Non-linear Adaptive Wiener Filter for Time Series Smoothing

Alexandre L. M. Levada

Computing Department, Federal University of São Carlos, Rod. Washington Luis, km. 235, São Carlos, 13565-905, SP, Brazil.

Contributing authors: alexandre.levada@ufscar.br;

Abstract

Often, in the analysis of time series and digital signals, a smoothing procedure is required to filter undesired random perturbations as noise and outliers in data. Among the most widely known techniques for time series smoothing, we have convolutional filters, simple exponential smoothing, triple exponential smoothing (Holt-Winters method) and linear adaptive filters, such as the Wiener filter. In this paper, we propose the NoLAW filter (*Non-Linear Adaptive Wiener filter*), a higher-order non-linear extension for the adaptive Wiener filter, which is a linear statistical smoothing technique that assumes the hypothesis that the underlying series is corrupted by a zero mean, additive and independent Gaussian noise. Numerical experiments show that the proposed method is a computationally efficient and viable approach for filtering time series. Quantitative metrics show that the NoLAW filter is capable of producing better results than the usual linear Wiener filter, simple exponential smoothing and Holt-Winters method. Moreover, the computational cost of the proposed NoLAW is linear in the number of samples, which means that, asymptotically, it is equivalent to the regular Wiener filter.

Keywords: Time series, smoothing, non-linear filtering, cubic regression, denoising

1 Introduction

From basic low-level feature extraction tasks to forecasting applications, time series filtering is a critical pre-processing step in a variety of statistical and data

science procedures [1]. As a result, being able to determine the correct underlying signal from a degraded corrupted version is a topic that many researchers in several domains of science are interested in learning [2]. In summary, the main purpose of time series smoothing is to eliminate or attenuate random disturbances in data while preserving as much useful signal information as possible [3]. From a statistical perspective, smoothing consists of removing fine-grained variance between time steps in order to reduce noise and reveal the signal of the underlying causal processes [4].

One of the most significant issues in smoothing is that general time series are usually non-stationary, requiring a locally adaptive filtering approach [5]. In time series smoothing and forecasting, common linear filters with convolutional kernels (Finite Impulse Response filter), such as weighted moving averages, are a simple and extensively used approach. However, because to their inability to adjust to local statistics, they frequently oversmooth the signal, destroying relevant information [6].

Minimum mean squared error filters (MMSE) are an important family of adaptive filters widely studied and employed with success in applied statistics and signal processing. Two classic methods that belong to this family are the Wiener filter [7] and the Kalman filter [8], with both of them being optimal when the noises/errors have a Gaussian distribution. In terms of computational efficiency, the linear pointwise adaptive Wiener filter is recognizable as one of the best approaches for signal processing and time series smoothing [9]. However, one potential drawback of these filters is the appearance of undesirable artifacts around abrupt discontinuities after smoothing, since the assumption that samples inside a local window come from the same ensemble becomes unrealistic if there are sharp fluctuations within that local window.

Exponential smoothing defines another kind of low-pass filters (Infinite Impulse Response filter) regularly employed to smooth data in signal processing, in order to remove high-frequency noise. Triple or third-order exponential smoothing (Holt Winters method) treats different types of seasonality in data: one additive and another multiplicative in nature. Briefly speaking, these filters compute a noise-free estimative using a weighted sum of past observations with a exponentially decreasing weight function [10, 11].

Recently, with advances in computer science and machine learning, deep neural networks have been used in time series smoothing and forecasting [13–15]. Recurrent neural networks (RNN's) are computational tools that control a internal state to analyze sequences of inputs with variable lenghts [16] and due to this characteristic are known to have memory. Among them, two kinds of networks play important roles: Long Short-Term Memory (LSTM) networks [18] and Convolutional Neural Networks (CNN's) [17]. Roughly speaking, in a time series and signal processing perspective, the term recurrent neural network refer to the category of networks with an infinite impulse response filters (IIR), whereas convolutional neural network refers to structures that uses finite impulse response filters (FIR). The main problem with deep neural networks are: 1) the computational cost is high as we have to train these models

using numerical methods such as stochastic gradient descent; 2) they usually require a large number of samples for convergence to good results, which is not always possible; 3) many researchers ague that deep neural networks lack interpretability, that is, they work as black boxes and often it is hard to clearly explain mathematically why and how the results are obtained.

