1=50 (f_1=0.02), τ_2=15 (f_2≈0.07) and τ_3=5 (f_3=0.2). The corresponding amplitudes are A_1=2, A_2=1 and A_3=0.5.

```
t = 1 : 1000;
x = 2*sin(2*pi*t/50) + sin(2*pi*t/15) + 0.5*sin(2*pi*t/5);

plot(t,x), axis([0 200 -4 4])
```

By restricting the t-axis to the interval [0,200], only one fifth of the original data series is displayed (Fig. 5.3 a). It is, however, recommended that long data series be generated, as in the example, in order to avoid edge effects when applying spectral analysis tools for the first time.

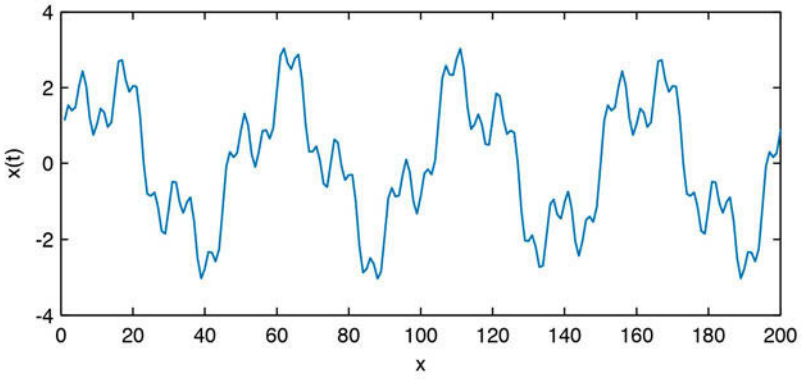
In contrast to our synthetic time series, real data also contain various disturbances, such as random noise and first or higher-order trends. In order to reproduce the effects of noise, a random-number generator can be used to compute Gaussian noise with mean of zero and standard deviation of one. The seed of the algorithm should be set to zero using `rng(0)`. One thousand random numbers are then generated using the function `randn`.

```
rng(0)
n = randn(1,1000);
```

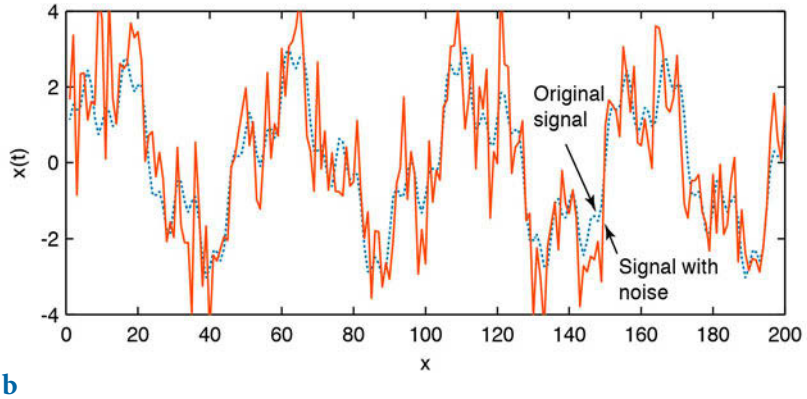
We add this noise to the original data, i.e., we generate a signal containing additive noise (Fig. 5.3 b). Displaying the data illustrates the effect of noise on a periodic signal. Since in reality no record is totally free of noise it is important to familiarize oneself with the influence of noise on power spectra.

Audio 5.1

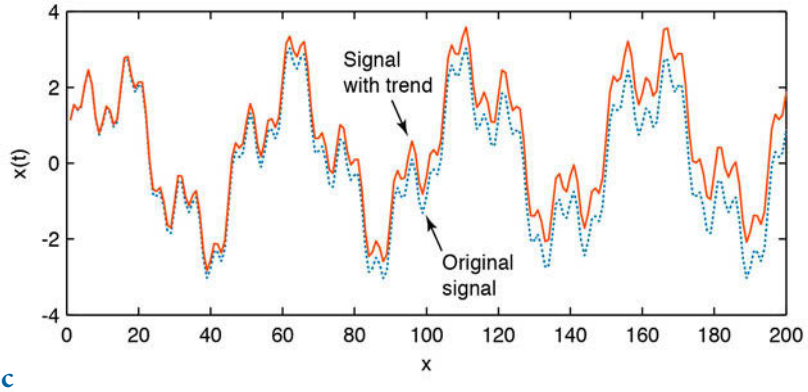
```
xn = x + n;

plot(t,x,'b-',t,xn,'r-'), axis([0 200 -4 4])
```



a



b



c

Fig. 5.3 **a** Synthetic signal with the periodicities τ_1=50, τ_2=15 and τ_3=5, with different amplitudes, and **b** the same signal overprinted with Gaussian noise. **c** The time series shows a significant linear trend.

Signal processing methods are often applied to remove a major part of the noise, although many filtering methods make arbitrary assumptions concerning the signal-to-noise ratio. Moreover, filtering introduces artifacts and statistical dependencies into the data, which may have a profound influence on the resulting power spectra.

Finally, we introduce a linear long-term trend to the data by adding a straight line with a slope of 0.005 and an intercept with the *y*-axis of zero (Fig. 5.3 c). Such trends are common in earth sciences. As an example, consider the glacial-interglacial cycles observed in marine oxygen isotope records, overprinted on a long-term cooling trend over the last six million years.

```
xt = x + 0.005*t;

plot(t,x,'b-',t,xt,'r-'), axis([0 200 -4 4])
```

In reality, more complex trends exist, such as higher-order trends or trends characterized by variations in gradient. In practice, it is recommended that such trends be eliminated by fitting polynomials to the data and subtracting the corresponding values. Our synthetic time series now contains many characteristics of a typical earth science data set. It can be used to illustrate the use of the spectral analysis tools that are introduced in the next section.


Audio 5.2

5.3 Auto-Spectral and Cross-Spectral Analysis

Auto-spectral analysis aims to describe the distribution of variance contained in a single signal $x(t)$ as a function of frequency or wavelength. A simple way to describe the variance in a signal over a time lag k is by means of the autocovariance. An unbiased estimator of the autocovariance cov_{xx} of the signal $x(t)$ with N data points sampled at constant time intervals Δt is

$$cov_{xx}(k) = \frac{1}{N-k-1} \sum_{t=1}^{N-k} (x_i - \bar{x})(x_{i+k} - \bar{x})$$

The autocovariance series clearly depends on the amplitude of $x(t)$. Normalizing the covariance by the variance σ^2 of $x(t)$ yields the autocorrelation sequence. Autocorrelation involves correlating a series of data with itself as a function of a time lag k.

$$corr_{xx}(k) = \frac{cov_{xx}(k)}{cov_{xx}(0)} = \frac{cov_{xx}(k)}{\sigma_x^{\,2}}$$

A popular method used to compute power spectra in earth sciences is the method introduced by Blackman and Tukey (1958). The *Blackman-Tukey method* uses the complex Fourier transform $X_{xx}(f)$ of the autocorrelation sequence $corr_{xx}(k)$,

$$X_{xx}(f) = \sum_{k=0}^{M} corr_{xx}(k)\, e^{i 2\pi fk/f_s}$$

where M is the maximum lag and f_s the sampling frequency. The Blackman-Tukey auto-spectrum is the absolute value of the Fourier transform of the autocorrelation function. In some fields, the *power spectral density* is used as an alternative way of describing the auto-spectrum. The Blackman-Tukey power spectral density *PSD* is estimated by

$$PSD = \frac{X^*_{xx}(f) X_{xx}(f)}{f_s} = \frac{\left|X_{xx}(f)\right|^2}{f_s}$$

where $X^*_{xx}(f)$ is the conjugate complex of the Fourier transform of the autocorrelation function $X_{xx}(f)$ and f_s is the sampling frequency. The actual computation of the power spectrum can only be performed at a finite number of different frequencies by employing a *Fast Fourier Transformation* (FFT). The FFT is a method of computing a discrete Fourier transform with reduced execution time. Most FFT algorithms divide the transform into two portions of size $N/2$ at each step of the transformation. The transform is therefore limited to blocks with dimensions equal to a power of two. In practice, the *spectrum* is computed by using a number of frequencies that is close to the number of data points in the original signal $x(t)$.

The discrete Fourier transform is an approximation of the continuous Fourier transform. The continuous Fourier transform assumes an infinite signal but discrete real data are limited at both ends, i.e., the signal amplitude is zero beyond either end of the time series. In the time domain, a finite signal corresponds to an infinite signal multiplied by a rectangular window that has a value of one within the limits of the signal and a value of zero elsewhere. In the frequency domain, the multiplication of the time series by this window is equivalent to a convolution of the power spectrum of the signal with the spectrum of the rectangular window (see Section 6.4 for a definition of convolution). The spectrum of the window, however, is a $\sin(x)/x$ function, which has a main lobe and numerous side lobes on either side of the main peak, and hence all maxima in a power spectrum *leak*, i.e., they lose power on either side of the peaks (Fig. 5.4).

A popular way to overcome the problem of *spectral leakage* is by windowing, in which the sequence of data is simply multiplied by a smooth bell-shaped curve with positive values. Several window shapes are available, e.g., *Bartlett* (triangular), *Hamming* (cosinusoidal) and *Hanning* (slightly different cosinusoidal) (Fig. 5.4). The use of these windows slightly modifies the equation for the Blackman-Tukey auto-spectrum to

$$X_{xx}(f) = \sum_{k=0}^{M} corr_{xx}(k) w(k)\, e^{i2\pi fk/f_s}$$

where $w(k)$ is the windowing function. The Blackman-Tukey method therefore performs auto-spectral analysis in three steps: calculation of the autocorrelation sequence $corr_{xx}(k)$, windowing and, finally, computation of the discrete Fourier transform. MATLAB allows power spectral analysis to be performed with a number of modifications to the above method. One useful modification is the Welch method (Welch 1967) (Fig. 5.5). This method involves dividing the time series into overlapping segments, computing the power spectrum for each segment, and then averaging the power spectra. The advantage of averaging the spectra is obvious: it simply improves the signal-to-noise ratio of a spectrum. The disadvantage is a loss of resolution in the spectra.

Cross-spectral analysis correlates two time series in the frequency domain.

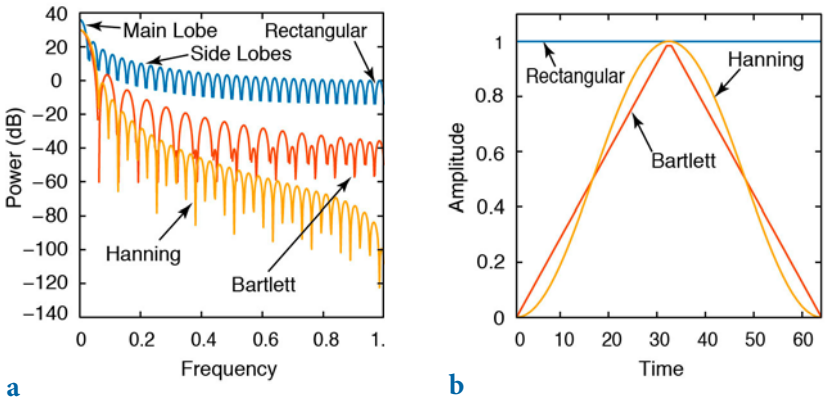


Fig. 5.4 Spectral leakage. **a** The amplitudes of the side lobes relative to that of the main lobe are reduced by multiplying the corresponding time series by **b** a smooth bell-shaped window function. A number of different windows with advantages and disadvantages are available for use instead of the default rectangular window, including *Bartlett* (triangular) and *Hanning* (cosinusoidal) windows. Graph generated using the function `wvtool`.

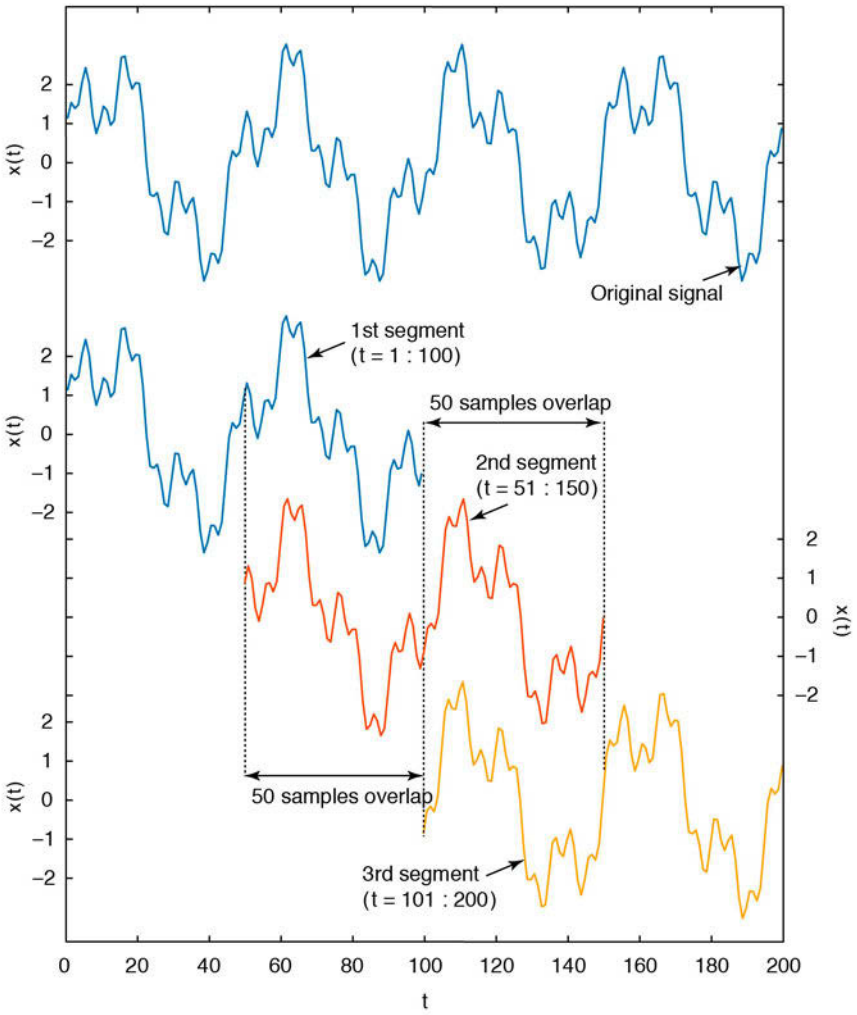


Fig. 5.5 Principle of Welch's power spectral analysis. The time series is first divided into overlapping segments; the power spectrum for each segment is then computed and all spectra are averaged to improve the signal-to-noise ratio of the power spectrum.

The cross-covariance is a measure of the variance between two signals over a time lag k. An unbiased estimator of the cross-covariance cov_{xy} of two signals, $x(t)$ and $y(t)$, with N data points sampled at constant time intervals Δt, is

$$cov_{xy}(k) = \frac{1}{N-k-1} \sum_{t=1}^{N-k} (x_i - \bar{x})(y_{i+k} - \bar{y})$$

The cross-covariance series again depends on the amplitudes of $x(t)$ and $y(t)$. Normalizing the covariance by the standard deviations of $x(t)$ and $y(t)$ yields the cross-correlation sequence.

$$corr_{xy}(k) = \frac{cov_{xy}(k)}{cov_{xy}(0)} = \frac{cov_{xy}(k)}{\sigma_x \sigma_y}$$

The *Blackman-Tukey method* uses the complex Fourier transform $X_{xy}(f)$ of the cross-correlation sequence $corr_{xy}(k)$

$$X_{xy}(f) = \sum_{k=0}^{M} corr_{xy}(k)\, e^{i2\pi fk/f_s}$$

where M is the maximum lag and f_s the sampling frequency. The absolute value of the complex Fourier transform $X_{xy}(f)$ is the cross-spectrum while the angle of $X_{xy}(f)$ represents the phase spectrum. The phase difference is important in calculating leads and lags between two signals, a parameter often used to propose causalities between two processes documented by the signals. The correlation between two spectra can be calculated by means of the coherence:

$$C_{xy} = \frac{\left|X_{xy}(f)\right|^2}{X_{xx}(f)X_{yy}(f)}$$

The coherence is a real number between 0 and 1, where 0 indicates no correlation and 1 indicates maximum correlation between $x(t)$ and $y(t)$ at the frequency f. A significant degree of coherence is an important precondition for computing phase shifts between two signals.

5.4 Examples of Auto-Spectral and Cross-Spectral Analysis

The Signal Processing Toolbox provides numerous methods for computing spectral estimators for time series. The introduction of object-oriented programming with MATLAB has led to the launch of a new set of functions performing spectral analyses. Type `help spectrum` for more information about object-oriented spectral analysis. The non-object-oriented functions to perform spectral analyses, however, are still available. One of the oldest functions in this toolbox is `periodogram(x,window,nfft,fs)` which computes the power spectral density `Pxx` of a time series `x(t)` using the periodogram

method. This method was invented by Arthur Schuster in 1898 for studying the climate and calculates the power spectrum by performing a Fourier transform directly on a sequence without requiring prior calculation of the autocorrelation sequence. The periodogram method can therefore be considered a special case of the Blackman and Tukey (1958) method, applied with the time lag k set to unity (Muller and Macdonald 2000). At the time of its introduction in 1958, the indirect computation of the power spectrum via an autocorrelation sequence was faster than calculating the Fourier transformation for the full data series `x(t)` directly. After the introduction of the Fast Fourier Transform (FFT) by Cooley and Tukey (1965), and subsequent faster computer hardware, the higher computing speed of the Blackman-Tukey approach compared to the periodogram method became relatively unimportant.

