European Journal of Operational Research 217 (2012) 589–599

Contents lists available at SciVerse ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Decision Support

A new wavelet-based denoising algorithm for high-frequency financial data mining

Edward W. Sun [a,*], Thomas Meinl [b]

[a] BEM Bordeaux Management School, France
[b] School of Economics and Business Engineering, Karlsruhe Institute of Technology (KIT), Germany

ARTICLE INFO

Article history:
Received 10 February 2011
Accepted 27 September 2011
Available online 21 October 2011

Keywords:
Time series
Data mining
Denoising
High-frequency financial data
Wavelets

ABSTRACT

Denoising analysis imposes new challenge for mining high-frequency financial data due to its irregularities and roughness. Inefficient decomposition of the systematic pattern (the trend) and noises of high-frequency data will lead to erroneous conclusion as the irregularities and roughness of the data make the application of traditional methods difficult. In this paper, we propose the local linear scaling approximation (in short, LLSA) algorithm, a new nonlinear filtering algorithm based on the linear maximal overlap discrete wavelet transform (MODWT) to decompose the systematic pattern and noises. We show several unique properties of this brand-new algorithm, that are, the local linearity, computational complexity, and consistency. We conduct a simulation study to confirm these properties we have analytically shown and compare the performance of LLSA with MODWT. We then apply our new algorithm with the real high-frequency data from German equity market to investigate its implementation in forecasting. We show the superior performance of LLSA and conclude that it can be applied with flexible settings and suitable for high-frequency data mining.

1. Introduction

Technical developments allow institutions to instantaneously collect massive tick-by-tick data from financial market. Mining tick-by-tick data from financial market then turns to be fundamental for financial informatics and stimulates the interest in analyzing information conveyed in this data collected at different frequency levels.

The ultra-high frequency data has the complex structure of irregularities and roughness that are caused by a large number of instantaneous changes of the markets and trading noises (see Engle (2000) and Ghysels (2000)). The noises conveyed in the high-frequency data usually illustrate heavy-tailedness (see Sun et al. (2007)), that is, the underlying time series data exhibits occasional jumps. Disregarding these irregularities will lead to erroneous conclusion when conducting data mining and statistical modeling (see Au et al. (2010)). Therefore, the statistical data mining methods (or models) require a denoising algorithm to clean the data in order to obtain reliable and significant result. Most data cleaning methods focus only on certain known type of irregularity. In high-frequency financial data, the irregularity is manifold, that is, the irregularity varies along with time and changes with different measuring scales (see Fan and Wang (2007) and Sun et al. (2008)). In consequence, finding an effective denoising algorithm turns to be critical for high-frequency data mining.

A classic assumption for data mining is that the data is generated by certain systematic patterns plus random noise (see Au et al. (2010), Bishop (2006)). Trend extraction of high-frequency time series data provides a fundamental tool to remove the noises (denoise) and obtain the long-run stable information (i.e., the systematic patterns) conveyed in the underlying dynamics. A specific problem arises when trend component exhibits occasional jumps or steep slopes that are in contrast to the slow evolving long-term trend. These occasional jumps are often caused by some exceptional singular external events (for example, unexpected large transaction or extreme prices) and should not be contributed to the normal short-term variations (since they are often considered as noises) but indeed to the long-run trend. In this paper we use the term jump in its original meaning as well as for other sudden changes like steep slopes, roofs, and valleys, see Joo and Qiu (2009) (without considering spikes). Traditional trend extraction methods like linear moving average filters usually fail to capture this information accurately as these linear methods tend to blur out jumps. Nonlinear filters, on the other hand, are not appropriate to smooth out high-frequency fluctuations sufficiently since the trends extracted by these methods are not smooth enough (i.e., usually with kicks) to present the long-run dynamic information, as they still partially contain information from the short-term variation.

In literature, several methodologies have been proposed to handle this issue, for example, Joo and Qiu (2009) suggest to use

* Corresponding author. Address: Professeur Senior en Finance, BEM Bordeaux Management School, 680 Cours de la Libération, 33405 Talence Cedex, France. Tel.: +33 556 842277; fax: +33 556 845500.
E-mail address: edward.sun@bem.edu (E.W. Sun).

0377-2217/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2011.09.049

derivatives, Qiu (2003) suggests the local polynomial estimators, and Wang (1998) applies wavelets. These methods focus either on pure jump detection and modeling or require further assumption of the noise. In this paper, we propose a new wavelet-based denoising algorithm for mining high-frequency financial data, that is, the local linear scaling approximation (in short, LLSA), which is a new nonlinear filtering algorithm based on the linear maximal overlap discrete wavelet transform (MODWT). This algorithm avoids the disadvantages of both linear and nonlinear methodologies in data denoising. Additionally, it displays all kinds of sudden changes including steep slopes, valleys, and roofs (defined by Joo and Qiu (2009)) with a good resolution and provides relatively smooth outputs.

The basic idea of our approach is to obtain a relatively smooth systematic pattern (i.e., denoising to obtain the trend of the underlying dynamics) and subsequently restore the information, particularly the information of sudden market change, that has been lost during the initial low-pass filtering process of the usual denoising methods. This can be done by adopting specific features of wavelets (i.e., the wavelet coefficient step response structure) and the wavelet specific multiresolution analysis that gives us the flexibility to improve the resolution of rapid changes according to the actual time series structure. This procedure yields several desirable properties, such as the local linearity feature (i.e., the ability to maintain control over frequency passbands, which is usually not feasible for nonlinear filters) and consistency of the algorithm.

The proposed algorithm in this paper enables the user to heuristically set the configuration parameters for specific problem. The underlying wavelet methods have been proven very efficient in handling all kinds of singularities for both continuous and discrete signals. Studies of applying wavelets in denoising data and construction can be found in Gençay et al. (2005, 2002), Keinert (2004), Mallat and Hwang (1992), and Percival and Walden (2006) among others. Ramsey (2002) highlights some research areas where wavelet analysis might be applied in economics and Crowley (2007) provides a survey about how wavelet methods have been used in the economics and finance literature. Gençay et al. (2003) propose a method based on a wavelet multiscaling approach for decomposing time series data. Lada and Wilson (2006) develop a wavelet-based spectral method for steady-state simulation analysis. Laukaitis (2008) applies wavelet transforms for high-frequency data denoising in the study of credit card intraday cash flow and intensity of transactions. Gençay and Gradojevic (2011) introduce a wavelet approach to estimate the parameters of a linear regression model. Sun et al. (2011) propose a wavelet approach in analyzing volatility dynamics of foreign exchange rate based on the second-by-second data.

In this paper, we show that our algorithm maintains the original wavelet transform's computational complexity and its approximation errors are bounded. In order to illustrate the computational reliability and consistency we have analytically shown, we further investigate the performance of LLSA by conducting two independent experiments. One is based on a simulation study and the other on the forecasting implementation of the LLSA with real high-frequency data from German DAX 30 stocks. The results of our simulation study confirm the analytical properties of LLSA. In addition, the simulation results in this paper coincided with the results reported by Meinl and Sun (2010) that illustrate how LLSA performs significantly superior than the benchmark filter (i.e., the MODWT) under four goodness-of-fit tests with high-frequency data. When applying LLSA for forecasting with the high-frequency data, we find that LLSA provides reliable and significant results and performs better than MODWT.

The paper is organized as follows. In Section 2, we briefly introduce the basic concepts of wavelets and discrete wavelet transforms. Then we show the derivation of the local linear scaling approximation (LLSA) algorithm based on wavelet transforms. We prove the properties of LLSA in Section 3. We show the performance of LLSA with two experiments (i.e., the simulation study and empirical investigation of forecasting based on real data of German DAX 30 stocks) in Sections 4 and 5, respectively. We summarize in Section 6.

2. Local linear scaling approach

2.1. Wavelets and wavelet transforms

Wavelets are bases of $L^2(\mathbb{R})$ that enable a localized time-frequency analysis. Since unlike trigonometric bases used in the Fourier transform, wavelet functions usually have either compact support or decay exponentially fast to zero. These bases provide a multiresolution analysis of a signal that allows to analyze it simultaneously on different (usually dyadic) scales. An analog discrete formulation is given by discrete wavelet filter banks $h = h_1, \ldots, h_L$ of length L. In this paper, we focus on the popular wavelets applied in practice (the Haar and *Daubechies D*4). Discussion about the constructional condition and mathematical background can be found in Daubechies (1992).

