MR/Finance
Multiresolution Analysis of Time Series

Documentation

J-L. Starck and F. Murtagh

September 2001

Contents

1   Introduction

The wavelet transform (WT) has been proposed for time series analysis in many papers over the last few years. In [3, 9] wavelet transform analysis was shown to be important for financial time series prediction in the case where the market can be modeled by fractional Brownian motion (fBm), a $1/f$ fractal process, which implies the presence of correlations across time. Wavelets are important also for the analysis of non-stationary signals [15], and a link between wavelets and a difference operator has been made in [16].

Several approaches have been proposed for time series filtering and prediction by the WT, based on a neural network [17, 2], Kalman filtering [4, 8], or an AR (autoregression) model [10]. In [17, 10] the undecimated Haar transform was used. This choice of the Haar transform was motivated by the fact that the wavelet coefficients are calculated only from data obtained previously in time, and the choice of an undecimated wavelet transform avoids aliasing problems.

Section 2 presents the à trous Haar wavelet transform, and section 3 shows how this transform is well suited for the design of a multiresolution AR model (MAR).

2   Wavelets and Prediction

2.1   Introduction

The continuous wavelet transform of a continuous function produces a continuum of scales as output. However input data are usually discretely sampled, and furthermore a "dyadic" or two-fold relationship between resolution scales is both practical and adequate. The latter two issues lead to the discrete wavelet transform.

The output of a discrete wavelet transform can take various forms. Traditionally, a triangle (or pyramid in the case of 2-dimensional images) is often used to represent all that is worth considering in the sequence of resolution scales. Such a triangle comes about as a result of "decimation" or the retaining of one sample out of every two. The major advantage of decimation is that just enough information is kept to allow exact reconstruction of the input data. Therefore decimation is ideal for an application such as compression. It can be easily shown too that the storage required for the wavelet transformed data is exactly the same as is required by the input data. The computation time for many wavelet transform methods is also linear in the size of the input data, i.e. O(n) for an n-length input time series.

A major disadvantage of the decimated form of output is that we cannot simply – visually or graphically – relate information at a given time point at the different scales. With somewhat greater difficulty, however, this goal is possible. What is not possible is to have shift invariance. This means that if we had deleted the first few values of our input time series, then the output wavelet transformed, decimated, data would not be the same as heretofore. We can get around this problem at the expense of a greater storage requirement, by means of a redundant or non-decimated wavelet transform.

A redundant transform based on an n-length input time series, then, has an n-length resolution scale for each of the resolution levels that we consider. It is easy, under these circumstances, to relate information at each resolution scale for the same time point. We do have shift invariance. Finally, the extra storage requirement is by no means excessive.

The redundant, discrete wavelet transform described in the next section is one used in Aussem et al. [1]. The successive resolution levels are formed by convolving with an increasingly dilated wavelet function which looks rather like a Mexican sombrero (central bump,

symmetric, two negative side lobes). Alternatively these resolution levels may be constructed by (i) smoothing with an increasingly dilated scaling function looking rather like a Gaussian function defined on a fixed interval (support) – the function used is in fact a B_3 spline; and (ii) taking the difference between successive versions of the data which are smoothed in this way.

2.2 The à trous wavelet transform

Our input data is decomposed into a set of band-pass filtered components, the wavelet coefficients, plus a low-pass filtered version of our data, the continuum (or background or residual).

We consider a signal or time series, $\{c_{0,l}\}$, defined as the scalar product at samples l of the function $f(x)$ – our input data – with a scaling function $\phi(x)$ which corresponds to a low-pass filter:

$$c_0(k) = <f(x), \phi(x-k)> \tag{1}$$

The scaling function is chosen to satisfy the dilation equation:

$$\frac{1}{2}\phi\left(\frac{x}{2}\right) = \sum_k h(k)\phi(x-k) \tag{2}$$

where h is a discrete low-pass filter associated with the scaling function. This means that a low-pass filtering of the signal is, by definition, closely linked to another resolution level of the signal. The distance between levels increases by a factor 2 from one scale to the next.

The smoothed data $\{c_{j,l}\}$ at a given resolution j and at a position k is the scalar product

$$c_{j,l} = \frac{1}{2^j} < f(x), \phi(\frac{x-k}{2^j}) > \tag{3}$$

This is consequently obtained by the convolution:

$$c_{j+1,l} = \sum_k h(k) \; c_{j,l+2^j k} \tag{4}$$

The signal difference between two consecutive resolutions is:

$$w_{j+1,l} = c_{j,l} - c_{j+1,l} \tag{5}$$

which we can also, independently, express as:

$$w_{j,l} = \frac{1}{2^j} < f(x), \psi(\frac{x-k}{2^j}) > \tag{6}$$

Here, the wavelet function is defined by:

$$\frac{1}{2}\psi(\frac{x}{2}) = \phi(x) - \frac{1}{2}\phi(\frac{x}{2}) \tag{7}$$

Equation 6 defines the discrete wavelet transform, for a resolution level j.

A series expansion of the original signal, , in terms of the wavelet coefficients is now given as follows. The final smoothed signal is added to all the differences :

$$c_{0,l} = c_{J,l} + \sum_{j=1}^{J} w_{j,l} \tag{8}$$

4

This equation provides a reconstruction formula for the original signal. At each scale j, we obtain a set which we call a wavelet scale. The wavelet scale has the same number of samples as the signal, i.e. it is redundant, and decimation is not used.

In view of the gaps manifest in equation 2, a good choice for h is a spline which leads to $h = (\frac{1}{16}, \frac{1}{4}, \frac{3}{8}, \frac{1}{4}, \frac{1}{16})$. The associated wavelet function, ψ, is similar to a Mexican hat function, i.e. central peak and negative side lobes.

Equation 4 also is relevant for the name of this transform ("with holes"). Unlike widely used non-redundant wavelet transforms, it retains the same computational requirement (linear, as a function of the number of input values). Redundancy (i.e. each scale having the same number of samples as the original signal) is helpful for detecting fine features in the detail signals since no aliasing biases arise through decimation.

Our application – prediction – points to the critical importance for us of the final values. Our time series is finite, and values n, n-1, n-2, ..., are of greatest interest for us. Any symmetric wavelet function is problematic for the handling of such a boundary (or edge). We also cannot use wavelet coefficients if these coefficients have been calculated from "future" data values. An asymmetric filter can allow us to get around this problem. Such a wavelet function can deal properly with the edge of importance to us. The first values of our time series, which also constitute a boundary, may be artibrarily treated as a result, but this is of no practical consequence.

2.3 The time-based à trous wavelet transform

The à trous wavelet transform with a wavelet function related to a spline function, as described above, is not consistent with a directed (time-varying) data stream. We now keep the wavelet function, but alter the wavelet transform, to make of it a multiscale transform which is appropriate for a data stream. We consider a signal s(1), s(2),..., s(n), where n is the present time-point.

1. For index k sufficiently large, carry out an à trous wavelet transform on $\{s(1), s(2), ..., s(k)\}$.

2. Retain the detail coefficient values, and the continuum value, for the kth time-point only (cf. equation 8): $w_{1,k}, w_{2,k}, w_{J,k}, c_{J,k}$. Note that summing these values gives $s(k)$.

3. If k is less than n, set k to k+1 and return to Step 1.

This produces an additive decomposition of the signal, which is similar to the à trous wavelet transform decomposition with the B_3 spline on $\{s(1), s(2), ..., s(k), ..., s(n)\}$. The computational time is evidently greater, $O(n^2)$ as against $O(n)$.

We have not touched on an important matter in regard to equation 2: how to handle signal boundaries. Although other strategies could be envisaged, we use a mirror approach. This is tantamount, of course, to redefining the discrete filter associated with the scaling function in the signal boundary region; and to redefining the associated wavelet function in this region. This strategy is of particular relevance when we work on an ordered data stream. We hypothesize future data based on values in the immediate past. Not surprisingly there is discrepancy in fit in the succession of scales, which grows with scale as larger numbers of immediately past values are taken into account.

