7 Causality

7.1 Introduction

In the previous chapters we decomposed multichannel data into its components and we saw that this can be useful to detect structure in complex data sets. Another question that is often posed concerns the causal structure between channels or components: that is, does one channel generates another? In neuroscience the underlying question is: does an area in the brain activate other ones? A typical example is when an epileptologist examines multichannel recordings of brain electrical activity and attempts to find the source (focus) of the epileptic seizures. Often this task is accomplished by finding the signals that lead or lag; the leading signals are then considered as causing the lagging ones. Cross-correlation or nonlinear equivalents can be used to formalize and quantify timing differences between signal pairs in multichannel data sets (see van Drongelen, 2007, chapter 8 to review cross-correlation).

We have to start pessimistically by pointing out that translation from lead−lag to causality is strictly not possible—the example in Fig. 7.1 demonstrates this. If we record from areas A and B in Fig. 7.1A, our method of interpreting lead−lag as a causal relationship A→B is correct. However, if we measure signals from A, B, and C (Fig. 7.1B), we conclude that A→B, A→C, and B→C. We are only partly correct: the two former relationships are correctly inferred but the latter is not. It even would get worse if we had not recorded from A in this example: then we only find B→C and we would be 100% incorrect. So equating lead−lag with causality/connectivity can be incorrect. Having said this, in many studies in neuroscience this is (conveniently) ignored, and timing in signals is frequently used as an argument for connectivity. Often, authors use terms such as "functional connectivity" or "synaptic flow" to (implicitly) indicate the caveats above. Because we often know the typical conduction velocity and delays caused by synaptic transmission, we can (at least) recognize unrealistic delays. For example, if we know that areas B and C in Fig. 7.1B are 10 cm apart and that conduction velocities of the fibers between B and C are ∼1 m/s, we can expect delays ∼100 ms plus a few milliseconds for each synapse involved. Now suppose that in this example the delay between B and C ($\Delta t_2 - \Delta t_1$) is ∼5 ms: such a value is far below the expected delay of over 100 ms, which is an indication that direct connectivity does not play a role in the observed lead−lag between B and C.

Signal Processing for Neuroscientists, A Companion Volume. DOI: 10.1016/B978-0-12-384915-1.00007-3

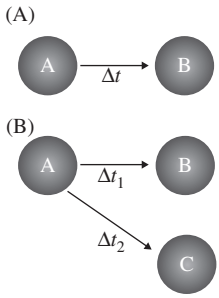

Figure 7.1 Example of areas in the brain A, B, C and their connections, indicated by the arrows. The delays, due to conduction velocity in the connections, between these areas are Δt, Δt_1, Δt_2 with $\Delta t_2 > \Delta t_1$. (A) Due to the delay, activity in area B lags relative to the activity in A. (B) Both B and C lag relative to A, but since $\Delta t_2 > \Delta t_1$, activity in C also lags relative to B.

7.2 Granger Causality

In the 1950s, Norbert Wiener proposed that one signal causes another if your prediction of the latter signal improves by including knowledge of the former. About a decade later, Clive Granger (1969) formalized this concept for linear regression of stochastic processes. To explain the principle, we consider an example of two signals x and y, and we suppose that we can characterize them with the following autoregressive (AR) models:

$$x_n = a_1 x_{n-1} + a_2 x_{n-2} + \cdots + a_i x_{n-i} + \cdots + b_1 y_{n-1} + \cdots + b_i y_{n-i} + \cdots + ex_n \tag{7.1a}$$

$$y_n = c_1 y_{n-1} + c_2 y_{n-2} + \cdots + c_i y_{n-i} + \cdots + d_1 x_{n-1} + \cdots + d_i x_{n-i} + \cdots + ey_n \tag{7.1b}$$

Here $a_{\cdots}$, $b_{\cdots}$, $c_{\cdots}$, and $d_{\cdots}$ are the coefficients and ex_n and ey_n are error terms. If the variance of error ex_n is reduced by including any value of y, y_{n-i} (i.e., $b_i \neq 0$), then we say that y causes x. Similarly, if the variance of error ey_n is reduced by including any value of x, x_{n-i} (i.e., $d_i \neq 0$), then we say that x causes y. In this example, if $b_i \neq 0$ and/or $d_i \neq 0$, we may use the term "Granger causality" to describe the relationship between x and y; we add the qualification "Granger" to indicate that we may not be dealing with a true causal relationship. Similar to the examples given above and in Section 1.3 (where we used the Hilbert transform to detect lead and lag between channels), there may be alternative explanations why x helps to predict y or vice versa.

7.3 Directed Transfer Function

In the following we will link the Granger causality to the z-domain and frequency domain approach of the directed transfer function (DTF), first described by Kamiński and Blinowska (1991). The explanation here will be informal; a formal proof of the relationship between Granger causality and DTF is presented in Kamiński et al. (2001).