In this paper, we generalize the linear adaptive Wiener filter by proposing the NoLAW filter (Non-Linear Adaptive Wiener filter) as a higher-order extension that recursively uses lower order filters to compute smooth approximations for the underlying signal in the estimation of the local statistics. The main contribution of the proposed method is twofold: 1) as the NoLAW filter is a statistical approach and not a deep learning technique, it works well when the number of samples is small or large; 2) besides improving the performance of the original Wiener filter, the NoLAW filter is a computationally efficient algorithm for time series smoothing, as its complexity is linear in the number of samples (equivalent to the original Wiener filter).

The remaining of the paper is organized as follows: Section 2 presents the mathematical derivation of the original linear adaptive Wiener filter in details by solving a least square problem. Section 3 proposes the NoLAW filter, which is a non-linear generalization of original where the noise-free sample is estimated as a cubic function of the observation. Section 4 presents the complexity analysis of the proposed NoLAW filter. Section 5 shows the computational experiments and the obtained results. Finally, Section 6 presents our conclusions and final remarks.

2 Adaptive Linear Wiener Filtering

The formulation of the adaptive linear Wiener filter is given in statistical terms, where the main goal is to find, for each time step of the series, the optimum linear estimative for a sample of the signal corrupted by a zero mean, additive and uncorrelated Gaussian noise. In mathematical terms, we have:

$$f(n) = s(n) + e(n) \tag{1}$$

where $f(n)$ denotes the noisy observation, $s(n)$ denotes the desired underlying noise-free observation and $e(n)$ is a Gaussian random variable with distribution $N(0, \sigma_n^2)$, given by:

$$p(e(n);\ \mu, \sigma_n^2) = \frac{1}{\sqrt{2\pi\sigma_n^2}} exp\left\{-\frac{e(n)^2}{2\sigma_n^2}\right\} \tag{2}$$

Knowing that we want the optimal linear estimator $\hat{s}(n)$, we have:

$$\hat{s}(n) = \alpha f(n) + \beta \tag{3}$$

where α is the slope and β is the intercept. By the initial hypothesis, we know that the signal and the noise are uncorrelated and by the linearity of the expectation, we have:

$$E[f(n)] = E[s(n)] \quad \rightarrow \quad \mu_f = \mu_s \tag{4}$$
$$Var[f(n)] = Var[s(n)] + Var[e(n)] \quad \rightarrow \quad \sigma_f^2 = \sigma_s^2 + \sigma_n^2 \tag{5}$$

Using least squares, it is possible to estimate the optimal parameters of the linear estimators. The objective function is given by:

$$J(\alpha, \beta) = E\left[(\hat{s}(n) - s(n))^2\right] = E\left[(\alpha f(n) + \beta - s(n))^2\right] \tag{6}$$

Differentiating with respect to α and β and setting the result to zero leads to:

$$\frac{\partial}{\partial \alpha} J(\alpha, \beta) = E\left[(\alpha f(n) + \beta - s(n))\, f(n)\right] = 0 \tag{7}$$

$$\frac{\partial}{\partial \beta} J(\alpha, \beta) = E\left[(\alpha f(n) + \beta - s(n))\right] = 0 \tag{8}$$

By solving equation (7), we can write:

$$E\left[\alpha f(n)^2 + \beta f(n) - s(n) f(n)\right] = 0 \tag{9}$$
$$\alpha E\left[f(n)^2\right] + \beta E[f(n)] - E[s(n) f(n)] = 0 \tag{10}$$
$$\alpha E\left[f(n)^2\right] + \beta E[f(n)] - E[s(n)^2] - E[s(n) e(n)] = 0 \tag{11}$$

As the signal and the noise are uncorrelated, $E[s(n)e(n)] = 0$. Moreover, it is known that $Var[x] = E[x^2] - E^2[x]$, which leads to:

$$\alpha(\sigma_f^2 + \mu_f^2) + \beta \mu_f - (\sigma_s^2 + \mu_s^2) = 0 \tag{12}$$