2.4 The redundant Haar wavelet transform

The Haar wavelet transform was first described in the early years of this century and is described in almost every text on the wavelet transform. As already mentioned, the asymmetry
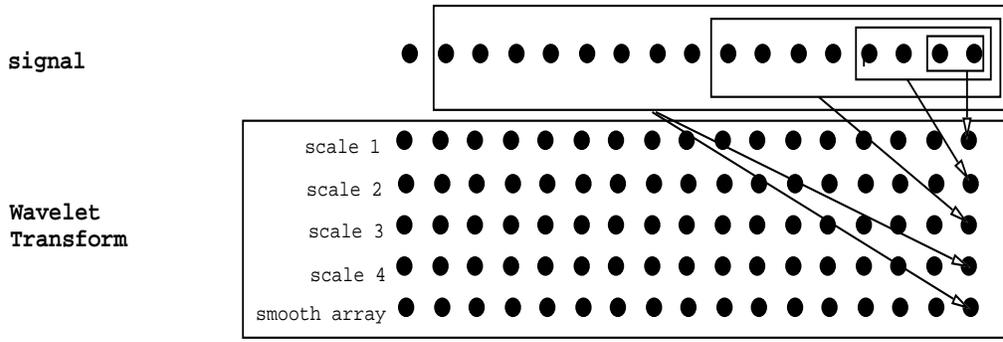
Figure 1: This figure shows which pixels of the input signal are used to calculate the last wavelet coefficient in the different scales.

of the wavelet function used makes it a good choice for edge detection, i.e. localized jumps. The usual Haar wavelet transform, however, is a decimated one. We now develop a non-decimated or redundant version of this transform. This will be an à trous algorithm, but with a different pair of scaling and wavelet functions compared to those used previously.

The non-decimated Haar algorithm is exactly the same as the à trous algorithm, except that the low-pass filter h, $(\frac{1}{16}, \frac{1}{4}, \frac{3}{8}, \frac{1}{4}, \frac{1}{16})$, is replaced by the simpler filter $(\frac{1}{2}, \frac{1}{2})$. There h is now non-symmetric. Consider the creation of the first wavelet resolution level. We have created it from by convolving the original signal with h. Then:

$$c_{j+1,l} = 0.5(c_{j,l-2^j} + c_{j,l}) \tag{9}$$

and

$$w_{j+1,l} = c_{j,l} - c_{j+1,l} \tag{10}$$

At any time point, l, we never use information after l in calculating the wavelet coefficient. Figure 1 shows which pixels of the input signal are used to calculate the last wavelet coefficient in the different scales. A wavelet coefficient at a position t is calculated from the signal samples at positions less than or equal to t, but never larger.

3 Autoregressive Multiscale Prediction

3.1 Stationary signal

Assuming a stationary signal $D = (d_1, \ldots, d_N)$, the AR (autoregressive) multiscale prediction model is:

$$d_{t+1} = \sum_{j=1}^{J} \sum_{k=1}^{A_j} a_{j,k} w_{j,t-2^j(k-1)} + \sum_{k=1}^{A_{J+1}} a_{J+1,k} c_{J,t-2^J(k-1)} + \epsilon(t+1) \tag{11}$$

where $\mathcal{W} = w_1, \ldots, w_J, c_J$ represents the Haar à trous wavelet transform of D ($D = \sum_{j=1}^{J} w_j + c_J$). For example, choosing $A_j = 1$ for all resolution levels j leads to the equation

$$d_{t+1} = \sum_{j=1}^{J} a_j w_{j,t} + a_{J+1} c_{J,t} + \epsilon(t+1) \tag{12}$$
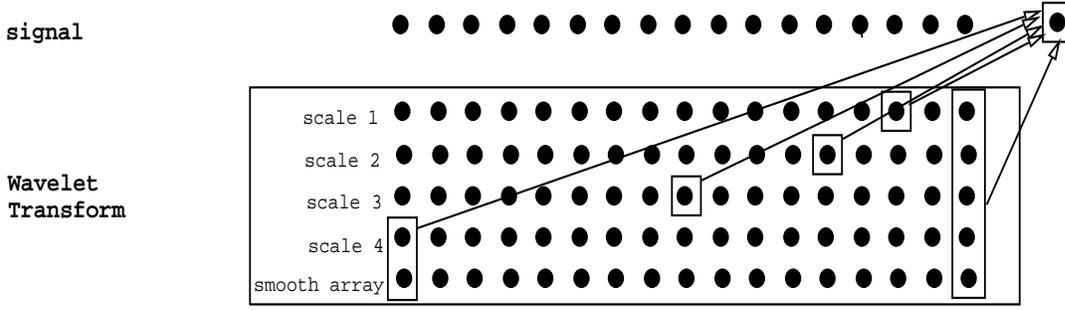
6

Figure 2: Wavelet coefficients which are used for the prediction of the next value.

Figure 2 shows which wavelet coefficients are used for the prediction using $A_j = 2$ for all resolution levels j, and a wavelet transform with five scales (four wavelet scales + the smoothed array). In this case, we can see that only ten coefficients are used, but taking into account low resolution information. This means that a long term prediction can easily be introduced, either by increasing the number of scales in the wavelet transform, or by increasing the AR order in the last scales, but with a very small additional number of parameters. To find the $Q = \sum_{j=1}^{J+1} A_j$ unknown parameters:

$$\{\{a_{1,1}, ..., a_{1,A_1}\}, ..., \{a_{j,1}, ..., a_{j,A_j}\}, ..., \{a_{J,1}, ..., a_{j,A_J}\}, \{a_{J+1,1}, ..., a_{j,A_J+1}\}\}$$

of our model, we need to resolve the following equation: $AX = S$, where A, X, W are defined by:

$$
\begin{aligned}
A^t &= (L_{N-1}, \ldots, L_{N-P}) \\
L_i &= (w_{1,i}, \ldots, w_{1,i-2A_1}, \ldots, w_{2,i}, \ldots, w_{2,i-2^2 A_2}, \ldots, w_{J,i}, \ldots, w_{J,i-2^J A_J}, c_{J,i}, \ldots, c_{J,i-2^J A_{J+1}}) \\
X^t &= (a_{1,1}, \ldots, a_{1,A_1}, a_{2,1}, \ldots, a_{2,A_2}, \ldots, a_{J,1}, \ldots, a_{J,A_j}, \ldots, a_{J+1,1}, \ldots, a_{J+1,A_{J+1}}) \\
S^t &= (d_N, \ldots, d_{i+1}, \ldots, d_{N-P+1})
\end{aligned}
$$

We have P equations and Q unknowns, so A is a $Q \times P$ matrix (P rows L_i, each with Q elements), X and S are respectively Q- and P-sized vectors. When Q is larger than P, many minimization methods may be used to find X. In our experiments, we used the standard SVD method.

3.2 Non-stationary signal

When the signal is not stationary, the previous method will not correctly model our data. However, in many cases, the non-stationary part affects the low frequency components, while the high frequencies may still give rise to stationary behavior. Then we can separate our signal D into two parts, the low and the high frequencies L and H:

$$
\begin{aligned}
L &= c_J \\
H &= D - L = \sum_{j=1}^{J} w_j \\
d_{t+1} &= l_{t+1} + h_{t+1}
\end{aligned}
$$

7

The smoothed array of the wavelet transform is first subtracted from the data, and we consider now that the signal H is stationary. Our prediction will be the coaddition of two predicted values, one on the signal H by the AR Multiscale model, and the second on the low frequency component by some other method. The AR-Multiscale model gives:

$$h_{t+1} = \sum_{j=1}^{J} \sum_{k=1}^{A_j} a_{j,k} w_{j,t-2^j(k-1)} + \epsilon(t+1) \tag{13}$$

We have now $Q = \sum_{j=1}^{J} A_j$ unknown parameters:

$$\{\{a_{1,1}, ..., a_{1,A_1}\}, ..., \{a_{j,1}, ..., a_{j,A_j}\}, ..., \{a_{J,1}, ..., a_{J,A_J}\}\}$$

and we need to resolve the following equation: $AX = S$, where A, X, W are defined by:

$$
\begin{aligned}
A^t &= (L_{N-1}, \ldots, L_{N-P}) \\
L_i &= (w_{1,i}, \ldots, w_{1,i-2A_1}, \ldots, w_{2,i}, \ldots, w_{2,i-2^2 A_2}, \ldots, w_{J,i}, \ldots, w_{J,i-2^J A_J}) \\
X^t &= (a_{1,1}, \ldots, a_{1,A_1}, a_{2,1}, \ldots, a_{2,A_2}, \ldots, a_{J,1}, \ldots, a_{J,A_J}) \\
S^t &= (h_N, \ldots, h_{i+1}, \ldots, h_{N-P+1})
\end{aligned}
$$

Many methods may be used for the prediction of l_{t+1}. The problem is simplified by the fact that L is very smooth. We used a polynomial fitting of degree 3 in our experiments.