For this next example we again use the synthetic time series `x`, `xn` and `xt` generated in Section 5.2 as the input:

```
clear

t = 1 : 1000; t = t';
x = 2*sin(2*pi*t/50) + sin(2*pi*t/15) + 0.5*sin(2*pi*t/5);

randn('seed',0)
n = randn(1000,1);
xn = x + n;

xt = x + 0.005*t;
```

We then compute the periodogram by calculating the Fourier transform of the sequence `x`. The fastest possible Fourier transform using `fft` computes the Fourier transform for `nfft` frequencies, where `nfft` is the next power of two closest to the number of data points `n` in the original signal `x`. Since the length of the data series is `n=1000`, the Fourier transform is computed for `nfft=1024` frequencies, while the signal is padded with `nfft-n=24` zeros.

```
Xxx = fft(x,1024);
```

If `nfft` is even, as in our example, then `Xxx` is symmetric. For example, as the first `(1+nfft/2)` points in `Xxx` are unique, the remaining points are symmetrically redundant. The power spectral density is defined as `Pxx2=(abs(Xxx).^2)/Fs`, where `Fs` is the sampling frequency. The function `periodogram` also scales the power spectral density by the length of the data series, i.e., it divides by `Fs=1` and `length(x)=1000`.

```
Pxx2 = abs(Xxx).^2/1000;
```

We now drop the redundant part in the power spectrum and use only the first `(1+nfft/2)` points. We also multiply the power spectral density by two to keep the same energy as in the symmetric spectrum, except for the first data point.

```
Pxx = [Pxx2(1); 2*Pxx2(2:512)];
```

The corresponding frequency axis runs from `0` to `Fs/2` in `Fs/(nfft-1)` steps, where `Fs/2` is the Nyquist frequency. Since `Fs=1` in our example, the frequency axis is

```
f = 0 : 1/(1024-1) : 1/2;
```

We then plot the power spectral density `Pxx` in the Nyquist frequency range from `0` to `Fs/2`, which in our example is from `0` to `1/2`. The Nyquist frequency range corresponds to the first 512 or `nfft/2` data points. We can plot the power spectral density over the frequency by typing

```
plot(f,Pxx), grid
```

The graphical output shows that there are three significant peaks at the positions of the original frequencies of the three sine waves (1/50, 1/15, and 1/5). Alternatively, we can also plot the power spectral density over the period by typing

```
plot(1./f,Pxx), axis([0 100 0 1000]), grid
```

where we observe the three periods 50, 15, and 5, as expected. Since the values on the *x*-axis of this plot are not evenly spaced (in constrast to those on the frequency axis), we find the long periods poorly resolved and a broad peak at a period of 50 in this graphics. The code for the power spectral density can be rewritten to make it independent of the sampling frequency,

```
Fs = 1;

t = 1/Fs :1/Fs : 1000/Fs; t = t';
x = 2*sin(2*pi*t/50) + sin(2*pi*t/15) + 0.5*sin(2*pi*t/5);

nfft = 2^nextpow2(length(t));
Xxx = fft(x,nfft);

Pxx2 = abs(Xxx).^2 /Fs /length(x);
Pxx = [Pxx2(1); 2*Pxx2(2:512)];
f = 0 : Fs/(nfft-1) : Fs/2;

plot(f,Pxx), grid
axis([0 0.5 0 max(Pxx)])
```

where the function `nextpow2` computes the next power of two closest to the length of the time series `x(t)`. This code allows the sampling frequency to be modified and the differences in the results to be explored. We can now compare the results with those obtained using the function `periodogram(x,window,nfft,fs)`.

```
[Pxx,f] = periodogram(x,[],1024,1);
```

This function allows the windowing of the signals with various window shapes to overcome spectral leakage. However, we use the default rectangular window by choosing an empty vector `[]` for `window` to compare the results with the above experiment. The power spectrum `Pxx` is computed using an FFT of length `nfft=1024`, which is the next power of two closest to the length of the series `x(t)` and which is padded with zeros to make up the number of data points to the value of `nfft`. A sampling frequency `fs` of one is used within the function in order to obtain the correct frequency scaling for the *f*-axis. We display the results by typing

```
plot(f,Pxx), grid
xlabel('Frequency')
ylabel('Power')
title('Auto-Spectrum')
```

or alternatively

```
plot(1./f,Pxx), axis([0 100 0 1000]), grid
xlabel('Period')
ylabel('Power')
title('Auto-Spectrum')
```

The graphical output is almost identical to our Blackman-Tukey plot and again shows that there are three significant peaks at the positions of the original frequencies (or periods) of the three sine waves. The same procedure can also be applied to the noisy data:

```
[Pxx,f] = periodogram(xn,[],1024,1);

plot(f,Pxx), grid
xlabel('Frequency')
ylabel('Power')
title('Auto-Spectrum')
```

Let us now increase the noise level by introducing Gaussian noise with a mean of zero and a standard deviation of five.

```
rng(0)
n = 5 * randn(size(x));
xn = x + n;
```

```
[Pxx,f] = periodogram(xn,[],1024,1);

plot(f,Pxx), grid
xlabel('Frequency')
ylabel('Power')
title('Auto-Spectrum')
```

This spectrum now resembles a real data spectrum in the earth sciences and the spectral peaks are set against a significant background noise level. The peak of the highest frequency even disappears into the noise and cannot be distinguished from maxima that are attributed to noise. Both spectra can be compared on the same plot (Fig. 5.6):

```
[Pxx,f] = periodogram(x,[],1024,1);
[Pxxn,f] = periodogram(xn,[],1024,1);

subplot(1,2,1)
plot(f,Pxx), grid
xlabel('Frequency')
ylabel('Power')

subplot(1,2,2)
plot(f,Pxxn), grid
xlabel('Frequency')
ylabel('Power')
```

Next, we explore the influence of a linear trend on a spectrum. Long-term trends are common features in earth science data. We will see that this trend is misinterpreted as a very long period by the FFT, producing a large peak

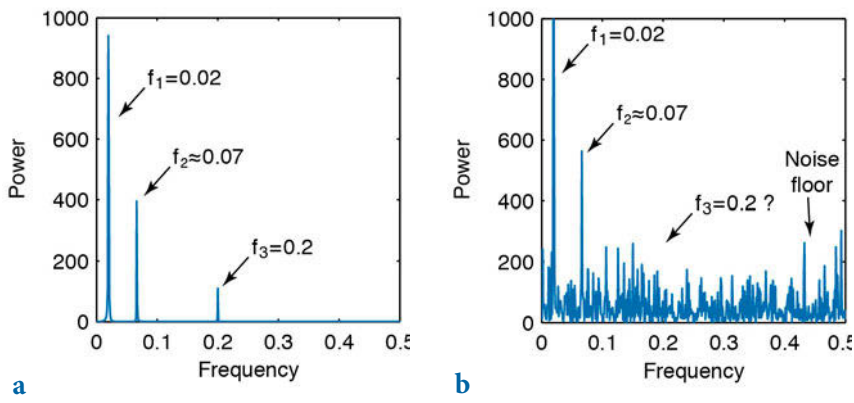


Fig. 5.6 Comparison of the auto-spectra for **a** the noise-free, and **b** the noisy synthetic signals with the periods τ_1=50 (f_1=0.02), τ_2=15 (f_2≈0.07) and τ_3=5 (f_3=0.2). The highest frequency peak disappears completely into the background noise and cannot be distinguished from peaks attributed to the Gaussian noise.

with a frequency close to zero (Fig. 5.7).

```
[Pxx,f] = periodogram(x,[],1024,1);
[Pxxt,f] = periodogram(xt,[],1024,1);

subplot(1,2,1)
plot(f,Pxx), grid
xlabel('Frequency')
ylabel('Power')

subplot(1,2,2)
plot(f,Pxxt), grid
xlabel('Frequency')
ylabel('Power')
```

To eliminate the long-term trend, we use the function `detrend`. This function removes linear trends, defined as either a single straight-line fit from the vector `x`, or a continuous, piecewise linear trend from `x` with one or more breakpoints defined by the user.

```
xdt = detrend(xt);

subplot(2,1,1)
plot(t,x,'b-',t,xt,'r-'), grid
axis([0 200 -4 4])

subplot(2,1,2)
plot(t,x,'b-',t,xdt,'r-'), grid
axis([0 200 -4 4])
```

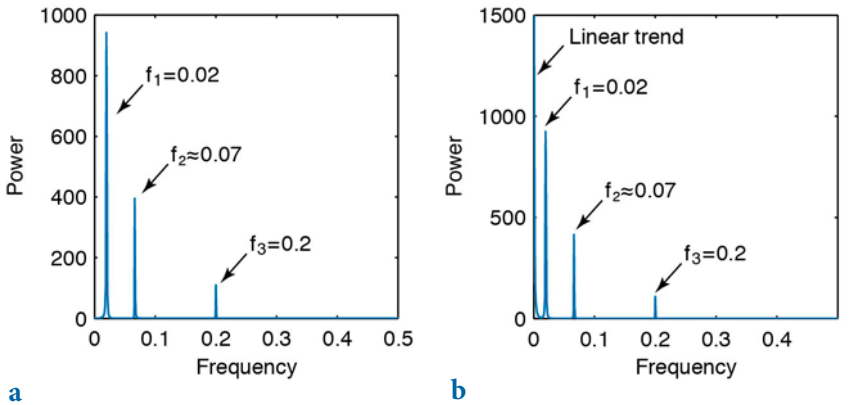


Fig. 5.7 Comparison of the auto-spectra for **a** the original noise-free signal with the periods τ_1=50 (f_1=0.02), τ_2=15 (f_2≈0.07) and τ_3=5 (f_3=0.2), and **b** the same signal overprinted on a linear trend. The linear trend is misinterpreted by the FFT as a very long period with a high amplitude.

The resulting spectrum no longer shows the low-frequency peak.

```
[Pxxt,f] = periodogram(xt,[],1024,1);
[Pxxdt,f] = periodogram(xdt,[],1024,1);

subplot(1,2,1)
plot(f,Pxx), grid
xlabel('Frequency')
ylabel('Power')

subplot(1,2,2)
plot(f,Pxxdt), grid
xlabel('Frequency')
ylabel('Power')
```

Some data contain a high-order trend that can be removed by fitting a higher-order polynomial to the data and subtracting the corresponding $x(t)$ values.

We now use two sine waves with identical periodicities τ=5 (equivalent to f=0.2) and amplitudes equal to two to compute the cross-spectrum of two time series. The sine waves show a relative phase shift of t=1. In the argument of the second sine wave this corresponds to $2\pi/5$, which is one fifth of the full wavelength of τ=5.

```
clear

t = 1 : 1000;
x = 2*sin(2*pi*t/5);
y = 2*sin(2*pi*t/5 + 2*pi/5);

plot(t,x,'b-',t,y,'r-')
axis([0 50 -2 2]), grid
```

The cross-spectrum is computed by using the function `cpsd`, which uses Welch's method for computing power spectra (Fig. 5.8). `Pxy` is complex and contains both amplitude and phase information.

```
[Pxy,f] = cpsd(x,y,[],0,1024,1);

plot(f,abs(Pxy)), grid
xlabel('Frequency')
ylabel('Power')
title('Cross-Spectrum')
```

The function `cpsd(x,y,window,noverlap,nfft,fs)` specifies the number of FFT points `nfft` used to calculate the cross power spectral density, which is 1024 in our example. The parameter `window` is empty in our example and the default rectangular window is therefore used to obtain eight sections of `x` and `y`. The parameter `noverlap` defines the number of overlapping samples, which is zero in our example. The sampling frequency `fs` is 1 in this example. The

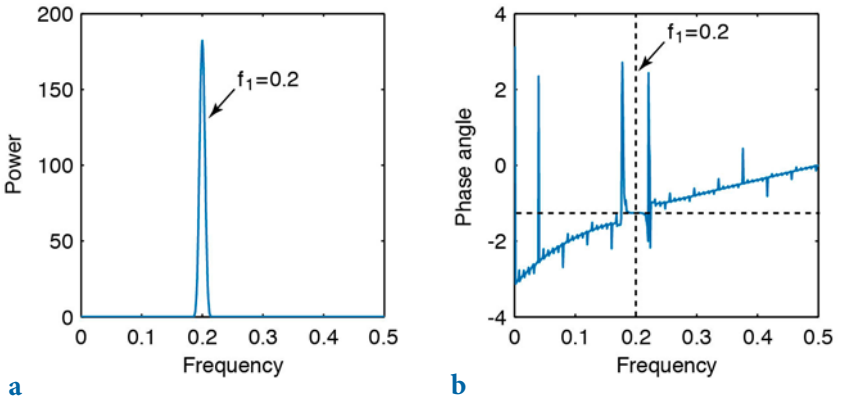


Fig. 5.8 Cross-spectrum of two sine waves with identical periodicities of τ=5 (equivalent to f=0.2) and amplitudes of 2. The sine waves show a relative phase shift of t=1. In the argument of the second sine wave this corresponds to $2\pi/5$, which is one fifth of the full wavelength of τ=5. **a** The magnitude shows the expected peak at f=0.2. **b** The corresponding phase difference in radians at this frequency is 1.2566, which equals $(1.2566 \cdot 5)/(2 \cdot \pi) = 1.0000$, which is the phase shift of 1 that we introduced initially.

coherence of the two signals is one for all frequencies, since we are working with noise-free data.

```
[Cxy,f] = mscohere(x,y,[],0,1024,1);

plot(f,Cxy), grid
xlabel('Frequency')
ylabel('Coherence')
title('Coherence')
```

We use the function `mscohere(x,y,window,noverlap,nfft,fs)` which specifies the number of FFT points `nfft`=1024, the default rectangular window (`window`=`[]`), and no overlapping data points (`noverlap`=0). The complex part of `Pxy` is required for computing the phase shift between the two signals using the function `angle`.

```
phase = angle(Pxy);

plot(f,phase), grid
xlabel('Frequency')
ylabel('Phase Angle')
title('Phase Spectrum')
```

The phase shift at a frequency of f=0.2 (period τ=5) can be interpolated from the phase spectrum

```
interp1(f,phase,0.2)
```

which produces the output

```
ans =
  -1.2566
```

The phase spectrum is normalized to one full period $\tau=2\pi$ and the phase shift of −1.2566 therefore equals $(-1.2566 \cdot 5)/(2 \cdot \pi) = -1.0000$, which is the phase shift of one that we introduced initially.

We now use two sine waves with different periodicities to illustrate cross-spectral analysis. Both signals, `x` and `y`, have a periodicity of 5 but a phase shift of 1.

```
clear

t = 1 : 1000;
x = sin(2*pi*t/15) + 0.5*sin(2*pi*t/5);
y = 2*sin(2*pi*t/50) + 0.5*sin(2*pi*t/5+2*pi/5);

plot(t,x,'b-',t,y,'r-')
axis([0 100 -3 3]), grid
```

We can now compute the cross-spectrum `Pxy`, which clearly shows the common period of $\tau=5$ (or frequency of $f=0.2$).

```
[Pxy,f] = cpsd(x,y,[],0,1024,1);

plot(f, abs(Pxy)), grid
xlabel('Frequency')
ylabel('Power')
title('Cross-Spectrum')
```

The coherence shows a high value that is close to one at $f=0.2$.

```
[Cxy,f] = mscohere(x,y,[],0,1024,1);

plot(f,Cxy), grid
xlabel('Frequency')
ylabel('Coherence')
title('Coherence')
```

The complex part of the cross-spectrum `Pxy` is required for calculating the phase shift between the two sine waves.

```
[Pxy,f] = cpsd(x,y,[],0,1024,1);
phase = angle(Pxy);

plot(f,phase), grid
```

The phase shift at a frequency of $f=0.2$ (period $\tau=5$) is

```
interp1(f,phase,0.2)
```

which produces the output of

```
ans =
  -1.2572
```

The phase spectrum is normalized to one full period $\tau=2\pi$ and the phase shift of −1.2572 therefore equals $(-1.2572\cdot5)/(2\cdot\pi)=-1.0004$, which is again the phase shift of one that we introduced initially.

5.5 Interpolating and Analyzing Unevenly-Spaced Data

We can now use our experience in analyzing evenly-spaced data to run a spectral analysis on unevenly-spaced data. Such data are very common in earth sciences, for example in the field of paleoceanography, where deep-sea cores are typically sampled at constant depth intervals. The transformation of evenly-spaced length-parameter data to time-parameter data in an environment with changing length-time ratios results in unevenly-spaced time series. Numerous methods exist for interpolating unevenly-spaced sequences of data or time series. The aim of these *interpolation techniques* for $x(t)$ data is to estimate the x-values for an equally-spaced t vector from the irregularly-spaced $x(t)$ actual measurements. *Linear interpolation* predicts the x-values by effectively drawing a straight line between two neighboring measurements and by calculating the x-value at the appropriate point along that line. However, this method has its limitations. It assumes linear transitions in the data, which introduces a number of artifacts including the loss of high-frequency components of the signal and the limiting of the data range to that of the original measurements.