A discrete wavelet transform (DWT) is an orthogonal transform of a time series (or any discrete signal) X of length N (which must be a multiple of 2^J) into J wavelet coefficient vectors $W_j \in \mathbb{R}^{N/2^j}, 1 \leqslant j \leqslant J$ and one scaling coefficient vector $V_J \in \mathbb{R}^{N/2^J}$. This can be obtained from a matrix operation

$$[W_1, \ldots, W_J, V_J] = \mathcal{W}X$$

with the transformation matrix $\mathcal{W}$ determined by the wavelet filter banks h. It is more convenient to use the pyramid algorithm developed by Mallat (1989), with a computational complexity of $\mathcal{O}(N)$ (i.e., the amount of calculation depends linearly on the length N of the time series). Applying the inverse transform on these vectors, we have

$$[W_1, \ldots, W_J, V_J]\mathcal{W}^T p = S_J + \sum_{j=1}^{J} D_j = X,$$

that is, an additive decomposition of the original time series. As

$$S_{\underset{\sim}{J-1}} = S_J + \sum_{\underset{\sim}{j=J}}^{J} D_j$$

holds, S_j is approximation (i.e., moving weighted average) of X on different dyadic scales, while D_j denotes the detail vectors, which is the details we lose at every coarser approximation level, yielding a multiscale decomposition. For each scale j we then have a separation into high and low frequencies by a wavelet filter with the bandwidth determined by j.

In this paper we use a variation of the traditional DWT, the maximal overlap discrete wavelet transform (MODWT) to develop the new optimal algorithm for denoising analysis. The MODWT differs from the DWT in the sense that all vectors are in $\mathbb{R}^N$ and has a higher computational complexity of $\mathcal{O}(N\log_2 N)$ (see, for example, Percival and Walden (2006)). Additionally, for the MODWT, above mentioned constraint on N can be released. The reason why we favor this redundant transform over the one-to-one and onto DWT is that only the MODWT can be associated with zero phase filters. This provides shift-invariant V_J, W_j, S_J, and D_j series. Shifting the signal X by a certain amount results in likewise shifted output vectors. This is a vital feature and requirement for our approach.

2.2. The local linear scaling approximation (LLSA)

Other nonlinear filters often combine nonlinear operations to preserve the edges with linear operations which further smooth out details having passed the nonlinear operation. The multiscale decomposition of the MODWT enables us to use a different approach. The main idea is that we start with the very smooth approximation S_J and reconstruct the lost details near the jumps that have been smoothed out, by using the information conveyed in the detail series D_j.

Additionally, to the scaling level J and the wavelet, we set K, the expected amount of jumps in X, along with $0 \leqslant \Lambda \leqslant J$ which determines up to which scale details should be reconstructed. LLSA is then capable to detect automatically the regions to be refined and restore details. This improves the shape of jumps and slopes that are blurred out in S_J. In the following sections, we show the algorithm by deriving it step by step. As the computations for each scale j depend on the results derived on scale $j+1$, we first show how to align the wavelet coefficient vectors in Section 2.2.1. In Section 2.2.2, we explain our rule of detecting the first jump on scale J. In Section 2.2.3 we explain how to determine the areas where lost details should be reconstructed. In Section 2.2.4, by applying the same rules, we determine the remaining $K-1$ jumps and their areas. Having all jumps on scale J determined, in Section 2.2.5 we apply a modified rule on the lower scales $1 \leqslant j < J$. Finally, we show how to reconstruct lost jump details in Section 2.2.6, and summarize the complete algorithm in Section 2.2.7.

2.2.1. Aligning the wavelet coefficient vectors

When applying the MODWT onto X, J wavelet coefficient vectors $\overline{W}_j$ are obtained, which are not exactly aligned with the events in X, as derived through the computationally efficient MODWT pyramid algorithm (first introduced for the DWT by Mallat (1989)). Since the reconstruction of jump details on lower scales depends on the local information derived on the higher scales, the LLSA is sensitive to coefficient vectors not being aligned correctly. We derive the correct aligned vectors W_j by circularly shifting each $\overline{W}_j$ with the effective filter width $-L_j^{\text{wvlt}}/2$ given by

$$L_j^{\text{wvlt}} = (2^j - 1)(L^{\text{wvlt}} - 1) + 1$$

with L^{wvlt}, the basic filter length as stated in Table 1. For further details about the correct alignment of the wavelet coefficient vectors for different wavelets can be found in Percival and Walden (2006).

2.2.2. First jump detection

There exist a number of works that consider jump detection, for example Wang (1995), Wang (1998), Qiu and Yandell (1998), Qiu (2003), Gijbels et al. (2007), Joo and Qiu (2009), and Sun and Qiu (2007). In this paper, we consider a simple rule of determining jump locations, which is based on the observation that the wavelet coefficients on scale J denote the differences between the weighted moving averages in S_J. As we are interested in refining successively only those jumps containing the highest potential of improvement for S_J, we determine the first index indicating the biggest jump by

$$l^W := \text{argmax}_t(|W_{J,t}|).$$

Table 1
n_α, n_β and L^{wvlt} for different wavelets.

Wavelet	n_α	n_β	L^{wvlt}
Haar	1	1	2
$D4$	2	4	4

This gives us potential jumps from the point of view of the original smoothed signal S_J, ignoring the wavelet coefficient vectors W_j associated with higher frequencies on the lower scales $1 \leqslant j < J$. This step is independent of the wavelet used.

2.2.3. First reconstruction boundary determination

Having the first jump determined, we need to fix the boundaries around it where details shall be restored. We suppose that these boundaries are set accordingly to ensure that (1) the complete jump should be captured, (2) no unnecessary details beyond the jump should be added, and (3) the transition between the modified sections and the original output S_J should be smooth (i.e., without adding artificial jumps).

The heuristic rule we derived, coping with these requirements by determining the boundaries to each side of the jump, depends on the wavelet. The intuition behind this rule is that, for the observation in respect to jumps, the general structure of the wavelet coefficients remains the same. This is due to the fact that higher scales almost only contain low frequencies and are hardly affected by high-frequency noise. If we analyze the jump coefficient structure for a particular wavelet and transfer it to the real signal, we can approximately capture the noisy jump as well. We first define some parameters for LLSA as follows.

Definition 1. Let X^s be a signal containing a single step (and being constant otherwise), and $W_j^s, 1 \leqslant j \leqslant J$, the wavelet coefficient vectors of the MODWT applied on X^s (exemplarily depicted for the Haar and $D4$ wavelets in Fig. 1). With

$$l_j^s := \text{argmax}_t\left(|W_{j,t}^s|\right) \tag{1}$$

and

$$l_{\min}^{js} := \min\left\{t | t \in \text{supp}\left(W_j^s\right)\right\}, \tag{2}$$

$$l_{\max}^{js} := \max\left\{t | t \in \text{supp}\left(W_j^s\right)\right\}, \tag{3}$$

we define

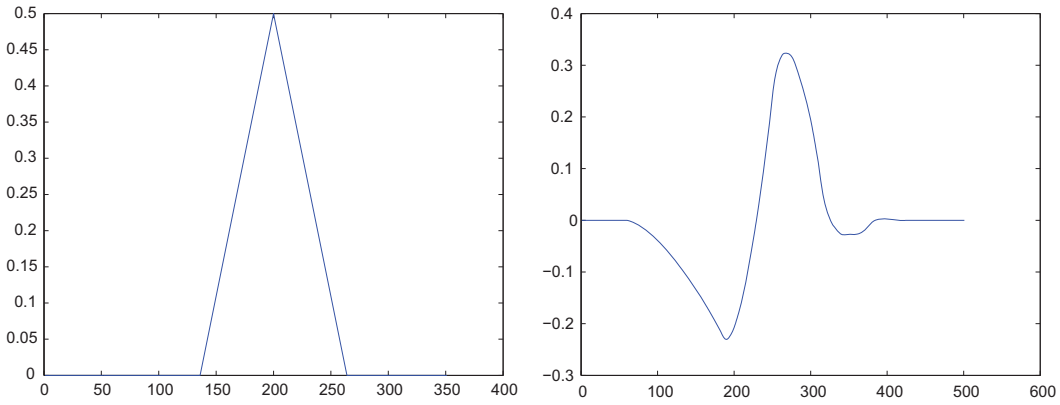
$$n_{\alpha,j}^{\text{wvlt}} := \sum_{t=l_{\min,j}^s}^{l_j^s - 1} \frac{\left|\text{sign}W_{j,t+1}^s - \text{sign}W_{j,t}^s\right|}{2}, \tag{4}$$

$$n_{\beta,j}^{\text{wvlt}} := \sum_{t=l_j^s+1}^{l_{\max,j}^s} \frac{\left|\text{sign}W_{j,t-1}^s - \text{sign}W_{j,t}^s\right|}{2}. \tag{5}$$

The idea of this definition is that, starting from the largest absolute wavelet coefficient (turning point of a sudden change or the exact location where the jump occurs), the general shape of this jump wavelet coefficient structure can be determined by change of signs the coefficient vectors exhibit to the right and left side of this point, that is, $n_{\alpha,j}^{\text{wvlt}}$ and $n_{\beta,j}^{\text{wvlt}}$, respectively. As this structure remains same for all $1 \leqslant j \leqslant J$ and all wavelets considered, we can omit the scale and wavelet identifiers and simply write n_α and n_β. We state these values for the Haar and $D4$ wavelets in Table 1.