7.3.1 Autoregression in the Frequency Domain

To link time and frequency domains, we start from a 1D AR model and its frequency domain presentation. Next, we extend the approach to a multidimensional model and define the DTF. Because we want to follow the derivation by Kamiński and Blinowska (1991), we start our explanation in the time domain and transform our findings into the frequency domain via the z-transform.

7.3.1.1 1D Example

A one-channel model of order p can be represented by:

$$x_n = a_1 x_{n-1} + a_2 x_{n-2} + \cdots + a_i x_{n-i} + \cdots + a_p x_{n-p} + e_n \tag{7.2a}$$

Here x_n are the signal samples, a_i are the AR model's coefficients, and e_n is the error term, which we represent by zero mean GWN.

Equation (7.2a) can be rewritten in the form:

$$x_n - a_1 x_{n-1} - a_2 x_{n-2} - \cdots - a_i x_{n-i} - \cdots - a_p x_{n-p} = e_n \tag{7.2b}$$

Because we deal with a discrete time system, we can transform from the time domain into the z-domain (to review the z-transform, see chapter 9 in van Drongelen, 2007). We define the following transform pairs:

$$\begin{aligned} x_n &\leftrightarrow X(z) \\ e_n &\leftrightarrow E(z) \end{aligned}$$

Applying the z-transform to Equation (7.2b) we get:

$$X(z) - a_1 z^{-1} X(z) - a_2 z^{-2} X(z) - \cdots - a_i z^{-i} X(z) - \cdots - a_p z^{-p} X(z) = E(z) \tag{7.2c}$$

$$\rightarrow X(z)[1 - a_1 z^{-1} - a_2 z^{-2} - \cdots - a_i z^{-i} - \cdots - a_p z^{-p}] = E(z)$$

$$\rightarrow X(z) = \underbrace{\left[\frac{1}{1 - a_1 z^{-1} - a_2 z^{-2} - \cdots - a_i z^{-i} - \cdots - a_p z^{-p}}\right]}_{H(z)} E(z) = H(z)E(z) \tag{7.2d}$$

In Equation (7.2d), we defined $H(z)$ as the transfer function of the system with input noise e_n and output signal x_n.

Recall that the z-transform is the Laplace transform applied to discrete time series (chapter 9 in van Drongelen, 2007); the complex variable z for a time series sampled with interval Δ is defined as $z = e^{s\Delta}$, where s is the complex variable of

the Laplace transform. To get from the z-domain to the frequency domain, we now use the imaginary part of exponent $s = \sigma + j\omega = \sigma + j2\pi f$ of variable z—that is:

$$z^{-1} = e^{-s\Delta} = e^{-(\sigma + j2\pi f)\Delta} \rightarrow z^{-1} = e^{-j2\pi f\Delta}$$

Using this, we can rewrite Equation (7.2d) as a function of frequency f:

$$X(f) = H(f)E(f) \tag{7.2e}$$

Now we have an expression in the frequency domain, where $X(f)$ is the Fourier transform of signal x_n, $E(f)$ is the Fourier transform of the noise input e_n, and $H(f)$ is the frequency response characterizing the system with input e_n and output x_n. Since $X(f)$ is the discrete Fourier transform of x_n, the unscaled power spectrum S for x_n (chapter 7 in van Drongelen, 2007) is:

$$S(f) = X(f)X(f)^* \tag{7.3a}$$

The * indicates the complex conjugate. Equation (7.3a) combined with (7.2e) gives:

$$S(f) = [H(f)E(f)][H(f)E(f)]^*$$

Since this expression is the product of four complex numbers (for each frequency), we may remove the brackets and rearrange the terms:

$$S(f) = H(f)E(f)E(f)^*H(f)^* \tag{7.3b}$$

If the input process e_n is zero mean GWN, its unscaled power spectrum $E(f)E(f)^* = N\sigma^2$, where N is the number of measurements of x_n, and σ^2 is the variance of the noise process.

Note: The equality $E(f)E(f)^* = N\sigma^2$ is directly related to Parseval's theorem (appendix 7.1 in van Drongelen, 2007), stating that the energy of a signal e in the time domain equals the energy of its power spectrum: $\sum(EE^*/N) = \sum e^2$. If the signal has zero mean we may use the following (biased) expression for variance: $\sigma^2 = (1/N)\sum e^2$. Combining the two latter expressions, we get a formula for the sum of the bins in the power spectrum: $\sum(EE^*/N) = N\sigma^2$. Now, if signal e is GWN, the power is equally distributed across the N bins of its spectrum—in other words, the power in each bin (the power for each frequency f) of the normalized spectrum is $N\sigma^2/N = \sigma^2$. Finally, for the non-normalized spectrum EE^* (instead of the normalized one EE^*/N) we find that the value for each frequency bin is $N \times \sigma^2 : E(f)E(f)^* = N\sigma^2$.