3.3 AR order determination

The AR order at the different scales must now be defined. It can be a user parameter, but an automatic method is generally preferable. At each scale j, we need to know how many coefficients should be used. This value A_j may be determined by looking at how the wavelet coefficients at the scale j are correlated. Therefore each scale is first analyzed separately, and the best AR order A_j at scale j minimizes:

$$J(A_j) = \log \sigma_{A_j}^2 + \mathcal{P}(A_j)$$

where σ_{A_j} is the prediction error, and $\mathcal{P}$ is a penalty function, which increases with the AR order. Examples of penalty functions are:

- AIC: $AIC = \log \sigma_{A_j}^2 + \frac{2A_j}{N}$

- AICC: $AICC = \log \sigma_{A_j}^2 + \frac{N+A_j}{N-A_j-2}$

- SIC: $SIC = \log \sigma_{A_j}^2 + \frac{A_j \log N}{N}$

4 Smoothing Time Series by the Wavelet Transform

4.1 Statistical significance test

Signals generally contain noise. Hence the wavelet coefficients are noisy too. For filtering, it is necessary to know if a coefficient is due to signal (i.e. it is significant) or to noise. We introduce a statistical significance test for wavelet coefficients. Let $\mathcal{H}_0$ be the hypothesis that

8

$$P = Prob(W_N > w_j(x, y))$$

and if the coefficient value is negative

$$P = Prob(W_N < w_j(x, y))$$

Given a threshold, ϵ, if $P > \epsilon$ the null hypothesis is not excluded. Although non-null, the value of the coefficient could be due to noise. On the other hand, if $P < \epsilon$, the coefficient value cannot be due only to the noise alone, and so the null hypothesis is rejected. In this case, a significant coefficient has been detected.

Our noise modeling in the wavelet space is based on the assumption that the noise in the data follows a distribution law, which can be:

- a Gaussian distribution

- a Poisson distribution

- a Poisson + Gaussian distribution (e.g., noise in CCD detectors)

- Poisson noise with few events (e.g., low-valued arrival counts)

- Speckle noise

- Root Mean Square map: we have a noise standard deviation of each data value.

If the noise does not follow any of these distributions, we can derive a noise model from any of the following assumptions:

- it is stationary, and we have a subimage containing a realization of the noise,

- it is additive, and non-stationary,

- it is multiplicative and stationary,

- it is multiplicative, but non-stationary,

- it is undefined but stationary,

- it is additive, stationary, and correlated.

4.2 Noise modeling

We summarize here the different noise modeling strategies implemented. The term "pixel" can be replaced by "time step" or "time interval" in the case of a signal representing a time series.

1. Gaussian noise
   Given stationary Gaussian noise, it suffices to compare $w_j(x, y)$ to $k\sigma_j$.

$$\begin{aligned} \text{if } \mid w_j \mid \ \geq \ k\sigma_j \quad &\text{then } w_j \text{ is significant} \\ \text{if } \mid w_j \mid \ < \ k\sigma_j \quad &\text{then } w_j \text{ is not significant} \end{aligned} \qquad (14)$$

9

2. Poisson noise
   If the noise in the data I is Poisson, the transform

$$t(I(x,y)) = 2\sqrt{I(x,y) + \frac{3}{8}} \qquad (15)$$

acts as if the data arose from a Gaussian white noise model (Anscombe, 1948), with $\sigma = 1$, under the assumption that the mean value of I is large. The signal is first transformed, and the same processing is performed as in the Gaussian case. This processing works if the number of events per pixel is greater than 30. Otherwise the detection levels will be over-estimated, and the case "Poisson noise with few events" should instead be used.

3. Poisson noise + Gaussian
   The generalization of the variance stabilizing is:

$$t(I(x,y)) = \frac{2}{\alpha}\sqrt{\alpha I(x,y) + \frac{3}{8}\alpha^2 + \sigma^2 - \alpha g}$$

where α is the gain of the detector, and g and σ are the mean and the standard deviation of the read-out noise.

4. Multiplicative noise
   The signal is first log-transformed. Then the transformed signal is treated as a signal with Gaussian additive noise.

5. Non-stationary additive noise
   The noise is assumed to be locally Gaussian. So we must consider one noise standard deviation per pixel. The Root Mean Square (RMS) map $R_\sigma(x,y)$ can be furnished by the user, or automatically calculated by estimating for each pixel the standard deviation in a box around it.

   From $R_\sigma(x,y)$, we have to compute the noise standard deviation $\sigma_j(x,y)$ for any wavelet coefficient $w_j(x,y)$. $w_j(x,y)$ is obtained by the correlation product between the signal I and a function g_j: $w_j(x,y) = \sum_k \sum_l I(x,y)g_j(x+k, y+l)$.

   Then we have: $\sigma_j^2(x,y) = \sum_k \sum_l R_\sigma^2(x,y)g_j^2(x+k, y+l)$.
   In the case of the à trous algorithm, the coefficients $g_j(x,y)$ are not known exactly, but they can easily be computed by taking the wavelet transform of a Dirac w^δ. The map σ_j^2 is calculated by correlating the square of the wavelet scale j of w^δ by $R_\sigma^2(x,y)$.

6. Non-stationary multiplicative noise
   The signal is first log-transformed. Then the transformed signal is treated as an signal with non-stationary additive noise.

7. Undefined stationary noise
   A k-sigma clipping is applied at each scale.

8. Undefined noise
   The standard deviation is estimated for each wavelet coefficient, by considering a box around it, and the calculation of σ is done in the same way as for non-stationary additive noise. The latter determines a map of variances for the signal, and then derives the variances for the wavelet coefficients. "Undefined noise" does not assume additivity of the noise, and so calculates the noise from local variance in the resolution scales.

9. Stationary correlated noise

   The noise is stationary, but correlated. This noise modeling requires a noise map, containing a realization of the noise. The threshold at a scale j S_j is found by computing the wavelet transform of the noise map, and using the histogram of S_j to derive the noise probability density function, PDF, of S_j.

10. Poisson noise with few events

    This case corresponds to noise with a very small number of events per pixel. This special case requires more processing time, due to the fact that a set of autoconvolutions of the histogram of the wavelet function must be calculated.

4.3 Filtering Methods

Hard and soft thresholding

Many filtering methods have been proposed in the last ten years. *Hard thresholding* consists of setting to 0 all wavelet coefficients which have an absolute value lower than a threshold T_j:

$$\tilde{w}_{j,k} = \begin{cases} w_{j,k} & \text{if } \mid w_j \mid \geq T_j \\ 0 & \text{otherwise} \end{cases}$$

where $w_{j,k}$ is a wavelet coefficient at scale j and at spatial position k.