Doing the same for equation (8), we have:

$$\alpha E[f(n)] + \beta - E[s(n)] = 0 \tag{13}$$
$$\alpha \mu_f + \beta - \mu_s = 0 \tag{14}$$
$$\alpha \mu_f + \beta - \mu_f = 0 \tag{15}$$
$$\tag{16}$$

which leads to:

$$\beta = (1 - \alpha)\mu_f \tag{17}$$

Plugging β into equation (7), we have:

$$\alpha(\sigma_f^2 + \mu_f^2) + (1-\alpha)\mu_f^2 - \sigma_s^2 - \mu_s^2 = 0 \tag{18}$$

which leads to:

$$\alpha = \frac{\sigma_s^2}{\sigma_f^2} = \frac{\sigma_s^2}{\sigma_s^2 + \sigma_n^2} \tag{19}$$

Therefore, the final linear least squares estimator (adaptive Wiener filter) is a convex combination between the observation in the current time step and the local mean, given by:

$$\hat{s}(n) = \frac{\sigma_s^2}{\sigma_s^2 + \sigma_n^2} f(n) + \left(1 - \frac{\sigma_s^2}{\sigma_s^2 + \sigma_n^2}\right) \mu_f \tag{20}$$

$$= \mu_f + \frac{\sigma_s^2}{\sigma_s^2 + \sigma_n^2}(f(n) - \mu_f) \tag{21}$$

where $\sigma_s^2 = max\{0, \sigma_f^2 - \sigma_n^2\}$. In practice, in the non-causal filter, the mean μ_f and the variance σ_f^2 are estimated in a local window around the current observation. It is possible to adopt a causal neighborhood to use only the current and past samples. A neighborhood of order n is comprised by $2n+1$ samples, being the current one plus the $2n$ previous samples. Figure 1 illustrates second, third and fourth order causal and non-causal neighborhood systems.

Fig. 1 Second, third and fourth causal and non-causal neighborhood systems.

3 NoLAW: Non-linear Adaptive Wiener Filter

For simplicity, we begin by considering the quadratic case, which is the second-order NoLAW filter. In this case, the idea is to have the estimator of the noise-free value as a quadratic function of the observation:

$$\hat{s}(n) = \alpha f(n)^2 + \beta f(n) + \gamma \tag{22}$$

Hence, the least squares problem consists in minimizing the following function of the parameter models:

$$J(\alpha, \beta, \gamma) = E\left[\left(\alpha f(n)^2 + \beta f(n) + \gamma - s(n)\right)^2\right] \tag{23}$$

Differentiating the objective funciton with respect to α and setting the result to zero leads to the first equation:

$$\frac{\partial}{\partial \alpha} J(\alpha, \beta, \gamma) = E\left[\left(\alpha f(n)^2 + \beta f(n) + \gamma - s(n)\right) f(n)^2\right] = 0 \quad \rightarrow \tag{24}$$

$$\alpha E\left[f(n)^4\right] + \beta E\left[f(n)^3\right] + \gamma E\left[f(n)^2\right] = E\left[f(n)^2 s(n)\right] \tag{25}$$

Similarly, differentiating the objective funciton with respect to β and setting the result to zero leads to the second equation:

$$\frac{\partial}{\partial \beta} J(\alpha, \beta, \gamma) = E\left[\left(\alpha f(n)^2 + \beta f(n) + \gamma - s(n)\right) f(n)\right] = 0 \quad \rightarrow \tag{26}$$

$$\alpha E\left[f(n)^3\right] + \beta E\left[f(n)^2\right] + \gamma E\left[f(n)\right] = E\left[f(n) s(n)\right] \tag{27}$$

Similarly, differentiating the objective funciton with respect to γ and setting the result to zero leads to the third equation:

$$\frac{\partial}{\partial \gamma} J(\alpha, \beta, \gamma) = E\left[\left(\alpha f(n)^2 + \beta f(n) + \gamma - s(n)\right)\right] = 0 \quad \rightarrow \tag{28}$$

$$\alpha E\left[f(n)^2\right] + \beta E\left[f(n)\right] + \gamma = E\left[s(n)\right] \tag{29}$$