In order to reconstruct the whole jump in the original signal X we need to determine the area inside W_J where the details should be preserved. Transferring the jump wavelet coefficient structure onto W_J the left boundary α is set to be the maximal index so that at least n_α change of signs occur between α and l^W. We set the right boundary β to be the minimal index according to n_β. Formally, these boundaries are given as follows:

$$\alpha = \max\left\{l \in [1, l^W - 1] \middle| \sum_{t=l}^{l^W-1} \frac{|\text{sign}W_{J,t+1} - \text{sign}W_{J,t}|}{2} \geqslant n_\alpha\right\}$$



(a) Haar (b) $D4$

Fig. 1. Wavelet coefficient structure for step functions.

and

$$\beta = \min\left\{ l \in [l^W + 1, N] \middle| \sum_{t=l^W+1}^{l} \frac{|\mathrm{sign}W_{J,t-1} - \mathrm{sign}W_{J,t}|}{2} \geqslant n_\beta \right\}.$$

We denote this area around the jump as $\Omega^W := [\alpha, \beta]$. For example, for Haar wavelet, Ω^W is determined by the first change of sign of the wavelet coefficients to the right-and left-hand side of the largest absolute wavelet coefficient l^W.

2.2.4. kth Jump detection and boundary determination on scale J

Detecting further jumps can follow the same procedure as we have shown in previous section, the difference is that we have to exclude the jumps already detected and their surrounding areas. We set $\Omega_1^W := \Omega^W$. Subsequently, for detecting the kth jump, $1 < k \leqslant K$, we determine

$$l_k^W := \mathrm{argmax}_t \left(|W_{J,t}| \setminus \bigcup_{i=1,\ldots,k-1} \Omega_i^W \right). \tag{6}$$

We set the corresponding $\Omega_k^W := [\alpha_k, \beta_k]$ and substitute in above equations l^W by l_k^W.

2.2.5. kth Jump detection and boundary determination on scales j < J

The above procedure yields $\Omega_k^W, 1 \leqslant k \leqslant K$ for the wavelet coefficients on scale J. For the lower scales $J - \Lambda \leqslant j < J$ we determine $l_{j,k}^W$ with

$$l_{j,k}^W := \mathrm{argmax}_t \left(|W_{j,t}| \middle| t \in \Omega_{j+1,k}^W \right).$$

This means that the jumps detected on each scale j depend on area of the jump already detected on the next higher scale $j+1$. This ensures the refinement of the same jump over different scales. The region

$$\Omega_{j,k}^W := [\alpha_{j,k}, \beta_{j,k}]$$

covering the jumps on these scales can be determined accordingly.

2.2.6. Detail reconstruction

Once all areas $\Omega_{j,k}^W$ have been determined, we set $1 \leqslant j \leqslant J$

$$\widetilde{W}_{j,t} = \begin{cases} W_{j,t} & \text{for } t \in \bigcup_{\substack{j=J-\Lambda,\ldots,J \\ k=1,\ldots,K}} \Omega_{j,k}^W, \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

In this way, we only keep the wavelet coefficients containing information about the jump details, and discard all other coefficients by setting them to zero. Applying the inverse MODWT onto the modified circularly backshifted wavelet coefficient vectors

$$\overline{\widetilde{W}}_j = \mathcal{T}^{L_j/2} \widetilde{W}_j$$

leads to the modified detail vectors $\widetilde{D}_j$ that now contain only the details near the jumps up to scale $J - \Lambda$. The adapted approximation is given by

$$S_J^{\mathrm{LLSA}} = S_J + \sum_{j=1}^{J} \widetilde{D}_j = S_J + \sum_{j=J-\Lambda}^{J} \widetilde{D}_j. \tag{8}$$

2.2.7. The LLSA algorithm: summary

Given the input (J, K, Λ) together with the wavelet, LLSA(J, K, Λ) determines for every $1 \leqslant k \leqslant K$ and $J - \Lambda \leqslant j \leqslant J$,

$$\Omega_{j,k}^W := [\alpha_{j,k}, \beta_{j,k}] \tag{9}$$

with

$$\alpha_{j,k} := \max\left\{ l \in \left[1, l_{j,k}^W - 1\right] \middle| \sum_{t=l}^{l_{j,k}^W - 1} \frac{|\mathrm{sign}W_{j,t+1} - \mathrm{sign}W_{j,t}|}{2} \geqslant n_\alpha \right\}, \tag{10}$$

$$\beta_{j,k} := \min\left\{ l \in \left[l_{j,k}^W + 1, N\right] \middle| \sum_{t=l_{j,k}^W + 1}^{l} \frac{|\mathrm{sign}W_{j,t-1} - \mathrm{sign}W_{j,t}|}{2} \geqslant n_\beta \right\} \tag{11}$$

and

$$l_{J,k}^W := \mathrm{argmax}_t \left(|W_{J,t}| \setminus \bigcup_{i=1,\ldots,k-1} \Omega_{J,i}^W \right), \tag{12}$$

$$l_{j,k}^W := \mathrm{argmax}_t \left(|W_{j,t}| \middle| t \in \Omega_{j+1,k}^W \right) \text{for } J - \Lambda \leqslant j < J. \tag{13}$$

The modified wavelet coefficient vectors are set according to Eq. (7). The inverse MODWT is then applied onto these vectors, with the final output signal given by Eq. (8). The pseudo code for the LLSA algorithm (assuming correctly aligned wavelet coefficient vectors) is stated in Algorithm 1.

Algorithm 1. LLSA

$w \leftarrow$ wavelet $\in \{Haar, D4\}$
$J \leftarrow$ initial scale
$\Lambda \leftarrow$ refinement scale
$K \leftarrow$ number of sections to be refined
$W_j \leftarrow \text{MODWT}(X, w, J)$
Compute n_α and n_β as given by Eq. (4) and Eq. (5) using Eqs. (1)–(3).
for $k \leftarrow 1$ **to** K **do**
  Compute $l^W_{J,k}$ and $\Omega^W_{J,k}$ as given by Eq. (12) and Eqs. (9)–(11), respectively.
for $k \leftarrow 1$ **to** K **do**
  for $j \leftarrow \Lambda - J$ **to** $J - 1$ **do**
  Compute $l^W_{j,k}$ and $\Omega^W_{j,k}$ as given by Eq. (13) and Eqs. (9)–(11), respectively.
for $j \leftarrow 1$ **to** J **do**
  Set $\widetilde{W}_j$ as given by Eq. (7).
$\widetilde{D}_j \leftarrow \text{IMODWT}(\widetilde{W}_j)$
Output S^{LLSA}_J as given by Eq. (8).

3. Properties

In this section, we discuss the properties of LLSA, i.e., the local linearity, computational complexity, and consistency (see also Meinl and Sun (2010)).

3.1. Local linearity

Although LLSA belongs to the class of nonlinear filters, it still retains the properties of a linear filter on certain subintervals. The reconstruction of prior lost details only affects the extracted trend (i.e., systematic pattern) in the immediate proximity of the jump. Outside of these regions, the output from the LLSA algorithm coincides with the output of the linear MODWT. Therefore, it can be controlled a priori in terms of frequency passbands.

Theorem 1. *Let X be a signal of length N to be filtered and S^{LLSA}_J the output generated by LLSA(J, K, Λ). The subintervals, which can be interpreted as the output of a linear filter, are given by*

$$S^{\text{LLSA}}_J \setminus \bigcup_{\substack{j = J-\Lambda,\dots,J \\ k = 1,\dots,K}} \Omega^S_{j,k}, \tag{14}$$

with

$$\Omega^S_{j,k} := \left[\max\{1, \alpha_{j,k} - L^{\text{wvlt}}_j + 1\}, \alpha_{j,k} + L^{\text{wvlt}}_j\right] \cup \left[\beta_{j,k} - L^{\text{wvlt}}_j, \min\left\{\beta_{j,k} + L^{\text{wvlt}}_j - 1, N\right\}\right]. \tag{15}$$

Proof. The subintervals of S^{LLSA}_J, which can be interpreted as the output of a linear filter, are those sections affected by either the wavelet coefficients we retained or by the ones we set to zero. For every refined section $k = 1,\dots,K$ and every level $j = J-\Lambda,\dots,J$, we need to determine the indices of $\widetilde{D}_j$, which are affected by both coefficient types at the same time. For each jump, the first coefficients of $\widetilde{W}_j$ set to zero are given by $\alpha_{j,k}$ and $\beta_{j,k}$ on the left and right-hand side, respectively. With L^{wvlt}_j denoting the respective wavelet filter lengths, the indices of $\widetilde{D}_j$ affected by both retained and discarded wavelet coefficients are given by $\left[\alpha_{j,k} - L^{\text{wvlt}}_j + 1, \alpha_{j,k} + L^{\text{wvlt}}_j\right]$ and $\left[\beta_{j,k} - L^{\text{wvlt}}_j, \beta_{j,k} + L^{\text{wvlt}}_j - 1\right]$. When including the natural boundaries $[1, N]$, we obtain Eq. (15). For the reconstructed signal S^{LLSA}_J in Eq. (8), mutually excluding all $\Omega^S_{j,k}$ leads to Eq. (14). □

Although we cannot make any statement about exact proportions of frequencies contained in $\Omega^S_{j,k}$, we still have a lower bound of frequencies not contained there. As we do not include any detail levels beyond $J - \Lambda$, this bound can be easily derived by analyzing the transfer functions of the linear MODWT for scales $1,\dots,J-\Lambda-1$.

