Wavelet Analysis of FMRI Time Series Data

B. Douglas Ward
Biophysics Research Institute
Medical College of Wisconsin
email: ward@mcw.edu

March 28, 2000

Abstract

Program 3dWavelets was developed to provide wavelet analysis of FMRI time series data. This program calculates the fast wavelet transform of the time series data, allowing the user to perform wavelet filtering and signal detection for each individual voxel of a 3d+time dataset.

Wavelet analysis is an extension of Fourier analysis in that wavelets provide localization in both time and frequency, whereas the sines and cosines of Fourier analysis localize the signal in frequency only. The capability of localization in both time and frequency makes wavelet analysis a powerful tool for signal analysis. By appropriate selection of a set of time-frequency windows, the user is able to determine which voxels have time series containing features of interest. Program 3dWavelets is described in Section 1.

Program 3dWavelets was developed for use in a "batch" processing mode. It should be used in conjunction with the interactive program plug_wavelets. Program plug_wavelets allows the user to visually inspect the wavelet-derived fit of the signal model to the measured data. The documentation for program plug_wavelets is contained in Section 2.

Contents

1 Program 3dWavelets

1.1 Purpose

Program 3dWavelets was developed to provide wavelet analysis of FMRI time series data. This program calculates the fast wavelet transform of the time series data, allowing the user to perform wavelet filtering and signal detection for each individual voxel of a 3d+time dataset.

Wavelet analysis is an extension of Fourier analysis in that wavelets provide localization in both time and frequency, whereas the sines and cosines of Fourier analysis localize the signal in frequency only. The capability of localization in both time and frequency makes wavelet analysis a powerful tool for signal analysis. By appropriate selection of a set of time-frequency windows, the user is able to determine which voxels have time series containing features of interest. Program 3dWavelets is described in Section 1.

Program 3dWavelets was developed for use in a "batch" processing mode. It should be used in conjunction with the interactive program plug_wavelets. Program plug_wavelets allows the user to visually inspect the wavelet-derived fit of the signal model to the measured data. The documentation for program plug_wavelets is contained in Section 2.

Several programs for time series analysis are included with the *AFNI* distribution. The beginning *AFNI* user may be confused as to which program is appropriate for analysis of a particular experiment. To aid in selecting the appropriate analysis tool, the table below

AFNI Time Series Analysis Programs

Program	Methodology	Experiment Paradigm	Signal Model
3dfim/ AFNI fim	cross-correlation	single stim function block-type design periodic functions short IRF duration	$y(t) = af(t - k \triangle t)$
3dDeconvolve / plug_deconvolve	deconvolution / multiple regression	multiple stim functions rapid presentation random stim functions medium IRF duration	$y(t) = h[0]f(t)$ $+h[1]f(t - \triangle t)$ $+h[2]f(t - 2 \triangle t)$ $+ \cdots + h[m]f(t - m \triangle t)$
3dNLfim / plug_nlfit	nonlinear - regression	single impulse *a priori* signal model long IRF duration	$y(t) = k(e^{-\alpha_1 (t-t_0)}$ $-e^{-\alpha_2 (t-t_0)})u(t - t_0)$
3dWavelets / plug_wavelets	wavelet analysis	single impulse no *a priori* signal model long IRF duration	$y(t) = \sum_i \sum_j c_{ij} \psi(2^i t - j)$

summarizes the *AFNI* time series analysis programs. Note that the table entries, particularly those entries in the "Experiment Paradigm" column, are intended to represent typical applications, and are not meant to be exhaustive listings of all potential applications.

Program 3dfim/AFNI fim uses cross-correlation for signal detection. The measured FMRI time series is cross-correlated with a reference waveform which represents the stimulus function. This program can be used when there is a single stimulus function. The typical experimental paradigm is a so-called block-type design, with an "on" period alternating with an "off" period. It is assumed that the system IRF (impulse response function) has a short duration as compared with the period of the stimulus function.

Program 3dDeconvolve/plug_deconvolve uses deconvolution to estimate the system impulse response function(s) for each voxel. The signal is then estimated by convolving the estimated IRF(s) with the input stimulus function(s). This program is particularly useful for determining the system response to rapid presentation, i.e., when the input impulses are spaced so close that the hemodynamic responses overlap, and for cases where there are multiple stimulus functions.

Program 3dNLfim/plug_nlfit uses nonlinear regression to fit the measured data to a user-defined model for the system response. A typical application is modeling of drug response, where the drug injection constitutes the (single) impulse input. In order for the user to define the system response model, it is helpful to have an *a priori* signal model. For example, drug response is often represented by the "difference-of-exponentials" model.

Program 3dWavelets/plug_wavelets uses wavelet analysis for filtering and signal detection. The measured FMRI time series is decomposed into a sum of orthogonal wavelet functions. By specifying the appropriate time-frequency windows, the user can test for the presence of the desired signal.

From the above discussion, it may be apparent that program 3dWavelets is closest to program 3dNLfim in terms of potential applications. In order to highlight the differences between these two programs, their relative advantages and disadvantages are listed below.

Advantages of using 3dNLfim (vs. 3dWavelets)

- If an *a priori* signal model is available, then the fitted signal model (from 3dNLfim) is assured to have the same functional form as this theoretical model.

- The *a priori* signal model uses relatively few parameters. Since 3dWavelets usually requires fitting of more parameters in order to represent the signal, this adversely effects the statistical power. Therefore, *assuming* that the *a priori* model is adequate, the statistical significance of the results from 3dNLfim is usually better than that obtained using 3dWavelets.

- The fitted signal model from 3dNLfim is smooth (assuming that the mathematical model itself is smooth). This facilitates calculation of various derived parameters (such as area under the curve, time to max signal, etc.).

- The estimated parameters for the *a priori* model may have inherent interest (such as the time of onset of the response, the rate of drug absorption, etc.).

Advantages of using 3dWavelets (vs. 3dNLfim)

- If no *a priori* model is available, then 3dNLfim would require the user to guess at an appropriate mathematical model to represent the data. Program 3dWavelets does not require the user to explicitly specify a particular mathematical function.

- Program 3dNLfim uses the same mathematical model (with varying parameters) for each voxel in the dataset. However, it may be the case that a single mathematical model is not be appropriate for every voxel. Since it does not use an explicit model, program 3dWavelets provides more flexibility in modeling the signal.

- The choice of baseline ("noise") model in 3dNLfim is limited to: constant, linear, quadratic. However, for detection of small features in the time series data, a more elaborate baseline model is useful. Program 3dWavelets provides more flexibility in modeling the "baseline".