Using the sample moments to approximate the population moments, we can write the least squares problem as the following system of equations:

$$\begin{bmatrix} \frac{1}{n_i}\sum_{n\in\eta_i} f(n)^4 & \frac{1}{n_i}\sum_{n\in\eta_i} f(n)^3 & \frac{1}{n_i}\sum_{n\in\eta_i} f(n)^2 \\ \frac{1}{n_i}\sum_{n\in\eta_i} f(n)^3 & \frac{1}{n_i}\sum_{n\in\eta_i} f(n)^2 & \frac{1}{n_i}\sum_{n\in\eta_i} f(n) \\ \frac{1}{n_i}\sum_{n\in\eta_i} f(n)^2 & \frac{1}{n_i}\sum_{n\in\eta_i} f(n) & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} = \begin{bmatrix} \frac{1}{n_i}\sum_{n\in\eta_i} f(n)^2 s(n) \\ \frac{1}{n_i}\sum_{n\in\eta_i} f(n) s(n) \\ \frac{1}{n_i}\sum_{n\in\eta_i} s(n) \end{bmatrix} \tag{30}$$

where η_i denotes the neighborhood of the i-th sample and n_i is the cardinality of this set (number of samples in the local window). We use a linear Wiener filter to smooth the the observed time series and approximate the underlying signal $s(n)$ in order to compute the cross-correlations. Note that using the same

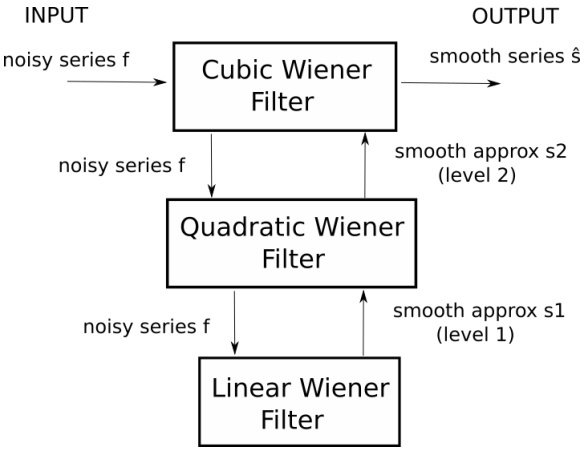

Fig. 2 Block diagram for the proposed NoLAW filtering strategy for time series smoothing.

strategy, it is possible to define a third-order Wiener filter (cubic function of the observation). For this, the estimation of the noise-free sample is given by a polynomial of degree 3:

$$\hat{s}(n) = \alpha f(n)^3 + \beta f(n)^2 + \gamma f(n) + \lambda \tag{31}$$

It is straightforward to see that the solution of the least squares problems is given by the following system of equations:

$$\begin{bmatrix} \frac{1}{n_i}\sum_{n\in\eta_i} f(n)^6 & \frac{1}{n_i}\sum_{n\in\eta_i} f(n)^5 & \frac{1}{n_i}\sum_{n\in\eta_i} f(n)^4 & \frac{1}{n_i}\sum_{n\in\eta_i} f(n)^3 \\ \frac{1}{n_i}\sum_{n\in\eta_i} f(n)^5 & \frac{1}{n_i}\sum_{n\in\eta_i} f(n)^4 & \frac{1}{n_i}\sum_{n\in\eta_i} f(n)^3 & \frac{1}{n_i}\sum_{n\in\eta_i} f(n)^2 \\ \frac{1}{n_i}\sum_{n\in\eta_i} f(n)^4 & \frac{1}{n_i}\sum_{n\in\eta_i} f(n)^3 & \frac{1}{n_i}\sum_{n\in\eta_i} f(n)^2 & \frac{1}{n_i}\sum_{n\in\eta_i} f(n) \\ \frac{1}{n_i}\sum_{n\in\eta_i} f(n)^3 & \frac{1}{n_i}\sum_{n\in\eta_i} f(n)^2 & \frac{1}{n_i}\sum_{n\in\eta_i} f(n) & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ \gamma \\ \lambda \end{bmatrix} = \begin{bmatrix} \frac{1}{n_i}\sum_{n\in\eta_i} f(n)^3 s(n) \\ \frac{1}{n_i}\sum_{n\in\eta_i} f(n)^2 s(n) \\ \frac{1}{n_i}\sum_{n\in\eta_i} f(n) s(n) \\ \frac{1}{n_i}\sum_{n\in\eta_i} s(n) \end{bmatrix} \tag{32}$$