Cubic-spline interpolation is another method for interpolating data that are unevenly spaced. Cubic splines are piecewise continuous curves requiring at least four data points for each step. The method has the advantage that it preserves the high-frequency information contained in the data. However, steep gradients in the data sequence, which typically occur adjacent to extreme minima and maxima, could cause spurious amplitudes in the interpolated time series. Since all these (and other) interpolation techniques might introduce artifacts into the data, it is always advisable to (1) keep the total number of data points constant before and after interpolation, (2) report the method employed for estimating the evenly-spaced data sequence, and (3) explore the effect of interpolation on the variance of the data.

Following this brief introduction to interpolation techniques we can apply the most popular linear and cubic spline interpolation techniques to unevenly-spaced data. Having interpolated the data we can then use the spectral tools that have previously been applied to evenly-spaced data

(Sections 5.3 and 5.4). We must first load the two time series:

```
clear

series1 = load('series1.txt');
series2 = load('series2.txt');
```

Both synthetic data sets contain a two-column matrix with 339 rows. The first column contains ages in kiloyears, which are unevenly spaced. The second column contains oxygen-isotope values measured on calcareous microfossils (foraminifera). The data sets contain 100, 40 and 20 kyr cyclicities and they are overlain by Gaussian noise. In the 100 kyr frequency band, the second data series has shifted by 5 kyrs with respect to the first data series. To plot the data we type

```
plot(series1(:,1),series1(:,2))
figure
plot(series2(:,1),series2(:,2))
```

The statistics for the spacing of the first data series can be computed by

```
intv1 = diff(series1(:,1));

plot(intv1)
```

The plot shows that the spacing varies around a mean interval of 3 kyrs, with a standard deviation of ca. 1 kyr. The minimum and maximum values for the time axis

```
min(series1(:,1))
max(series1(:,1))
```

of t_{min}=0 and t_{max}=997 kyrs provide some information about the temporal range of the data. The second data series

```
intv2 = diff(series2(:,1));

plot(intv2)

min(series2(:,1))
max(series2(:,1))
```

has a similar range, from 0 to 997 kyrs. We see that both series have a mean spacing of 3 kyrs and range from 0 to ca. 1000 kyrs. We now interpolate the data to an evenly-spaced time axis. While doing this, we follow the rule that the number of data points should not be increased. The new time axis runs from 0 to 996 kyrs, with 3 kyr intervals.

```
t = 0 : 3 : 996;
```

We can now interpolate the two time series to this axis with *linear* and *spline* interpolation methods, using the function `interp1`.

```
series1L = interp1(series1(:,1),series1(:,2),t,'linear');
series1S = interp1(series1(:,1),series1(:,2),t,'spline');

series2L = interp1(series2(:,1),series2(:,2),t,'linear');
series2S = interp1(series2(:,1),series2(:,2),t,'spline');
```

In the `linear` interpolation method the linear interpolant is the straight line between neighboring data points. In the `spline` interpolation the interpolant is a piecewise polynomial (the *spline*) between these data points. The method `spline` with `interp1` uses a piecewise cubic spline interpolation, i.e., the interpolant is a third-degree polynomial. The results are compared by plotting the first series before and after interpolation.

```
plot(series1(:,1),series1(:,2),'ko'), hold on
plot(t,series1L,'b-',t,series1S,'r-'), hold off
```

We can already observe some significant artifacts at ca. 370 kyrs. Whereas the linearly-interpolated points are always within the range of the original data, the spline interpolation method produces values that are unrealistically high or low (Fig. 5.9). The results can be compared by plotting the second data series.

```
plot(series2(:,1),series2(:,2),'ko'), hold on
plot(t,series2L,'b-',t,series2S,'r-'), hold off
```

In this series, only a few artifacts can be observed. The function `interp1` also provides an alternative to `spline`, which is `pchip`. The name `pchip` stands for

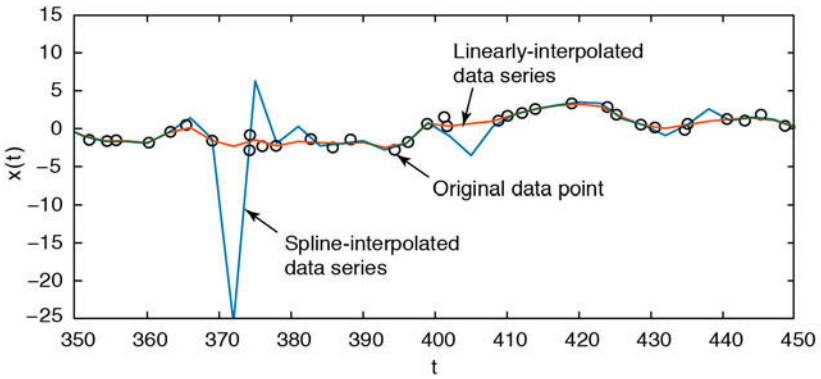


Fig. 5.9 Interpolation artifacts. Whereas the linearly interpolated points are always within the range of the original data, the spline interpolation method results in unrealistic high and low values.

Piecewise Cubic Hermite Interpolating Polynomial and this method performs a shape-preserving piecewise cubic interpolation. The function avoids the typical artifacts of the splines as it preserves the original shape of the data series. We can apply the function used above to calculate the power spectrum, computing the FFT for 256 data points with a sampling frequency of 1/3 kyr^{-1}.

```
[Pxx,f] = periodogram(series1L,[],256,1/3);

plot(f,Pxx)
xlabel('Frequency')
ylabel('Power')
title('Auto-Spectrum')
```

Significant peaks occur at frequencies of approximately 0.01, 0.025 and 0.05, corresponding approximately to the 100, 40 and 20 kyr cycles. Analysis of the second time series

```
[Pxx,f] = periodogram(series2L,[],256,1/3);

plot(f,Pxx)
xlabel('Frequency')
ylabel('Power')
title('Auto-Spectrum')
```

also yields significant peaks at frequencies of 0.01, 0.025 and 0.05 (Fig. 5.10). We now compute the cross-spectrum for both data series.

```
[Pxy,f] = cpsd(series1L,series2L,[],128,256,1/3);

plot(f,abs(Pxy))
xlabel('Frequency')
ylabel('Power')
title('Cross-Spectrum')
```

The correlation, as indicated by the high value for the coherence, is quite convincing.

```
[Cxy,f] = mscohere(series1L,series2L,[],128,256,1/3);

plot(f,Cxy)
xlabel('Frequency')
ylabel('Magnitude Squared Coherence')
title('Coherence')
```

We can observe a fairly high coherence at frequencies of 0.01, 0.025 and 0.05. The complex part of `Pxy` is required for calculating the phase difference for each frequency.

```
phase = angle(Pxy);
```

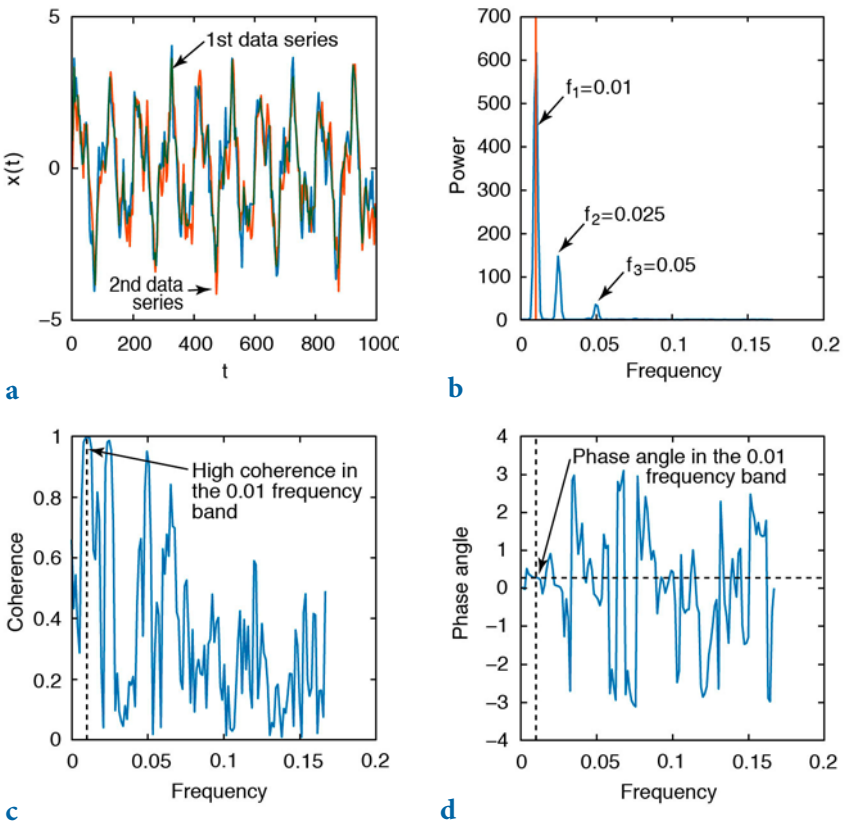


Fig. 5.10 Result from cross-spectral analysis of the two linearly-interpolated signals: **a** signals in the time domain, **b** cross-spectrum of both signals, **c** coherence of the signals in the frequency domain, and **d** phase spectrum in radians.

```
plot(f,phase)
xlabel('Frequency')
ylabel('Phase Angle')
title('Phase spectrum')
```

The phase shift at a frequency of f=0.01 is calculated using

```
interp1(f,phase,0.01)
```

which produces the output of

```
ans =
  -0.2796
```

The phase spectrum is normalized to a full period $\tau=2\pi$ and the phase

shift of –0.2796 therefore equals (–0.2796·100 kyrs)/(2·π)=–4.45 kyrs. This corresponds roughly to the phase shift of 5 kyrs introduced to the second data series with respect to the first series.

The Signal Processing Toolbox also contains a GUI function named `sptool` (for *Signal Processing Tool*), which is a more convenient tool for spectral analysis but is not described in any detail herein.


Movie 5.1

5.6 Evolutionary Power Spectrum

The amplitude of spectral peaks usually varies with time. This is particularly true for paleoclimate time series. Paleoclimate records usually show trends, not only in the mean and variance but also in the relative contributions of rhythmic components such as the Milankovitch cycles in marine oxygen-isotope records. Evolutionary power spectra have the ability to map such changes in the frequency domain. The evolutionary or windowed power spectrum is a modification of the method introduced in Section 5.3, which computes the spectrum of overlapping segments of the time series. These overlapping segments are relatively short compared to the windowed segments used by the Welch method (Section 5.3), which is used to increase the signal-to-noise ratio of power spectra. The evolutionary power spectrum method therefore uses the Short-Time Fourier Transform (STFT) instead of the Fast Fourier Transformation (FFT). The output from the evolutionary power spectrum is the short-term, time-localized frequency content of the signal. There are various methods to display the results. For instance, time and frequency can be plotted on the *x*- and *y*-axes, respectively, or vice versa, with the color of the plot being dependent on the height of the spectral peaks.

As an example we use a data set that is similar to those used in Section 5.5. The data series contains three main periodicities of 100, 40 and 20 kyrs and additive Gaussian noise. The amplitudes, however, change through time and this example can therefore be used to illustrate the advantage of the evolutionary power spectrum method. In our example the 40 kyr cycle appears only after ca. 450 kyrs, whereas the 100 and 20 kyr cycles are present throughout the time series. We first load from the file *series3.txt* and display the data (Fig. 5.11).

```
clear

series3 = load('series3.txt');
plot(series3(:,1),series3(:,2))
xlabel('Time (kyr)')
ylabel('d18O (permille)')
title('Signal with Varying Cyclicities')
```

Since both the standard and the evolutionary power spectrum methods require evenly-spaced data, we interpolate the data to an evenly-spaced time vector `t`, as demonstrated in Section 5.5.

```
t = 0 : 3 : 1000;
series3L = interp1(series3(:,1),series3(:,2),t,'linear');
```

We then compute a non-evolutionary power spectrum for the full length of the time series (Fig. 5.12). This exercise helps us to compare the differences between the results of the standard and the evolutionary power spectrum methods.

```
[Pxx,f] = periodogram(series3L,[],1024,1/3);
plot(f,Pxx)
xlabel('Frequency')
ylabel('Power')
title('Power Spectrum')
```

Audio 5.3

The auto-spectrum shows significant peaks at 100, 40 and 20 kyr cyclicities, as well as some noise. The power spectrum, however, does not provide any

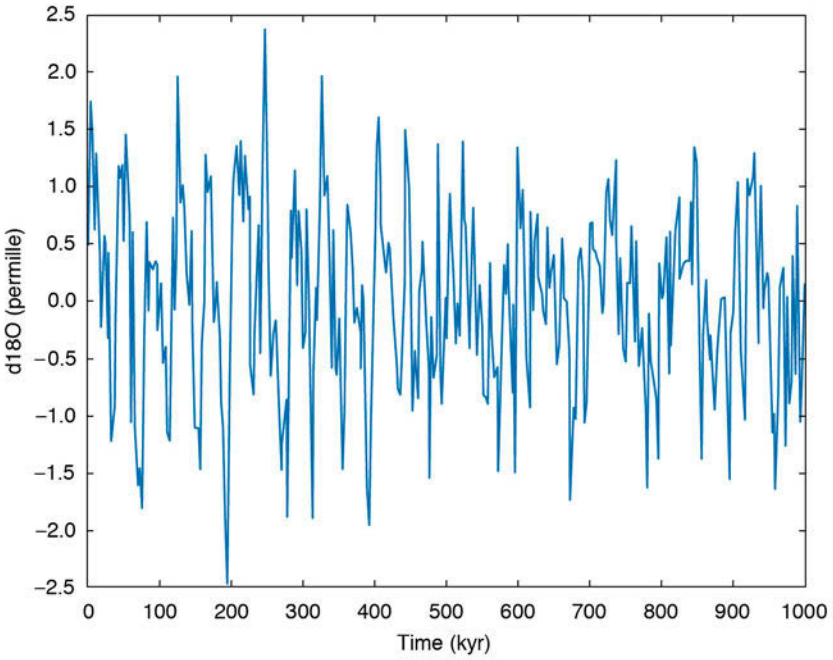


Fig. 5.11 Synthetic data set containing three main periodicities of 100, 40, and 20 kyrs and additive Gaussian noise. Whereas the 100 and 20 kyr cycles are present throughout the time series, the 40 kyr cycle only appears at around 450 kyrs before present.

information about fluctuations in the amplitudes of these peaks. The non-evolutionary power spectrum simply represents an average of the spectral information contained in the data.

We now use the function `spectrogram` to map the changes in the power spectrum with time. By default, the time series is divided into eight segments with a 50% overlap. Each segment is windowed with a Hamming window to suppress spectral leakage (Section 5.3). The function `spectrogram` uses similar input parameters to those used in `periodogram` in Section 5.3. We then compute the evolutionary power spectrum for a window of 64 data points with a 50 data point overlap. The STFT is computed for `nfft`=256. Since the spacing of the interpolated time vector is 3 kyrs, the sampling frequency is 1/3 kyr^{-1}.

```
spectrogram(series3L,64,50,256,1/3)
title('Evolutionary Power Spectrum')
xlabel('Frequency (1/kyr)')
ylabel('Time (kyr)')
colormap(jet)
```

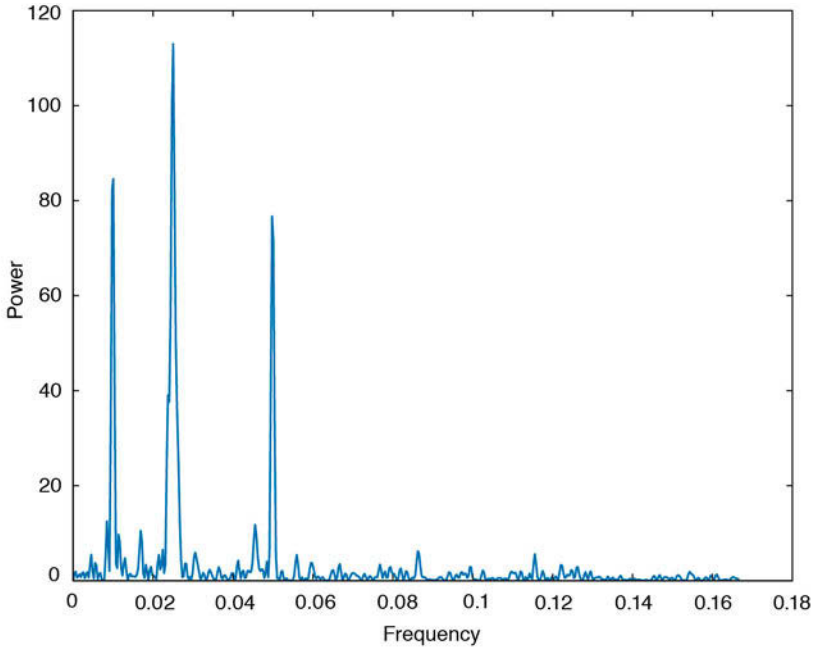


Fig. 5.12 Power spectrum for the complete time series. showing significant peaks at 100, 40 and 20 kyrs. The plot, however, does not provide any information on the temporal behavior of the cyclicities.