If we substitute this result in Equation (7.3b), we get:

$$S(f) = N\sigma^2[H(f)H(f)^*] = N\sigma^2|H(f)|^2 \tag{7.3c}$$

Thus, in this case the spectrum $S(f)$ is proportional with the power of the frequency response $|H(f)|^2 = [H(f)H(f)^*]$. Thus, in the frequency domain $H(f)$ relates input with output—that is, the noise with the signal.

7.3.1.2 Multidimensional Example

The next step is to generalize the above from a one-channel pth-order AR process to a k-channel one. As a first step let us consider a three-channel data set with a pth-order AR process. For this system, the equivalent for Equation (7.2a) becomes:

$$\underbrace{\begin{pmatrix} x(1) \\ x(2) \\ x(3) \end{pmatrix}_n}_{\vec{x}_n} = \underbrace{\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}_1}_{A_1} \underbrace{\begin{pmatrix} x(1) \\ x(2) \\ x(3) \end{pmatrix}_{n-1}}_{\vec{x}_{n-1}} + \underbrace{\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}_2}_{A_2} \underbrace{\begin{pmatrix} x(1) \\ x(2) \\ x(3) \end{pmatrix}_{n-2}}_{\vec{x}_{n-2}}$$
$$+ \cdots + \underbrace{\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}_i}_{A_i} \underbrace{\begin{pmatrix} x(1) \\ x(2) \\ x(3) \end{pmatrix}_{n-i}}_{\vec{x}_{n-i}} + \cdots$$
$$+ \underbrace{\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}_p}_{A_p} \underbrace{\begin{pmatrix} x(1) \\ x(2) \\ x(3) \end{pmatrix}_{n-p}}_{\vec{x}_{n-p}} + \underbrace{\begin{pmatrix} e(1) \\ e(2) \\ e(3) \end{pmatrix}_n}_{e_n} \tag{7.4}$$

or

$$\vec{x}_n = A_1\vec{x}_{n-1} + A_2\vec{x}_{n-2} + \cdots + A_i\vec{x}_{n-i} + \cdots + A_p\vec{x}_{n-p} + \vec{e}_n$$

Here vector:

$$\vec{x}_n = \begin{pmatrix} x(1) \\ x(2) \\ x(3) \end{pmatrix}_n$$

are the samples for the three channels $x(1)-x(3)$, the 3×3 matrix:

$$A_i = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}_i$$

are AR model's coefficients, and vector:

$$\vec{e}_n = \begin{pmatrix} e(1) \\ e(2) \\ e(3) \end{pmatrix}_n$$

are the errors for each channel. Note that we use arrows to indicate that x and e are vectors instead of scalars now. A useful feature is that the **AR coefficients A_i relate the x values across time and more importantly (in this context) across channels**. For instance, for:

$$A_2 = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}_2$$

we have $(a_{11})_2 \times x(1)_{n-2}$ relating past values of $x(1)$ (i.e., the value of $x(1)$ at sample $n-2$) to the present value of $x(1)$ (i.e., its value at sample n); $(a_{12})_2 \times x(2)_{n-2}$ relating past values of $x(2)$ (i.e., the value at sample $n-2$) to the present value of $x(1)$ (i.e., the value at n); $(a_{13})_2 \times x(3)_{n-2}$ relating past values of $x(3)$ (i.e., the value at $n-2$) to the present value of $x(1)$ (i.e., the value at n); $(a_{21})_2 \times x(1)_{n-2}$ relating past values of $x(1)$ (i.e., the value at $n-2$) to the present value of $x(2)$ (i.e., the value at n), etc. Here we can see the relationship between this approach and Granger causality: if one of the coefficients $a_{ij} \neq 0$ (with $i \neq j$), there is a causal relationship (following the definition of Granger causality as discussed in Section 7.2) between the channels i and j, such that $j \rightarrow i$. In the numerical example above, if $(a_{12})_2 \neq 0$, we find that $x(2) \rightarrow x(1)$ (i.e., a causal relationship between channels $2 \rightarrow 1$).

Following our approach for Equation (7.2a), Equation (7.4) can be rewritten in the form:

$$\vec{x}_n - A_1\vec{x}_{n-1} - A_2\vec{x}_{n-2} - \cdots - A_i\vec{x}_{n-i} - \cdots - A_p\vec{x}_{n-p} = \vec{e}_n \tag{7.5a}$$

We now repeat the same procedure as we employed for the one-dimensional case, and get to the frequency domain via the z-transform. First we define the following transform pairs:

$$\vec{x}_n \leftrightarrow \vec{X}(z)$$
$$\vec{e}_n \leftrightarrow \vec{E}(z)$$