Soft thresholding consists of replacing each wavelet coefficient by the value $\tilde{w}$ where

$$\tilde{w}_{j,k} = \begin{cases} sgn(w_{j,k})(\mid w_{j,k} \mid -T_j) & \text{if } \mid w_j \mid \geq T_j \\ 0 & \text{otherwise} \end{cases}$$

When the discrete orthogonal wavelet transform is used, it is interesting to note that the hard and soft thresholded estimators are solutions of the following minimization problems:

$$\tilde{w} = \arg_w \min \frac{1}{2} \parallel y - \mathcal{W}^{-1}w \parallel_{l^2}^2 + \lambda \parallel w \parallel_{l^0}^2 \qquad \textbf{hard threshold}$$

$$\tilde{w} = \arg_w \min \frac{1}{2} \parallel y - \mathcal{W}^{-1}w \parallel_{l^2}^2 + \lambda \parallel w \parallel_{l^2}^2 \qquad \textbf{soft threshold}$$

where y is the input data, $\mathcal{W}$ the wavelet transform operator, and l^0 indicates the limit of l^δ when $\delta \to 0$. This counts in fact the number of non-zero elements in the sequence.

Several approaches have been proposed for deriving the T_j thresholds.

k-Sigma thresholding

The k-Sigma approach consists of deriving T_j from the probability of false detection ϵ

$$Prob(w > T_j) < \epsilon$$

Given stationary Gaussian noise, it suffices to compare $w_{j,k}$ to $k\sigma_j$, where σ_j is the noise standard deviation in band j. Often k is chosen as 3, which corresponds approximately to $\epsilon = 0.002$.

11

Iterative filtering

When a redundant wavelet transform is used, the result after a simple hard thresholding can still be improved by iterating. Indeed, we want the wavelet transform of our solution s to reproduce the same significant wavelet coefficients (i.e., coefficients larger than T_j). This can be expressed in the following way:

$$(\mathcal{W}s)_{j,k} = w_{j,k} \text{ if } \mid w_{j,k} \mid > k\sigma_j \tag{16}$$

where $w_{j,k}$ are the wavelet coefficients of the input data y. Denoting M the multiresolution support (i.e. $M(j,k) = 1$ if $\mid w_{j,k} \mid > k\sigma_j$, and 0 otherwise), we want:

$$M.\mathcal{W}s = M.\mathcal{W}y$$

The solution can be obtained by the following Van Cittert iteration [13]:

$$
\begin{aligned}
s^{n+1} &= s^n + \mathcal{W}^{-1}(M.\mathcal{W}y - M.\mathcal{W}s^n) \\
&= s^n + \mathcal{W}^{-1}(M.\mathcal{W}R^n)
\end{aligned}
\tag{17}
$$

where $R^n = y - s^n$. Another approach consists of minimizing the functional

$$J(s) = \parallel M.\mathcal{W}y - M.\mathcal{W}s \parallel^2 \tag{18}$$

using a minization method such as the fixed step gradient or the conjugate gradient.

Universal threshold

Universal thresholding consists of using a threshold [7, 6] $T_j = \sqrt{2\log(n)}\sigma_j$, where n is the number of pixels in the input data. It ensures with a probability tending to one that all noise is removed. The drawback is that it tends to give an oversmoothed estimator.

SURE threshold

The SURE threshold minimizes Stein's unbiased risk estimator [5].

MULTI-SURE thresholding

The SURE method is applied independently on each band of the wavelet transform.

MAD thresholding

The Median Absolute Deviation (MAD) threshold on a given band j is:

$$T_j = k\sigma_{j,m}$$

where $\sigma_{j,m}$ is the Median Absolute Deviation ($\sigma_{j,m} = \text{MED}(\mid w_j \mid)/0.6745$, where MED is the median function). The MAD method does not requires any knowledge about the noise such as the noise standard deviation. Is is considered as a very good method to denoise data contaminated by correlated noise.

Multiscale entropy filtering

The multiscale entropy filtering method [14, 12] (MEF) consists of measuring the information h relative to wavelet coefficients, and of separating this into two parts h_s, and h_n. The expression h_s is called the signal information and represents the part of h which is certainly not contaminated by the noise. The expression h_n is called the noise information and represents the part of h which may be contaminated by the noise. We have $h = h_s + h_n$. Following this notation, the corrected coefficient $\tilde{w}$ should minimize:

$$J(\tilde{w}_j) = h_s(w_j - \tilde{w}_j) + \alpha h_n(\tilde{w}_j) \tag{19}$$

i.e. there is a minimum of information in the residual $(w - \tilde{w})$ which can be due to the significant signal, and a minimum of information which could be due to the noise in the solution $\tilde{w}_j$.

In order to verify a number of properties, the following functions have been proposed for h_s and h_n in the case of Gaussian noise [14]:

$$
\begin{aligned}
h_s(w_j) &= \frac{1}{\sigma_j^2} \int_0^{|w_j|} u \operatorname{erf}\left(\frac{|w_j| - u}{\sqrt{2}\sigma_j}\right) du \\
h_n(w_j) &= \frac{1}{\sigma_j^2} \int_0^{|w_j|} u \operatorname{erfc}\left(\frac{|w_j| - u}{\sqrt{2}\sigma_j}\right) du
\end{aligned}
\tag{20}
$$

Simulations have shown [11] that the MEF method produces a better result than the standard soft or hard thresholding, from both the visual aspect and PSNR (peak signal-to-noise ratio). Figures 3 and 4 show the filtering respectively on simulated noisy blocks and on a real spectrum.

5 Programs

5.1 Conversion: im1d_convert

Program *im1d_convert* converts a 1D signal from one data format to another. Currently supported signal formats are the ASCII format, the FITS format, and Excel format. Suffixes for these formats are respectively ".dat", ".fits", and ".csv"". The "-s" option allows the user to suppress a given number of lines at the beginning of the file. This option has an effect only for ASCII and Excel input formats. The ASCII format consists of a series of numbers separated by a space, a tab, or a new line.

USAGE: im1d_convert [-s] file_name_in file_name_out

Examples:

- im1d_convert sig.dat image.fits
  Converts an ASCII file to FITS format.

- im1d_convert -s 2 sig.dat image.fits
  Ditto, but the first lines are not taken into account.

5.2 Statistical information: im1d_info

im1d_info gives information about a signal:

- the number of pixels

- the minimum, the maximum

- the arithmetic mean: $\bar{x} = \frac{1}{N} \sum_k x_k$

- the standard deviation: $\sigma = \frac{1}{N} \sum_k (x_k - \bar{x})^2 = \sqrt{\bar{x^2} - \bar{x}^2}$

- the flux: $F = \sum_k x_k$

- the energy: $E = \sum_k x_k^2$

- the skewness: $S = \frac{1}{N\sigma^3} \sum_k (x_k - \bar{x})^3 = \frac{1}{\sigma^3}(\bar{x^3} - 3\bar{x}\bar{x^2} + 2\bar{x}^3)$
  S is zero if the data are symmetrically distributed around the mean. If a tail extends to the right (resp. left), S is positive (resp. negative).

- the kurtosis: $C = \frac{1}{N\sigma^4} \sum_k (x_k - \bar{x})^4 - 3 = \frac{1}{\sigma^4}(\bar{x^4} - 4\bar{x}\bar{x^3} + 6\bar{x^2}\bar{x}^2 - 3\bar{x}^4) - 3$
  Positive C implies a higher peak and larger wings than the Gaussian distribution with the same mean and variance. Negative C means a wider peak and shorter wings.

- Measure of dependence: calculate the autoregressive model which fits the data, for all orders between 1 and M, and select the AR model which minimizes the following equation:

$$J(p) = \log \sigma^2_{A(p)} + \mathcal{P}(A(p))$$

where σ_A is the prediction error, and $\mathcal{P}$ is a penalty function, which increases with the AR order. Examples of penalty functions are:

  - AIC: $AIC = \log \sigma^2_{A_j} + \frac{2A_j}{N}$
  - AICC: $AICC = \log \sigma^2_{A_j} + \frac{N+A_j}{N-A_j-2}$
  - SIC: $SIC = \log \sigma^2_{A_j} + \frac{A_j \log N}{N}$

When the "-a" option is set, then the autocorrelation function $\rho(h)$ is also calculated:

$$\rho(h) = \frac{\gamma(h)}{\gamma(0)} \tag{21}$$

where $\gamma(h)$ is the autocovariance function defined by:

$$\gamma(h) = \frac{1}{N} \sum_k (x_{k+h} - \bar{x})(x_k - \bar{x}) \tag{22}$$

The command line is:

USAGE: im1d_info file_name_in

where options are:

- **[-a Nbr_of_Lags]**
  Calculate the autocorrelation function (AF) with a given number of lags. The output AF is saved in a file of name "autocor". Default number of lags is 10.