The output from `spectrogram` is a color plot (Fig. 5.13) that displays red vertical stripes representing significant maxima at frequencies of 0.01 and 0.05 kyr^{-1} (i.e., every 100 and 20 kyrs). There is also a 40 kyr cycle (corresponding to a frequency of 0.025 kyr^{-1}), but this only occurs after ca. 450 kyrs, as documented by the vertical red stripe in the lower half of the graph.

To improve the visibility of the significant cycles, the colors used in the graph can be modified using the colormap editor.

Movie 5.2 `colormapeditor`

The colormap editor displays the colormap of the figure as a strip of rectangular cells. The nodes that separate regions of uniform slope in the RGB colormap can be shifted by using the mouse, which introduces distortions in the colormap and results in modification of the spectrogram colors. For example shifting the yellow node towards the right increases the contrast between the vertical peak areas at 100, 40 and 20 kyrs, and the background.

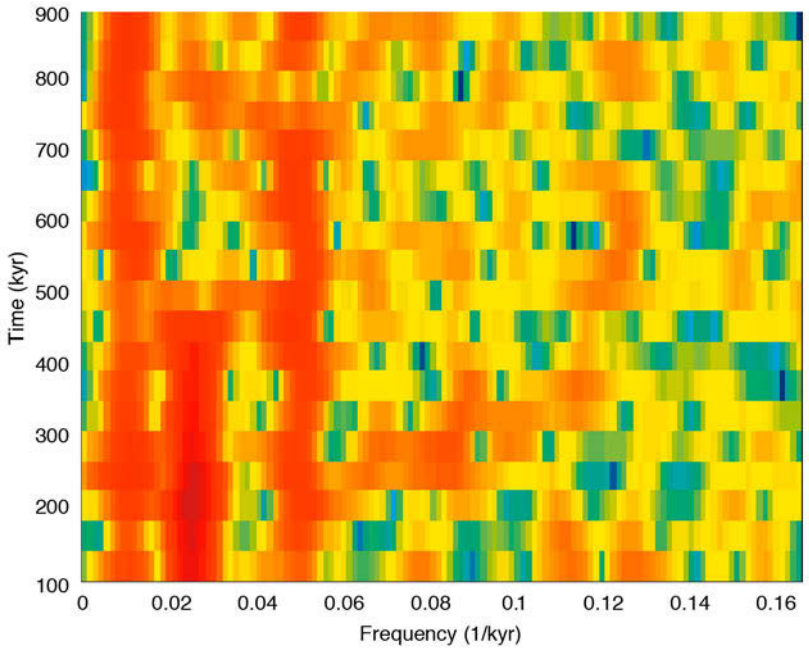


Fig. 5.13 Evolutionary power spectrum using `spectrogram`, which computes the short-time Fourier transform STFT of overlapping segments of the time series. We use a Hamming window of 64 data points and 50 data points overlap. The STFT is computed for `nfft`=256. Since the spacing of the interpolated time vector is 3 kyrs the sampling frequency is 1/3 kyr^{-1}. The plot shows the onset of the 40 kyr cycle at around 450 kyrs before present.

5.7 Lomb-Scargle Power Spectrum

The power spectrum methods introduced in the previous sections require evenly-spaced data. In earth sciences, however, time series are often unevenly spaced. Although interpolating the unevenly-spaced data to a grid of evenly-spaced times is one way to overcome this problem (Section 5.5), interpolation introduces numerous artifacts into the data, in both the time and frequency domains. For this reason an alternative method of time-series analysis has become increasingly popular in earth sciences, the Lomb-Scargle algorithm (e.g., Scargle 1981, 1982, 1989, 1990, Press et al. 1992, Schulz et al. 1998).

The Lomb-Scargle algorithm only evaluates the data of the time series at the times t_i that are actually measured. Assuming a series $y(t)$ of N data points, the Lomb-Scargle normalized periodogram P_x, as a function of angular frequency $\omega=2\pi f > 0$, is given by

$$P_x(\omega)=\frac{1}{2s^2}\left\{\frac{\left[\sum_j (y_i-\bar{y})\cos\omega(t_j-\tau)\right]^2}{\sum_j \cos^2\omega(t_j-\tau)}+\frac{\left[\sum_j (y_i-\bar{y})\sin\omega(t_j-\tau)\right]^2}{\sum_j \sin^2\omega(t_j-\tau)}\right\}$$

where

$$\bar{y}=\frac{1}{N}\sum_{i=1}^{N} y_i$$

and

$$s^2=\frac{1}{N-1}\sum_{i=1}^{N}(y_i-\bar{y})^2$$

are the arithmetic mean and the variance of the data (Section 3.2). The constant τ, which is defined by the relationship

$$\tan(2\omega\tau)=\frac{\sum_j \sin 2\omega t_j}{\sum_j \cos 2\omega t_j}$$

is an offset that makes $P_x(\omega)$ independent of shifting the t_i values by any

constant amount. Scargle (1982) showed that this particular choice of the offset τ has the consequence that the solution for $P_x(\omega)$ is identical to a least-squares fit of sine and cosine functions to the data series $y(t)$:

$$y(t) = A\cos\omega t + B\sin\omega t$$

The least-squares fit of harmonic functions to data series in conjunction with spectral analysis had previously been investigated by Lomb (1976), and hence the method is called the normalized Lomb-Scargle Fourier transform. The term *normalized* refers to the factor s^2 in the dominator of the equation for the periodogram.

Scargle (1982) has shown that the Lomb-Scargle periodogram has an exponential probability distribution with a mean equal to one, assuming that the noise is Gaussian distributed. The probability that $P_x(\omega)$ will be between some positive quantity z and $z+dz$ is $\exp(-z)dz$. If we scan M independent frequencies, the probability of none of them having a value larger than z is $(1-\exp(-z))M$. We can therefore compute the false-alarm probability of the null hypothesis (i.e., the probability that a given peak in the periodogram is not significant) using

$$P(>z) \equiv 1-(1-e^{-z})^M$$

Press et al. (1992) suggested using the Nyquist criterion (Section 5.2) to determine the number of independent frequencies M, assuming that the data were evenly spaced. In this case, the appropriate value for the number of independent frequencies is $M=2N$, where N is the length of the time series.

More detailed discussions of the Lomb-Scargle method are given in Scargle (1989) and Press et al. (1992). An excellent summary of the method and a TURBO PASCAL program to compute the normalized Lomb-Scargle power spectrum of paleoclimatic data have been published by Schulz and Stattegger (1998). A convenient MATLAB algorithm `lombscargle` for computing the Lomb-Scargle periodogram has been published by Brett Shoelson (The MathWorks, Inc.) and can be downloaded from *File Exchange* at

`http://www.mathworks.de/matlabcentral/fileexchange/993-lombscargle-m`

The following MATLAB code is based on the original FORTRAN code published by Scargle (1989). Significance testing uses the methods proposed by Press et al. (1992) explained above.

We first load the synthetic data that were generated to illustrate the use of the evolutionary or windowed power spectrum method in Section 5.6.

The data contain periodicities of 100, 40 and 20 kyrs, as well as additive Gaussian noise, and are unevenly spaced about the time axis. We define two new vectors `t` and `x` that contain the original time vector and the synthetic oxygen-isotope data sampled at times `t`.

```
clear

series3 = load('series3.txt');
t = series3(:,1);
x = series3(:,2);
```

We then generate a frequency axis `f`. Since the Lomb-Scargle method is not able to deal with the frequency of zero (i.e., with an infinite period) we start at a frequency value that is equivalent to the spacing of the frequency vector. The variable `ofac` is the oversampling parameter that influences the resolution of the frequency axis about the `N(frequencies)=N(datapoints)` case. We also need the highest frequency `fhi` that can be analyzed by the Lomb-Scargle algorithm: the Nyquist frequency `fnyq` that would be obtained if the `N` data points were evenly spaced over the same time interval is commonly used for `fhi`. The following code uses the input parameter `hifac`, which is defined by Press et al. (1992) as `hifac=fhi/fnyq`.

```
int = mean(diff(t));
ofac = 4; hifac = 1;
f = ((2*int)^(-1))/(length(x)*ofac): ...
    ((2*int)^(-1))/(length(x)*ofac): ...
    hifac*(2*int)^(-1);
```

where `int` is the mean sampling interval. We normalize the data by subtracting the mean.

```
x = x - mean(x);
```

We can now compute the normalized Lomb-Scargle periodogram `px` as a function of the angular frequency `wrun` using the translation of Scargle's FORTRAN code into MATLAB code.

```
for k = 1:length(f)
    wrun = 2*pi*f(k);
    px(k) = 1/(2*var(x)) * ...
       ((sum(x.*cos(wrun*t - ...
       atan2(sum(sin(2*wrun*t)),sum(cos(2*wrun*t)))/2))).^2) ...
       /(sum((cos(wrun*t - ...
       atan2(sum(sin(2*wrun*t)),sum(cos(2*wrun*t)))/2)).^2)) + ...
       ((sum(x.*sin(wrun*t - ...
       atan2(sum(sin(2*wrun*t)),sum(cos(2*wrun*t)))/2))).^2) ...
       /(sum((sin(wrun*t - ...
       atan2(sum(sin(2*wrun*t)),sum(cos(2*wrun*t)))/2)).^2));
end
```

The significance level for any peak in the power spectrum `px` can now be computed. The variable `prob` indicates the false-alarm probability for the null hypothesis: a low `prob` therefore indicates a highly significant peak in the power spectrum.

```
prob = 1-(1-exp(-px)).^(2*length(x));
```

We now plot the power spectrum and the probabilities (Fig. 5.14):

```
plot(f,px)
xlabel('Frequency')
ylabel('Power')
title('Lomb-Scargle Power Spectrum')

figure
plot(f,prob)
xlabel('Frequency')
ylabel('Probability')
title('Probabilities')
```

The two plots suggest that all three peaks are highly significant since the errors are extremely low at the cyclicities of 100, 40 and 20 kyrs.

An alternative way of displaying the significance levels was suggested by Press et al. (1992). In this method the equation for the false-alarm probability of the null hypothesis is inverted to compute the corresponding power of the significance levels. As an example we choose a significance level of 95%. However, this number can also be replaced by a vector of several significance levels such as `signif=[0.90 0.95 0.99]`. We can now type

```
m = floor(0.5*ofac*hifac*length(x));
effm = 2*m/ofac;
signif = 0.95;
levels = log((1-signif.^(1/effm)).^(-1));
```

where `m` is the true number of independent frequencies and `effm` is the effective number of frequencies using the oversampling factor `ofac`. The second plot displays the spectral peaks and the corresponding probabilities.

```
plot(f,px)
hold on
for k = 1:length(signif)
    line(f,levels(:,k)*ones(size(f)),'LineStyle','--')
end
xlabel('Frequency')
ylabel('Power')
title('Lomb-Scargle Power Spectrum')
hold off
```

All three spectral peaks at frequencies of 0.01, 0.025 and 0.05 kyr^{-1} exceed the

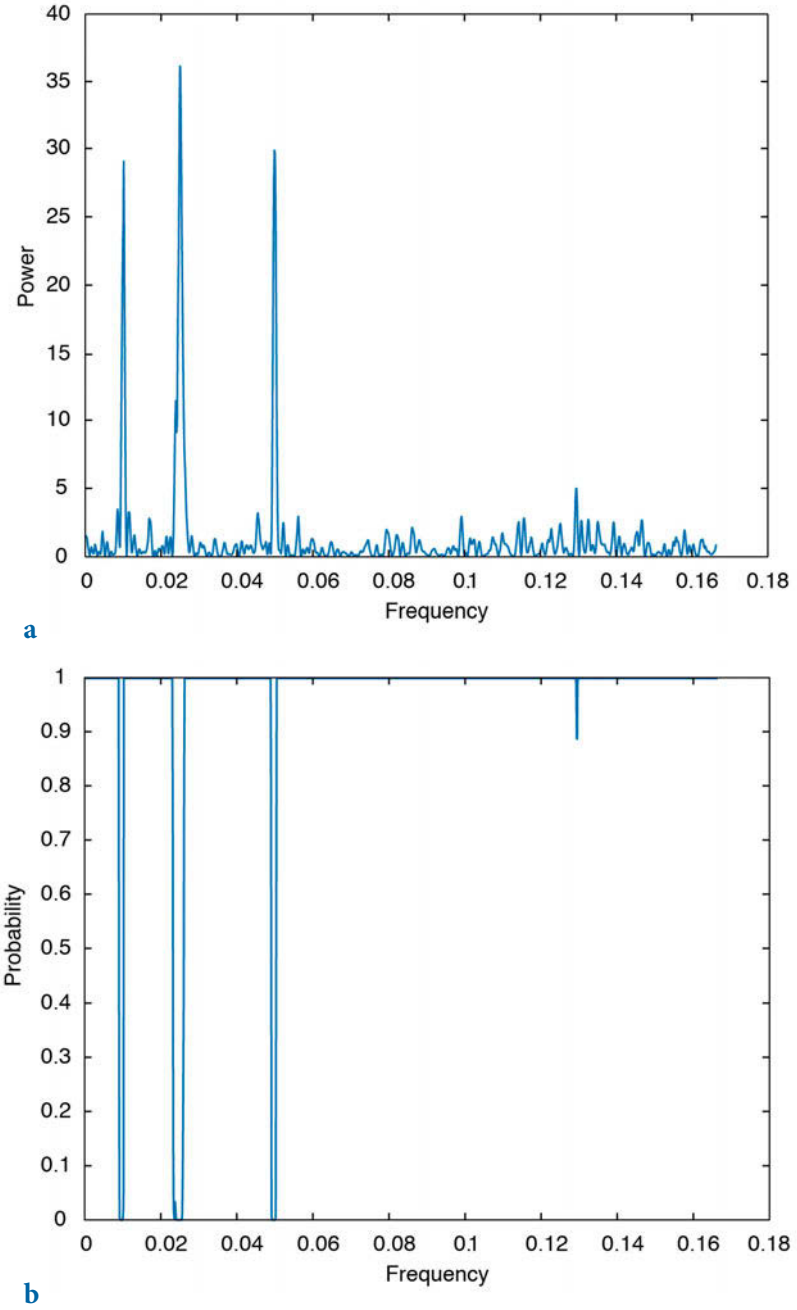


Fig. 5.14 **a** Lomb-Scargle power spectrum and **b** the false-alarm probability of the null hypothesis. The plot suggests that the 100, 40 and 20 kyr cycles are highly significant.

95% significant level, suggesting that they represent significant cyclicities. We have therefore obtained similar results to those obtained using the periodogram method. However, the Lomb-Scargle method has the advantage that is does not require any interpolation of unevenly-spaced data, as well as permitting quantitative significance testing.

5.8 Wavelet Power Spectrum

Section 5.6 demonstrated the use of a modification to the power spectrum method for mapping changes in cyclicity through time. A similar modification could, in theory, be applied to the Lomb-Scargle method, which would have the advantage that it could then be applied to unevenly-spaced data. Both methods, however, assume that the data are composites of sine and cosine waves that are globally uniform in time and have infinite time spans. The evolutionary power spectrum method divides the time series into overlapping segments and computes the Fourier transform of these segments. To avoid spectral leakage, the data are multiplied by windows that are smooth bell-shaped curves with positive values (Section 5.3). The higher the temporal resolution of the evolutionary power spectrum the lower the accuracy of the result. Moreover, short time windows contain a large number of high-frequency cycles whereas the low-frequency cycles are underrepresented.