- Since 3dWavelets uses the fast wavelet transform for parameter estimation, it is much faster than 3dNLfim, which uses a random search followed by the nonlinear Simplex algorithm, for determining the model parameters.

1.2 Theory

See Refs. [1] – [5]. The notation used below generally follows that of Ref.[2].

1.2.1 Scaling Functions and Wavelets

The objective is to analyze the FMRI time series for a particular voxel. Let this measured signal be represented by: $f(kT), k = 0, ..., N-1$, where T =1TR. By a suitable rescaling, we will assume that $f(t)$ is actually defined on the interval $[0, 1)$.

Fourier analysis uses one type of function, sine waves, as basis functions. Wavelet analysis requires a pair of functions, which are referred to as "scaling functions" and "wavelets", for basis functions. There are many types of wavelets (and their associated scaling functions). Here, we will primarily consider "Haar" wavelets and scaling functions, since they are mathematically the simplest to work with. However, it should be kept in mind that Haar wavelets are not necessarily the best for a particular application.

The Haar scaling function is defined by:

$$\phi_H(t) = \begin{cases} 1 & 0 \le t < 1 \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

and the Haar wavelet is given by:

$$\psi_H(t) = \begin{cases} 1 & 0 \le t < \frac{1}{2} \\ -1 & \frac{1}{2} \le t < 1 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

The Haar scaling function and Haar wavelet are illustrated in Figures 1a and 1b, respectively.

If, in the definition of the Haar scaling function, the t is replaced by $2t$, then $\phi_H(2t)$ is given by:

$$\phi_H(2t) = \begin{cases} 1 & 0 \leq t < \frac{1}{2} \\ \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

and the translated version $\phi_H(2t-1)$ is:

$$\phi_H(2t-1) = \begin{cases} 1 & \frac{1}{2} \leq t < 1 \\ \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

The functions $\phi_H(2t)$ and $\phi_H(2t-1)$ are depicted in Figure 1c.

We see that $\phi_H(2t)$ is a "small scale" version of $\phi_H(t)$, that is, $\phi_H(2t)$ is nonzero on $[0, \frac{1}{2})$, whereas $\phi_H(t)$ is nonzero on $[0, 1)$. As can be verified by inspection of Figure 2, the "small scale" functions $\phi_H(2t)$ and $\phi_H(2t-1)$ can be written as linear combinations of the "large scale" functions $\phi_H(t)$ and $\psi_H(t)$:

$$\begin{aligned} \phi_H(2t) &= \frac{1}{2}\left(\phi_H(t) + \psi_H(t)\right) \\ \phi_H(2t-1) &= \frac{1}{2}\left(\phi_H(t) - \psi_H(t)\right) \end{aligned} \tag{5}$$

It is also possible to go in the opposite direction; i.e., the "large scale" functions $\phi_H(t)$ and $\psi_H(t)$ can be written as linear combinations of the "small scale" functions $\phi_H(2t)$ and $\phi_H(2t-1)$ (see Figure 3):

$$\begin{aligned} \phi_H(t) &= \phi_H(2t) + \phi_H(2t-1) \\ \psi_H(t) &= \phi_H(2t) - \phi_H(2t-1) \end{aligned} \tag{6}$$

More generally, we can define the translated and dilated Haar scaling function by:

$$\phi_{H;m,k}(t) = \phi_H(2^m t - k) = \begin{cases} 1 & \dfrac{k}{2^m} \leq t < \dfrac{k+1}{2^m} \\ \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

and the translated and dilated Haar wavelet by:

$$\psi_{H;m,k}(t) = \psi_H(2^m t - k) = \begin{cases} 1 & \dfrac{k}{2^m} \leq t < \dfrac{k+\frac{1}{2}}{2^m} \\ \\ -1 & \dfrac{k+\frac{1}{2}}{2^m} \leq t < \dfrac{k+1}{2^m} \\ \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

Let the vector spaces $\Phi_{H;m}$ and $\Psi_{H;m}$ be defined:

$$\begin{aligned} \Phi_{H;m} &= span\left\{\phi_{H;m,k} | k = 0, \ldots, 2^m - 1\right\} \\ \Psi_{H;m} &= span\left\{\psi_{H;m,k} | k = 0, \ldots, 2^m - 1\right\} \end{aligned} \tag{9}$$

Then, eqns. (6) imply that $\Phi_{H;0} \subset \Phi_{H;1}$ and $\Psi_{H;0} \subset \Phi_{H;1}$; hence, $\Phi_{H;0} + \Psi_{H;0} \subset \Phi_{H;1}$. Also, eqns. (5) imply that $\Phi_{H;1} \subset \Phi_{H;0} + \Psi_{H;0}$, which, combined with the previous statement yields $\Phi_{H;1} = \Phi_{H;0} + \Psi_{H;0}$. Moreover, it is easy to see that every function in $\Phi_{H;0}$ is orthogonal to every function in $\Psi_{H;0}$; i.e., $\Phi_{H;0} \perp \Psi_{H;0}$.

The same relations hold true at smaller scales, and in fact,for $m = 0, 1, 2, \ldots$, it can be shown that:

$$\begin{cases} \Phi_{H;m+1} = \Phi_{H;m} + \Psi_{H;m} \\ \Phi_{H;m} \perp \Psi_{H;m} \end{cases} \tag{10}$$

which can be written more succinctly as:

$$\Phi_{H;m+1} = \Phi_{H;m} \oplus \Psi_{H;m} \tag{11}$$

In the following sections, we will consider general orthogonal wavelets ψ and scaling functions ϕ (not necessarily Haar wavelets and scaling functions). As above, we can define the translated and dilated scaling function by:

$$\phi_{m,k}(t) = \phi(2^m t - k) \tag{12}$$

and the translated and dilated wavelet by:

$$\psi_{m,k}(t) = \psi(2^m t - k) \tag{13}$$

Then the vector spaces Φ_m and Ψ_m are defined:

$$\begin{aligned} \Phi_m &= span\left\{\phi_{m,k} | k = 0, \ldots, 2^m - 1\right\} \\ \Psi_m &= span\left\{\psi_{m,k} | k = 0, \ldots, 2^m - 1\right\} \end{aligned} \tag{14}$$