14

- **[-O Estimation_AR_Order_Method]**

  1. AIC
  2. AICC
  3. BIC

  Default is BIC method.

- **[-M MaxAROrder]**
  Maximum AR model order. Default is 10.

Examples:

- im1d_info data.dat
  Gives information about the data.

- im1d_info -a 10 data.dat
  Ditto, but calculates also autocorrelation function with 10 lags.

5.3   Tendency estimation: im1d_tend

If the signal exhibits a tendency, it may be convenient to remove it before starting the analysis. A well-known fitting method is the polynomial one. A polynomial of order p is fitted around each pixel in a window of size T (p equals 2 in this program). In order to have smooth tendency estimation, it is recommended to weight the pixels with weights from 1 to zero for pixels in the middle to the border of the window.

USAGE: im1d_tend option in_data out_tend out_signal_no_tend

where options are:

- **[-T WindowSize_for_tendency_estimation]**
  Default is 100.

- **[-f FirstPixels]** Default is the input signal size. If this option is set, the tendency is estimated only on the first *FirstPixels* pixels.

Examples:

- im1d_tend sig.dat tend.dat sig_out
  Remove the tendency in the input data with all default options.

- im1d_tend -T 200 sig.dat tend.dat sig_out
  Ditto, but increase the window size.

Figure 5 shows the results when applying the *tend_est* program (with a window size equal to 400) to a time series containing a tendency and an AR(4).

15

5.4   Multiresolution transform: mr1d_trans

Program *mr1d_trans* applies a one-dimensional multiresolution transform to a signal. The ouput file contains an image, in which each row is a band of the multiresolution transform. Morlet, Mexican hat, and French hat wavelet transforms are non-dyadic (the resolution is not decreased by a factor two between two scales), and 12 voices (fixed value) are calculated (instead of one for the dyadic case) when the resolution is divided by two. Then the number of rows in the output image will be equal to $12*number\_of\_scales$. The Morlet wavelet is complex, so the wavelet transform is complex too. Using this transform, the first $12*number\_of\_scales$ lines represent the real part of the transform, and the $12*number\_of\_scales$ last lines the imaginary part. Four transforms are non-redundant: the bi-orthogonal, the lifting scheme, and the wavelet packet methods (transforms 16 and 17). In this case, the output is not an image but a signal (i.e. 1D rather than 2D), which has the same size as the original one. Position and length of a given band in the output signal can be found by reading the file created using the "-w" option. For the lifting scheme based method, the type of lifting can be changed using the "-l" option, and for the (bi-) orthogonal and packet one, the filter can be changed by the "-f" option.

19 transforms are available, which are grouped into 5 classes

- Class 1: no decimation (transforms 1 to 7 and 11 to 14).

- Class 2: pyramidal transform (transforms 8 to 10).

- Class 3: orthogonal transform (15 and 16).

- Class 4: Wavelet packets (17 and 18).

- Class 5: Wavelet packets via the à trous algorithm (19).

Depending on the class, the transform does not contain the same number of pixels, and the data representation differs. By default, the number of scales is calculated from the length of the signal.

USAGE: mr1d_trans option signal_in image_out

where options are:

- **[-t type_of_multiresolution_transform]**

  1. Linear wavelet transform: à trous algorithm
  2. B_1-spline wavelet transform: à trous algorithm
  3. B_3-spline wavelet transform: à trous algorithm
  4. Derivative of a B_3-spline: à trous algorithm
  5. Undecimated Haar wavelet transform: à trous algorithm
  6. Morphological median transform
  7. Undecimated (bi-) orthogonal wavelet transform
  8. Pyramidal linear wavelet transform
  9. Pyramidal B_3-spline wavelet transform
  10. Pyramidal median transform

11. Morlet's wavelet transform
    Continuous wavelet transform with a complex wavelet which can be decomposed into two parts, one for the real part, and the other for the imaginary part:

$$\psi_r(x) = \frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{x^2}{2\sigma^2}}\cos(2\pi\nu_0\frac{x}{\sigma})$$

$$\psi_i(x) = \frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{x^2}{2\sigma^2}}\sin(2\pi\nu_0\frac{x}{\sigma})$$

First scale σ_0 is chosen equal to $2\nu_0$ and $\nu_0 = 0.8$, using 12 voices per octave.

12. Mexican hat wavelet transform
    Continuous wavelet transform with the Mexican hat wavelet. This is the second derivative of a Gaussian

$$\psi(x) = (1 - \frac{x^2}{\sigma^2})e^{-\frac{x^2}{2\sigma^2}} \tag{23}$$

First scale σ_0 is chosen equal to $\frac{1}{\sqrt{3}}$, using 12 voices per octave.

13. French hat wavelet transform
    Continuous wavelet transform with the French hat wavelet

$$\psi(x) = \begin{cases} 0 & \text{if } |\frac{x}{\sigma}| > 1 \\ -1 & \text{if } |\frac{x}{\sigma}| \in [\frac{1}{3}, 1] \\ 2 & \text{if } |\frac{x}{\sigma}| < \frac{1}{3} \end{cases} \tag{24}$$

First scale σ_0 equal to 0.66, and 12 voices per octave.

14. Gaussian derivative wavelet transform
    Continuous wavelet transform. The wavelet is the first derivative of a Gaussian

$$\psi(x) = -xe^{-\frac{1}{2}x^2} \tag{25}$$

First scale σ_0 equal to $\frac{1}{\sqrt{3}}$, and 12 voices per octave.

15. (bi-) orthogonal transform.

16. (bi-) orthogonal transform via lifting scheme.

17. Wavelet packets.

18. Wavelet packets via lifting scheme.

19. Wavelet packets using the à trous algorithm.

- **[-n number_of_scales]**
  Number of scales used in the multiresolution transform.

- **[-r]**
  Rebin all scales to the input signal size (for pyramidal transforms only).

- **[-k]**
  Set to 0 pixels contaminated by the border problem.

- **[-T type_of_filters]**

  1. Antonini 7/9 filters.
  2. Daubechies filter 4.
  3. Biorthogonal 2/6 Haar filters.
  4. Biorthogonal 2/10 Haar filters.

5. Odegard 7/9 filters.

    6. User's filters.

Default is Antonini 7/9 filters.
This option is only available if the chosen transform method is the (bi-) orthogonal
transform (-t option in [7,15,17]).
For option 6, the format used is that of the Bath Wavelet Warehouse: see Example
below.

- **[-L]**
  Use an L_2 normalization. Default is L_1.

- **[-L]**
  Use an L_2 normalization. Default is L_1.

- **[-l type_of_lifting_transform]**

    1. Lifting scheme: CDF WT.

    2. Lifting scheme: median prediction.

    3. Lifting scheme: integer Haar WT.

    4. Lifting scheme: integer CDF WT.

    5. Lifting scheme: integer (4,2) interpolating transform.

    6. Lifting scheme: Antonini 7/9 filters.

    7. Lifting scheme: integer Antonini 7/9 filters.

Default is Lifting scheme: integer Haar WT.
This option is only available if the chosen transform method is the lifing scheme (-t 24).