In contrast to the Fourier transform, the *wavelet transform* uses base functions (*wavelets*) that have smooth ends *per se* (Lau and Weng 1995, Mackenzie et al. 2001). Wavelets are small packets of waves; they are defined by a specific frequency and decay towards either end. Since wavelets can be stretched and translated in both frequency and time, with a flexible resolution, they can easily map changes in the time-frequency domain. We use the functions for wavelet analysis that are included in the Wavelet Toolbox (MathWorks 2014b). There is also, however, a very popular wavelet toolbox produced by Christopher Torrence and Gilbert P. Compo (1998), which is freely available online from

`http://paos.colorado.edu/research/wavelets/`

A wavelet transformation mathematically decomposes a signal $y(t)$ into elementary functions $\psi_{a,b}(t)$ derived from a *mother wavelet* $\psi(t)$, by dilation and translation,

$$\psi_{a,b} = \frac{1}{(a)^{1/2}} \psi\left(\frac{t-b}{a}\right)$$

where b denotes the position (translation) and a (>0) the scale (dilation) of the wavelet (Lau and Weng 1995). The wavelet transform of the signal $y(t)$ about the mother wavelet $\psi(t)$ is defined as the convolution integral

$$W(b,a) = \frac{1}{(a)^{1/2}} \int \psi^* \left(\frac{t-b}{a} \right) y(t)\, dt$$

where ψ^* is the complex conjugate of ψ. There are many mother wavelets available in the literature, such as the classic *Haar* wavelet, the *Morlet* wavelet, or the *Daubechies* wavelet. The most popular wavelet in geosciences is the Morlet wavelet introduced by French geophysicist Jean Morlet (1931–2007), which is defined by

$$\psi_0(\eta) = \pi^{-1/4} \exp(i\omega_0 \eta) \exp(-\eta^2/2)$$

where η is the time and ω_0 is the wave number (Torrence and Compo 1998). The wave number is the number of oscillations within the wavelet itself. We can easily compute a discrete version of the Morlet wavelet `wave` by translating the above equation into MATLAB code, where `eta` is the non-dimensional time and `w0` is the wave number. Changing `w0` produces wavelets with different wave numbers. Note that it is important not to use `i` for index in `for` loops, since it is used here for imaginary unit (Fig. 5.15).

```
clear

eta = -10 : 0.1 : 10;
w0 = 6;
wave = pi.^(-1/4) .* exp(i*w0*eta) .* exp(-eta.^2/2);

plot(eta,wave)
xlabel('Position')
ylabel('Scale')
title('Morlet Mother Wavelet')
```

In order to familiarize ourselves with wavelet power spectra, we use a pure sine wave with a period five and additive Gaussian noise.

```
clear

rng(0)
t = 0 : 0.5 : 50;
x = sin(2*pi*t/5) + randn(size(t));
```

As a first step, we need to define the mother wavelet and its wave number `w0`.

```
mother = 'morl';
w0 = 6;
```

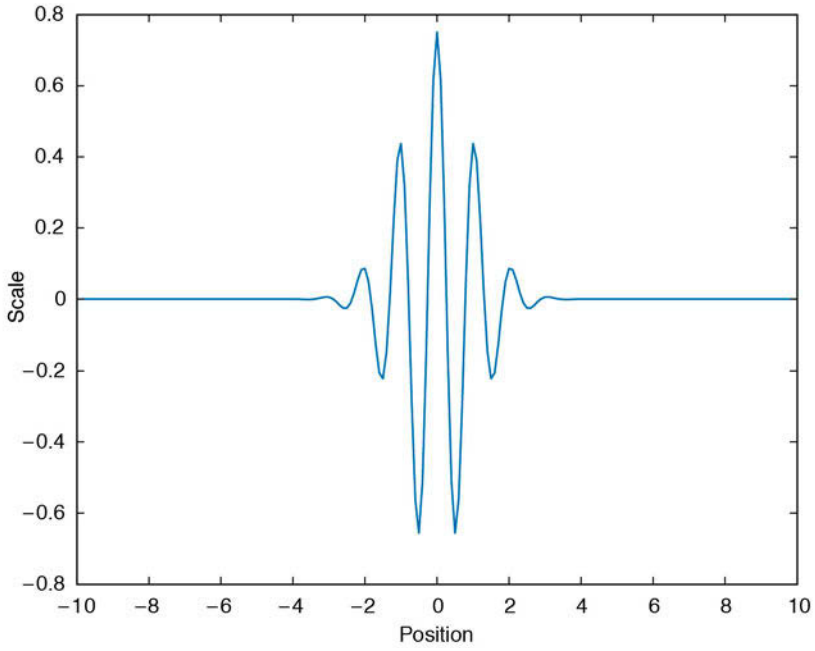


Fig. 5.15 Morlet mother wavelet with wave number 6.

We then need to define the values of the scales for which the wavelet transform will be computed. These values define how much a wavelet is stretched or compressed to map the variability of the time series at different wavelengths. Scales with smaller values correspond to higher frequencies and can therefore map rapidly-changing details, whereas those with higher values can map the long-term variations. The definition of the scales first requires the sampling interval `dt` of our time series `x`. We then use the default spacing `ds` of 0.4875 for a Morlet wavelet, following the instructions contained in the Wavelet Toolbox manual (MathWorks 2014b). The smallest value for the scales `s0` is usually chosen to be twice the sampling interval, i.e., `2*dt`. We next calculate the number of scales `nb`, which depends on the length of the time series and the spacing of the scales. Finally, we calculate the scales `scales` themselves depending on the smallest scale, the number of scales, and the spacing of the scales, using equations provided in the Wavelet Toolbox manual (MathWorks 2014b).

```
dt = 0.5;
ds = 0.4875;
s0 = 2*dt;
nb = fix(log2(length(x))/ds)+1;
scales = s0*2.^((0:nb-1)*ds);
```

In the next step we compute the real or complex continuous wavelet coefficients using the function `cwt` contained in the Wavelet Toolbox.

```
coefs = cwt(x,scales,mother);
```

The function `scal2frq` converts scales `scales` to pseudo-frequencies, using the mother wavelet `mother` and the sampling period `dt`.

```
f = scal2frq(scales,mother,dt);
```

We use a filled contour plot to portray the power spectrum, i.e., the absolute value of the wavelet coefficients (Fig. 5.16 a).

```
contour(t,f,abs(coefs),...
   'LineStyle','none',...
   'LineColor',[0 0 0],...
   'Fill','on')
xlabel('Time')
ylabel('Frequency')
title('Wavelet Power Spectrum')
set(gcf,'Colormap',jet)
set(gca,'YLim',[0 0.9],...
   'XGrid','On',...
   'YGrid','On')
```

Alternatively, we can compute the wavelet transform using the fast Fourier transform (FFT) algorithm implemented in the function `cwtft`. This approach is used in the freely available wavelet toolbox produced by Torrence and Compo (1998). We first define the scales using the values for the smallest scales `s0`, the sampling interval `ds` of the scales, and the number of scales `nb` from above, merged into a structure array `sc`.

```
sc.s0 = s0;
sc.ds = ds;
sc.nb = nb;
```

Then, we create a structure array `sig` that contains the signal `x`, the sampling interval (or period) `dt`, the mother wavelet `mother`, and the scales `sc`.

```
sig = struct('val',x,...
    'period',dt,...
    'wavelet',mother,...
    'scales',sc);
```

The output from `cwtft` is a structure array `cwtstruct` that includes the wavelet coefficients `cfs` and the scales `scales`. The default mother wavelet is the Morlet wavelet.

```
cwtstruct = cwtft(sig);
```

We convert the scales to pseudo-frequencies using the equation for the Morlet wavelet, which we find in the wavelet definitions in the `cwtft` help section.

```
f = 1./(4*pi*cwtstruct.scales/(w0+sqrt(2+w0^2)));
```

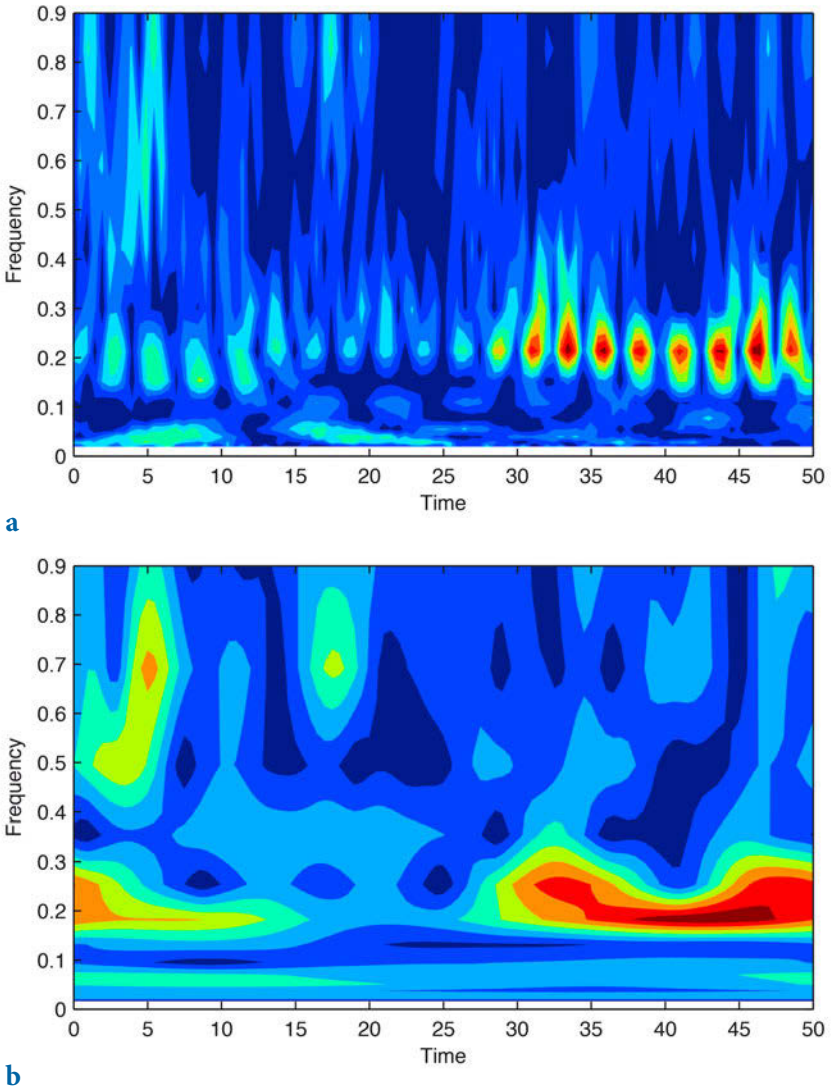


Fig. 5.16 Wavelet power spectrum showing a significant period at 5 cycles that persists throughout the full length of the time vector. The wavelet power spectrum has been calculated using **a** the continuous 1D wavelet transform `cwt` and **b** the continuous wavelet transform using the FFT algorithm `cwtft`.

We again use a filled contour plot to portray the power spectrum, i.e., the absolute value of the wavelet coefficients (Fig. 5.16 b).

```
contour(t,f,abs(cwtstruct.cfs),...
    'LineStyle','none',...
    'LineColor',[0 0 0],...
    'Fill','on')
xlabel('Time')
ylabel('Frequency')
title('Wavelet Power Spectrum Using FFT Algorithm')
set(gcf,'Colormap',jet)
set(gca,'YLim',[0 0.9],...
    'XGrid','On',...
    'YGrid','On')
```

As we can see, the wavelet power spectrum derived using `cwtft` is much smoother than that computed with `cwt`, since `cwtft` uses sinusoids to smooth the coefficients. However, the smoothing causes a significant loss of detail in the contour plot.

We now apply this concept to the synthetic data from the example to demonstrate the windowed power spectrum method and load the synthetic data contained in file `series3.txt`, remembering that the data contain periodicities of 100, 40, and 20 kyrs as well as additive Gaussian noise, and that they are unevenly spaced about the time axis.

```
clear

series3 = load('series3.txt');
```

As for the Fourier transform and in contrast to the Lomb-Scargle algorithm, the wavelet transform requires evenly-spaced data, and we therefore interpolate the data using `interp1`.

```
t = 0 : 3 : 1000;
series3L = interp1(series3(:,1),series3(:,2),t,'linear');
```

Again, we first need to define the mother wavelet and its wave number `w0`.

```
mother = 'morl';
w0 = 6;
```

We then define the scales, as demonstrated in the first example. Unlike the previous example the sampling interval `dt` of our time series is now 3.

```
dt = 3;
ds = 0.4875;
s0 = 2*dt;
nb = fix(log2(length(series3L))/ds)+1;
scales = s0*2.^((0:nb-1)*ds);
```

We compute the wavelet coefficients using `cwt`.

```
coefs = cwt(series3L,scales,mother);
```

We convert the scales `scales` to pseudo-frequencies using the mother wavelet `mother` and the sampling period `dt`.

```
f = scal2frq(scales,mother,dt);
```

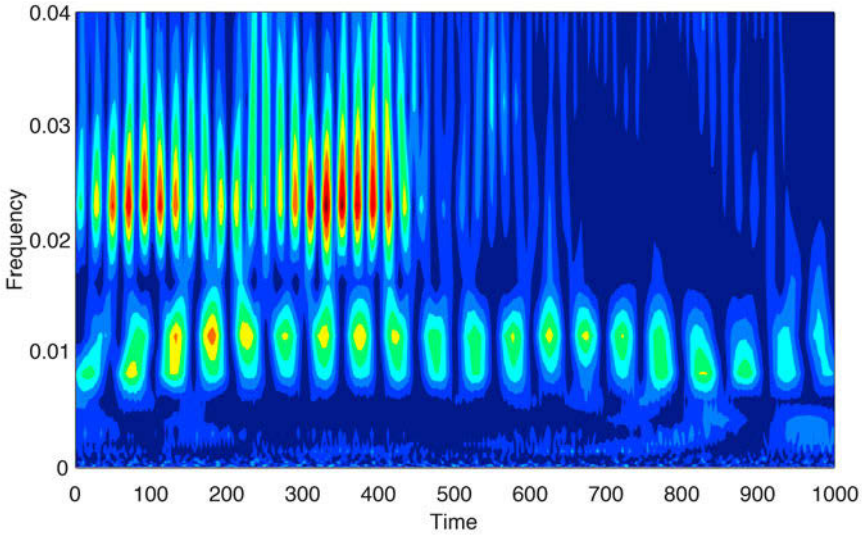
We use a filled contour plot to portray the power spectrum (Fig. 5.17 a).

```
contour(t,f,abs(coefs),...
   'LineStyle','none',...
   'LineColor',[0 0 0],...
   'Fill','on')
xlabel('Time')
ylabel('Frequency')
title('Wavelet Power Spectrum')
set(gcf,'Colormap',jet)
set(gca,'YLim',[0 0.04],...
   'XGrid','On',...
   'YGrid','On')
```

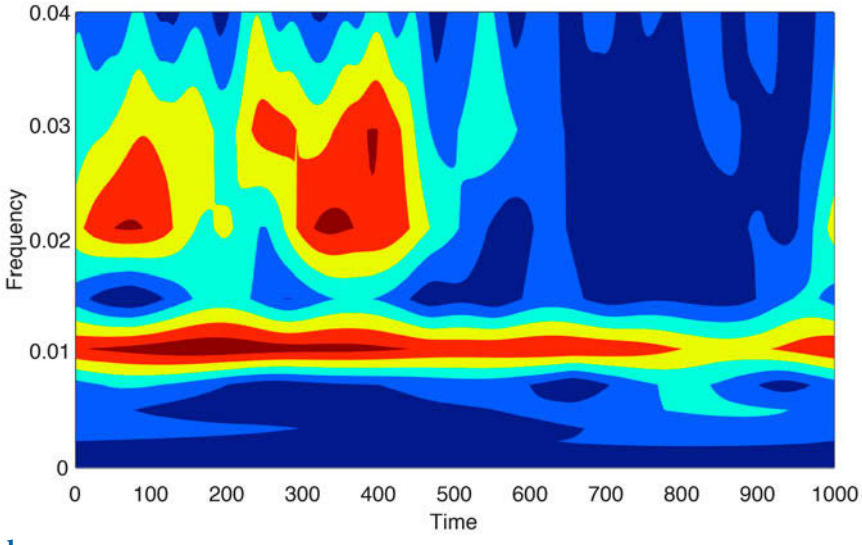
The graph shows horizontal clusters of peaks at around 0.01 and 0.025 kyr^{-1}, corresponding to 100 and 40 kyr cycles. The 40 kyr cycle (a frequency of 0.025 kyr^{-1}) only appears at ca. 450 kyrs before present. Using `cwtft` instead of `cwt` again creates a much smoother result (Fig. 5.17 b).

```
sc.s0 = s0;
sc.ds = ds;
sc.nb = nb;
sig = struct('val',series3L,...
    'period',dt,...
    'wavelet',mother,...
    'scales',sc);
cwtstruct = cwtft(sig);
scales = cwtstruct.scales

f = 1./(4*pi*cwtstruct.scales/(w0+sqrt(2+w0^2)));

contour(t,f,abs(cwtstruct.cfs),...
    'LineStyle','none',...
    'LineColor',[0 0 0],...
    'Fill','on')
xlabel('Time')
ylabel('Frequency')
title('Wavelet Power Spectrum Using FFT Algorithm')
set(gcf,'Colormap',jet)
set(gca,'YLim',[0 0.04],...
    'XGrid','On',...
    'YGrid','On')
```

Compared to the windowed power spectrum method, the wavelet power



a



b

Fig. 5.17 Wavelet power spectrum for the synthetic data series contained in *series_3.txt*. The plot clearly shows significant periodicities at frequencies of 0.01, 0.025, and 0.05 kyr^{-1} corresponding to the 100, 40, and 20 kyr cycles. The 100 kyr cycle is present throughout the entire time series, whereas the 40 kyr cycle only appears at around 450 kyrs before present. The 20 kyr cycle is relatively weak but is probably present throughout the entire time series. The wavelet power spectrum has been calculated using **a** the continuous 1D wavelet transform `cwt` and **b** the continuous wavelet transform using FFT algorithm `cwtft`.

spectrum clearly shows a much higher resolution on both the time and the frequency axes. Instead of dividing the time series into overlapping segments and computing the power spectrum for each segment, the wavelet transform uses short packets of waves that better map temporal changes in the cyclicities. The disadvantage of both the windowed power spectrum and the wavelet power spectrum is, however, the requirement for evenly-spaced data. The Lomb-Scargle method overcomes this problem but (as with the power spectrum method) has limitations in its ability to map temporal changes in the frequency domain.