We will assume that these vector spaces permit an orthogonal decomposition, as was true for the Haar wavelets:

$$\Phi_{m+1} = \Phi_m \oplus \Psi_m \tag{15}$$

1.2.2 Multiresolution Analysis

The actual physiological response signal $f(t)$ lives in $\mathbf{L}^2$, the space of square-integrable functions, since $f(t)$ has finite energy. The measured FMRI signal $\tilde{f}(t)$ lives in the lower dimensional space Φ, since $\tilde{f}(t)$ samples $f(t)$ only at discrete times. We will assume that $\tilde{f}(t)$ can be well approximated by the signal $f_n(t)$ which is a member of the sub-space Φ_n.

$$
\begin{array}{lll}
\mathbf{L}^2 & & \longleftarrow \text{ Actual signal lives here} \\
\vdots & & \\
\Phi & & \longleftarrow \text{ Measured signal lives here} \\
\vdots & & \\
\Phi_n & & \longleftarrow \text{ Approximated signal lives here} \\
\downarrow & \searrow & \\
\Phi_{n-1} & \Psi_{n-1} & \\
\downarrow & \searrow & \\
\Phi_{n-2} & \Psi_{n-2} & \\
\downarrow & \searrow & \\
\Phi_{n-2} & \Psi_{n-2} & \\
\vdots & & \\
\Phi_1 & & \\
\downarrow & \searrow & \\
\Phi_0 & \Psi_0 &
\end{array} \tag{16}
$$

The above decomposition yields the nested sequence of subspaces:

$$\Phi_0 \subset \Phi_1 \subset \Phi_2 \subset \cdots \subset \Phi_{n-1} \subset \Phi_n \subset \cdots \subset \Phi \subset \cdots \subset \mathbf{L}^2 \tag{17}$$

By repeated application of eqn.(15), we have:

$$
\begin{aligned}
\Phi_n &= \Phi_{n-1} \oplus \Psi_{n-1} && (18) \\
&= (\Phi_{n-2} \oplus \Psi_{n-2}) \oplus \Psi_{n-1} \\
&= (\Phi_{n-3} \oplus \Psi_{n-2}) \oplus \Psi_{n-2} \oplus \Psi_{n-1} \\
&= \ldots \\
&= \Phi_0 \oplus \Psi_0 \oplus \Psi_1 \oplus \cdots \oplus \Psi_{n-2} \oplus \Psi_{n-1}
\end{aligned}
$$

Therefore, for $y_n(t) \in \Phi_n$, we can write:

$$y_n(t) = y_0(t) + f_0(t) + f_1(t) + \cdots + f_{n-2}(t) + f_{n-1}(t), \tag{19}$$

where $y_0(t) \in \Phi_0$, $f_0(t) \in \Psi_0$, $f_1(t) \in \Psi_1$, , $f_{n-1}(t) \in \Psi_{n-1}$.

1.2.3 Fast Wavelet Transform

Since $f_i(t) \in \Psi_i$, it follows that:

$$f_i(t) = \sum_{j=0}^{2^i-1} c_{ij}\psi_{ij}(t) \tag{20}$$

where the c_{ij} are the wavelet coefficients. Also, since $y_0(t) \in \Phi_0$, we can write

$$y_0(t) = d_{00}\phi_{00}(t) \tag{21}$$

Hence, the time series $y_n(t)$ can be written as:

$$y_n(t) = d_{00}\phi_{00}(t) + \sum_{i=0}^{n-1}\sum_{j=0}^{2^i-1} c_{ij}\psi_{ij}(t) \tag{22}$$

Program 3dWavelets calculates the wavelet coefficients, using the fast wavelet transform, at each voxel location. These wavelet coefficients are stored in a vector, in the following order:

$$d_{00}, c_{00}, c_{10}, c_{11}, c_{20}, c_{21}, c_{22}, c_{23}, c_{30}, c_{31}, c_{32}, \ldots, c_{37}, c_{40}, c_{41}, \ldots, c_{n-1,2^{n-1}-1} \tag{23}$$

See Figure 4 for a graphical depiction of the wavelet parameter space (for the particular case of a 64 point time series).

If requested by the user, program 3dWavelets saves the wavelet coefficients for each voxel into an *AFNI* 3d+time dataset.

1.2.4 Wavelet Filtering

By abusing the notation, we will define $c_{-10} = d_{00}$, and $\psi_{-10}(t) = \phi_{00}(t)$. Then we can write

$$y_n(t) = \sum_{(i,j)\in\mathbf{X}} c_{ij}\psi_{ij}(t) \tag{24}$$

where

$$\mathbf{X} = \{(-1,0),(0,0),(1,0),(1,1),(2,0),(2,1),(2,2),(2,3),\ldots,(n-1,2^{n-1}-1)\} \tag{25}$$

Now, if certain of the c_{ij} in eqn.(24) are set to zero, then the inverse wavelet transform will not yield $y_n(t)$, but rather a filtered version of $y_n(t)$, say $y_f(t)$. If we let $\mathbf{F} \subset \mathbf{X}$ be the set of indices for the wavelet coefficients which are to be "zeroed out", then the filtered time series can be written:

$$y_f(t) = \sum_{(i,,j)\in\mathbf{X}-\mathbf{F}} c_{ij}\psi_{ij}(t) \tag{26}$$

1.2.5 Signal Detection

Test of Hypotheses Determining whether the time series for a particular voxel corresponds to a given signal waveform can be expressed in terms of a statistical hypothesis test. The null hypothesis is:

$$H_o : \text{time series is “baseline + noise”} \tag{27}$$

and the alternative hypothesis is:

$$H_a : \text{time series is “signal + baseline + noise”} \ . \tag{28}$$

Define the sets $\mathbf{B}$ and $\mathbf{S}$, $\mathbf{B} \subset \mathbf{X}$, $\mathbf{S} \subset \mathbf{X}$, $\mathbf{B} \cap \mathbf{S} = \emptyset$:

$$\begin{aligned} \mathbf{B} &= \big\{(i,j) | \psi_{ij}(t) \text{ is a basis function for the baseline model}\big\} \\ \mathbf{S} &= \big\{(i,j) | \psi_{ij}(t) \text{ is a basis function for the signal model}\big\} \end{aligned} \tag{29}$$

Then the baseline model time series is given by:

$$b(t) = \sum_{(i,j) \in \mathbf{B}} c_{ij} \psi_{ij}(t) \tag{30}$$

and the signal model time series is given by:

$$s(t) = \sum_{(i,j) \in \mathbf{S}} c_{ij} \psi_{ij}(t) \tag{31}$$

Therefore, the above test of hypotheses can be written as:

$$\begin{aligned} H_o &: \ y(t) = b(t) + \varepsilon(t) \\ \text{vs. } H_a &: \ y(t) = b(t) + s(t) + \varepsilon(t) \end{aligned} \tag{32}$$

where $\varepsilon(t) \sim N(0, \sigma^2)$. Note that the alternative hypothesis includes the baseline model in addition to the “pure” signal waveform.