- **[-w InfoFileName]**
  Write in a file the size and the starting index of each band. This file contains a 2D float
  array (Array[2, NbrBand+3]).

```
info[0,0] = transform number
info[1,0] = number of scales
info[0,1] = transform class number (5 classes)
info[1,1] = number of bands
            it is not equal to the number of scales
            for wavelet packets transform.
info[0,2] = number of pixels
info[1,2] = lifting scheme type
info[0,3] = type of filter
info[1,3] = type of normalization
for i=4 to Number_of_bands + 3
info[0,i] = number of pixels in the band i
info[1,i] = position number of the pixel of the band
```

If a user filter file is given (i.e. -T 6,filename), with a filename of L characters, L lines
are added to the array:

```
        info[1,Number_of_bands + 4] = number of characters of the filter file name
        for i=Number_of_bands+4 to  Number_of_bands+4+L-1
info[0,i] = ascii number of the ith character.
```

Examples:

- mr1d_trans exch.dat tr_out
  Transform input is in file exch.dat, and output is in file tr_out.dat. For the latter, the number of rows equals the lenght of the time series in exch.dat. The number of columns equals the number of resolution scales.

- mr1d_trans -n 7 -t 3 sig.dat outsig.dat
  À trous algorithm with 7 scales.

- mr1d_trans -n 7 -T 5 -t 15 sig.dat outsig.dat
  Bi-orthogonal wavelet transform (Odegard 7/9 filters).

- mr1d_trans -n 7 -T 5 -t 15 -w info.dat sig.dat outsig.dat
  Ditto, but an information file is written. This information is necessary for reconstruction.

- mr1d_trans -n 7 -T 5 -t 15 -w info.dat sig.dat outsig.dat
  Ditto, but an information file is written. This information is necessary for reconstruction.

- mr1d_trans -n 7 -T 6,dau4 -t 15 -w info.dat sig.dat outsig.dat
  Ditto, but the filters defined in the file "dau4.wvf" are used instead of the Odegard filters. The format is that of the Bath Wavelet Warehouse,
  http://dmsun4.bath.ac.uk/wavelets/warehouse.html

5.5   Wavelets and autocorrelation function: mr1d_acor

Program *mr1d_acor* calculates the autocorrelation at each scale of the wavelet transform.

USAGE: mr1d_acor option signal_in autocor_out

where options are:

- **[-n number_of_scales]**
  Number of scales used in the multiresolution transform.

- **[-S Nbr_of_Lags]**
  Default is 10.

5.6   Filtering: mr1d_filter

Program *mr1d_filter* filters a signal using different methods.

USAGE: mr1d_filter option signal_in signal_out

where options are:

- **[-t type_of_multiresolution_transform]**

  1. Linear wavelet transform: à trous algorithm
  2. B_1-spline wavelet transform: à trous algorithm
  3. B_3-spline wavelet transform: à trous algorithm
  4. Derivative of a B_3-spline: à trous algorithm
  5. Undecimated Haar wavelet transform: à trous algorithm
  6. morphological median transform
  7. Undecimated (bi-) orthogonal wavelet transform
  8. pyramidal linear wavelet transform
  9. pyramidal B_3-spline wavelet transform

19

10. pyramidal median transform

Default is 3.

- **[-T type_of_filters]**

  1. Antonini 7/9 filters.
  2. Daubechies filter 4.
  3. Biorthogonal 2/6 Haar filters.
  4. Biorthogonal 2/10 Haar filters.
  5. Odegard 7/9 filters.
  6. User's filters.

  Default is Antonini 7/9 filters.
  This option is only available if the chosen transform method is the (bi-) orthogonal transform (-t 7).
  For option 6, see description under mr1d_trans.

- **[-m type_of_noise]**

  1. Gaussian Noise
  2. Poisson Noise
  3. Poisson Noise + Gaussian Noise
  4. Multiplicative Noise
  5. Non-stationary additive noise
  6. Non-stationary multiplicative noise
  7. Undefined stationary noise
  8. Undefined noise
  9. Stationary correlated noise
  10. Poisson noise with few events

  Default is Gaussian noise.

- **[-g sigma]**

- **[-c gain,sigma,mean]**

- **[-E Epsilon]**
  Epsilon = precision for computing thresholds (only used in case of poisson noise with few events). Default is $1e - 03$.

- **[-n number_of_scales]**

- **[-s NSigma]**

- **[-i number_of_iterations]**

- **[-e epsilon]**
  Convergence parameter. Default is $1e - 4$.

- **[-K]**
  Suppress the last scale. Default is no.

- **[-k]**
  Suppress isolated pixels in the support. Default is no.

- **[-v]**
  Verbose. Default is no.

Examples:

- mr1d_filter sig.dat filter_sig.dat
  Filtering using the à trous algorithm, and a Gaussian noise model.

- mr1d_filter -m 2 sig.dat filter_sig.dat
  Ditto, but assuming Poisson noise.

5.7 Multiscale entropy filtering: mw1d_filter

The program *mw1d_filter* filters a one-dimensional signal using the multiscale entropy method (N2-MSE approach).

USAGE: mw1d_filter options signal_in signal_out

where options are :

- **[-t type_of_multiresolution_transform]**

- **[-m type_of_noise]**

- **[-g sigma]**

- **[-c gain,sigma,mean]**

- **[-s NSigma]**

- **[-n number_of_scales]**

- **[-e epsilon]**
  Convergence parameter. Default is $1e^{-4}$.

- **[-i number_of_iterations]**
  Maximum number of iterations. Default is 10.

- **[-G RegulParam]**
  Regularization parameter. Default is 1.

- **[-D]**
  The regularization parameter is a function of the SNR in the data. Default is no.

- **[-w FilterCoefFileName]**
  Write to disk the filtered wavelet coefficient.

- **[-v]**
  Verbose. Default is no.

Examples:

- mw1d_filter sig_in.dat sig_out.dat
  filters a signal by the multiscale entropy method, assuming Gaussian noise (its standard deviation is automatically estimated).

- mw1d_filter -G 2 sig_in.dat sig_out.dat
  Same as before, but the regularization will be stronger, and the solution more smooth.

- mw1d_filter -G 2 -D sig_in.dat sig_out.dat
  The regularization is adaptive, depending on the wavelet SNR.

- mw_filter -G 2 -D -s 5 sig_in.dat sig_out.dat
  Same as before, preserving feature in the wavelet space greater than 5σ instead of the default 3σ value.

5.8   Transition detection: mr1d_nowcast

Program *mr1d_nowcast* detects the transitions in all scales at a given position. The wavelet transform used is the Haar transform, so a given wavelet coefficient at position x and at scale j ($j = 1..P$, P being the number of scales) is calculated from pixel values between positions $x - 2^j + 1$ and x. Only pixels in the signal which are on the left of a given position x (or before a given time for temporal signal) are used for the calculation of the wavelet coefficients at position x. This allows us to detect a new event in a temporal series irrespective of the time scale of the event. By default, the analysed position is the last one of the signal, but other positions can equally well be analyzed using the "-x" option. The program prints for each scale j the following information corresponding to the position x:

- **No detection**
  if the wavelet coefficient $\mid w_j(x) \mid < k\sigma_j$

- **New upward detection**
  if $w_j(x) > k\sigma_j$ and $\mid w_j(x-1) \mid < k\sigma_j$

- **New downward detection**
  if $w_j(x) < -k\sigma_j$ and $\mid w_j(x-1) \mid < k\sigma_j$

- **Positive significant structure**
  if $w_j(x) > k\sigma_j$ and $\mid w_j(x-1) \mid > k\sigma_j$
  The first detected coefficient of the structure is also given.

- **Negative significant structure**
  if $w_j(x) < -k\sigma_j$ and $\mid w_j(x-1) \mid > k\sigma_j$
  The first detected coefficient of the structure is also given.

- **End of significant structure**
  if $\mid w_j(x) \mid < k\sigma_j$ and $\mid w_j(x-1) \mid > k\sigma_j$

Furthermore the signal to noise ratio of the wavelet coefficient is given.

USAGE: mr1d_nowcast option signal_in

where options are:

- **[-m type_of_noise]**

  1. Gaussian Noise
  2. Poisson Noise
  3. Poisson Noise + Gaussian Noise
  4. Multiplicative Noise
  5. Non-stationary additive noise
  6. Non-stationary multiplicative noise
  7. Undefined stationary noise
  8. Undefined noise

  Description in section 4. Default is Gaussian noise.