5.9 Detecting Abrupt Transitions in Time Series

A number of methods are available to detect abrupt changes in time series in the time domain. An example of such such methods for use in climate time series is the *rampfit method* (Mudelsee and Stattegger 1997, Mudelsee 2000), and examples suitable for use in the frequency domain are the evolutionary *power spectrum* and the *wavelet power spectrum* (e.g., Lau and Weng 1995, Mackenzie et al. 2001). In most cases, trends and events in both time and frequency domains are detected by computing the statistical parameters of the data (e.g., measures of central tendency and dispersion) contained in a sliding window of length L. The precision of these parameters depends on the length of the window, i.e., an accurate value for the mean and the variance is obtained if L is large. However, a larger window reduces the accuracy of the estimated changes in these parameters. This problem is often referred to as *Grenander's uncertainty principle of statistics* (Grenander 1958). Performing a statistical test to assess differences in central tendency and dispersion between two different sliding windows, however, partly overcomes this problem, provided only the location of a sharp transition in statistical parameters is required.

The classic t-test and F-test statistics are often used to compare the means and variances of two sets of measurements and could therefore be used to detect changes in the location and dispersion between two sliding windows. These two tests, however, make the basic assumption that the samples came from a population with a Gaussian distribution (Sections 3.7 and 3.8). The non-parametric Mann-Whitney and Ansari-Bradley tests provide a solution to this problem that is independent of the distribution (Sections 3.11 and 3.12). The Mann-Whitney test (Mann and Whitney 1947, Lepage 1971) performs a two-sided rank sum test of the null hypothesis that two samples come from identical continuous distributions with identical medians, against the alternative that they do not have identical medians. The Ansari-Bradley test performs a two-sided test that two independent samples come from the

same distribution, against the alternative that they come from distributions that have the same median and shapes but different dispersions (Ansari and Bradley 1960, Lepage 1971).

The example below demonstrates the Mann-Whitney and Ansari-Bradley tests on two synthetic records that contain significant changes in the central tendency (mean, median, mode) and dispersion (range, variance, quantiles) in the middle of the time series (Fig. 5.18). The time axis runs from 0.1 to 500 kyr at sampling intervals of 0.1 kyr. At 250 kyr the mean of the log-normal distributed data changes abruptly from 1.0 to 1.5 and the standard deviation changes from 0.5 to 1.3 (Fig. 5.18 a).

```
clear
rng(0)
t = 0.1 : 0.1 : 500;
y1 = 0.1 * random('logn',1,   0.5, 1, length(t),1);
y2 = 0.1 * random('logn',1.5, 1.3, 1, length(t),1);
y = y1(1:length(t)/2);
y(length(t)/2+1:length(t)) = y2(length(t)/2+1:length(t));
```

We first use a Mann-Whitney test with paired sliding windows of three different lengths, in order to detect any abrupt change in the mean. We choose sliding window lengths of 300, 500, and 1,000 data points, i.e., in each step we apply the Mann-Whitney test to two samples of 150 data points, two samples of 250 data points, and two samples of 500 data points. Note that when running a Mann-Whitney test on different sets of data the length of the window needs to be adjusted to the length of the time series, and to the required accuracy with which the transition in the mean is to be identified.

```
w = [300 500 1000];
```

We use the function `ranksum` introduced in Section 3.11 to perform the Mann-Whitney test.

```
for j = 1:length(w)
na = w(j);
nb = w(j);
for i = w(j)/2+1:length(y)-w(j)/2
    [p,h] = ranksum(y(i-w(j)/2:i-1),y(i+1:i+w(j)/2));
    mwreal(j,i) = p;
end
mwreal(j,1:w(j)/2) = mwreal(j,w(j)/2+1) * ones(1,w(j)/2);
mwreal(j,length(y)-w(j)/2+1:length(y)) = ...
        mwreal(j,length(y)-w(j)/2) * ones(1,w(j)/2);
end
```

We then display the results.

```
subplot(2,1,1)
```

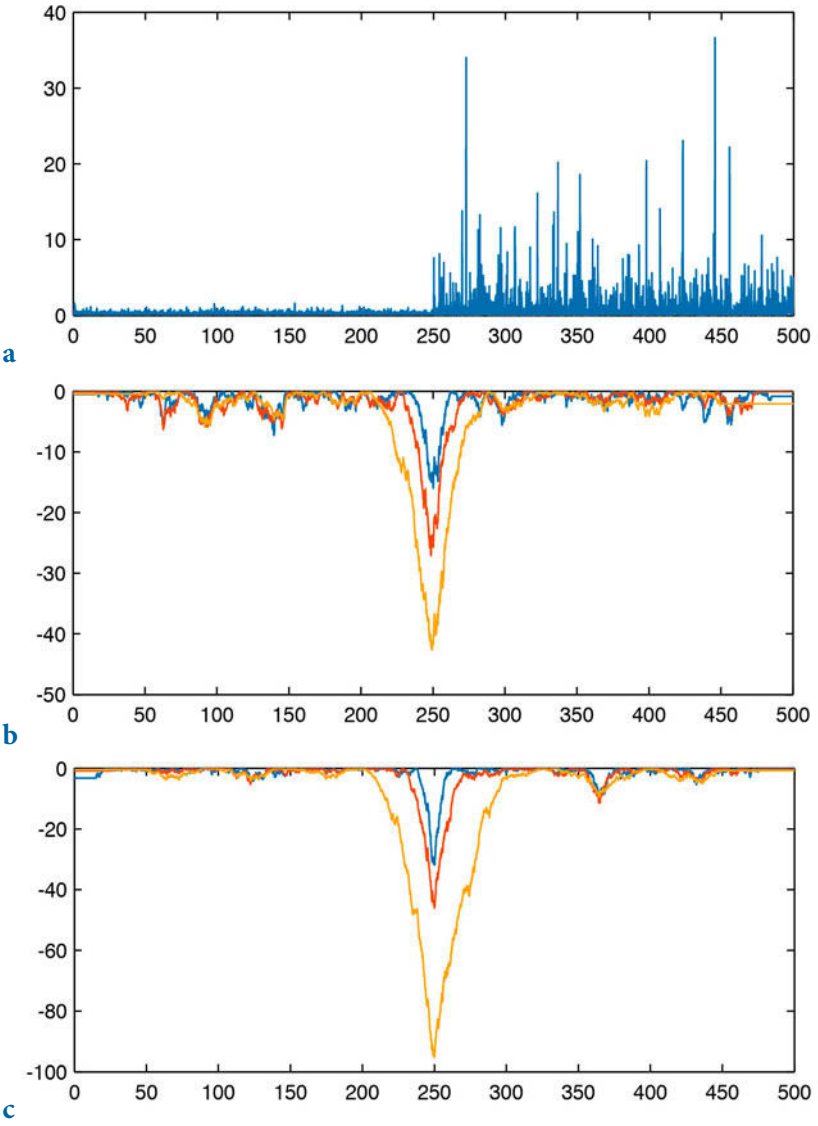


Fig. 5.18 Mann-Whitney and Ansari-Bradley tests on synthetic dust flux record. **a** Lognormal distributed noise. After 250 kyrs the mean and variance of the data shifts towards a lower value. **b** Result of a Mann-Whitney test for three different lengths of the paired sliding windows (150, 250 and 500 data points, equivalent to 15, 25 and 50 kyrs). The length of the window clearly influences the amplitudes and widths of the parameter maxima, whereas the location of the transition in the mean is well defined. **c** Result of a Ansari-Bradley test for three different lengths of the paired sliding windows (150, 250 and 500 data points, equivalent to 15, 25 and 50 kyrs). The length of the window clearly influences the amplitudes and widths of the parameter maxima, and the location of the transition in the dispersion is well defined.

```
plot(t,y)
title('Synthetic signal of lognormal distributed noise')
subplot(2,1,2)
plot(t,log(mwreal))
title('Results from Mann-Whitney U-test')
```

The result from the Mann-Whitney test reveals that the length of the window influences the amplitudes and widths of the maxima of the test parameter, whereas the location of the transition in the means is well defined (Fig. 5.18 b). We next use an Ansari-Bradley test for the same three different lengths of paired sliding windows (150, 250 and 500 data points) to detect any abrupt change in the standard deviation. We use the function `ansaribradley` introduced in Section 3.12 to perform the Ansari-Bradley test.

```
for j = 1:length(w)
df1 = w(j) - 1;
df2 = w(j) - 1;
for i = w(j)/2+1:length(y)-w(j)/2
    [h,p] = ansaribradley(y(i-w(j)/2:i-1),y(i+1:i+w(j)/2));
    abreal(j,i) = p;
end
abreal(j,1:w(j)/2) = abreal(j,w(j)/2+1) * ones(1,w(j)/2);
abreal(j,length(y)-w(j)/2+1:length(y)) = ...
    abreal(j,length(y)-w(j)/2) * ones(1,w(j)/2);
end
```

We then display the results.

```
subplot(2,1,1)
plot(t,y)
title('Synthetic signal of lognormal distributed noise')
subplot(2,1,2)
plot(t,log(abreal))
title('Results from Ansari-Bradley test')
```

The length of the window again clearly influences the amplitudes and widths of the maxima of the test parameters, and the location of the transition in the dispersion is again well defined (Fig. 5.18 c). This method has been successfully applied to records of terrigenous dust flux preserved in marine sediments offshore subtropical West Africa, the eastern Mediterranean Sea, and the Arabian Sea, in order to detect trends, rhythms and events in the African Plio-Pleistocene climate (Trauth et al. 2009).

5.10 Nonlinear Time-Series Analysis (by N. Marwan)

The methods described in the previous sections detect linear relationships in the data. However, natural processes on the earth often show a more complex and chaotic behavior, and methods based on linear techniques may

therefore yield unsatisfactory results. In recent decades, new techniques for nonlinear data analysis derived from chaos theory have become increasingly popular. Such methods have been employed to describe nonlinear behavior by, for example, defining the scaling laws and fractal dimensions of natural processes (Turcotte 1997, Kantz and Schreiber 1997). However, most methods of nonlinear data analysis require either long or stationary data series and these requirements are rarely satisfied in the earth sciences. While most nonlinear techniques work well on synthetic data, these methods are unable to describe nonlinear behavior in real data.

During the last decades, *recurrence plots* have become very popular in science and engineering as a new method of nonlinear data analysis (Eckmann 1987, Marwan 2007). Recurrence is a fundamental property of dissipative dynamical systems. Although small disturbances in such systems can cause exponential divergence in their states, after some time the systems will return to a state that is close to a former state and then pass again through a similar evolution. Recurrence plots allow such recurrent behavior of dynamical systems to be visually portrayed. The method is now widely accepted as a useful tool for the nonlinear analysis of short and nonstationary data sets.

Phase Space Portrait

The starting point for most nonlinear data analyses is the construction of a phase space portrait for a system. The state of a system can be described by its state variables $x_1(t)$, $x_2(t)$, …, $x_d(t)$. As an example, suppose the two variables *temperature* and *pressure* are used to describe the thermodynamic state of the earth's mantle as a complex system. The d state variables at time t form a vector in a d-dimensional space, which is known as the phase space. The state of a system typically changes with time and the vector in the phase space therefore describes a trajectory representing the temporal evolution (i.e., the dynamics) of the system. The trajectory provides essential information on the dynamics of the system, such as whether systems are periodic or chaotic.

In many applications the observation of a natural process does not yield all possible state variables, either because they are not known or because they cannot be measured. However, due to coupling between the system's components, we can reconstruct a phase space trajectory from a single observation u_i:

$$x_i = (u_i, u_{i+\tau}, \ldots, u_{i+(m-1)\tau})^T$$

where m is the embedding dimension and τ is the time delay (index based;

the real time delay is $\tau=\Delta t$). This reconstruction of the phase space is called *time delay embedding*. The reconstruction of the phase space is not exactly the same as the original phase space, but its topological properties are preserved provided that the embedding dimension is sufficiently large. In practice, the embedding dimension must be more than twice the dimension of the attractor (i.e., $m>2d+1$). The reconstructed trajectory is then sufficiently accurate for subsequent data analysis.

As an example we now explore the phase space portrait of a harmonic oscillator such as an undamped pendulum. We first create the position vector `x1` and the velocity vector `x2`

```
clear

t = 0 : pi/10 : 3*pi;
x1 = sin(t);
x2 = cos(t);
```

The phase space portrait

```
plot(x1,x2)
xlabel('x_1')
ylabel('x_2')
```

is a circle, suggesting an exact recurrence of each state after one complete cycle (Fig. 5.19). Using the time delay embedding we can reconstruct this phase space portrait using only a single observation, e.g., the velocity vector, and a time delay of five, which corresponds to a quarter of the period of our pendulum.

```
tau = 5;
plot(x2(1:end-tau),x2(1+tau:end))
xlabel('x_1')
ylabel('x_2')
```

As we can see, the reconstructed phase space is almost the same as the original phase space. Next, we compare this phase space portrait with one for a typical nonlinear system, the Lorenz system (Lorenz 1963). Weather patterns often to not change in a predictable manner. In 1963, Edward Lorenz introduced a simple three-dimensional model to describe the chaotic behavior exhibited by turbulence in the atmosphere. The variables defining the Lorenz system are the intensity of atmospheric convection, the temperature difference between ascending and descending currents, and the distortion of the vertical temperature profiles from linearity. Small variations in the initial conditions can cause dramatically divergent weather patterns, a behavior often referred to as the butterfly effect. The dynamics of the Lorenz

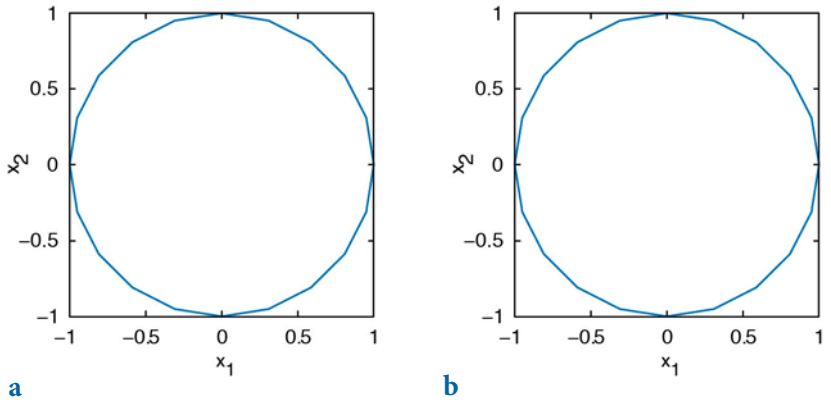


Fig. 5.19 a Original, and **b** reconstructed phase space portrait for a periodic system. The reconstructed phase space is almost the same as the original phase space.

system are described by three coupled nonlinear differential equations:

$$\frac{dx}{dt} = s(y(t) - x(t)),$$
$$\frac{dy}{dt} = -x(t)z(t) + rx(t) - y(t),$$
$$\frac{dz}{dt} = x(t)y(t) - bz(t).$$

Integrating the differential equation yields a simple MATLAB code for computing the *xyz* triplets of the Lorenz system. As system parameters controlling the chaotic behavior we use `s=10`, `r=28` and `b=8/3`; the time delay is `dt=0.01`. The initial values for the position vectors are `x1=8`, `x2=9` and `x3=25`. These values, however, can be changed to any other values, which of course will then change the behavior of the system.

```
clear

dt = .01;
s = 10;
r = 28;
b = 8/3;
x1 = 8; x2 = 9; x3 = 25;
for i = 1 : 5000
   x1 = x1 + (-s*x1*dt) + (s*x2*dt);
   x2 = x2 + (r*x1*dt) - (x2*dt) - (x3*x1*dt);
   x3 = x3 + (-b*x3*dt) + (x1*x2*dt);
   x(i,:) = [x1 x2 x3];
end
```

Typical traces of a variable (such as the first variable) can be viewed by plotting `x(:,1)` over time (Fig. 5.20).

```
t = 0.01 : 0.01 : 50;
plot(t,x(:,1))
xlabel('Time')
ylabel('Temperature')
```

We next plot the phase space portrait for the Lorenz system (Fig. 5.21).

```
plot3(x(:,1),x(:,2),x(:,3))
grid, view(70,30)
xlabel('x_1')
ylabel('x_2')
zlabel('x_3')
```

In contrast to the simple periodic system described above, the trajectories of the Lorenz system obviously do not precisely follow the previous course, but recur very close to it. Moreover, if we follow two very close segments of the trajectory, we see that they run into different regions of the phase space with time. The trajectory is obviously circling around a fixed point in the phase space and then, after a random time period, circling around another. The curious orbit of the phase states around fixed points is known as the Lorenz attractor.