Note that all statistics involved in the non-linear Wiener filter can be directly computed from a local window in the observed time series $f(n)$ and a local window in a pre-smoothed version of it. A reasonable and intuitive choice for the application of higher-order Wiener filters is to apply a cascading strategy, that is, the linear Wiener filter is used to build the smooth approximation for the quadratic Wiener filter. Then the quadratic Wiener filter can be employed to build the approximation for the cubic Wiener filter, and so on. Figure 2 shows a block diagram illustrating the process of the third-order adaptive Wiener filtering. The general case in which the estimator is a polynomial function of degree K of the observation, we have:

$$\hat{s}(n) = \alpha_K f(n)^K + \alpha_{K-1} f(n)^{K-1} + \alpha_{K-2} f(n)^{K-2} + \ldots + \alpha_0 \tag{33}$$

where the vector $\vec{\alpha} = [\alpha_0, \alpha_1, \alpha_2, ..., \alpha_K]$ stores the coefficients of the desired non-linear estimator. In this case, the system of equations for the non-linear Wiener filter of order k is represented by a $(k+1) \times (k+1)$ matrix:

$$\begin{bmatrix} \frac{1}{n_i}\sum_{n\in\eta_i} f(n)^{2K} & \frac{1}{n_i}\sum_{n\in\eta_i} f(n)^{2K-1} & \ldots & \frac{1}{n_i}\sum_{n\in\eta_i} f(n)^{K} \\ \frac{1}{n_i}\sum_{n\in\eta_i} f(n)^{2K-1} & \frac{1}{n_i}\sum_{n\in\eta_i} f(n)^{2K-2} & \ldots & \frac{1}{n_i}\sum_{n\in\eta_i} f(n)^{K-1} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{n_i}\sum_{n\in\eta_i} f(n)^{K} & \frac{1}{n_i}\sum_{n\in\eta_i} f(n)^{K-1} & \ldots & 1 \end{bmatrix} \begin{bmatrix} \alpha_K \\ \alpha_{K-1} \\ \vdots \\ \alpha_1 \\ \alpha_0 \end{bmatrix} = \begin{bmatrix} \frac{1}{n_i}\sum_{n\in\eta_i} f(n)^{K-1} s(n) \\ \frac{1}{n_i}\sum_{n\in\eta_i} f(n)^{K-2} s(n) \\ \vdots \\ \frac{1}{n_i}\sum_{n\in\eta_i} s(n) \end{bmatrix} \tag{34}$$

For some points of the series, the matrix defined by the system of equations can be ill-conditioned and some regularization may be required. Often, a small positive increment along its main diagonal is enough to overcome this problem. In this work, we add $\delta = 0.001$ to all elements of the main diagonal. The algorithm for the proposed NoLAW filter is presented in the following. The recursive function NoLAW has four parameters: ts (the time series), k (the order of the filter, i.e., if $k = 3$ we have the cubic filter), w_size (the size of the local windows) and σ_n^2 (the variance of the noise).

In order to better understand why the computational cost of the proposed NoLAW filter is equivalent to the computational cost of the regular Wiener filter, we must compute the complexity of the NoLAW filter using asymptotic analysis with the Big-O notation. In the next section, we provide a detailed discussion about this issue.

Algorithm 1 Non-Linear Adaptive Wiener Filter

```
1: function NoLAW(ts, k, w_size, σ_n^2)
2:     if k = 1 then
3:         smooth = LinearWiener(ts, w_size, σ_n^2)
4:         return smooth
5:     else
6:         smooth = NoLAW(ts, k − 1, w_size, σ_n^2)
7:         n = len(ts)
8:         for i = 1 to n do
9:             Build the matrix of moments M using equation (34)
10:            Build the vector of cross-covariances c⃗ using equation (34)
11:            Regularize the matrix M (add a small value to its main diagonal)
12:            Solve the linear system Mα⃗ = c⃗
13:            for j = 0 to k do
14:                filtered[i] += α[j] ∗ ts[i]^j
15:            end for
16:        end for
17:        return filtered
18:    end if
19: end function
```

4 Complexity Analysis

In order to analyze the complexity of the proposed NoLAW filter, first note that the *LinearWiener* function has complexity $O(w\_size \times n)$, where w_size is the size of the local windows and n in the length of the time series. Recall that, as the linear Wiener filter has a closed-form solution, it enables direct computation of the noise-free estimative in $O(w\_size)$ for each sample of the time series. The *LinearWiener* function is called only one time, when $k = 1$. For values of k ranging from 2 to the order of the NoLAW filter, the *else* command block is executed.