See Figure 5 for a graphical depiction of wavelet filtering and signal detection.

F-statistic A test of the null hypothesis is made by first determining the parameters which yield the least squares fit for the baseline model, and the parameters which yield the least squares fit for the signal plus baseline model (also called the “full” model). The error sum of squares from fitting the baseline model is defined:

$$\begin{aligned} SSE(B) &= \sum_t (y(t) - b(t))^2 \\ &= \sum_t \left(y(t) - \sum_{(i,j) \in \mathbf{B}} c_{ij} \psi_{ij}(t) \right)^2 \end{aligned} \tag{33}$$

and the error sum of squares from fitting the full model is:

$$\begin{aligned} SSE(F) &= \sum_t (y(t) - b(t) - s(t))^2 \\ &= \sum_t \left(y(t) - \sum_{(i,j)\in\mathbf{B}\cup\mathbf{S}} c_{ij}\psi_{ij}(t) \right)^2 \end{aligned} \tag{34}$$

We have reason to reject the null hypothesis if $SSE(F)$ is much less than $SSE(B)$. However, if $SSE(F)$ is only slightly smaller than $SSE(B)$, then we do not have reason to reject the null hypothesis. Consider the test statistic F^* :

$$F^* = \frac{MS(\text{Regression})}{MS(\text{Error})} = \frac{\dfrac{SSE(B) - SSE(F)}{df_B - df_F}}{\dfrac{SSE(F)}{df_F}} \tag{35}$$

where df_B is the number of degrees of freedom for the baseline model, and df_F is the number of degrees of freedom for the full model. Specifically, we have:

$$\begin{aligned} df_B &= N - b, \\ df_F &= N - (b + s), \\ \text{so } df_B - df_F &= s \end{aligned} \tag{36}$$

where $b = Card(\mathbf{B})$ and $s = Card(\mathbf{S})$.

By the above reasoning, we see that a large value for F^* indicates that signal is present, whereas a small value for F^* suggests that only the baseline plus noise is present. The statistic F^* has the $F(df_B - df_F, df_F)$ distribution under the null hypothesis (Ref. [4]).

Note: If the input data is filtered (see Section 1.2.4), then the degrees of freedom are adjusted as follows:

$$\begin{aligned} df_B &= N - f - b, \\ df_F &= N - f - (b + s), \end{aligned} \tag{37}$$

where $f = Card(\mathbf{F})$.

Program 3dWavelets calculates the F^* statistic for each voxel, and appends these values as one of the sub-bricks of an *AFNI* "bucket" dataset (if so requested by the user).

Coefficient of Multiple Determination The coefficient of multiple determination, R^2, can be used as an indicator for how well the full model fits the data. We define R^2:

$$R^2 \equiv 1 - \frac{SSE(F)}{SSE(B)} \tag{38}$$

Roughly speaking, R^2 is the proportion of the variation in the data (about the baseline) that is explained by the full model. Note that, for every voxel, $0 \leq R^2 \leq 1$. (R^2 is a generalization of the square of the correlation coefficient computed in the fim programs).

Program 3dWavelets calculates R^2 for each voxel, and appends these values as one of the sub-bricks of an *AFNI* "bucket" dataset (if so requested by the user).

1.3 Usage

The syntax for execution of program 3dWavelets is as follows:
3dWavelets [**-type** *wavelet*] [**-input** *fname* | **-input1D** *dname*] [**-mask** *mname*] [**-nfirst** *fnum*] [**-nlast** *lnum*] [**-fdisp** *fval*] [**-filt_stop** *band* min *tr* max *tr*] [**-filt_base** *band* min *tr* max *tr*] [**-filt_sgnl** *band* min *tr* max *tr*] [**-coefts** *cprefix*] [**-fitts** *fprefix*] [**-sgnlts** *sprefix*] [**-errts** *eprefix*] [**-fout**] [**-rout**] [**-cout**] [**-vout**] [**-stat_first**] [**-bucket** *bprefix*]

The different command line options are explained below.

1.4 Options

-type *wname*
The -type command specifies that *wname* is the name of the type of wavelet to be used in the analysis. At present, there are only two choices for *wname*:
Haar — Haar wavelets
Daub — Daubechies wavelets
The default is $wname = Haar$.

-input *fname*
The -input command specifies that *fname* is the filename of the *AFNI* 3d + time data set to be used as input for the wavelet analysis program. The -input command is mandatory *except* when the -input1D command is used in its place.

-input1D *dname*
The -input1D command specifies that *dname* is the filename of the *AFNI* .1D time series data file to be used as input for the wavelet analysis program. That is, instead of a 3d+time dataset, the input consists of only a single FMRI time series as the measured data. Commands which would otherwise generate output files (such as the -bucket command) are ignored. This option allows analysis of, e.g., time series obtained from selected voxels, or time series obtained as the average over an ROI, or a completely artificial time series.
In addition to the screen output, the following ".1D" files are written to the disk:

File	Contents
WA.coefts.1D	forward wavelet transform coefficients
WA.fitts.1D	full model fit to the input time series data
WA.sgnlts.1D	signal model fit to the input time series data
WA.errts.1D	residual errors (i.e., error = (filtered) input data - full model fit)

Warning: This program will overwrite pre-existing ".1D" files which have the same names as those listed above.

-mask *mname*
The optional -mask command specifies that *mname* is the filename of the *AFNI* 3d dataset to be used for "masking " the input data. That is, if a voxel in the mask dataset has value zero, then the corresponding voxel in the 3d+time input dataset will be ignored for computational purposes. All output corresponding to that particular voxel will be set to zero. If the mask dataset represents the brain, i.e., if the mask contains 1's only at locations inside the brain, and 0's at locations outside the brain, this will greatly improve the program execution speed. (See program 3dIntracranial.)

Of course, the mask dataset must have the same voxel dimensions as the input 3d+time dataset.

-nfirst *fnum*
The optional -nfirst command specifies that $fnum$ is the number of the first image to be used in the analysis. (Note: the first image in the dataset is numbered 0.) The default value is $fnum = 0$.