These observed properties are typical of chaotic systems. While small disturbances in such a system cause exponential divergences in its state, the system returns approximately to a previous state through a similar course. The reconstruction of the phase space portrait using only the first state and a time delay of six

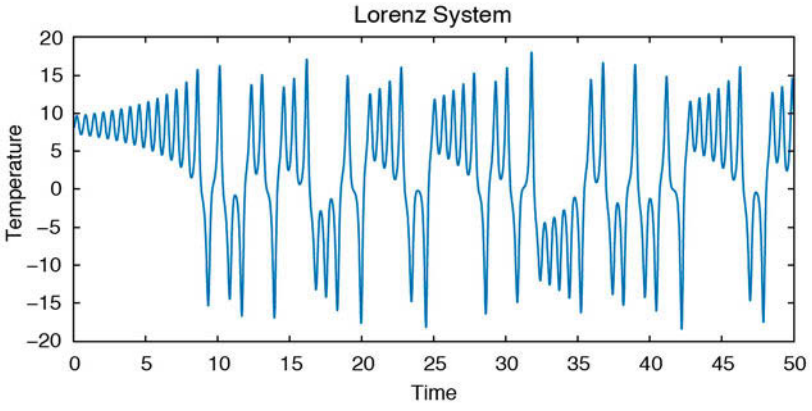


Fig. 5.20 The Lorenz system. As system parameters we use `s=10`, `r=28` and `b=8/3`; the time delay is `dt=0.01`.

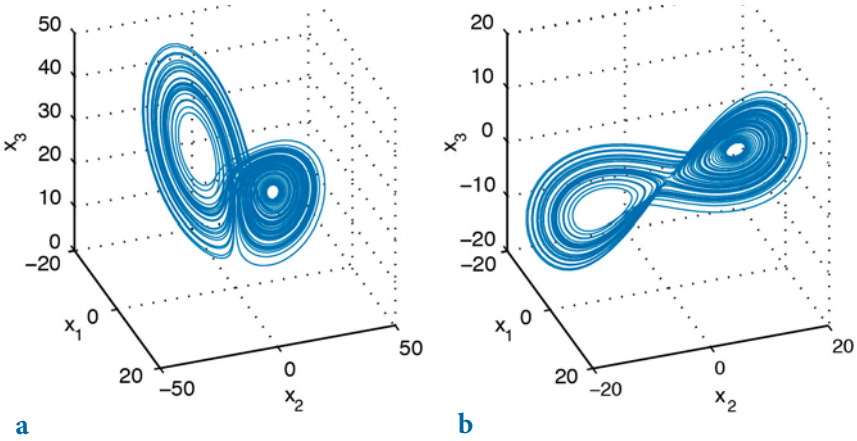


Fig. 5.21 a The phase space portrait for the Lorenz system. In contrast to the simple periodic system, the trajectories of the Lorenz system obviously do not follow precisely the previous course, but recur very close to it. **b** The reconstruction of the phase space portrait using only the first state and a time delay of 6 reveals a topologically similar phase portrait to a, with the two typical ears.

```
tau = 6;
plot3(x(1:end-2*tau,1),x(1+tau:end-tau,1),x(1+2*tau:end,1))
grid, view([100 60])
xlabel('x_1'), ylabel('x_2'), zlabel('x_3')
```

Movie 5.3

reveals a similar phase portrait with the two typical ears (Fig. 5.21). The characteristic properties of chaotic systems can also be observed in this reconstruction.

The time delay and embedding dimension need to be chosen from a previous analysis of the data. The delay can be estimated with the help of the autocovariance or autocorrelation function. For our example of a periodic oscillation,

```
clear

t = 0 : pi/10 : 3*pi;
x = sin(t);
```

we compute and plot the autocorrelation function

```
for i = 1 : length(x) - 2
    r = corrcoef(x(1:end-i),x(1+i:end));
    C(i) = r(1,2);
end

plot(C)
```

```
xlabel('Delay'), ylabel('Autocorrelation')
grid on
```

We now choose a delay such that the autocorrelation function for the first time period equals zero. In our case this is five, which is the value that we have already used in our example of phase space reconstruction. The appropriate embedding dimension can be estimated using the false nearest neighbors method, or more simple, using recurrence plots, which are introduced in the next subsection. The embedding dimension is gradually increased until the majority of the diagonal lines are parallel to the line of identity.

The phase space trajectory or its reconstruction is the basis of several measures defined in nonlinear data analysis, such as *Lyapunov exponents*, *Rényi entropies*, or *dimensions*. The book on nonlinear data analysis by Kantz and Schreiber (1997) is recommended for more detailed information on these methods. Phase space trajectories or their reconstructions are also necessary for constructing recurrence plots.

Recurrence Plots

The phase space trajectories of dynamic systems that have more than three dimensions are difficult to portray visually. *Recurrence plots* provide a way of analyzing systems with higher dimensions. They can be used, e.g., to detect transitions between different regimes, or to detect interrelationships or synchronisations between different systems (Marwan 2007). The method was first introduced by Eckmann and others (1987). The recurrence plot is a tool that displays the recurrences of states in the phase space through a two-dimensional plot.

$$R_{i,j} = \begin{cases} 0 & \left\| x_i - x_j \right\| > \varepsilon \\ 1 & \left\| x_i - x_j \right\| \leq \varepsilon \end{cases}$$

If the distance between two states, i and j , on the trajectory is smaller than a given threshold ε, the value of the recurrence matrix R is one; otherwise it is zero. This analysis is therefore a pairwise test of all states. For N states we compute N^2 tests. The recurrence plot is then the two-dimensional display of the N-by-N matrix, where black pixels represent $R_{i,j}$=1 and white pixels indicate $R_{i,j}$=0, with a coordinate system representing two time axes. Such recurrence plots can help to find a preliminary characterization of the dynamics of a system or to find transitions and interrelationships within a system (cf. Fig. 5.22).

As a first example we load the synthetic time series containing 100 kyr, 40 kyr and 20 kyr cycles already used in the previous sections. Since the data are unevenly spaced, we need to linearly interpolate the data to an evenly-spaced time axis.

```
clear

series1 = load('series1.txt');
t = 0 : 3 : 996;
series1L = interp1(series1(:,1),series1(:,2),t,'linear');
```

We start with the assumption that the phase space is only one-dimensional. Calculating the distances between all points of the phase space trajectory produces the distance matrix `S`.

```
N = length(series1L);
```

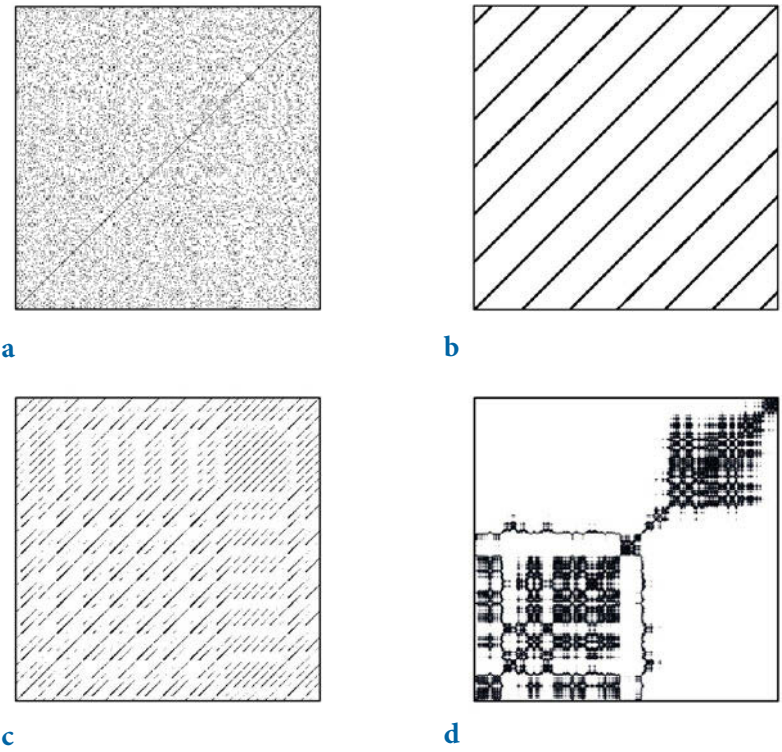


Fig. 5.22 Recurrence plots representing typical dynamical behaviors: **a** stationary uncorrelated data (white noise), **b** periodic oscillation, **c** chaotic data (Roessler system), and **d** non-stationary data with abrupt changes.

```
S = zeros(N, N);

for i = 1 : N,
    S(:,i) = abs(repmat(series1L(i), N, 1 ) - series1L(:));
end
```

We can now plot the distance matrix

```
imagesc(t,t,S)
colormap jet
colorbar
xlabel('Time'), ylabel('Time')
axis xy
```

for the data set, where a colorbar provides a quantitative measure of the distances between states (Fig. 5.23). We now apply a threshold ε to the distance matrix to generate the black/white recurrence plot (Fig. 5.24).

```
imagesc(t,t,S<1)
colormap([1 1 1;0 0 0])
xlabel('Time'), ylabel('Time')
axis xy
```

Both plots reveal periodically recurring patterns. The distances between these periodically recurring patterns represent the cycles contained in the time series. The most significant periodic patterns have periods of 200 and 100 kyrs. The 200 kyr period is the most significant because of the superposition of the 100 and 40 kyr cycles, which are common divisors of 200 kyrs. Moreover, there are smaller substructures within the recurrence plot that have periods of 40 and 20 kyrs.

As a second example we now apply the method of recurrence plots to the Lorenz system. We again generate *xyz* triplets from the coupled differential equations.

```
clear

dt = .01;
s = 10;
r = 28;
b = 8/3;
x1 = 8; x2 = 9; x3 = 25;
for i = 1 : 5000
   x1 = x1 + (-s*x1*dt) + (s*x2*dt);
   x2 = x2 + (r*x1*dt) - (x2*dt) - (x3*x1*dt);
   x3 = x3 + (-b*x3*dt) + (x1*x2*dt);
   x(i,:) = [x1 x2 x3];
end
```

We then choose the resampled first component of this system and reconstruct a phase space trajectory by using an embedding of m=3 and τ=2.

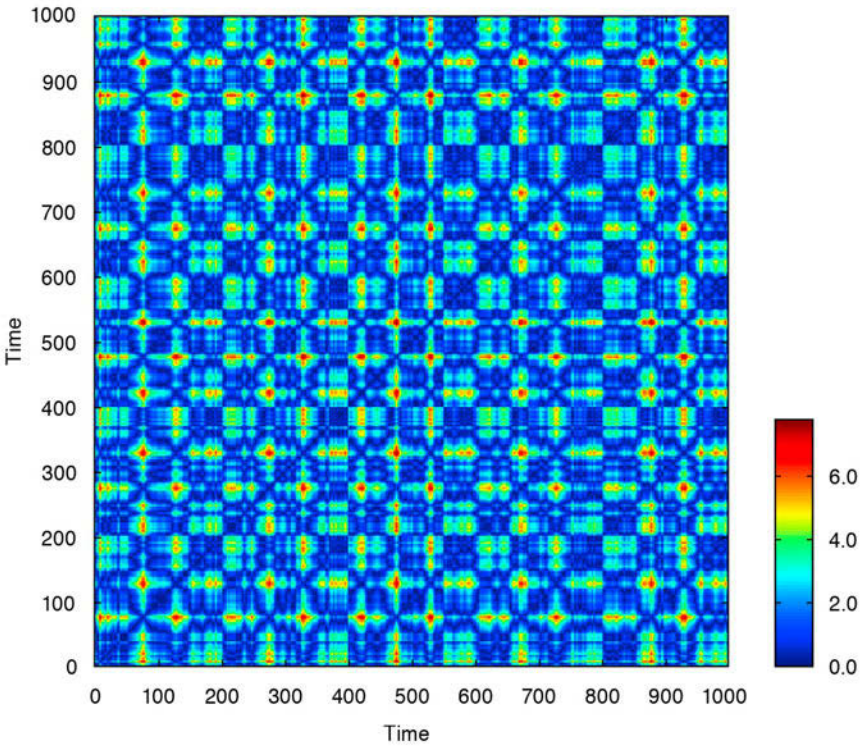


Fig. 5.23 Display of the distance matrix from the synthetic data, providing a quantitative measure for the distances between states at particular times; blue colors indicate small distances and red colors represent large distances.

```
t = 0.01 : 0.05 : 50;
y = x(1:5:5000,1);
m = 3; tau = 2;

N = length(y);
N2 = N - tau*(m - 1);
```

The original data series had a length of 5,000 data points, reduced to 1,000 data points (equivalent to 50 seconds), but because of the time delay method the reconstructed phase space trajectory has a length of 996 data points. We can create the phase space trajectory with

```
for mi = 1:m
   xe(:,mi) = y([1:N2] + tau*(mi-1));
end
```

We can accelerate the pair-wise test between each pairs of points on the

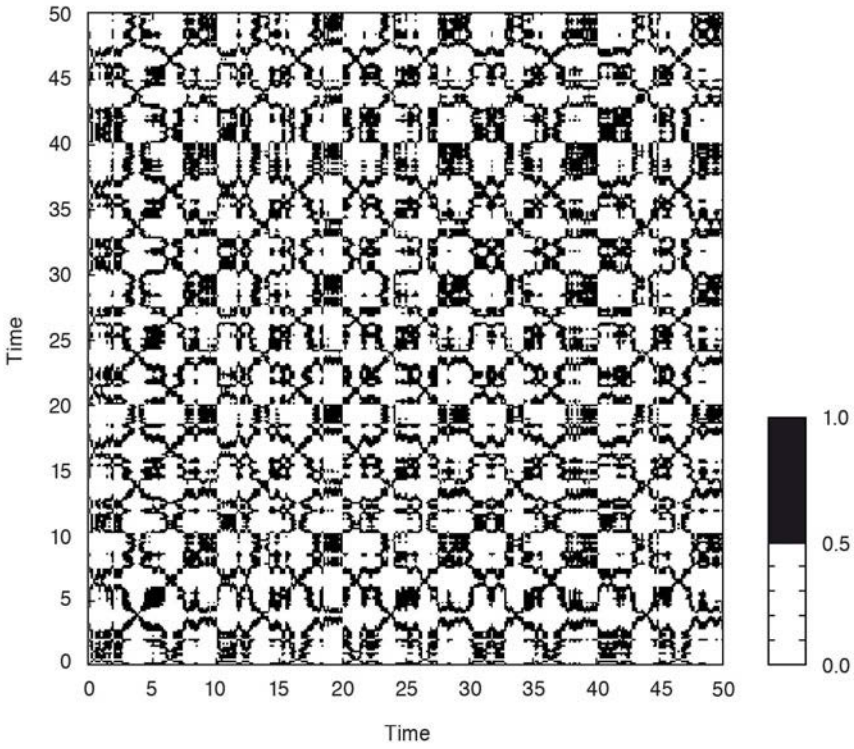


Fig. 5.24 The recurrence plot for the synthetic data derived from the distance matrix, as shown in Fig. 5.23, after applying a threshold of ε=1.

trajectory with a fully vectorized algorithm supported by MATLAB. For this we need to transfer the trajectory vector into two test vectors whose element-wise test will provide the pair-wise test of the trajectory vector:

```
x1 = repmat(xe,N2,1);
x2 = reshape(repmat(xe(:),1,N2)',N2*N2,m);
```

From these vectors we calculate the recurrence plot using the Euclidean norm without any `FOR` loop (see Section 9.4 for details on Euclidean distances).

```
S = sqrt(sum((x1 - x2).^ 2,2 ));
S = reshape(S,N2,N2);

imagesc(t(1:N2),t(1:N2),S<10)
colormap([1 1 1;0 0 0])
xlabel('Time'), ylabel('Time')
axis xy
```

This recurrence plot reveals many short diagonal lines (Fig. 5.25). These

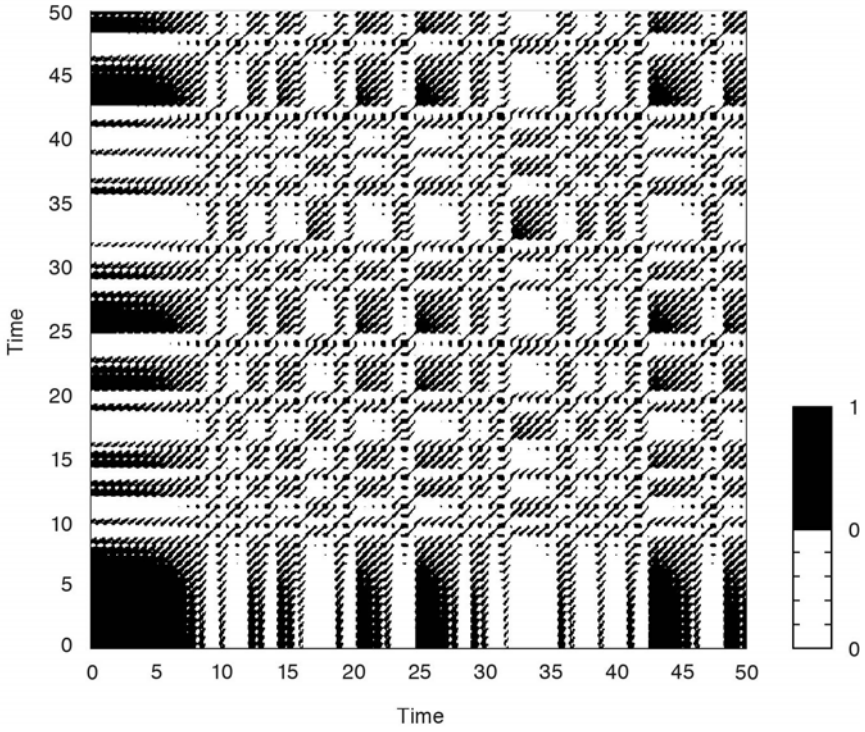


Fig. 5.25 The recurrence plot for the Lorenz system using a threshold of ε=2. The regions with regular diagonal lines reveal unstable periodic orbits, typical of chaotic systems.