- **[-g sigma]**

- **[-c gain,sigma,mean]**

- **[-n number_of_scales]**

- **[-s NSigma]**

- **[-x Position]**
  Position to analyse. Default is the last point.

Examples:

- mr1d_nowcast sig.dat
  Analyse the last point of the signal with all default option.

- mr1d_nowcast -x 55 -s 10 sig.dat
  Analyse the point at position 55, and detect the transition with a signal to noise ratio equal to 10.

5.9   Prediction: mr1d_fcast

Program *mr1d_fcast* performs a forecasting by four different methods: the standard AR model (AR), an AR method per scale (and the prediction is the coaddition of the predicted wavelet coefficients), the multiresolution AR model (MAR), and a neural network. The program can be used in two modes: the evaluation and the prediction mode. In the evaluation mode, the first part of the time series is used to predict the second part, and the output file contains the same number of values as the input file. The initial values (corresponding to the initial part of the signal) are identical, and the other values are the predicted ones. The error prediction is calculated and printed on the standard output device (screen window). In the prediction mode, the output file contains more values than the input file. The last values correspond to the predicted values. For the AR and MAR models, the order of the model can either be fixed by the user ("-a option"), or automatically calculated. In case of an automatic MAR model order estimation, the order can be different at each scale.

By default, the signal is assumed to be stationary. If the "-h" option is set, we assume a non-stationary signal. In that case, the last scale of the wavelet transform is analyzed differently (i.e. not with the AR model). Several methods can be selected by the "-B" option. The default is the polynomial extrapolation of order 2.

If the "-L" option is set, only the last pixels will be used in the analysis.

USAGE: mr1d_fcast option signal_in signal_out

where options are:

- **[-P predict_method]**

  1. Autoregressive model.
  2. Autoregressive model per scale.
  3. Multiresolution Autoregressive model.
  4. Neural network.

  Default is Multiresolution Autoregressive Model.

- **[-n number_of_scales]**
  Number of scales to be used in the Multiresolution AR model. Default is 5.

- **[-a AR_Order]**
  AR oder used for the prediction. Default is automatically estimated.

- **[-O Estimation_AR_Order_Method]**

  1. AIC
  2. AICC
  3. BIC

  Default is BIC method.

- **[-p Number_of_Predict]**
  Number of prediction. Default is 0.

- **[-m MinAROrder]**
  Minimum AR model order. Default is 1.

- **[-M MaxAROrder]**
  Maximum AR model order. Default is 10.

- **[-L NPix]]**
  Analyse the last NPix pixels. Default is all pixels.

- **[-h]**
  Non stationary signal. Default is stationary.

- **[-B extrapol_type]**

  1. Constant border ($c_J(N + i) = c_J(N)$, where c_J is the last scale, N the number of pixels).
  2. Mirror border ($c_J(N + i) = c_J(N - i)$).
  3. Double mirror border ($c_J(N + i) = 2c_J(N) - c_J(N - i)$).
  4. Polynomial extrapolation (deg 1).
  5. Polynomial extrapolation (deg 2).

  Default is 5. Only used if the "-h" option is set.

- **[-T Poly_Nbr_Pix]**
  Number of pixels used for the polynomial extrapolation. Default is 5. Only used if the "-h" option is set and if a polynomial extrapolation is used.

- **[-w InfoFileName]**
  Write to disk some infomation about the prediction error and the AR model order. The file contains a 1D table of size $N + 2$, where N is the number of scales ($N = 1$ when the MAR model is not used).

  ```
  T[0] = prediction error
  T[1] = Number_of_scale
  for j = 0 to Number_of_scale-1 do T[j] = AR order at scale j
  ```

Examples:

- mr1d_fcast sig.dat eval.dat
  Evaluate the prediction by MAR method.

- mr1d_fcast -p 1 sig.dat pred.dat
  Make a prediction at position N+1.

Hints on forecasting

Using input file `sp.csv`, assess the overall quality as given by the standard deviation prediction error.
  `mr1d_fcast -P2 -v sp.csv spout` gives an error (rounded) of 8.26.
  `mr1d_fcast -P3 -v sp.csv spout` gives an error of 7.71.
  `mr1d_fcast -P1 -v sp.csv spout` gives an error of 7.75.
  Using the MAR model (`-P3`, default option) gives predicted values as follows.
  `mr1d_fcast -P3 -p3 sp.csv spout`
  Prediction =
  +1 ==> 1214.73
  +2 ==> 1216.89
  +3 ==> 1217.54

References

[1] A. Aussem and F. Murtagh. A neuro-wavelet strategy for web traffic forecasting. *Journal of Official Statistics*, 1:65–87, 1998.

[2] Z. Bashir and M.E. El-Hawary. Short term load forecasting by using wavelet neural networks. In *Canadian Conference on Electrical and Computer Engineering*, pages 163–166, 2000.

[3] V Bjorn. Multiresolution methods for financial time series prediction. In *Proceedings of the IEEE/IAFE 1995 on Computational Intelligence for Financial Engineering*, page 97, 1995.

[4] R. Cristi and M. Tummula. Multirate, multiresolution, recursive Kalman filter. *Signal Processing*, 80:1945–1958, 2000.

[5] D. L. Donoho and I. M. Johnstone. Adapting to unknown smoothness via wavelet shrinkage. *Journal of the American Statistical Association*, 90:1200–1224, 1995.

[6] D.L. Donoho. Nonlinear wavelet methods for recovery of signals, densities, and spectra from indirect and noisy data. *Proceedings of Symposia in Applied Mathematics*, 47, 1993.

[7] D.L. Donoho and I.M. Johnstone. Ideal spatial adaptation by wavelet shrinkage. Technical Report 400, Stanford University, 1993.

[8] L. Hong, G. Chen, and C.K. Chui. A filter-bank based Kalman filter technique for wavelet estimation and decomposition of random signals. *IEEE Trans. on Circuit Systems - II Analog and Digital Signal Processing.*, 45(2):237–241, 1998.

[9] J. Moody and W. Lizhong. What is the "true price"? State space models for high frequency FX data. In *Proc. IEEE/IAFE 1997 Conference on Computational Intelligence for Financial Engineering (CIFEr)*, pages 150–156, 1997.

[10] S. Soltani, D. Boichu, P. Simard, and S. Canu. The long-term memory prediction by multiscale decomposition. *Signal Processing*, 80:2195–2205, 2000.

[11] J.L. Starck and F. Murtagh. Image filtering from multiple vision model combination. Technical report, CEA, January 1999.

[12] J.L. Starck and F. Murtagh. Multiscale entropy filtering. *Signal Processing*, 76(2):147–165, 1999.

[13] J.L. Starck, F. Murtagh, and A. Bijaoui. *Image Processing and Data Analysis: The Multiscale Approach*. Cambridge University Press, Cambridge (GB), 1998.

[14] J.L. Starck, F. Murtagh, and R. Gastaud. A new entropy measure based on the wavelet transform and noise modeling. Special Issue on Multirate Systems, Filter Banks, Wavelets, and Applications of *IEEE Transactions on CAS II*, 45(8), 1998.

[15] E.G.T. Swee and S. Elangovan. Applications of symmlets for denoising and load forecasting. In *Proceedings of the IEEE Signal Processing Workshop on Higher-Order Statistics*, pages 165–169, 1999.

[16] K. Xizheng, J. Licheng, Y.Tinggao, and W. Zhensen. Wavelet model for the time scale. In *Proceedings of the 1999 Joint Meeting of the European Frequency and Time Forum, 1999 and the IEEE International Frequency Control Symposium, 1999*, volume 1, pages 177–181, 1999.