-nlast *lnum*
The optional -nlast command specifies that $lnum$ is the number of the last image to be used in the analysis. (Note: the first image in the dataset is numbered 0). The default value is $lnum$ = number of the last image in the dataset. (The program 3dinfo can be used to print out information about a dataset, including the number of time points.)

-fdisp *f*
The optional -fdisp command is used to control output to the user's terminal during program execution. For each voxel in the data set, if the regression $F-$statistic is greater than or equal to f, then the estimated baseline and signal model wavelet coefficients are written to the screen; otherwise, nothing is written to the screen for that particular voxel. Note that the -fdisp command effects screen output only, and has absolutely no effect upon the data file output generated by the program. By default, this option is disabled.

-filt_stop *band* mintr maxtr The optional -filt_stop command is used to specify which wavelet transform coefficients are to be set to zero. The integer *band* specifies the frequency band, and the integers $mintr$ and $maxtr$ specify the time interval. Any wavelet coefficients c_{ij} whose corresponding time-frequency window falls (completely) within the specified band and time interval are set to zero. Thus, this command effectively defines the set **F** of indices of wavelet coefficients as described in Section 1.2.4.

-filt_base *band* mintr maxtr The optional -filt_base command is used to specify which wavelet transform coefficients should be used in constructing the baseline model. The integer *band* specifies the frequency band, and the integers $mintr$ and $maxtr$ specify the time interval. Any wavelet coefficients c_{ij} whose corresponding time-frequency window falls (completely) within the specified band and time interval are included in the baseline time series model. Thus, this command effectively defines the set **B** of indices of baseline model wavelet coefficients as described in Section 1.2.5.

-filt_sgnl *band* min *tr* max *tr* The optional -filt_sgnl command is used to specify which wavelet transform coefficients should be used in constructing the signal model. The integer *band* specifies the frequency band, and the integers *mintr* and *maxtr* specify the time interval. Any wavelet coefficients c_{ij} whose corresponding time-frequency window falls (completely) within the specified band and time interval are included in the signal time series model. Thus, this command effectively defines the set **S** of indices of signal model wavelet coefficients as described in Section 1.2.5.

-coefts ***cprefix***

The optional -coefts command instructs program 3dWavelets to save the forward wavelet transform coefficients for each voxel. See Section 1.2.3 for a description of the order in which the wavelet coefficients are stored. The wavelet coefficients are stored (as if a time series) into an *AFNI* 3d+time dataset. The output dataset has prefix filename cprefix.

Note: If the -filt_stop command is used, the corresponding wavelet coefficients are set to zero.

-fitts ***fprefix***

The optional -fitts command instructs program 3dWavelets to save the full model fit to the input time series data for each voxel. Recall that the full model is the signal model plus the baseline model. The full model time series are stored into an *AFNI* 3d+time dataset having prefix filename fprefix.

Note: If neither the baseline model nor the signal model is defined, then the -fitts command writes out the filtered input data, i.e., the input data after the wavelet coefficients specified by the -filt_stop command have been removed.

-sgnlts ***sprefix***

The optional -sgnlts command instructs program 3dWavelets to save the signal model fit to the input time series data for each voxel. Recall that the signal model does not include the baseline model. The signal model time series are stored into an *AFNI* 3d+time dataset having prefix filename sprefix.

-errts ***eprefix***

The optional -errts command instructs program 3dWavelets to save the residual errors (i.e., error = (filtered) input data - full model fit) into an *AFNI* 3d+time dataset. The output dataset has prefix filename eprefix.

Note: If neither the baseline model nor the signal model is defined, then the -errts command writes out the difference between the input data and the filtered input data (i.e., error = input data - filtered input data).

-bucket *bprefix*

The -bucket command is used to create a single *AFNI* "bucket" type dataset having multiple sub-bricks. The output is written to the file with the user specified prefix filename bprefix. Each of the individual sub-bricks can then be accessed for display within program afni. See Examples 3 and 4 below for illustrations of the format of the bucket dataset.

The following commands control the contents of the output bucket dataset:

-fout	Flag to output the full model F-statistics
-rout	Flag to output the full model R^2
-cout	Flag to output the full model wavelet coefficients
-vout	Flag to output the sample variance (MSE) map
-stat_first	Flag to specify that the full model statistics will appear first (prior to the wavelet coefficients)

1.5 Problems, Limitations, Future Improvements

- At the present time, the number of data points used in the wavelet analysis must be a power of 2. If the number N of (selected) time series data points is not a power of 2, the program automatically truncates the time series to the largest power of 2 less than N.

- The Daubechies wavelet coefficients are calculated by convolution with a finite length filter. As a result, the time window corresponding to a particular Daubechies wavelet coefficient is not sharply defined (in contrast with the Haar wavelet coefficients). Also, periodic extension of the data is used for calculation of the Daubechies wavelet coefficients. This makes statistical interpretation of the results more difficult. Therefore, it is safer to use the Haar wavelets for statistical analysis of time series data. However, since the Daubechies wavelets are continuous, they usually yield better curve fits to the measured data. Therefore, the particular application should determine which type of wavelet is used.

- Other types of wavelets may be included in future releases of this program.

- This program applies wavelet analysis to 1D (time series) data. Future programs may apply multidimensional wavelet analysis to 2D or 3D (image) data, or even to 4D (image+time) data.

1.6 Examples

Example 1. Wavelet filtering in the frequency domain

Here, we consider a possible application of wavelet filtering for the purpose of removing the high frequency noise components of a dataset. Suppose that Barbara+orig is a 3d+time dataset having 260 time points. The following batch command file will produce a filtered version of the data.

Program 3dWavelets Batch Command File for Example 1

```
3dWavelets \
-input Barbara+orig \
-nfirst 2 \
-nlast 257 \
-type Haar \
-filt_stop   6  0  300 \
-filt_stop   7  0  300 \
-fitts Barbara.filtrd
```

■

The output 3d+time dataset, Barbara.filtrd+orig, contains the wavelet filtered version of Barbara+orig. The number of selected time series data points is $N = 257 - 2 + 1 = 256$. Since $N = 2^8$, the highest frequency band is $8 - 1 = 7$. Thus, the two -filt_stop commands remove the 2 highest frequency bands. This should get rid of most of the high frequency "noise" in each time series. The output filtered time series has length 256.