The z-transform of Equation (7.5a) is:

$$\vec{X}(z) - A_1 z^{-1}\vec{X}(z) - A_2 z^{-2}\vec{X}(z) - \cdots - A_i z^{-i}\vec{X}(z) - \cdots - A_p z^{-p}\vec{X}(z) = \vec{E}(z) \tag{7.5b}$$

$$\rightarrow \vec{X}(z)[I - A_1 z^{-1} - A_2 z^{-2} - \cdots - A_i z^{-i} - \cdots - A_p z^{-p}] = \vec{E}(z)$$

$$\rightarrow \vec{X}(z) = \underbrace{\left[\frac{I}{I - A_1 z^{-1} - A_2 z^{-2} - \cdots - A_i z^{-i} - \cdots - A_p z^{-p}}\right]}_{H(z)} \vec{E}(z) = H(z)\vec{E}(z) \quad (7.5c)$$

In the above, I is the identity matrix and $H(z)$ is defined as the transfer function matrix between the input noise:

$$\vec{E}(z) = \begin{pmatrix} E(1) \\ E(2) \\ E(3) \end{pmatrix}_z$$

and output signal:

$$\vec{X}(z) = \begin{pmatrix} X(1) \\ X(2) \\ X(3) \end{pmatrix}_z$$

Similar to the procedure we followed in the 1D example above, we now use the imaginary part of exponent $s = \sigma + \mathrm{j}\omega = \sigma + \mathrm{j}2\pi f$ of variable z to get to the frequency domain—that is:

$$z^{-1} = \mathrm{e}^{-s\Delta} = \mathrm{e}^{-(\sigma + \mathrm{j}2\pi f)\Delta} \rightarrow z^{-1} = \mathrm{e}^{-\mathrm{j}2\pi f \Delta}$$

We then rewrite Equation (7.5c) as a function of frequency f:

$$\vec{X}(f) = H(f)\vec{E}(f) \quad (7.5d)$$

The unscaled power spectrum S for $\vec{x_n}$ is:

$$S(f) = \vec{X}(f)\vec{X}(f)^* \quad (7.6a)$$

The * indicates the adjoint, **both** the complex conjugate and the transpose. Note that for each frequency f in Equation (7.6a) S is a 3×3 matrix. Equation (7.6a) combined with Equation (7.5d) gives:

$$S(f) = [H(f)\vec{E}(f)][H(f)\vec{E}(f)]^* = H(f)\vec{E}(f)\vec{E}(f)^* H(f)^* \quad (7.6b)$$

In the above, we used the identity $[H(f)\vec{E}(f)]^* = \vec{E}(f)^* H(f)^*$. If the input noise processes $\vec{e_n}$ are independent white Gaussian with zero mean and variance σ^2, we get

$\vec{E}(f)\vec{E}(f)^* = N\sigma^2 I$, where N is the number of measurements of $\vec{x_n}$ and I is the identity matrix. This generates:

$$S(f) = N\sigma^2[H(f)H(f)^*] \tag{7.6c}$$

Thus, in this case the spectrum $S(f)$ is proportional with the power of the frequency response $[H(f)H(f)^*] = |H(f)|^2$. Thus, in the frequency domain $H(f)$ relates input with output, and it is by definition inversely proportional to $A(f)$ (see Equation (7.5c)).

7.3.1.3 The Directed Transfer Function

Unlike in Equation (7.3c), H in the multichannel version in Equation (7.6c) is not a single value but a matrix for each frequency f (in this example a 3×3 matrix). **Each element H_{ij} in H represents a transfer value between channels j and i.** Again we can see the relationship between this approach and Granger causality: if the transfer value between j and i is nonzero, there is an input−output (causal) relationship. Kamiński and Blinowska (1991) use a normalized version of H, which they call the DTF, to study interrelationships between channels in their data sets. They normalize each component H_{ij} by dividing it by the sum of all elements of H in the same row of the H matrix. Because H_{im} represents the effect of channel m on channel i, you can, in a K-channel recording, normalize by division by the contributions from all channels: $\sum_{m=1}^{K} H_{im}$. Therefore, the normalized version of transfer function element H_{ij} becomes $H_{ij} / \sum_{m=1}^{K} H_{im}$. Because H_{ij} is usually a complex number, this ratio is further simplified to the squared magnitude, generating the commonly used definition of the DTF γ_{ij}:

$$\boxed{\gamma_{ij} = \frac{|H_{ij}|^2}{\sum_{m=1}^{K} |H_{im}|^2}} \tag{7.7}$$

Thus, the DTF can be determined from transfer matrix H, which can be determined using the matrix of AR coefficients A (Equation (7.5c)). A recent study by Wilke et al. (2009) describes how DTF can be combined with adaptive parameter estimation; this adaptive version of DTF allows time-variant coefficients of the AR model to deal with nonstationarity of the EEG signal.