3.2. Computational complexity

The computational complexity of the MODWT is preserved for LLSA and there is no additional computation to be considered when applying the LLSA algorithm.

Corollary 1. *The computational complexity of LLSA is given by $\mathcal{O}(N\log_2 N)$.*

Proof. As proved by Percival and Walden (2006), the computational complexity of the MODWT, which LLSA is built upon, is given by $\mathcal{O}(N\log_2 N)$. The determination of $\alpha_{j,k}$ and $\beta_{j,k}$ as defined in Theorem 1, can be obtained at most N comparisons for every $j = 1,\dots,J-\Lambda$ and $k = 1,\dots,K$. Furthermore, manipulation of the wavelet coefficients requires at most N additional operations for each of all J levels. Consequently, the computational complexity of $\mathcal{O}(N\log_2 N)$ is preserved. □

3.3. Consistency

The following theorem provides an upper bound for the error of the estimated trend to the real one. We now show the asymptotic convergence of LLSA towards the MODWT.

Theorem 2. *Given any signal X of length N with the trend $\vartheta(X)$, the error of LLSA's estimated S^{LLSA}_J is bounded, that is, for any choice of (J, K, Λ) there exists a constant $A > 0$ such that*

$$\sum_{t=1}^{N} \left|\vartheta_t(X) - S^{\text{LLSA}}_{J,t}\right| < A \tag{16}$$

holds. Furthermore, for K fixed and $N \to \infty$,

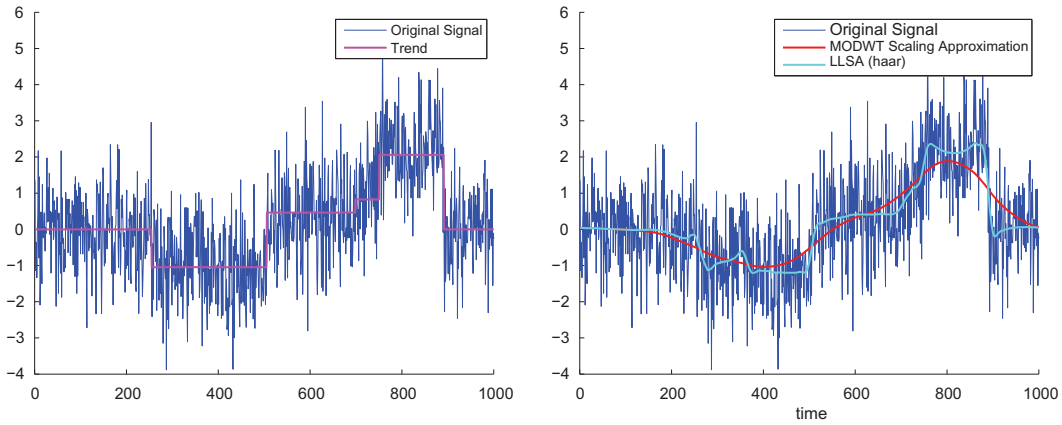
$$\lim_{N\to\infty} \frac{1}{N} \sum_{t=1}^{N} \left|S^{\text{MODWT}}_{J,t} - S^{\text{LLSA}}_{J,t}\right| = 0 \tag{17}$$

holds, i.e., LLSA converges asymptotically towards the estimated trend of the MODWT.

Proof. Let J be fixed. Considering $0 \leqslant \Lambda \leqslant J$ and $0 \leqslant K \leqslant N$, by setting either $\Lambda = 0$ or $K = 0$ we have $S^{\text{LLSA}}_J = S^{\text{MODWT}}_J$. For any other choice of K and Λ more details will be added to the estimator S^{LLSA}_J, according to $\Omega^W_{j,k}$. After the reconstruction (see Eq. (8)) these additional details correspond to the estimator $S^{\text{MODWT}}_{J-\Lambda}$ are bounded as well. Unfortunately, as during the refinement process several wavelet coefficients are set to zero, this match only holds approximately, i.e., no exact description of $S^{\text{LLSA}}_{J,t}$ is available, since this procedure causes the information contained in the wavelet coefficients on different levels to be intermixed. However, knowing that the MODWT approximation of any level is bounded by the signal itself, we can set up an ϵ-tube around the initially estimated trend by

$$\epsilon := |\max_t X_t - \min_t X_t|. \tag{18}$$

As we have



(a) Original signal (b) Filtered signal

Fig. 2. Simulated original and filtered signal.

Table 2
Mean and variance (the latter in parenthesis, both given in units of 10^{-5}), of the MSE over 512 simulation runs with 5 jumps.

	$\sigma = 0.01$	$\sigma = 0.5$	$\sigma = 1$	$\sigma = 2$	$\sigma = 5$
$\text{LLSA}_{(\text{Haar})}$	0.0142	0.0239	0.0333	0.0521	0.1089
	(0.23)	(0.41)	(0.44)	(0.77)	(14.1)
$\text{MODWT}_{(\text{Haar})}$	0.0974	0.1003	0.1033	0.1082	0.1245
	(15.4)	(18.5)	(28.2)	(33.2)	(13.0)
$\text{LLSA}_{(D4)}$	0.0177	0.0316	0.0459	0.0746	0.1611
	(0.46)	(1.99)	(2.08)	(1.08)	(4.95)
$\text{MODWT}_{(D4)}$	0.0922	0.0949	0.0982	0.1045	0.1232
	(25.8)	(49.3)	(25.0)	(15.4)	(15.6)

$$\min_t X_t \leqslant S_{J,t}^{\text{MODWT}} \leqslant \max_t X_t$$

for all $1 \leqslant j \leqslant J$, it follows that

$$\left|\vartheta_t(X) - S_{J,t}^{\text{MODWT}}\right| \leqslant \epsilon.$$

This should also hold for LLSA, that is,

$$\min_t X_t \leqslant S_{J,t}^{\text{LLSA}} \leqslant \max_t X_t \quad \text{and} \quad |\vartheta_t(X) - S_{J,t}^{\text{LLSA}}| \leqslant \epsilon.$$

Therefore, we can estimate an upper bound for the error by $A = N \cdot \epsilon$, which suffices Eq. (16). To prove the asymptotic consistency we assume that for a fixed K the $\Omega_{j,k}^W$ are ordered in time, that is

$$\alpha_{j,k+1} \geqslant \beta_{j,k}$$

for all $1 \leqslant j \leqslant J$ and $1 \leqslant k \leqslant K-1$. Hence, with

$$t_K := \max_{t_j}\{t \in \Omega_{j,K}^S\},$$

we have $S_{J,t}^{\text{LLSA}} = S_{J,t}^{\text{MODWT}}$ for all $t > t_K$. As the prior deviations between both estimators are bounded by the same ϵ given in Eq. (18), we have

$$\sum_{t=1}^{N}\left|S_J^{\text{MODWT}} - S_J^{\text{LLSA}}\right| = \sum_{t=1}^{t_K}\left|S_J^{\text{MODWT}} - S_J^{\text{LLSA}}\right| \leqslant t_K \cdot \epsilon$$

for all N, which leads to Eq. (17). □

The consistency boundaries shown in this theorem are in a general setting. As we have pointed out, LLSA provides the possibility of using specific settings in order to solve the practical problem optimally. In the following two sections, we conduct two studies to investigate the performance of LLSA and empirically confirm its properties we have analytically proved.

4. Simulation study

In order to investigate the performance of LLSA, we conduct a simulation study. We set up a signal of length 2^{10}. Jump occurrences in this signal are uniformly distributed (coinciding with a Poisson arrival rate observed in many systems), with the jump heights being a random number drawn from a normal distribution with mean 0 and variance 1. The signal is constant between the jumps. White Gaussian noise is added to the signal afterwards. We run the simulation with different settings, varying number of jumps $\widetilde{K}$, noise variance σ, and wavelets. We run 2^9 times simulation for each setting, with m different time series generated for the mth run. Totally 131,328 different time series are generated in our simulation. For each run we measure the mean squared error (MSE) aggregated over all m outcomes and compare them to the errors of the linear MODWT applied to the same series.[1]

For both filters, we choose the same bandwidth $2^J = 2^7$. At first we assume the amount of jumps to be known for our algorithm, i.e., we have $K = \widetilde{K}$. We then set $\Lambda = 3$ independent of the noise variance. These parameter combinations are certainly not optimal for all different settings, but nevertheless allow us to compare the robustness and performance of our algorithm to the MODWT without giving any specific data dependent input other than K.