Movie 5.4

lines represent periods of time during which the phase space trajectory runs parallel to earlier or later sequences in this trajectory, i.e., periods of times during which the states and dynamics were similar. The distances between these diagonal lines represent the periods of the cycles, which vary and are not constant, in contrast to those for a harmonic oscillation (Fig. 5.22).

Recurrence Quantification

The structure of recurrence plots can also be described by a suite of quantitative measures. Several measures are based on the distribution of the lengths of diagonal or vertical lines, as well as on the local proximity configuration. These measures can be used to trace hidden transitions within a process. As an example we will consider two measures: the recurrence rate and the transitivity coefficient. The recurrence rate is the density of points in the recurrence plot and corresponds to the recurrence probability

of the system. The transitivity coefficient has its roots in graph theory and characterizes the regularity or complexity of the system.

We load the synthetic time from the file *series3.txt*, interpolate the data to an annual time axis, and reconstruct its phase space trajectory using an embedding dimension of 5 and a time delay of 3 (Fig. 5.26).

```
clear

series3 = load('series3.txt');

t = 0 : 1 : 996;
series3L = interp1(series3(:,1),series3(:,2),t,'linear');
plot(t,series3L)
xlabel('Time')

N = length(series3L);
tau = 3; m=5;
N2 = N - tau*(m - 1);

xe = zeros(N2,m);
for mi = 1:m
   xe(:,mi) = series3L([1:N2] + tau*(mi-1));
end
```

Using the vectorized approach we calculate the recurrence plot by applying a threshold of 1.2 to the distance matrix (Fig. 5.27).

```
x1 = repmat(xe,N2,1);
x2 = reshape(repmat(xe(:),1,N2)',N2*N2,m);

S = sqrt(sum((x1 - x2).^ 2,2));
S = reshape(S,N2,N2);
```

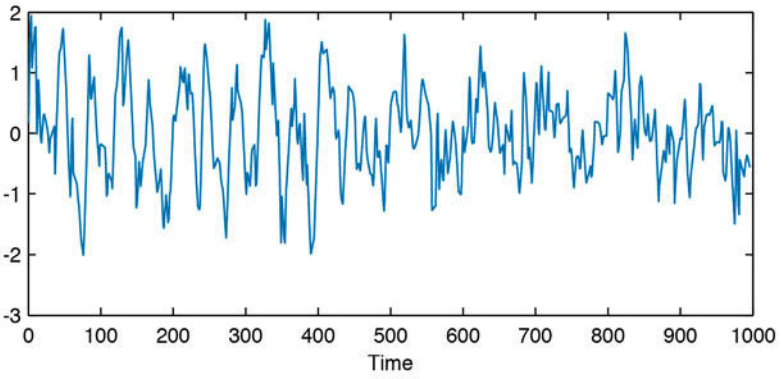


Fig. 5.26 Time series of the synthetic data used in the example of quantitative measures of recurrence plots.

```
R = S<1.2;

imagesc(t(1:N2),t(1:N2),R)
colormap([1 1 1;0 0 0])
xlabel('Time'), ylabel('Time')
axis square xy
```

To calculate the recurrence rate we can simply compute the mean of the matrix `R`

```
RR = mean(R(:))
```

which yields

```
RR =
  0.1399
```

The probability that the system returns to a randomly selected previous state is therefore about 14%.

The transitivity coefficient is a graph-theoretical measure of the probability that three connected network nodes (triples) are completely interconnected,

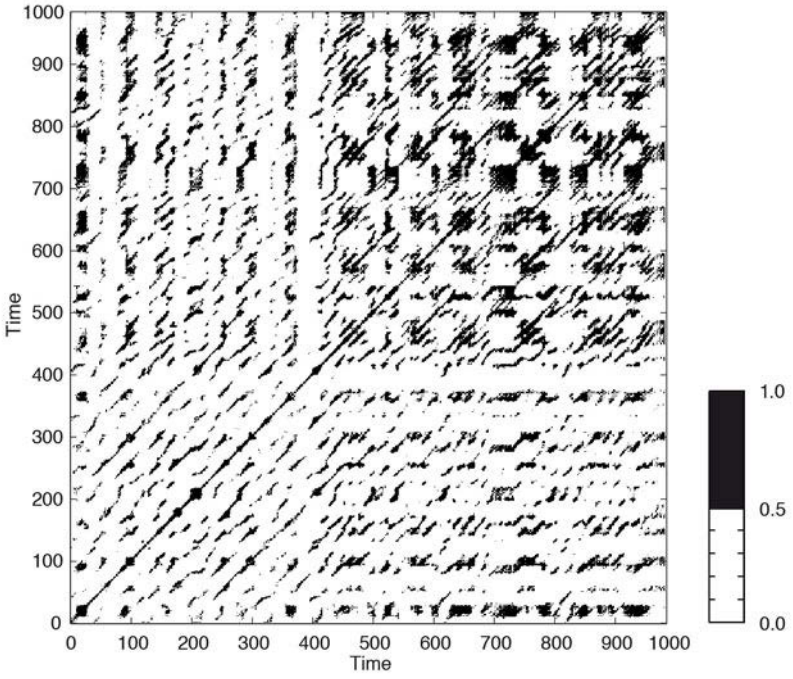


Fig. 5.27 Recurrence plot for the synthetic data in Fig. 5.26, using an embedding of m=5 and τ = 3 and applying a threshold of ε=1.2.

i.e., that they form a closed triangle:

$$\tau = \frac{\sum_{i,j,k}^{N} A_{ij} A_{ij} A_{ik}}{\sum_{i,j,k=1}^{N} A_{ij} A_{ik} \left(1 - \delta_{ik}\right)}$$

This measure can be intuitively understood with respect to recurrences in the phase space. We identify a recurrence of states by a network link: close points on the phase space trajectory are connected by a link. Three connected points form a triple, but only if all three points recur closely to each other, thus forming a triangle. Such a triangular configuration will remain along the phase space trajectory if the dynamic is very regular (recurring states remain recurring over a long period of time). However, if the dynamic is chaotic, then parts of the phase space trajectory that were initially close will subsequently diverge and the triangular configuration will break down, although the corresponding triple nodes might remain interconnected for some time. The probability of finding triangles is therefore higher for regular dynamics but lower for chaotic dynamics. This explanation is, of course, rather simplified but a theoretically substantiated explanation can be found in Donner et al. (2011).

In order to calculate the probability that triples also form triangles we need to compute the number of connected triples and the number of triangles, which can be achieved directly from the recurrence plot but excluding the main diagonal.

```
A = R - eye(size(R));
```

The number of triangles and triples is then

```
numTripl = sum(sum(A * A));
numTria = trace(A * A * A);
```

and finally, the transitivity coefficient is the fraction

```
Trans = numTria/numTripl
```

which yields

```
Trans =
   0.5930
```

This number means that the system does not have regular dynamics (which would yield a transitivity coefficient close to one).

Changes in the dynamics, such as transition points and regime changes,

are often of interest. Recurrence analysis can be used to detect different types of such transitions. Applying moving windows along the main diagonal of the recurrence plot, we divide it into sub-recurrence plots and calculate the recurrence measures of these sub-plots. In our example we choose a moving window length of 150 and an overlap of 20%

```
w = 150;
```

Movie 5.5

We then calculate the recurrence rate and transitivity coefficient within these moving windows (Fig. 5.28).

```
w = 150;
Trans = zeros(length(R)-w,1);
RR = zeros(length(R)-w,1);
for i = 1:w/5:length(R)-w
   subR = R(i:i+w,i:i+w);
   RR(i) = mean(subR(:));
   subA = A(i:i+w,i:i+w);
   numTripl = sum(sum(subA * subA));
   numClosTria = trace(subA * subA * subA);
   Trans(i) = numClosTria/numTripl;
end

plot(t(round(w/2) + (1:w/5:length(RR))), RR(1:w/5:end),...
  t(round(w/2) + (1:w/5:length(RR))), Trans(1:w/5:end))
xlabel('Time')
legend('recurrence rate','transitivity coeff',4)
```

The results suggest slight changes in the dynamics with respect to recurrence probability (due to the visible amplitude variations in the time series) and regularity. For a reliable interpretation of the variations in the recurrence

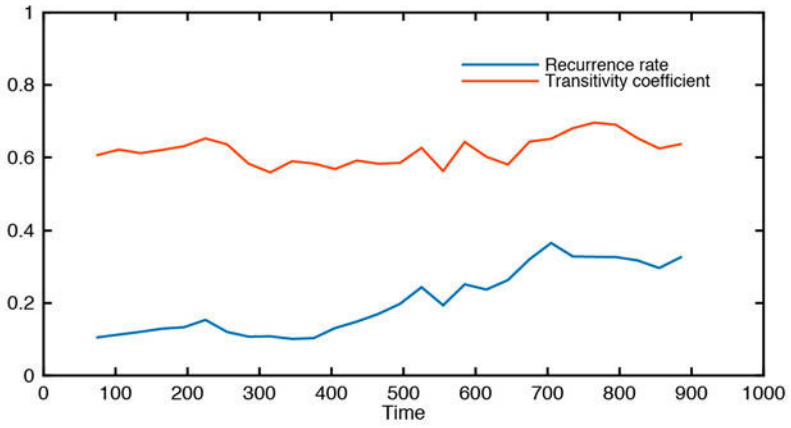


Fig. 5.28 Recurrence rate and transitivity coefficient for the synthetic data in Fig. 5.26, using a moving window of 150 data points and an overlap of 20%.

measures, a statistical test should be applied (Marwan 2011). Other more complex measures that quantify other aspects of the dynamics (e.g., predictability, or laminar phases) are included in the Cross Recurrence Plot Toolbox for MATLAB, available from

http://tocsy.pik-potsdam.de/CRPtoolbox/

Bivariate and multivariate extensions of recurrence plots allow nonlinear correlation tests and synchronization analyses to be carried out. A detailed introduction to methods based on recurrence plots can be found on the following web site:

http://www.recurrence-plot.tk

The analysis of recurrence plots has already been applied to many problems in earth sciences. The comparison of the dynamics of modern precipitation data with paleo-rainfall data inferred from annual-layered lake sediments in the northwestern Argentine Andes provides a good example of such analyses (Marwan et al. 2003). In this instance the recurrence plot method was applied to red-color intensity transects across varved lake sediments that were approximately 30 kyrs old (Section 8.7). Comparing the recurrence plots from the sediments with those from modern precipitation data revealed that the reddish layers document the more intense rainy seasons that occurred during La Niña years. The application of linear techniques was, however, not able to link the increased flux of reddish clays with either the El Niño or La Niña phase of the El Niño/Southern Oscillation. Moreover, recurrence plots helped to prove the hypothesis that longer rainy seasons, enhanced precipitation, and the stronger influence of the El Niño/Southern Oscillation caused an increase in the number of landslides 30 kyrs ago (Marwan et al. 2003, Trauth et al. 2003).

Recommended Reading

Ansari AR, Bradley RA (1960) Rank-Sum Tests for Dispersion. Annals of Mathematical Statistics, 31:1174–1189. [Open access]

Blackman, RB, Tukey, JW (1958) The Measurement of Power Spectra. Dover NY

Cooley JW, Tukey JW (1965) An Algorithm for the Machine Calculation of Complex Fourier Series. Mathematics of Computation 19(90):297–301.

Donner RV, Heitzig J, Donges JF, Zou Y, Marwan N, Kurths J (2011) The Geometry of Chaotic Dynamics – A Complex Network Perspective. European Physical Journal B, 84:653–672

Eckmann JP, Kamphorst SO, Ruelle D (1987) Recurrence Plots of Dynamical Systems. Europhysics Letters 5:973–977

Grenander U (1958) Bandwidth and variance in estimation of the spectrum. Journal of the Royal Statistical Society Series B 20:152–157

Holschneider M (1995) Wavelets, an Analysis Tool. Oxford University Press, Oxford
Kantz H, Schreiber T (1997) Nonlinear Time Series Analysis. Cambridge University Press, Cambridge
Lau KM, Weng H (1995) Climate Signal Detection Using Wavelet Transform: How to make a Time Series Sing. Bulletin of the American Meteorological Society 76:2391–2402
Lepage Y (1971) A combination of Wilcoxon's and Ansari-Bradley's statistics. Biometrika 58:213–271
Lomb NR (1972) Least-Squared Frequency Analysis of Unequally Spaced Data. Astro-physics and Space Sciences 39:447–462
Lorenz EN (1963) Deterministic Nonperiodic Flow. Journal of Atmospheric Sciences 20:130–141
Mackenzie D, Daubechies I, Kleppner D, Mallat S, Meyer Y, Ruskai MB, Weiss G (2001) Wavelets: Seeing the Forest and the Trees. Beyond Discovery, National Academy of Sciences, December 2001, available online at http://www.beyonddiscovery.org
Mann, HB, Whitney, DR (1947) On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other. Annals of Mathematical Statistics 18:50–60
Marwan N, Thiel M, Nowaczyk NR (2002) Cross Recurrence Plot Based Synchronization of Time Series. Nonlinear Processes in Geophysics 9(3/4):325–331
Marwan N, Trauth MH, Vuille M, Kurths J (2003) Nonlinear Time-Series Analysis on Present-Day and Pleistocene Precipitation Data from the NW Argentine Andes. Climate Dynamics 21:317–332
Marwan N, Romano MC, Thiel M, Kurths J (2007) Recurrence Plots for the Analysis of Complex Systems. Physics Reports, 438: 237–329
Marwan N (2011) How to avoid potential pitfalls in recurrence plot based data analysis. International Journal of Bifurcation and Chaos 21:1003–1017
MathWorks (2014a) Signal Processing Toolbox – User's Guide. The MathWorks, Inc., Natick, MA
MathWorks (2014b) Wavelet Toolbox – User's Guide. The MathWorks, Inc., Natick, MA
Mudelsee M, Stattegger M (1997) Exploring the structure of the mid-Pleistocene revolution with advanced methods of time-series analysis. International Journal of Earth Sciences 86:499–511
Mudelsee M (2000) Ramp function regression: A tool for quantifying climate transitions. Computers and Geosciences 26:293–307
Muller RA, MacDonald GJ (2000) Ice Ages and Astronomical Causes – Data, Spectral Analysis and Mechanisms. Springer Verlag, Berlin Heidelberg New York
Press WH, Teukolsky SA, Vetterling WT (2007) Numerical Recipes: The Art of Scientific Computing – Third Edition. Cambridge University Press, Cambridge
Romano M, Thiel M, Kurths J, von Bloh W (2004) Multivariate Recurrence Plots. Physics Letters A 330(3–4):214–223
Scargle JD (1981) Studies in Astronomical Time Series Analysis. I. Modeling Random Processes in the Time Domain. The Astrophysical Journal Supplement Series 45:1–71
Scargle JD (1982) Studies in Astronomical Time Series Analysis. II. Statistical Aspects of Spectral Analysis of Unevenly Spaced Data. The Astrophysical Journal 263:835–853
Scargle JD (1989) Studies in Astronomical Time Series Analysis. III. Fourier Transforms, Autocorrelation Functions, and Cross-Correlation Functions of Unevenly Spaced Data. The Astrophysical Journal 343:874–887
Schulz M, Stattegger K (1998) SPECTRUM: Spectral Analysis of Unevenly Spaced Paleoclimatic Time Series. Computers & Geosciences 23:929–945
Schuster A (1898) On the investigation of hidden periodicities with application to a supposed

26 day period of meteorological phenomena. Terrestrial Magmetism and Atmospheric Electricity 3:13–41

Takens F (1981) Detecting Strange Attractors in Turbulence. Lecture Notes in Mathematics, 898:366–381

Torrence C, Compo GP (1998) A Practical Guide to Wavelet Analysis. Bulletin of the American Meteorological Society 79:61–78

Trulla LL, Giuliani A, Zbilut JP, Webber Jr CL (1996) Recurrence Quantification Analysis of the Logistic Equation with Transients. Physics Letters A 223(4):255–260

Turcotte DL (1992) Fractals and Chaos in Geology and Geophysics. Cambridge University Press, Cambridge

Trauth MH, Bookhagen B, Marwan N, Strecker MR (2003) Multiple Landslide Clusters Record Quaternary Climate Changes in the NW Argentine Andes. Palaeogeography Palaeoclimatology Palaeoecology 194:109–121

Trauth MH, Larrasoaña JC, Mudelsee M (2009) Trends, rhythms and events in Plio-Pleistocene African climate. Quaternary Science Reviews 28:399–411

Weedon G (2003) Time-Series Analysis and Cyclostratigraphy – Examining Stratigraphic Records of Environmental Change. Cambridge University Press, Cambridge

Welch PD (1967) The Use of Fast Fourier Transform for the Estimation of Power Spectra: A Method Based on Time Averaging over Short, Modified Periodograms. IEEE Trans. Audio Electroacoustics AU-15:70–73