The question arises: Is it legitimate to use this filtered time series as input to another time series analysis program, say 3dfim or 3dDeconvolve? The answer depends on the particular application. Each of the time series analysis programs assumes that every time point represents an independent measurement, i.e., that the measurement errors are independent. However, the effect of filtering the data in the wavelet domain was to set 192 of the 256 wavelet coefficients to 0. Therefore, each time series has only 64 degrees of freedom. Thus, it is not proper to use the filtered time series data as input to another time series analysis program for the purpose of *statistical analysis of the time series itself.*

However, this is only one possible application of the filtered data. Another potential application is parameter estimation. The user may be interested in estimating various parameters related to the signal response. Filtering the data may help reduce the error in the parameter estimates. One could then use the parameter estimates as input to statistical analysis across subjects and across experimental conditions. In this case, the error degrees of freedom depends on the number of subjects, number of runs, etc., but does *not* depend on the number of independent measurements within each time series. As long as each time series is treated equally, there is no theoretical problem in performing statistical analysis of the parameters estimated from the filtered data.

Example 2. Wavelet filtering in both frequency and time domains

In the previous example, wavelet filtering was applied equally at all time points of the input 3d+time dataset. However, in certain cases it may be advantageous to use the capability of wavelets to discriminate in both frequency and in time. For example, suppose that the 3d+time dataset Hugh+orig shows a large perturbation of the time series, at each voxel location, at about time = 151 TR, and a second perturbation at around 215 TR. Also, assume that after each perturbation, the time series returns to its previous behavior (i.e., the same offset, linear drift, etc.). One possible solution would be to remove the offending time points by cutting them out of the 3d+time dataset. However, this would upset the correspondence between consecutive images and consecutive points in time. This would have negative consequences for programs such as 3dDeconvolve, for which the timing of events is important.

Another possible solution would be to low-pass filter the entire time series, in order to remove the high frequency noise, as in Example 1. However, that would effect all of the data, not just the time points near the perturbed data. Furthermore, as was mentioned in the previous example, this would significantly reduce the error degrees of freedom, which invalidates the time series statistical analysis.

A third possibility is to filter the time series, but only in the immediate neighborhood of the disturbances. This is possible using wavelet filtering, as indicated by the following batch command file.

Program 3dWavelets Batch Command File for Example 2

```
3dWavelets \
-input Hugh+orig \
-nfirst 2 \
-nlast 513 \
-type Haar \
-filt_stop   8  150 152 \
-filt_stop   8  214 216 \
-fitts Hugh.filtrd
```

■

Here, the highest frequency band has been filtered, but only near the two disturbances. Since only a few time points have been filtered (smoothed), this should have negligible effect on the error degrees of freedom.

Warning: As always, great caution should be exercised in any manipulation of data that is subject to further statistical analysis. When in doubt, don't.

Example 3. Wavelet signal detection in the frequency domain

The measured FMRI data is contained in the 3d+time dataset Tony+orig (.HEAD and .BRIK). In order to calculate the wavelet transform and estimate the signal function, at each voxel location, program 3dWavelets can be executed with the following batch commands:

Program 3dWavelets Batch Command File for Example 3

```
3dWavelets \
-input Tony+orig \
-nfirst 2 \
-nlast 513 \
-type Haar \
-filt_base  -1  0  600 \
-filt_base   0  0  600 \
-filt_sgnl   1  0  600 \
```

```
-filt_sgnl   2  0  600 \
-filt_sgnl   3  0  600 \
-fdisp 5.0 \
-fout -rout -cout \
-bucket Tony.bucket \
-coefts Tony.coefts \
-fitts Tony.fitts \
-errts Tony.errts
```

■

The first batch command specifies that the input 3d+time dataset is to be read from file Tony+orig (.HEAD and .BRIK). The command -nfirst specifies that image #2 is the first to be used in the analysis, i.e., to allow for transients, the first three data points in each time series are to be discarded. The command -nlast specifies that image #513 is the last to be used in the analysis. Hence, $N = 513 - 2 + 1 = 512$ time points will be used in the analysis. Since $512 = 2^9$, the highest frequency band is $9 - 1 = 8$.

The command -type Haar indicates that Haar wavelets should be used in the wavelet analysis.

The next 5 commands are used to specify the baseline and signal models. The first 2 of these commands, -filt_base, specify that the baseline model contains wavelets whose time-frequency window indices are in the set:

$$\mathbf{B} = \{(-1, 0), (0, 0)\}$$

The next 3 of these commands, -filt_sgnl, specify that the signal model contains wavelets whose time-frequency window indices are in the set:

$$\mathbf{S} = \left\{ \begin{array}{c} (1,0), (1,1), (2,0), (2,1), (2,2), (2,3), (3,0), \\ (3,1), (3,2), (3,3), (3,4), (3,5), (3,6), (3,7) \end{array} \right\}$$

The command -fdisp 5.0 is used to specify that, during execution of the program, screen output is generated for voxels whose F-statistic exceeds 5.0.

The -fout, -rout, and -cout commands indicate that the F-statistics, R^2, and full model wavelet coefficients are to be included in the bucket dataset output. The command -bucket Tony.bucket is then used to generate the "bucket" type dataset Tony.bucket+orig (.HEAD and .BRIK) containing the full model wavelet coefficients, and the F-statistic and R^2 for significance of the full model.

The last three commands specify the prefix filenames to be used for 3d+time dataset output. The command -coefts Tony.coefts generates the 3d+time dataset Tony.coefts+orig (.HEAD and .BRIK), which contains the complete set of forward wavelet transform coefficients for each voxel. The command -fitts Tony.fitts generates the 3d+time dataset Tony.fitts+orig (.HEAD and .BRIK), which contains the estimated full model time series for each voxel. Finally, the command -errts Tony.errts generates the 3d+time dataset Tony.errts+orig (.HEAD and .BRIK), which contains the residual error (original data - full model fit) for each voxel.

During program execution, output is written to the screen for those voxels whose F-statistic for significance of the regression exceeds 5.0, as illustrated below.