Although the choice of the wavelet is not as important for the MODWT as for the DWT (see Percival and Walden (2006)), we test the robustness of LLSA's performance with the above introduced wavelets (i.e., Haar and $D4$). The wavelet dependent parameters required for the LLSA algorithm are reported in Table 1. We choose $\sigma \in [0.01, 0.5, 1, 2, 5]$ for the noise variance. 0.01 signifies that there is almost no any noise in the signal, while the jumps and the noise share the same distribution for variance 1. Given a noise variance of 5 or higher, the jumps are hardly discernible, that is, the signal is completely dominated by the noise. The amount of jumps $\widetilde{K}$ is fixed for all 512 simulation runs and chosen from the set $[5, 10]$. These are reasonable parameter combinations for our algorithm, being designed for extracting smooth trends with occasional jumps and slopes.

An exemplary plot of one simulation run is depicted in Fig. 2, which contains two panels. In panel (a) we show the predetermined trend and the simulated noisy data (referred to as the original data). In panel (b) we show the outcome of the original data denoised by both of the two methods (i.e., MODWT and LLSA).

[1] We compute the mean absolute error as well. Since the results only differ in scale and lead to completely same conclusion, we do not report them here.

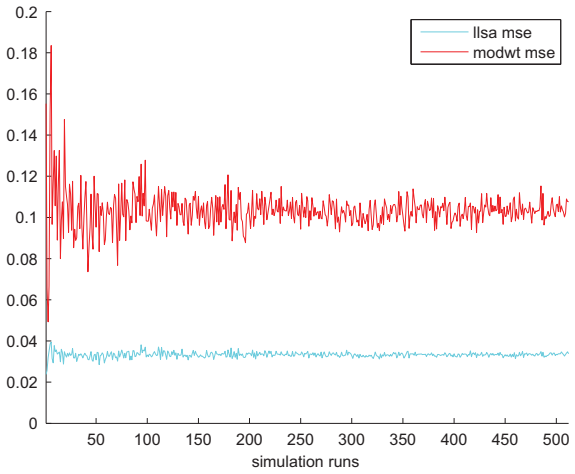


Fig. 3. MSE for LLSA and MODWT based on Haar wavelet with 5 jumps and $\sigma = 1$ in the simulation study. The figure shows that the error measure for LLSA is always strictly smaller than for the MODWT.

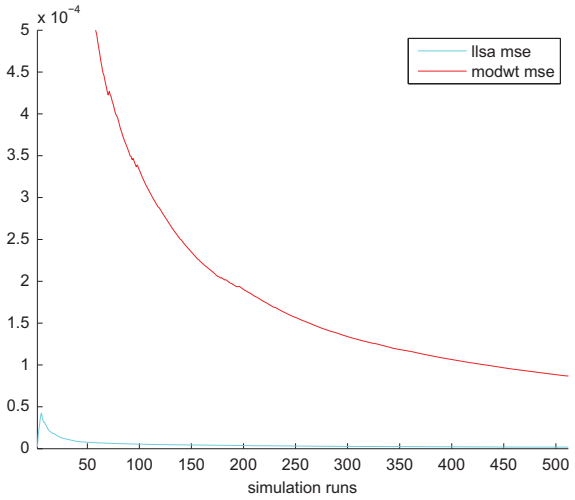


Fig. 4. Variance of the MSE for LLSA and MODWT based on Haar wavelet with 5 jumps and $\sigma = 1$ in the simulation study. It can be seen that the variance of the MSE, when applying LLSA, turns out to be strictly smaller than the one obtained by using MODWT. Additionally, this figure shows that when increasing the number of simulation runs, the error variance decreases. The speed of the decrease in variance (i.e., the speed the error variance converges to its limit) of LLSA is faster than that of MODWT.

By comparing both extracted trends in Fig. 2, we can easily identify that the trend extracted by LLSA is much closer to our predetermined trend than the outcome of MODWT.

In Table 2, we report the mean of the MSE and the corresponding sample variances (in parenthesis) over all 512 runs. We observe that the errors of all algorithms increase with a higher noise variance. Independent of the choice of the wavelet, for $\sigma \in \{0.01, 0.5, 1, 2\}$ LLSA always outperforms the MODWT (utilizing the same wavelet). When $\sigma = 5$, the errors of LLSA tend to be larger,

Table 4
Mean and variance (the latter in parenthesis, both given in units of 10^{-5}), of the MSE over 512 simulation runs with 10 jumps and $\sigma = 1$.

	$p = 0.3$	$p = 0.5$	$p = 1$
$\text{LLSA}_{(\text{Haar})}$	0.0606	0.0607	0.1128
	(1.84)	(3.40)	(32.0)
$\text{MODWT}_{(\text{Haar})}$	0.2133	0.2139	0.2146
	(51.0)	(70.9)	(53.0)
$\text{LLSA}_{(D4)}$	0.0829	0.0712	0.1318
	(7.82)	(2.76)	(87.6)
$\text{MODWT}_{(D4)}$	0.2055	0.2056	0.2051
	(69.4)	(51.6)	(60.6)

Table 5
MSE 99% bootstrapping confidence intervals over 512 simulation runs with 10 jumps and $\sigma = 1$. UCB < 0 denotes a significant better performance of LLSA over MODWT, LCB > 0 a superior performance of the MODWT. For LCB < 0 and USB > 0 there is no significant conclusion.

	$p = 0.3$	$p = 0.5$	$p = 1$
Haar	$(-0.154, -0.151)$	$(-0.154, -0.151)$	$(-0.104, -0.097)$
$D4$	$(-0.124, -0.121)$	$(-0.135, -0.133)$	$(-0.075, -0.070)$

because for this high level of noise, LLSA restores details in areas where actually no jump has occurred. In this way, the LLSA reconstructs jumps that deviate from the original signal, generating larger error compared to the MODWT, which simply ignores those jumps caused by noise. In all other cases, LLSA yields lower errors, independent of the wavelet utilized.

We find that the variances of the errors of LLSA are lower than that of the MODWT. Among the different wavelets used in LLSA, the Haar wavelets generate the lowest variance of the error, except for $\sigma = 5$. We also find that the speed of convergence of the error variances is much faster for our algorithm than that for MODWT, illustrating LLSA's better performance due to the consistency properties we have proved in Section 3.3. Figs. 3 and 4 illustrate the errors and variances, respectively.

As pointed out before, a signal with high noise level and the inherent jumps can generate larger error for LLSA. When the number of expected jumps K exceeds (or simply deviates from) the number of jumps $\widetilde{K}$ that actually occurred, we might also have large error for LLSA. We run simulations in which we allow K to vary, i.e., we choose K as a uniformly distributed random integer from the interval $[(1-p)\widetilde{K}, (1+p)\widetilde{K}]$ with p (the percentage of deviation) and $\widetilde{K} = 10$. When $p = 1$, the interval ranges from 0 (being equivalent to the MODWT) up to twice as much as $\widetilde{K}$ being restored. The results are summarized in Table 4. When $p = 0.3$ and $p = 0.5$, we observe that our algorithm still performs better than the MODWT, according to the mean value of the errors as well as the variance. For the very large deviation (i.e., $p = 1$), we can still see LLSA performs better based on the measure of MSE.

We conduct a statistical significance test based on the bootstrapping method proposed by Sun et al. (2008). We bootstrap the confidence intervals for the mean difference between LLSA and MODWT. LCB and UCB stand for the lower and upper confidence bounds, respectively. If we have both LCB < 0 and UCB < 0, there is a significant improvement of LLSA over the MODWT. If

Table 3
MSE 99% bootstrap confidence intervals (LCB, UCB) for the mean difference of LLSA and MODWT (for 512 simulation run with 5 predetermined jumps). UCB < 0 denotes a significant better performance of LLSA over MODWT, LCB > 0 a superior performance of the MODWT. For LCB < 0 and USB > 0 there is no significant conclusion.

	$\sigma = 0.01$	$\sigma = 0.5$	$\sigma = 1$	$\sigma = 2$	$\sigma = 5$
Haar	$(-0.084, -0.082)$	$(-0.077, -0.075)$	$(-0.071, -0.069)$	$(-0.057, -0.055)$	$(-0.017, -0.014)$
$D4$	$(-0.075, -0.073)$	$(-0.064, -0.062)$	$(-0.053, -0.051)$	$(-0.030, -0.029)$	$(0.037, 0.038)$

Table 6
Summary of the MAE of ARMA (1,1) and ARMA (2,1) model one-step-ahead forecasting (1387 times) after prior denoising with LLSA and MODWT based on the number of jumps (i.e., the value of K) allowed.