7.3.2 Implementation

After obtaining the above results, it is appropriate to start thinking about an algorithm to determine γ_{ij} from measured data (e.g., a multichannel EEG recording). One practical approach to find the transfer matrix H in a measured data set is to fit an AR model to the data and determine matrix A of the AR coefficients (Section 7.3.1.2). The inverse of A gives H (Equation (7.5c)). This is a parametric

approach, because the basis is to fit parameters of an AR model to the data and the DTF is derived from there. Fitting a multichannel AR model to a data set is an art in itself that is beyond the scope of this chapter. If you want to play with fitting models, there are several Web sites with MATLAB scripts available; one example is the ARfit toolbox from http://www.gps.caltech.edu/~tapio/arfit/.

If you want to evaluate this arfit code you can create your AR model and evaluate the performance of the estimator's output with the known coefficients in your model. The following script, Pr7_1.m, is an example of this procedure. Note that this works only if you download and install the arfit toolbox! Recall to include the arfit directory in the path by using "Set Path . . ." in the "File" menu.

```
% Pr7_1.m
% A test for identifying coefficients from a time series

% The program prints the coefficients we use (a and b)
% plus their estimates (vector A) obtained with the arfit function
% from the ARfit toolbox:http://www.gps.caltech.edu/~tapio/arfit/

clear;
close all;

% Set coefficients a and b
a=0.95;
b=-.55;
% Parameters & initial values for the time series
N=1000;
e=randn(1,N+3); % GWN input
x(1)=0;e(1)=0;
x(2)=0;e(2)=0;

% create the time series using autoregression
for i=3:N+3
   x(i)=a*x(i-1)+b*x(i-2)+e(i);
end;
% Remove the 1st 2nd zero-valued-points from x
x=x(3:N+3);
% Make e the same length
e=e(3:N+3);
% normalize x & e
x=x-mean(x);
x=x/std(x)^2;
e=e-mean(e);
e=e/std(e)^2;

% arfit toolbox
[w,A,C,SBC,FPE,th]=arfit(x',0,4);      % arfit function
% Show the results
```

```
('coefficients')
('in')
[a b]
('estimated')
A
```

When you run the above script you will find that the arfit routine finds reasonable estimates for the coefficients a and b (i.e., your estimates for a and b will be close to 0.95 and −0.55, respectively).

For the implementation we can also go the **nonparametric** route by assuming that the noise sources in our model are zero mean GWN signals. This gives us the expression in Equation (7.6c), which shows us that the unscaled power spectrum $S(f)$ is proportional to $[H(f)H(f)^*] = |H|^2$. Under this assumption, we may determine the power spectrum directly from the data without having to estimate parameters by using the proportionality between S and $|H|^2$ to estimate the DTF.

7.3.2.1 Examples

Let us simulate an example. First we create three signals S1, S2, and S3 with causal relationships S1 → S2 and S1 → S3.

In MATLAB we accomplish this by creating a delay between S1 and S2 (in our example 5 samples delay) and S1 and S3 (10 samples delay). The signal-to-noise ratio (SNR) of our signal in S1 is determined by the variable SNR, and the strength of the coupling of the signal in S1 to S2 and S3 is determined by K2 and K3, respectively (see top diagram in Fig. 7.2).

```
function [S1 S2 S3]=Simulated_Signal(SNR, K2, K3);
% Output signals S1, S2, S3
% the signal-to-noise for S1 is determined by SNR
% the coupling from S1 to S2 and S3 are determined by K2 and K3
% Delay between S2 and S1 is dly2 and between S3 and S1 is dly3
% Linear vs. nonlinear coupling is determined by flag NL
% parameters

sample_rate=400;        % sr in Hz
freq=30;                % f in Hz
tim=40;                 % time in seconds
dly2=5;                 % # points delay and coupling strength signal 1 –>2
dly3=10;                % # points delay and coupling strength signal 1 –>3

% Nonlinear Flag (coupling can be linear (0) or nonlinear (1))
NL=0;
```

```
% derived parms
f_Nyq=sample_rate/2;          % Nyquist
dt=1/sample_rate;             % time step
N=tim*sample_rate;            % no of points
t=0:dt:tim;                   % time axis
noise=randn(1,length(t));     % noise component
null=zeros(1,length(t));      % basline of zeros

% Source Signal
S1=sin(2*pi*freq*t).*SNR*std(noise);
% Create the derived signals S2 and S3
S2=null; S3=null;
for k=dly2+1:N
   if (NL==0);
      S2(k)=S1(k-dly2).*K2+randn;              % linear coupling
   else;
      S2(k)=(S1(k-dly2).^2).*K2+randn;         % nonlinear coupling
   end;
end;
for k=dly3+1:N
   if (NL==0);
      S3(k)=S1(k-dly3).*K3+randn;              % linear coupling
   else;
      S3(k)=(S1(k-dly3).^2).*K3+randn;         % nonlinear coupling
   end;
end;
% Noise added to S1
S1=S1+noise;
% Normalize all signals
S1=(S1-mean(S1))/std(S1);
S2=(S2-mean(S2))/std(S2);
S3=(S3-mean(S3))/std(S3);
```