Program 3dWavelets Screen Output for Example 3

⋮

Results for Voxel #2337:

Full Model Wavelet Coefficients:

$B($	$-1)[$	$2,$	$513] =$	80.402344
$B($	$0)[$	$2,$	$513] =$	0.703125
$S($	$1)[$	$2,$	$257] =$	-11.527344
$S($	$1)[$	$258,$	$513] =$	5.847656
$S($	$2)[$	$2,$	$129] =$	-3.171875
$S($	$2)[$	$130,$	$257] =$	-2.960938
$S($	$2)[$	$258,$	$385] =$	3.156250
$S($	$2)[$	$386,$	$513] =$	-0.664062
$S($	$3)[$	$2,$	$65] =$	-2.531250
$S($	$3)[$	$66,$	$129] =$	-15.718750
$S($	$3)[$	$130,$	$193] =$	-3.359375
$S($	$3)[$	$194,$	$257] =$	-3.062500
$S($	$3)[$	$258,$	$321] =$	0.203125
$S($	$3)[$	$322,$	$385] =$	3.015625
$S($	$3)[$	$386,$	$449] =$	-5.187500
$S($	$3)[$	$450,$	$513] =$	6.140625

Baseline:

# params	=	2
SSE	=	279636.000
MSE	=	548.306

Full Model:

# params	=	16
SSE	=	210857.500
MSE	=	425.116

Summary Statistics:

R^2	=	0.246
F[14,496]	=	11.556

⋮

Results for Voxel #2342

etc.

■

After program 3dWavelets has finished execution, program afni can be used to view the output files.

The format for the bucket dataset Tony.bucket is illustrated below.

Brick #	Label		Model	Freq.	Time
				Band	Interval
0	B(-1)[2,513]		Baseline	-1	$2 \leq t \leq 513$
1	B(0)[2,513]		Baseline	0	$2 \leq t \leq 513$
2	S(1)[2,257]		Signal	1	$2 \leq t \leq 257$
3	S(1)[258,513]		Signal	1	$258 \leq t \leq 513$
4	S(2)[2,129]		Signal	2	$2 \leq t \leq 129$
5	S(2)[130,257]		Signal	2	$130 \leq t \leq 257$
6	S(2)[258,385]		Signal	2	$258 \leq t \leq 385$
7	S(2)[386,513]		Signal	2	$386 \leq t \leq 513$
8	S(3)[2, 65]		Signal	3	$2 \leq t \leq 65$
9	S(3)[66,129]		Signal	3	$66 \leq t \leq 129$
10	S(3)[130,193]		Signal	3	$130 \leq t \leq 193$
11	S(3)[194,257]		Signal	3	$194 \leq t \leq 257$
12	S(3)[258,321]		Signal	3	$258 \leq t \leq 321$
13	S(3)[322,385]		Signal	3	$322 \leq t \leq 385$
14	S(3)[386,449]		Signal	3	$386 \leq t \leq 449$
15	S(3)[450,513]		Signal	3	$450 \leq t \leq 513$
16	Full R^2		Coefficient of multiple determination R^2		
17	Full F-stat		F-statistic for significance of the full model		

As indicated above, sub-bricks containing wavelet coefficients are either labeled "B" for baseline, or "S" for signal model. This is followed by a single number in parenthesis which refers to the frequency band. This is followed by two more numbers inside square brackets, which specify the time window.

Example 4. Wavelet signal detection in both frequency and time domains

In the previous example, wavelets were used to test for the presence of the signal. However, only the frequency content was used to discriminate between signal and noise. In many cases, it is appropriate to use both frequency and time to determine whether the signal is present in a particular voxel.

Program 3dWavelets Batch Command File for Example 4

```
3dWavelets \
-input Jerry+orig \
-nfirst 2 \
-nlast 513 \
-type Haar \
-filt_base  -1  0  600 \
-filt_base   0  0  600 \
-filt_sgnl   1  0  600 \
-filt_sgnl   2  0  300 \
-filt_sgnl   3  0  200 \
```

```
-filt_sgnl   4 25  100 \
-fdisp 5.0 \
-fout -rout -cout \
-bucket Jerry.bucket \
-coefts Jerry.coefts \
-fitts Jerry.fitts \
-errts Jerry.errts
```

■

The format for the output bucket dataset Jerry.bucket is illustrated below.

Brick #	Label		Model	Freq. Band	Time Interval
0	B(-1)[2,513]		Baseline	-1	$2 \le t \le 513$
1	B(0)[2,513]		Baseline	0	$2 \le t \le 513$
2	S(1)[2,257]		Signal	1	$2 \le t \le 257$
3	S(1)[258,513]		Signal	1	$258 \le t \le 513$
4	S(2)[2,129]		Signal	2	$2 \le t \le 129$
5	S(2)[130,257]		Signal	2	$130 \le t \le 257$
6	S(3)[2, 65]		Signal	3	$2 \le t \le 65$
7	S(3)[66,129]		Signal	3	$66 \le t \le 129$
8	S(3)[130,193]		Signal	3	$130 \le t \le 193$
9	S(4)[34, 65]		Signal	4	$34 \le t \le 65$
10	S(4)[66, 97]		Signal	4	$66 \le t \le 97$
11	Full R^2		Coefficient of multiple determination R^2		
12	Full F-stat		F-statistic for significance of the full model		

From the above, we see that the set of time-frequency window indices for the baseline model are the same as in the previous example:

$$\mathbf{B} = \{(-1, 0), (0, 0)\}$$

However, the set of time-frequency window indices for the signal model is different:

$$\mathbf{S} = \{(1, 0), (1, 1), (2, 0), (2, 1), (3, 0), (3, 1), (3, 2), (4, 1), (4, 2)\}$$

Note that the signal model now includes coefficients from a higher frequency band than in the previous example. However, due to the time-windowing, the total number of wavelet coefficients that are included in the signal model has actually been reduced from 14 to 9.

2 Program plug_wavelets

2.1 Purpose

Program plug_wavelets is an *AFNI* "plug-in" which can display either the fitted output waveform (on top of the actual time series data), or the residuals from fitting the output waveform, or the forward wavelet transform coefficients, for voxels of interest. Program plug_wavelets is the interactive version of the batch command program 3dWavelets. The reader is strongly advised to consult the documentation for program 3dWavelets first.

2.2 Usage

To use plug_wavelets, first one must be running afni. Display the Image and Graph for Axial, Sagittal, or Coronal views, for the measured FMRI 3d+time dataset. Choose Define Datamode. This will popup the datamode menu. From the last line of the menu, choose Plugins. This presents a menu of the different *AFNI* plugins that are available. Choose Wavelets.

This opens the Wavelets popup control box. At the top are four control buttons: Quit, to close the popup without using the plugin; Run + Keep, to run the plugin and keep the popup window open; Run + Close, to run the plugin and close the popup window; and Help, to popup a help window. Below this, there is the Control option line, followed by twenty Filter option lines.