	ARMA(1,1)						ARMA(2,1)					
	$K=1$		$K=2$		$K=3$		$K=1$		$K=2$		$K=3$	
	LLSA	MODWT	LLSA	MODWT	LLSA	MODWT	LLSA	MODWT	LLSA	MODWT	LLSA	MODWT
ADS	4.6480	4.6686	4.6336	4.6686	4.6521	4.6686	4.6605	4.6802	4.6463	4.6802	4.6678	4.6802
ALV	5.5858	5.5802	5.5592	5.5802	5.5344	5.5802	5.5914	5.5856	5.5634	5.5856	5.5366	5.5856
BAS	3.2772	3.2536	3.2277	3.2536	3.2150	3.2536	3.3228	3.2907	3.2689	3.2907	3.2557	3.2907
BAY	5.5455	5.5118	5.5284	5.5118	5.5213	5.5118	5.5544	5.5196	5.5354	5.5196	5.5284	5.5196
BEI	4.8267	4.8577	4.8253	4.8577	4.8154	4.8577	4.8346	4.8632	4.8318	4.8632	4.8211	4.8632
BMW	5.3135	5.3154	5.3064	5.3154	5.2897	5.3154	5.3277	5.3307	5.3225	5.3307	5.3079	5.3307
CBK	6.5162	6.5074	6.4548	6.5074	6.4670	6.5074	6.5199	6.5082	6.4616	6.5082	6.4751	6.5082
DAI	4.4306	4.4181	4.4341	4.4181	4.3993	4.4181	4.4450	4.4324	4.4541	4.4324	4.4185	4.4324
DBK	5.0453	5.0493	5.0156	5.0493	5.0036	5.0493	5.0662	5.0702	5.0349	5.0702	5.0243	5.0702
DB1	4.8311	4.8323	4.8133	4.8323	4.8197	4.8323	4.8345	4.8357	4.8173	4.8357	4.8250	4.8357
DPB	5.7655	5.7769	5.7766	5.7769	5.7720	5.7769	5.7870	5.7988	5.7978	5.7988	5.7937	5.7988
DPW	4.2808	4.2913	4.2821	4.2913	4.2488	4.2913	4.2877	4.2951	4.2891	4.2951	4.2584	4.2951
DTE	4.1490	4.1439	4.1344	4.1439	4.1188	4.1439	4.1417	4.1352	4.1276	4.1352	4.1102	4.1352
EPC	6.5854	6.4594	6.4799	6.4594	6.4219	6.4594	6.6043	6.4948	6.4943	6.4948	6.4333	6.4948
FME	5.3449	5.3830	5.3449	5.3830	5.3275	5.3830	5.3460	5.3839	5.3463	5.3839	5.3294	5.3839
LHA	5.5580	5.5626	5.5176	5.5626	5.5098	5.5626	5.5668	5.5725	5.5257	5.5725	5.5172	5.5725
HNK	4.5328	4.5526	4.5224	4.5526	4.5181	4.5526	4.5386	4.5590	4.5265	4.5590	4.5221	4.5590
IFX	6.1535	6.1283	6.1300	6.1283	6.0622	6.1283	6.1791	6.1496	6.1569	6.1496	6.0820	6.1496
SDF	10.784	10.787	10.887	10.787	10.942	10.787	10.841	10.818	10.934	10.818	10.981	10.818
LIN	5.1221	5.1392	5.1198	5.1392	5.0932	5.1392	5.1300	5.1460	5.1274	5.1460	5.0982	5.1460
MAN	4.5561	4.5460	4.4896	4.5460	4.4920	4.5460	4.5609	4.5522	4.4928	4.5522	4.4967	4.5522
MRC	4.6282	4.6142	4.6040	4.6142	4.5936	4.6142	4.6497	4.6431	4.6247	4.6431	4.6144	4.6431
MEO	4.1639	4.1553	4.1452	4.1553	4.1041	4.1553	4.1696	4.1637	4.1510	4.1637	4.1095	4.1637
MUV	5.4272	5.4321	5.4027	5.4321	5.4008	5.4321	5.4363	5.4441	5.4129	5.4441	5.4081	5.4441
RWE	4.3271	4.3530	4.3368	4.3530	4.3429	4.3530	4.3364	4.3606	4.3448	4.3606	4.3499	4.3606
SZG	7.4782	7.4401	7.3843	7.4401	7.4048	7.4401	7.4594	7.4139	7.3606	7.4139	7.3803	7.4139
SAP	5.4023	5.4308	5.4033	5.4308	5.3560	5.4308	5.4040	5.4360	5.3990	5.4360	5.3501	5.4360
SIE	4.4264	4.4356	4.4273	4.4356	4.3988	4.4356	4.4405	4.4511	4.4404	4.4511	4.4081	4.4511
TKA	6.0097	6.0066	5.9789	6.0066	5.9659	6.0066	6.0142	6.0081	5.9836	6.0081	5.9675	6.0081
VOW	29.335	29.075	29.042	29.075	28.897	29.075	30.139	30.078	29.912	30.078	29.702	30.078

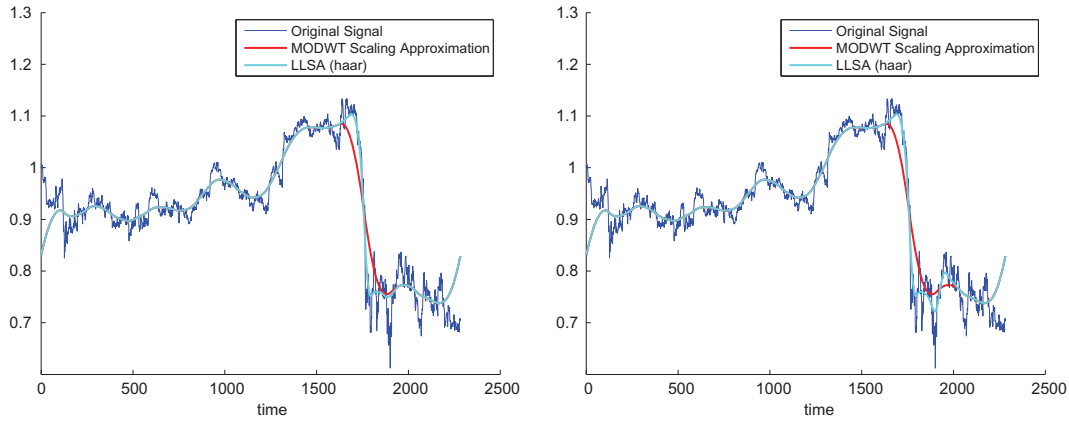


Fig. 5. LLSA(Haar) filtered trend of SAP.

we have LCB > 0 and UCB > 0, the MODWT is significantly better. In case of LCB< = 0 and UCB> = 0 there is no statistically significant conclusion. We report the 99% confidence intervals in Tables 3 and 5. The results show that the LLSA performs significantly better than MODWT in our simulation.

We conclude that in our simulation, the proposed algorithm LLSA has stable performance in every case, although it may be outperformed by the MODWT for certain choices of wavelets and extreme (i.e., very high and low) noise levels. In addition, LLSA remains stable and performs better, when we deviate K from the actual number of jumps in the signal. Based on these results, we conclude that the performance of LLSA is generally better by using the Haar and $D4$ wavelets.

5. Empirical study

In this empirical study, we investigate the LLSA performance in forecasting with the high-frequency data.

5.1. The empirical methodology

The analysis is conducted with high-frequency German DAX 30 component stock prices in 2008. In our empirical study, the homogeneous (i.e., equally spaced) time series data is aggregated at 60 minutes level using the linear interpolation method (suggested by Dacorogna et al. (2001)). In order to clarify the potential

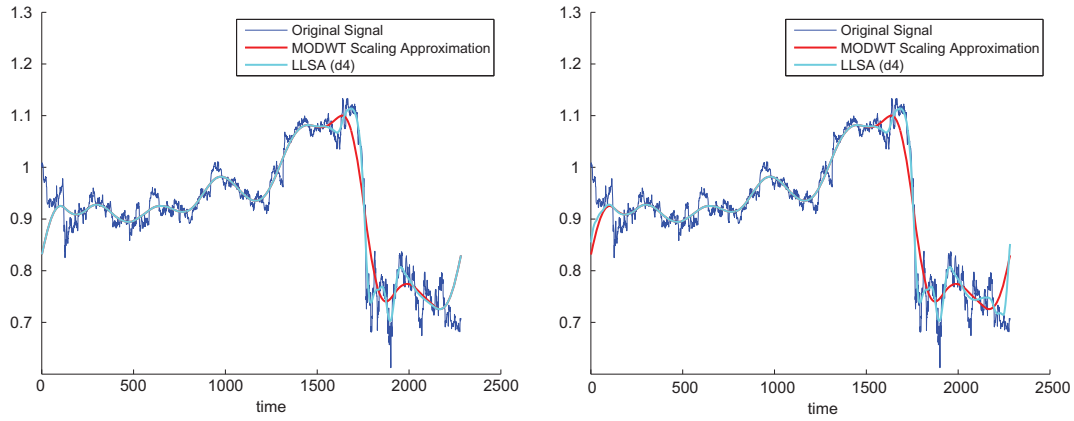


Fig. 6. LLSA($D4$) filtered trend of SAP.

Table 7
Conditional mean 99% bootstrap confidence intervals (LCB, UCB) for ARMA (1,1) and ARMA (2,1), given in units of 10^{-5}. UCB < 0 denotes a significant better performance of LLSA over MODWT, LCB > 0 a superior performance of the MODWT. For LCB < 0 and USB > 0 no significant statement can be made.