In our example we run the above function by typing

```
[el1 el2 el3]=Simulated_Signal(3,0.1,0.001);
```

and the result is three traces simulating signals from electrodes 1–3 (el1, el2, and el3) with causal relationships; we then save the result in a file test.mat:

```
save test el1 el2 el3
```

Next, we use the file named test to investigate the causal relation between the signals with the script Pr7_2.m.

```
% Pr7_2.m
% Demo DTF based on signals generated with function Simulated_Signal,
% These signals are saved in File test.mat
% !! the function regres must be in the directory to detrend the data !!

% This program steps with 20 s windows (duration) through
% each of three 40 s signals (el1, el2, el3).
% These 20 second windows move ahead with steps of 5 s (increment).
% So there is 15 s overlap between the 20 s analysis windows)
% Within each window of 20 seconds, the average (cross)spectra
% are computed from fft-analysis epochs of 128 points (step).

% NOTE: THIS PROGRAM IS NOT OPTIMIZED.

clear;
load test                       % load the data with 40 s input traces el1 - el3

% Parameters
cmd0=['N=length(el1)'];         % Determine the length of the signal
eval([cmd0 ';'])
sample_rate=400;                % 400 Hz sample rate
duration=20;                    % duration of the total analysis window in
                                % seconds
step=128;                       % # of points in the FFT analysis window
increment=5;                    % steps of the analysis window in seconds
dt=1/sample_rate;               % sample interval
fNyq=sample_rate/2;             % Nyquist frequency
df=1/(step*dt);                 % Frequency step for the FFT
f=0:df:fNyq;                    % Frequency axis for the FFT

% Plot the three signals el1 - el3 in the top panels
figure
subplot(4,3,1);
plot(el1);hold;axis([0 N min(el1) max(el1)]);
t=['el1'];title(t);
axis('off');
subplot(4,3,2);
plot(el2);hold;axis([0 N min(el2) max(el2)]);
t=['el2'];title(t);
axis('off');
subplot(4,3,3);
plot(el3);hold;axis([0 N min(el3) max(el3)]);
t=['el3'];title(t);
axis('off');
```

```
% MAIN LOOP: STEPPING THROUGH THE DATA
% AND COMPUTING THE (CROSS)SPECTRA
count=0;
for w=1:increment*sample_rate:N-duration*sample_rate
   % Move data window into x, y, z
   x=el1(w:w+duration*sample_rate-1);
   y=el2(w:w+duration*sample_rate-1);
   z=el3(w:w+duration*sample_rate-1);

   % Initialize the Cross-Spectral arrays for averaging
   Sxx=zeros(1,step);
   Syy=Sxx;
   Szz=Sxx;
   Sxy=Sxx;
   Sxz=Sxx;
   Syz=Sxx;

   % SECOND LOOP TO COMPUTE AVERAGE (CROSS)SPECTRA
   xtemp=0:step-1;
   for i=1:step:sample_rate*duration-step;
      % pre-processing x
      [m,b,r]=regres(xtemp,x(i:i+step-1));   % Use regression to compute trend
      trend=m*xtemp+b;
      x(i:i+step-1)=x(i:i+step-1)-trend;                  % detrend
      x(i:i+step-1)=x(i:i+step-1)-mean(x(i:i+step-1));    % demean
      fx=fft(x(i:i+step-1).*hann(step)');                 % windowed fft
      % pre-processing y
      [m,b,r]=regres(xtemp,y(i:i+step-1));
      trend=m*xtemp+b;
      y(i:i+step-1)=y(i:i+step-1)-trend;