[17] G. Zheng, J.L. Starck, J. Campbell, and F. Murtagh. The wavelet transform for filtering financial data streams. *Journal of Computational Intelligence in Finance*, 7(3), 1999.
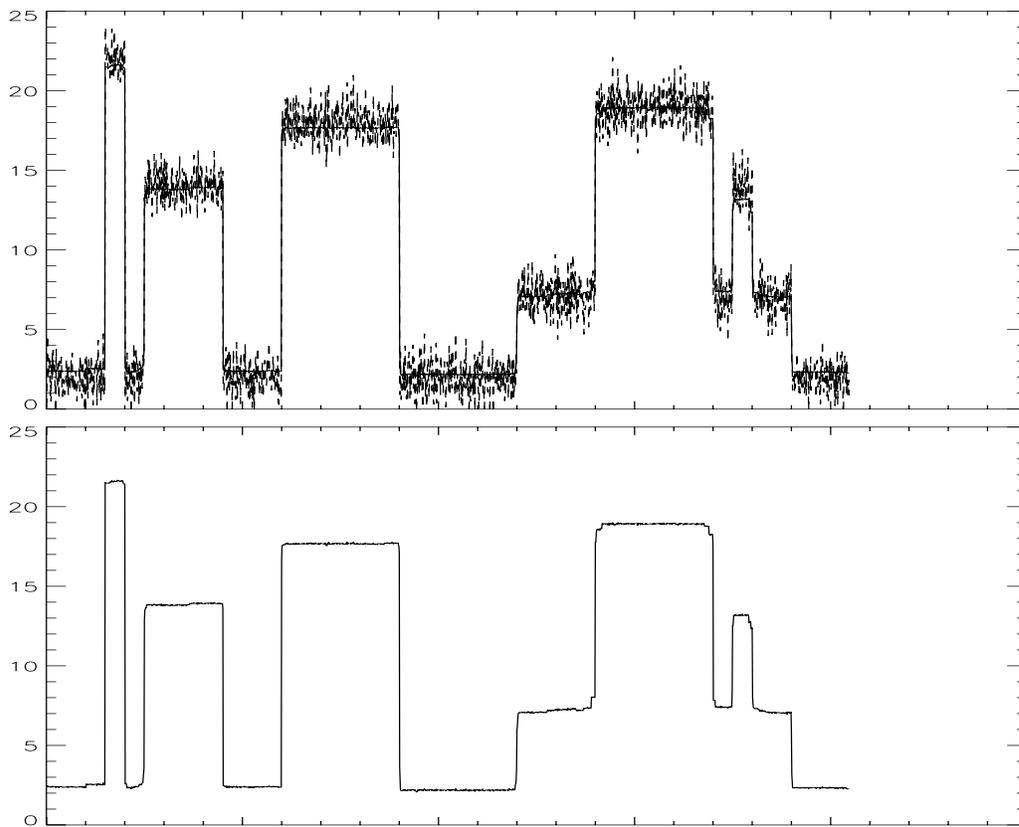
Figure 3: top, noisy blocks and filtered blocks overplotted. Bottom, filtered blocks.
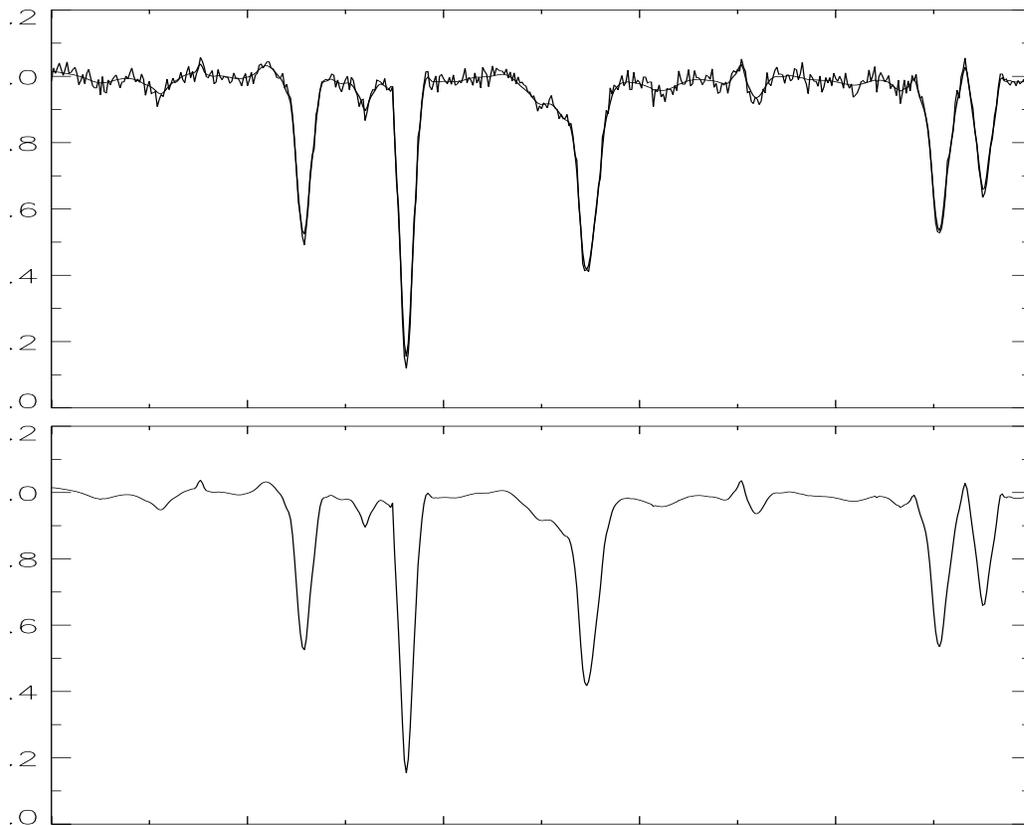

Figure 4: Top, real spectrum and filtered spectrum overplotted. Bottom, filtered spectrum.
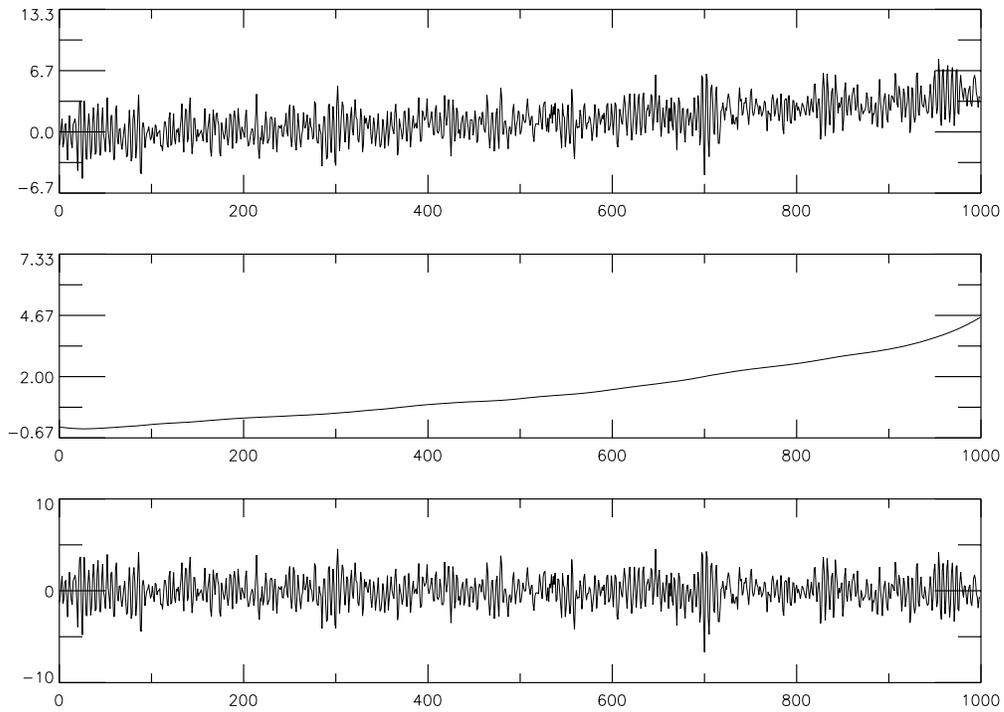
Figure 5: Top, input signal containing an AR(4) signal and a tendency. Middle, estimated tendency. Bottom, difference between the two previous signals.