	ARMA(1,1)			ARMA(2,1)		
	$K=1$	$K=2$	$K=3$	$K=1$	$K=2$	$K=3$
ADS	(−4.1559, −0.0129)	(−6.3628, −0.6737)	(−5.3445, 2.0884)	(−4.0998, 0.1127)	(−6.3054, −0.4890)	(−5.0472, 2.4887)
ALV	(−2.5553, 3.6524)	(−5.6367, 1.4802)	(−8.9334, −0.2226)	(−2.6061, 3.7488)	(−5.8480, 1.4292)	(−9.3005, −0.4395)
BAS	(−7.0152, 13.313)	(−12.298, 9.2152)	(−13.991, 8.6170)	(−4.8147, 14.376)	(−10.590, 9.7261)	(−12.292, 9.2332)
BAY	(−0.4398, 7.1443)	(−2.6490, 5.9989)	(−4.3382, 6.1821)	(−4.7238, 7.4712)	(−2.9401, 6.0870)	(−4.6422, 6.3850)
BEI	(−5.7376, −4.5630)	(−6.8522, 0.4430)	(−8.0347, −0.3929)	(−5.4997, −0.2557)	(−6.7796, 0.5100)	(−7.9734, −0.3811)
BMW	(−2.4248, 2.0962)	(−4.2676, 2.5501)	(−6.2053, 1.1071)	(−2.5417, 1.9428)	(−4.1842, 2.5419)	(−5.9016, 1.3230)
CBK	(−1.7524, 3.5674)	(−8.6374, −1.8677)	(−8.2126, 0.0809)	(−1.5090, 3.8656)	(−8.0653, −1.1875)	(−7.5410, 0.9558)
DAI	(−1.2928, 3.7345)	(−1.8374, 5.0088)	(−5.7396, 1.9246)	(−1.4475, 3.9062)	(−1.4662, 5.7683)	(−5.4459, 2.6256)
DBK	(−3.2218, 2.3894)	(−6.7330, −0.0528)	(−8.1058, −1.0216)	(−3.3966, 2.5608)	(−7.0426, −0.0175)	(−8.3526, −0.8725)
DB1	(−1.9851, 1.7063)	(−5.1366, 1.3334)	(−5.3390, 2.8307)	(−2.0430, 1.7407)	(−5.1730, 1.4539)	(−5.2396, 3.1382)
DPB	(−4.2457, 1.9140)	(−3.7751, 3.7234)	(−5.1885, 4.1175)	(−4.4992, 2.1204)	(−4.1590, 3.9322)	(−5.4676, 4.4875)
DPW	(−3.8453, 1.6935)	(−4.8438, 2.9342)	(−8.2517, −0.1902)	(−3.3964, 1.9420)	(−4.3543, 3.1612)	(−7.5475, 0.2346)
DTE	(−1.0307, 2.0793)	(−3.4109, 1.5143)	(−5.3948, 0.4335)	(−0.9184, 2.2349)	(−3.2110, 1.7045)	(−5.4146, 0.3785)
EPC	(5.4875, 20.219)	(−6.4562, 10.899)	(−13.599, 6.6042)	(3.6915, 18.506)	(−8.8558, 8.9746)	(−16.309, 4.5508)
FME	(−5.8829, −1.7668)	(−7.0977, −0.4984)	(−9.7120, −1.4143)	(−5.8902, −1.7132)	(−7.0959, −0.4826)	(−9.5711, −1.3951)
LHA	(−3.2310, 2.2760)	(−7.6565, −1.2670)	(−8.8052, −1.7291)	(−3.3434, 2.1522)	(−7.8972, −1.4507)	(−9.1173, −1.9452)
HNK	(−3.8802, −0.0368)	(−5.9203, −0.1271)	(−7.1299, 0.2661)	(−3.9804, −0.0754)	(−6.1556, −0.3205)	(−7.3862, 0.0375)
IFX	(−0.6089, 5.6608)	(−4.3721, 4.7448)	(−11.906, −1.3058)	(−0.4062, 6.2602)	(−4.1231, 5.6460)	(−12.320, −1.1782)
SDF	(−21.120, 24.278)	(−13.111, 39.320)	(−11.921, 49.759)	(−18.665, 27.602)	(−12.132, 40.102)	(−10.760, 50.347)
LIN	(−4.8114, 1.4089)	(−5.7122, 1.8089)	(−8.7956, −0.4148)	(−4.6931, 1.5355)	(−5.6252, 1.9267)	(−8.9728, −0.6274)
MAN	(−1.6058, 3.6291)	(−8.9263, −2.3597)	(−9.1908, −1.6247)	(−1.8623, 3.6030)	(−9.3241, −2.5296)	(−9.4367, −1.6433)
MRC	(−1.6093, 4.4479)	(−4.4113, 2.3793)	(−6.2323, 2.1227)	(−2.3764, 3.6891)	(−5.2637, 1.6041)	(−7.0719, 1.3195)
MEO	(−1.4676, 3.1336)	(−4.1664, 2.0853)	(−8.7598, −1.5892)	(−1.7690, 2.8881)	(−4.4834, 1.8897)	(−9.1329, −1.8095)
MUV	(−3.2201, 2.2138)	(−6.7602, 0.8248)	(−7.6464, 1.3664)	(−3.6504, 2.0787)	(−7.1435, 0.8677)	(−8.2854, 1.1084)
RWE	(−4.7550, −0.4468)	(−4.8356, 1.5472)	(−5.4034, 3.3013)	(−4.6124, −0.2574)	(−4.8032, 1.5742)	(−5.3647, 3.2107)
SZG	(−1.1580, 8.8742)	(−11.558, 0.4498)	(−11.316, 4.2228)	(−0.8008, 9.8409)	(−11.703, 1.0263)	(−11.750, 5.0011)
SAP	(−8.2440, 2.6255)	(−8.8755, 3.3352)	(−14.205, −0.7066)	(−9.1525, 2.7121)	(−10.312, 2.7723)	(−15.771, −1.5082)
SIE	(−3.9026, 2.1138)	(−4.6889, 3.0838)	(−8.3742, 1.1349)	(−4.0709, 1.9435)	(−4.9203, 2.7864)	(−9.0410, 0.4776)
TKA	(−3.1074, 3.6934)	(−7.7639, 2.1766)	(−10.163, 1.8512)	(−2.9578, 4.1669)	(−7.6007, 2.7090)	(−10.254, 2.2145)
VOW	(−24.172, 68.169)	(−59.620, 41.000)	(−82.940, 34.982)	(−62.028, 49.061)	(−89.520, 29.192)	(−127.13, 17.698)

economic implementation of our algorithm, we apply LLSA and MODWT in forecasting with this data set.

We first apply LLSA and MODWT for denoising the real data. Since the choice of wavelet does not influence the performance of MODWT, in our empirical study we choose the Haar wavelet. It is the wavelet with the smallest support and generates the smallest regions around the jumps, which must be considered as the output from a nonlinear filter. It will lead to less ripples in the whole trend. In addition, the small support minimizes the boundary distortions (see Percival and Walden (2006)).

For the 60 minute data we select an approximation level $J = 7$ providing a relatively smooth trend of the data with relatively few ripples. This choice could be adjusted based on particular requirements for practical issues. The extracted trend is then associated with a weighted average of bandwidth of $2^7 \cdot 60$ minutes. We set $\Lambda = 2$, which can improve the performance of MODWT by avoiding excessive ripples near any occurring jumps. The algorithm is configured to detect from one jump only up to three, i.e., $K \in \{1,2,3\}$. Since we did not specify the combination of parameters depending on each time series separately, this choice of (Λ, K) cannot be optimal for all stocks at the same time. In practice, it should be chosen individually for each stock data.

In order to increase the statistical significance, the filters are applied on the whole time series not only once, but successively in a moving window with length of $\mathcal{X}$ times the filter size, i.e., $\mathcal{X} \cdot 2^J$ and $\mathcal{X} = 7$, which provides a sufficiently large number of subsamples. We do not consider the first and last 2^J data points for each estimated trend in the moving window in order to avoid the boundary

effect of signal (see Percival and Walden (2006)). The effective filtering window size then equals to $5 \cdot 2^J$. To compare the performance of these two algorithms, ARMA(1,1) and ARMA(2,1) models are used on the detrended series to calculate the conditional mean for the one-step-ahead forecasting.[2] The performance is evaluated by computing the mean absolute error (MAE) of the forecasted values comparing to the true values observed in the original data.

5.2. Empirical results for forecasting

The mean absolute errors (MAE) of the forecasting performance when $K \in \{1,2,3\}$ are reported in Table 6. Increasing value of K means that more jumps are captured by the algorithm. From Table 6, we can compare the denoising performance of LLSA and MODWT based on the average value of MAE. The smaller the average value, the better the performance the underlying algorithm has in forecasting. We can see that (1) for the ARMA(1,1) model, when $K = 1$, $K = 2$, and $K = 3$, there are 16, 25 and 28, respectively, out of total 30 cases, showing that the denoising performance of LLSA is better than MODWT, and (2) for the ARMA(2,1) model, when $K = 1$, $K = 2$, and $K = 3$, 15, 26 and 28 cases, respectively, showing that LLSA is better than the MODWT. Figs. 5 and 6 illustrate the example of the performance of LLSA when denoising the stock price data of SAP with Haar and $D4$ respectively, and we can easily identify that LLSA performs better than MODWT.