      y(i:i+step-1)=y(i:i+step-1)-mean(y(i:i+step-1));
      fy=fft(y(i:i+step-1).*hann(step)');
      % pre-processing z
      [m,b,r]=regres(xtemp,z(i:i+step-1));
      trend=m*xtemp+b;
      z(i:i+step-1)=z(i:i+step-1)-trend;
      z(i:i+step-1)=z(i:i+step-1)-mean(z(i:i+step-1));
      fz=fft(z(i:i+step-1).*hann(step)');
      % compute all 9 spectra which are proportinal with |H|^2, Eq (7.6c)
      Sxx=Sxx+fx.*conj(fx);
      Syy=Syy+fy.*conj(fy);
      Szz=Szz+fz.*conj(fz);
      Sxy=Sxy+fx.*conj(fy);
```

```
    Sxz=Sxz+fx.*conj(fz);
    Syz=Syz+fy.*conj(fz);
    Syx=conj(Sxy);
    Szx=conj(Sxz);
    Szy=conj(Syz);
end;

% Compute the power
S11=abs(Sxx).^2;
S12=abs(Sxy).^2;
S13=abs(Sxz).^2;
S21=abs(Syx).^2;
S22=abs(Syy).^2;
S23=abs(Syz).^2;
S31=abs(Szx).^2;
S32=abs(Szy).^2;
S33=abs(Szz).^2;

% Normalize
NS11=S11./max(S11);          % on diagonal the normalized power spectrum
NS12=S12./(S11+S12+S13); % Eq (7.7)
NS13=S13./(S11+S12+S13); % Eq (7.7)
NS21=S21./(S21+S22+S23); % Eq (7.7)
NS22=S22./max(S22);          % on diagonal the normalized power spectrum
NS23=S23./(S21+S22+S23); % Eq (7.7)
NS31=S31./(S31+S32+S33); % Eq (7.7)
NS32=S32./(S31+S32+S33); % Eq (7.7)
NS33=S33./max(S33);          % on diagonal the normalized power spectrum

count=count+1;

% Plot the results in the corresponding panels and
% superimpose the results for different epochs

% Titles for the panels
ttle1=[' ' num2str(count) ' ' 'Spectrum el1'];
ttle2=' el2 -> el1';
ttle3=' el3 -> el1';
ttle4=' el1 -> el2';
ttle5=' Spectrum el2';
ttle6=' el3 -> el2';
ttle7=' el1 -> el3';
ttle8=' el2 -> el3';
ttle9=' Spectrum el3';
```

```
    % Draw a red horizontal line for each 20 s analysis window
    Y=[0 0];
    X=[w w+duration*sample_rate];
    XP=[w+1-increment*sample_rate w];
    subplot(4,3,1);plot(X,Y,'r');if (count > 1);plot(XP,Y);end;
    subplot(4,3,2);plot(X,Y,'r');if (count > 1);plot(XP,Y);end;
    subplot(4,3,3);plot(X,Y,'r');if (count > 1);plot(XP,Y);end;
    % Plot the (cross)spectral information in the lower 3 × 3 panels
    subplot(4,3,4);hold on; plot(f(1:step/4),NS11(1:step/4),'k');axis([0 60 0 1]);
    title(ttle1);
    subplot(4,3,5);hold on; plot(f(1:step/4),NS12(1:step/4),'k');axis([0 60 0 1]);
    title(ttle2);
    subplot(4,3,6);hold on; plot(f(1:step/4),NS13(1:step/4),'k');axis([0 60 0 1]);
    title(ttle3);
    subplot(4,3,7);hold on; plot(f(1:step/4),NS21(1:step/4),'k');axis([0 60 0 1]);
    title(ttle4);
    subplot(4,3,8);hold on; plot(f(1:step/4),NS22(1:step/4),'k');axis([0 60 0 1]);
    title(ttle5);
    subplot(4,3,9);hold on; plot(f(1:step/4),NS23(1:step/4),'k');axis([0 60 0 1]);
    title(ttle6);
    subplot(4,3,10);hold on; plot(f(1:step/4),NS31(1:step/4),'k');axis([0 60 0 1]);
    title(ttle7);
    subplot(4,3,11);hold on; plot(f(1:step/4),NS32(1:step/4),'k');axis([0 60 0 1]);
    title(ttle8);
    subplot(4,3,12);hold on; plot(f(1:step/4),NS33(1:step/4),'k');axis([0 60 0 1]);
    title(ttle9);

    % Force the script to draw the plots and pause for 1 second
    drawnow;
    pause(1);

    % END MAIN LOOP
end;
```