On the Control option line, there are three option choosers: Wavelet, NFirst, and NLast. The Wavelet option lets the user specify the type of wavelet to be used in the analysis. At this time, there are only two choices: Haar or Daubechies wavelets. The NFirst box allows the user to specify the initial time series data point to include when performing the wavelet analysis. The third number chooser on the Control option line is labeled NLast. This option allows the user to specify the number of the last image to use in the wavelet analysis. If NLast is set to a number greater than the number of the last image, then NLast is reset by the program (internally) to be the last image number. Note that, whatever the values for NFirst and NLast, the wavelet analysis is applied only to the largest power of 2 points contained within the range of NFirst to NLast. For example, if NFirst $= 5$ and NLast $=$ 300, then NLast - NFirst $+ 1 = 296$. The largest power of 2 which is not greater than 296 is 256 ($= 2^8$). Hence, the *effective* NLast is 5+256-1=260.

Below the Control option line are multiple Filter option lines. As the name implies, the Filter option line allows the user to select the filter to be applied to the time series data. There are four options on each Filter option line. The first option, Type, gives the user 3 choices for the type of filtering: Stop, Base, or Signal. If the user chooses Stop, then the wavelet coefficients that fall within the specified time-frequency window will be set to zero. If the user chooses Base, then the wavelet coefficients that fall within the specified time-frequency window will be used to define the baseline model. If the user chooses Signal, then the wavelet coefficients that fall within the specified time-frequency window will be used to define the signal model. The second option, Band, lets the user specify the frequency band. The lowest frequency band, -1, refers to the constant offset or average value of the time series. The highest frequency band is $n-1$, where $2^n =$ the number of time series points (actually used in the analysis). The third and fourth options, Min TR and Max TR, allow the user to specify the minimum and maximum times, in units of TR, for the time window.

2.3 Examples

Example 4 (continued).

We will assume that the current subdirectory contains the *AFNI* 3d+time dataset of interest, which we will take to be Jerry+orig, plus the bucket dataset file Jerry.bucket+orig. To start the program, type afni. First, from the main menu, click on Switch Anatomy. From the Anatomy submenu, choose Jerry. Then, from the main menu, click on Axial (or Sagittal, or Coronal) Image. Once the image is displayed, it can be resized or moved to another (more convenient) location. Next, click on Switch Function. From the pop-up menu, choose Jerry.bucket [fbuc], which contains the *AFNI* statistical parametric maps which were generated by program 3dWavelets. Click on Set to select this option. Next, click on Define Function. For the Func sub-brick choose the sub-brick labeled S(2)[130,257] (i.e., the wavelet coefficient for: Signal model, frequency band 2, time interval $130 \leq t \leq 257$), and for the Thr sub-brick, choose the Full F-stat (F-statistic for significance of the full model). Now, click on See Function. In the axial (or sagittal, or coronal) image view, those voxels whose fit to the data for the full model is significant at the specified probability threshold will light up. You can adjust the F-statistic probability threshold using the vertical bar. The color coding for those voxels which light up indicates the sign and magnitude of the selected wavelet coefficient. (Note that any parameter subbrick can be selected as the Func subbrick for color coding, and any statistical subbrick can be selected as the Thr subbrick for thresholding).

To view the observed time series at a particular location, use the mouse to change the placement of the crosshairs within the image. To see the corresponding time series for voxels indicated by the crosshairs, click on Axial (or Sagittal, or Coronal) Graph. The time series plots for a 3×3 grid of voxels pops up (see Figure 2). The time series can be vertically rescaled by using the '+', '-', and 'a' keys.

To initialize the wavelet analysis plugin, first click on Define Datamode. From the popup box, click on Plugins. This pops up a list of the different plugin programs that are available. From this list, choose Wavelets. This pops up the Wavelet Analysis control box.

To duplicate the results from the batch program 3dWavelets, we will use 6 time-frequency windows. So, press the button next to the Filter option on six separate lines. Set the Band numbers on the consecutive lines to be: -1, 0, 1, 2, 3, and 4. For the first two filter windows (bands -1 and 0), set Type to Baseline (since these time-frequency windows will be used to define the baseline model), and for the last four filter windows (bands 1, 2, 3, and 4), set Type to Signal (since these time-frequency windows will be used to define the signal model).

For each Filter option, set the Min TR and Max TR inputs to the corresponding values of the time intervals used in Example 4.

Now, press Run + Keep. The program will then write to the text window the following information:

Program:	plug_wavelets
Author:	B. Douglas Ward
Date:	22 March 2000

Controls:

Wavelet	=	Haar
NFirst	=	2
NLast	=	513

Baseline Filter:	Band =	-1	Min. TR =	0	Max. TR =	600
Baseline Filter:	Band =	0	Min. TR =	0	Max. TR =	600
Signal Filter:	Band =	1	Min. TR =	0	Max. TR =	600
Signal Filter:	Band =	2	Min. TR =	0	Max. TR =	300
Signal Filter:	Band =	3	Min. TR =	0	Max. TR =	200
Signal Filter:	Band =	4	Min. TR =	25	Max. TR =	100

■

Check to make sure that the option choices listed agree with the choices that you intended to make.

To overlay a plot of the wavelet analysis estimate of the signal + baseline time series on top of the observed time series, do the following: Click on Opt, and select Double Plot. Then, click on Opt again, and select Tran 1D, and select WA_Fit. For each voxel whose time series is displayed, the program writes relevant statistical information into the text window.

To view the signal model itself, i.e., the fitted time series without the baseline, first click on Opt and switch Double Plot off. Then, click on Opt again, and select Tran 1D, and select WA_Sgnl. To view the residual errors, select Tran 1D, and select WA_Err. To view the forward wavelet transform coefficients, select Tran 1D, and select WA_FWT.

To view the full model fit statistics for a particular voxel, move the cursor into the box containing the time series for that voxel, and press the right-most button on the mouse (i.e., Button 3). This pops-up a display window with the information corresponding to that voxel.

3 References

[1] C. K. Chui, *An Introduction to Wavelets*, Academic Press, San Diego. (1992).

[2] C. K. Chui, *Wavelets: A Mathematical Tool for Signal Analysis*, SIAM, Philadelphia. (1997).

[3] G. Kaiser, *A Friendly Guide to Wavelets,* Birkhauser, Boston. (1994).

[4] J. Neter, W. Wasserman, M. H. Kutner, *Applied Linear Statistical Models*, 2nd edition. Homewood, Illinois: Irwin (1985).

[5] Y. Nievergelt, *Wavelets Made Easy,* Birkhauser, Boston. (1999).