In order to confirm this observation, we run significant tests based on the same bootstrapping method described in Section 4 and report the results in Table 7. Based on the result, we find that: (1) when $K = 1$, for ARMA (1,1) model, 5 cases from the whole 30 samples confirm that the denoising performance of LLSA is significantly better than MODWT, while only one case confirms the MODWT has a better denoising performance, and for ARMA(2,1) model, 4 cases confirm LLSA is better while 1 confirms MODWT is better in denoising performance ; (2) when $K = 2$, for both ARMA(1,1) and ARMA(2,1) model, there is no significant evidence confirming that MODWT performs better than that of LLSA while 7 cases of the sample significantly confirm that LLSA is superior than MODWT in the denoising performance; and (3) when $K = 3$, for both ARMA(1,1) and ARMA(2,1) model, there is no significant evidence confirming that MODWT performs better than LLSA while 11 and 10 cases of the sample significantly confirm that LLSA is superior than MODWT respectively. As Sun et al (2008) and the reference therein point out that the underlying significance test based on bootstrapping can be applied effectively to confirm the goodness-of-fit test (the average MAE in our study) undertaken, we can then conclude that in our empirical study, the denoising performance of LLSA is generally better than that of the MODWT.

Our findings confirm that the proposed LLSA algorithm is able to significantly outperform MODWT when analyzing the high-frequency financial data. In this study we did not even calibrate the input parameters (i.e., neither K nor Λ) specifically for every stock data, but we recommend to do it in practice (either by a priori or a posteriori analysis). In our empirical study setting K to a larger value does not necessarily improve the algorithm's performance on specific stocks (see, for example, ADS, BEI, CBK, HNK, and RWE for different K). In order to determine the optimal value of K, several works in the literature suggest different methodologies for jump detection, see, for example, Wang (1998), Sun and Qiu (2007), Joo and Qiu (2009).

[2] Since financial data illustrates auto-correlation, we run the ARMA(2,1) model after applying ARMA(1,1). Based on our empirical results, the performance of LLSA and MODWT are independent of the choice these models.

6. Conclusions

In this paper, we propose a new long-term trend extraction method named local linear scaling approximation (LLSA), which is a nonlinear filtering technique based on the linear maximal overlap discrete wavelet transform (MODWT). When applying it with the high-frequency financial data, this algorithm is able to preserve a smooth trend and accurately displays any occurring jumps. This is realized by reconstructing lost details of jumps whose information is still contained in the wavelet multiresolution decomposition, that is, the initial filtering procedure.

We define LLSA's requirements and its input parameters and prove analytically its consistency and local linear properties that are useful in practice for controlling the filter output's frequencies. We also define the LLSA's impulse and step response function that can characterize its nonlinear filtering property. Since the new algorithm has the same computational complexity as MODWT, it can be applied in practice without any further restrictions.

In order to show the reliability of LLSA, we first conduct an experiment based on simulation. In all simulation settings our algorithm illustrates its robustness independent of the wavelet and other input parameters. For the non-extreme simulation settings (i.e., high energy noise), LLSA performs better than the MODWT filter, which is used as an alternative method representing pure linear filtering methods. We then empirically show the potential application of LLSA by forecasting the high-frequency data (aggregated at the 60 minute-frequency level) of the German DAX 30 component stock price data. The results given by the statistical test based on the bootstrapping method confirm the significance of our conclusion.

Based on the analytically proved properties and the results of the simulation and empirical study, we conclude that LLSA is a robust algorithm which enriches the class of nonlinear filtering methods in data mining. The proposed algorithm is expected to provide a significant improvement of the accuracy of, for example, volatility modeling and forecasting in high-frequency financial time series, when the underlying trend contains jump components. Since denoising is fundamental for time series analysis, performing this task efficiently will help us to avoid the erroneous conclusion.

Acknowledgments

The authors thank I. Bomze, the editor, and two anonymous referees for their insights and valuable comments. We are also grateful to Pedro A. Morettin and Aluísio de Souza Pinheiro for their valuable suggestions. Meinl's research was partially supported and funded by the Karlsruhe House of Young Scientists (KHYS) and the German Ministry of Education and Research (BMBF).

References

Au, S., Duan, R., Hesar, S., Jiang, W., 2010. A framework of irregularity enlightenment for data pre-processing in data mining. Annals of Operations Research 174 (1), 47–66.

Bishop, C., 2006. Pattern Recognition and Machine Learning, Vol. 4. Springer, New York.

Crowley, P., 2007. A guide to wavelets for economists. Journal of Economic Surveys 21 (2), 207–267.

Dacorogna, M., Gençay, R., Mueller, U., Olsen, R., Pictet, O., 2001. An Introduction to High-frequency Finance. Academic Press.

Daubechies, I., 1992. Ten Lectures on Wavelets. Society for Industrial Mathematics, Philadelphia.

Engle, R., 2000. The econometrics of ultra-high-frequency data. Econometrica 68 (1), 1–22.

Fan, J., Wang, Y., 2007. Multi-scale jump and volatility analysis for high-frequency financial data. Journal of the American Statistical Association 102 (480), 1349–1362.

Gençay, R., Gradojevic, N., 2011. Errors-in-variables estimation with wavelets. Journal of Statistical Computation and Simulation 99999 (1), 1–20.

Gençay, R., Selçuk, F., Whitcher, B., 2002. An Introduction to Wavelets and Other Filtering Methods in Finance and Economics. Academic Press.
Gençay, R., Selçuk, F., Whitcher, B., 2003. Systematic risk and timescales. Quantitative Finance 3 (2), 108–116.
Gençay, R., Selçuk, F., Whitcher, B., 2005. Multiscale systematic risk. Journal of International Money and Finance 24 (1), 55–70.
Ghysels, E., 2000. Some econometric recipes for high-frequency data cooking. Journal of Business and Economic Statistics 18 (2), 154–163.
Gijbels, I., Lambert, A., Qiu, P., 2007. Jump-preserving regression and smoothing using local linear fitting: A compromise. Annals of the Institute of Statistical Mathematics 59 (2), 235–272.
Joo, J., Qiu, P., 2009. Jump detection in a regression curve and its derivative. Technometrics 51 (3), 289–305.
Keinert, F., 2004. Wavelets and Multiwavelets. Chapman & Hall/CRC.
Lada, E., Wilson, J., 2006. A wavelet-based spectral procedure for steady-state simulation analysis. European Journal of Operational Research 174 (3), 1769–1801.
Laukaitis, A., 2008. Functional data analysis for cash flow and transactions intensity continuous-time prediction using Hilbert-valued autoregressive processes. European Journal of Operational Research 185 (3), 1607–1614.
Mallat, S., 1989. A theory for multiresolution signal decomposition: The wavelet representation. IEEE Transactions on Pattern Analysis and Machine Intelligence 11 (7), 674–693.
Mallat, S., Hwang, W., 1992. Singularity detection and processing with wavelets. IEEE Transactions on Information Theory 38 (2), 617–643.
Meinl, T., Sun, E. 2010. A Nonlinear Filtering Algorithm based on Wavelet Transforms for High-Frequency Financial Data Analysis. Technical report, Karlsruhe Institute of Technology, Germany. Presentation at the Fourth Italian Econometrics Conference.
Percival, D., Walden, A., 2006. Wavelet Methods for Time Series Analysis. Cambridge University Press.
Qiu, P., 2003. A jump-preserving curve fitting procedure based on local piecewise-linear kernel estimation. Journal of Nonparametric Statistics 15 (4), 437–453.
Qiu, P., Yandell, B., 1998. A local polynomial jump-detection algorithm in nonparametric regression. Technometrics 40 (2), 141–152.
Ramsey, J., 2002. Wavelets in economics and finance: Past and future. Studies in Nonlinear Dynamics and Econometrics 6 (3), 1–27.
Sun, E., Rezania, O., Rachev, S., Fabozzi, F., 2011. Analysis of the intraday effects of economic releases on the currency market. Journal of International Money and Finance 30 (4), 692–707.
Sun, J., Qiu, P., 2007. Jump detection in regression surfaces using both first-order and second-order derivatives. Journal of Computational and Graphical Statistics 16 (2), 289–311.
Sun, W., Rachev, S., Fabozzi, F., 2007. Fractals or IID: Evidence of long-range dependence and heavy tailedness from modeling German equity market returns. Journal of Economics and Business 59 (6), 575–595.
Sun, W., Rachev, S., Fabozzi, F., Kalev, P., 2008. Fractals in trade duration: Capturing long-range dependence and heavy tailedness in modeling trade duration. Annals of Finance 4 (2), 217–241.
Wang, Y., 1995. Jump and sharp cusp detection by wavelets. Biometrika 82 (2), 385–397.
Wang, Y., 1998. Change curve estimation via wavelets. Journal of the American Statistical Association 93 (441), 163–172.