The final result of running the above scripts is a plot as depicted in Fig. 7.2. The top row of plots in Fig. 7.2 shows the three signals in the time domain and the bottom part (3 × 3 matrix of panels) shows the spectral plus "causal" information. The diagonal panels are the spectra of each of the three channels and the off-diagonal plots show the DTF. It can be seen that, as expected, the first columns shows a consistent energy peak around 30 Hz (the frequency of the test signal in el1) in the DTF, reflecting the causal relationships between el1 and the other two electrodes (el2 and el3). The spectral panels show superimposed traces because spectra and DTFs were determined a number of times for five subsequent epochs of the signals.

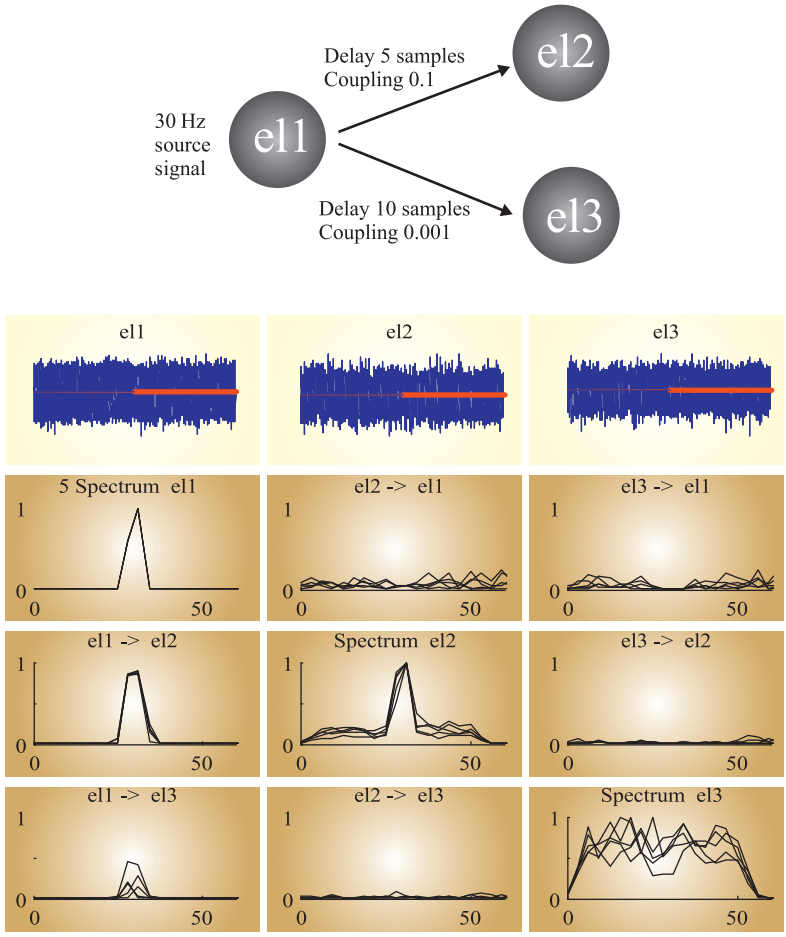

Figure 7.2 Test of the nonparametric DTF algorithm on simulated data. The top diagram shows how three signals el1, el2, and el3 relate to each other. The source el1 contains a 30 Hz sinusoidal signal that is coupled with delays to the other two electrode signals el2 and el3. Noise is added to all three channels. The top panels el1, el2, and el3 show the signals plus their noise components in the time domain and the bottom 3×3 panels are the result of the DTF analysis. These panels show multiple traces, the results from analyzing five overlapping epochs superimposed. Each epoch is 20 s and the overlap between subsequent epochs is 15 s (the red horizontal lines in the three upper panels represent the 20 s analysis windows used for the last of the five epochs). The diagonal panels in the 3×3 arrangement show the power spectra scaled between 0 and 1 of each electrode and the off-diagonal panels show the DTF according to Equation (7.7). It is clear that the first column contains energy around 30 Hz, confirming that el1 is a source for el2 and el3. The graphs were prepared with MATLAB scripts `Simulated_Signal.m` and `Pr7_2.m`; if you repeat this procedure your results may slightly differ (due to the effects of the added noise components).

Recording of brain electrical activity from the scalp (EEG) or the cortex (ECoG) is the clinical basis for the evaluation of patients with epilepsy. These recordings can capture interictal spikes and the epileptic seizures and they can be used to determine the temporal and spatial characteristics of these activity patterns. For surgical candidates, a precise localization of the region where seizures originate is highly significant because it determines the target for surgical resection. Therefore, it is clinical practice to monitor surgical candidates for several days. In such clinical recordings, even if we assume that the electrodes sufficiently cover the brain areas of interest, the determination of the origin of the ictal activity can be far from simple. First, the epileptologist must detect all seizures occurring in a large data set; second, within each seizure the origin of the epileptiform discharges must be determined. To determine the epileptic focus, the epileptologist will use multiple data sets reflecting brain structure and function (EEG, MRI, PET, etc.). The DTF analysis is a natural fit into this set of clinical data because it provides an indicator where activity may originate.

7.4 Combination of Multichannel Methods

Finally we discuss how several of the multichannel techniques can be employed to investigate brain activity. Gómez-Herrero et al. (2008) developed and applied a novel methodology based on multivariate AR modeling and ICA to determine the temporal activation of the intracerebral EEG sources as well as their approximate locations. First these authors used PCA to remove noise components (similar to our example with Lena's image in `Pr6_2.m`) and ICA to identify the EEG sources (as in our example in Fig. 6.15). The direction of synaptic flow between these EEG sources is then estimated using DTF (as we did in the example of Fig. 7.2). The reliability of their approach is assessed with simulations and evaluated by analyzing the EEG-alpha rhythm. Their results suggest that the major generation mechanism underlying EEG-alpha oscillations consists of a strong bidirectional feedback between thalamus and posterior neocortex. Altogether, the study suggests that the combined application of PCA, ICA, and DTF is a promising noninvasive approach for studying directional coupling between neural populations.