Now, we focus in the analysis of the *for* loop (lines 8 to 16). Note that this loop is executed n times, one for each sample. First, we have to build the squared matrix M, which is $(k+1) \times (k+1)$. Recall that each sample moment requires the summation of w_size samples and the matrix M has $(k+1)^2$ elements. Note that to simplify a little bit further our calculations, for large values of i, we have $i + 1 \approx i$. Knowing that by recursion these processes are executed for $i = 2, 3, ..., k$, we have to solve the following summation:

$$\sum_{i=2}^{k} (i+1)^2 \approx \sum_{i=2}^{k} i^2 \tag{35}$$

First note that $(i+1)^3 = i^3 + 3i^2 + 3i + 1$ and summing for $i = 2, ..., k$ we have:

$$\sum_{i=2}^{k}\left[(i+1)^3 - i^3\right] = 3\sum_{i=2}^{k} i^2 + 3\sum_{i=2}^{k} i + \sum_{i=2}^{k} 1 \tag{36}$$

As the he left hand side of the previous equation is a telescopic sum, we can solve it easily as:

$$(k+1)^3 - 8 = 3\sum_{i=2}^{k} i^2 + 3\sum_{i=2}^{k} i + (k-1) \tag{37}$$

To solve the summation of i from 2 to k, we adopt the same strategy. First, note that $(i+1)^2 = i^2 + 2i + 1$ and summing for $i = 2, ..., k$ we have:

$$\sum_{i=2}^{k}\left[(i+1)^2 - i^2\right] = 2\sum_{i=2}^{k} i + \sum_{i=2}^{k} 1 \tag{38}$$

Once again, the left hand side of the previous equation is a telescopic sum, which is easily solvable. Hence, we can write:

$$2\sum_{i=2}^{k} i = (k+1)^2 - 4 - k + 1 \tag{39}$$

which finally leads to:

$$\sum_{i=2}^{k} i = \frac{k^2 + k - 2}{2} = \frac{k(k+1)}{2} - 1 \tag{40}$$

Going back to equation (37), we can write:

$$(k+1)^3 - 8 = 3\sum_{i=2}^{k} i^2 + 3\left[\frac{k(k+1)}{2} - 1\right] + (k-1) \tag{41}$$

which leads to:

$$\begin{aligned} 3\sum_{i=2}^{k} i^2 &= (k+1)^3 - 8 - \frac{3k(k+1)}{2} + 3 - k + 1 \\ &= k^3 + 3k^2 + 3k + 1 - 4 - \frac{3k(k+1)}{2} - k \\ &= k^3 + 3k^2 + 3k - \frac{3k(k+1)}{2} - k - 3 \end{aligned} \tag{42}$$

Dividing both sides by 3, gives us:

$$\sum_{i=2}^{k} i^2 = \frac{1}{3}\left[k^3 + 3k^2 + 3k - \frac{3k^2}{2} - \frac{3k}{2} - k\right] - 1 \tag{43}$$

$$= \frac{1}{3}\left[k^3 + \frac{3k^2}{2} + \frac{k}{2}\right] - 1 = \frac{1}{6}\left(2k^3 + 3k^2 + k\right) - 1$$

Finally, by factoring the polynomial we have:

$$\sum_{i=2}^{k} i^2 = \frac{1}{6}k(k+1)(2k+1) - 1 \tag{44}$$

which is $O(k^3)$. Thus, as we have n samples in the series, the total computational cost to build the matrices M is $O(w\_size \times n \times k^3)$. In the following we have to compute the elements of the vector c. Similarly, by the same argument, we have to solve the summation:

$$\sum_{i=2}^{k}(i+1) \approx \sum_{i=2}^{k} i \tag{45}$$

which has been solved is the previous analysis and results in a $O(k^2)$ function. Thus, the total computational complexity for building the $\vec{c}$ vectors is $O(w\_size \times n \times k^2)$. Next, we have to compute the computational cost regarding the solution of the linear system. It is known that a system composed of k equations with k variables can be solved in $O(k^3)$. But as we have a system with $k+1$ equations and $k+1$ variables, and knowing that the process is done for $i = 2, 3, ..., k$ due to the recursion, we have to solve the following summation:

$$\sum_{i=2}^{k}(i+1)^3 \approx \sum_{i=2}^{k} i^3 \tag{46}$$

First, note that $(i+1)^4 = i^4 + 4i^3 + 6i^2 + 4i + 1$. Then, we have:

$$\sum_{i=2}^{k}\left[(i+1)^4 - i^4\right] = 4\sum_{i=2}^{k} i^3 + 6\sum_{i=2}^{k} i^2 + 4\sum_{i=2}^{k} i + \sum_{i=2}^{k} 1 \tag{47}$$

The left hand side of the previous equation is a telescopic sum, and plugging the expressions for the summations computed previously, we can write:

$$(k+1)^4 - 16 = 4\sum_{i=2}^{k} i^3 + 6\left[\frac{1}{6}k(k+1)(2k+1) - 1\right] + 4\left[\frac{k(k+1)}{2} - 1\right] + (k-1) \tag{48}$$

By symplifying the equation we reach the following equality:

$$4\sum_{i=2}^{k} i^3 = k^4 + 2k^3 + k^2 - 4 \tag{49}$$

which finally leads to:

$$\sum_{i=2}^{k} i^3 = \frac{k^2(k+1)^2}{4} - 1 \tag{50}$$

which is $O(k^4)$. Hence, as we have to solve a linear system for each sample of the series, the total computational complexity is $O(n \times k^4)$.

Finally, the cost for computing the non-linear estimatives is $O(n \times k^2)$. Hence, the total computational complexity of the NoLAW filter is given by:

$$O(w\_size \times n) + O(w\_size \times n \times k^3) + O(w\_size \times n \times k^2) + O(n \times k^4) + O(n \times k^2) \tag{51}$$

which can be reduced to $O(w\_size \times n \times k^3)$ when the the size of the local windows w_size is larger than the filter order k or reduced to $O(n \times k^4)$ when the size of the local windows w_size is smaller than the filter order k. Anyway, the most important fact is that both w_size and k are constants much smaller than n. Therefore, the computational cost of the proposed NoLAW filter is linear in n, which is equivalent to the linear Wiener filter. In all experiments in this paper, we use a cubic NoLAW filter, which means $k = 3$.

5 Experiments and results

In order to test and evaluate the proposed NoLAW filter in time series smoothing, we performed a series of computational experiments. We compared the performance of the proposed method against Gausian smoothing, Simple Exponential Smoothing, Holt-Winters method (Triple Exponential Smoothing) and the linear adaptive Wiener filter using local windows of first (size 3), second (size 5) and third-orders (size 7). To measure the performance of the smoothing, each time series was corrupted by an additive independent Gaussian noise with variance equal to 10% of the signal's variance. After the smoothing, five different quantitative metrics were computed: Mean Absolute Percentage Error (MAPE) [19], Root Mean Squared Error (RMSE) [20], Mean Absolute Error (MAE) [21], Median Absolute Error (MedAE) [22] and Peak Signal-to-Noise Ratio (PSNR), a measure often employed in digital signal processing [23]. In all experiments, the NoLAW filter parameter k was set to 3, which means a cubic Wiener filter. The estimation of the noise variance was performed by [12]:

$$\hat{\sigma}_n^2 = \frac{V_0 - V_{med}}{1 - q} \tag{52}$$

where V_0 is the variance of the input signal, V_{med} is the variance of the input singal after median filtering and q is a noise reduction coefficient. According to the authors, in case of Gaussian noise, the estimator can be expressed as [12]:

$$\hat{\sigma}_n^2 = \frac{2n}{2n - \pi}(V_0 - V_{med}) \tag{53}$$

In our computational experiments, we selected 16 different time series: 1) daily_female_birds; 2) bovespa; 3) dolar; 4) seatbelts; 5) beer; 6) mintemp; 7) shampoo; 8) Cali emissions; 9) monthly-sunspots; 10) ETTh1; 11) ETTh2; 12) mfeat-fourier; 13) eeg-eye-state; 14) phoneme; 15) texture; 16) optdigits. In the following, we present a brief description of each one of them:

1. *daily_female_births*: daily total female births for California in 1959;
2. *bovespa*: time series of the Brazilian stock market index showing the daily closing prices for the period 2018 to 2020.
3. *dolar*: dollar to Brazilian reais exchange rate from 2014 to 2020.
4. *seatbelts:* time series giving the monthly totals of car drivers in Great Britain killed or seriously injured from January 1969 to December 1984.
5. *beer*: montly beer production in Australia from Januray 1956 to August 1995.
6. *mintemp*: daily minimum temperatures in Melbourne from January 1981 to December 1990.
7. *shampoo*: sales of shampoo over a three year period.
8. *Cali emissions*: total carbon dioxide emissions from all sectors, all fuels in California from 1980 to 2017.
9. *monthly-sunspots*: monthly mean total sunspot number from January 1749 to December 1983.
10. *ETTh1*: hourly electricity transformer temperature collected from a county in China during an interval of 2 years.
11. *ETTh2*: hourly electricity transformer temperature collected from another county in China during an interval of 2 years.
12. *mfeat-fourier*: Fourier coefficients extracted from the shape of handwritten digits from 0 to 9.
13. *eeg-eye-state*: continuous eletroencephalogram measurements with the Emotiv EEG Neuroheadset during 117 seconds.
14. *phoneme*: amplitudes of the first harmonics normalised by the total energy (integrated on all the frequencies) during nasal and oral sounds.
15. *texture*: statistical measures from texture image patches.
16. *optdigits*: the variation of a feature extracted from optical recognition of handwritten digits along all samples of the dataset.

All data used in the experiments can be found free of charge in several open internet repositories. Table 1 shows the obtained results in terms of Mean Absolute Percentage Error. The bold value indicates the best result and the underline value represents the second best result. The metrics show that for these time series, in the majority of the cases, the best result is obtained by cubic NoLaW with first-order or third-order local windows. However, looking at the medians, the best performance was obtained by the first-order cubic NoLaW, followed by the first-order linear Wiener filter. To verify whether the performance of the proposed cubic NoLAW filter (best method) is superior to the linear Wiener filter (second best method) or not, we performed a Wilcoxon signed-rank test [24]. According to the test, there are strong evidences in favor

of the rejection of the null hypothesis that the performances are equivalent (p-value $= 9.15 \times 10^{-5}$), suggesting that the proposed NoLAW filter can produce better results.

Table 2 shows the obtained results in terms of another metric: the Root Mean Squared Error. The measurements show that for these time series, once again the best methods were the cubic NoLaW filter with first-order or third-order local windows. In terms of average performance, the best result was obtained by the third-order cubic NoLaW filter, followed by the third-order linear Wiener filter. According to a Wilcoxon signed-rank test, there are strong evidences in favor of the rejection of the null hypothesis that the performances are equivalent (p-value $= 0.0005$), suggesting that the proposed method performs better.

Table 3 shows the obtained results in terms of the Mean Absolute Error. In this case, the best methods were the cubic NoLaW filter and the linear Wiener filter with third-order local windows. According to a Wilcoxon signed-rank test, there are significant evidences in favor of the rejection of the null hypothesis that the performances are equivalent (p-value $= 0.001$), once again favoring the proposed method.

Table 4 shows the obtained results in terms of the Median Absolute Error. In general, the two best methods were the cubic NoLaW filter and the linear Wiener filter with first-order local windows. According to a Wilcoxon signed-rank test, for a significance level $\alpha = 0.05$, the null hypothesis should not be rejected (p-value $= 0.051$), suggesting that there are no significant differences between the linear Wiener filter and the cubic NoLAW filter for these series.

Finally, Table 5 shows the obtained results in terms of the Peak Signal-to-Noise Ratio. In general, the two best methods were the cubic NoLAW filter and the linear Wiener filter with second-order local windows. According to a Wilcoxon signed-rank test, there are strong evidences in favor of rejecting the null hypothesis that the linear Wiener filter and NoLAW are equivalent in these datasets (p-value $= 3.05 \times 10^{-5}$), suggesting that the proposed method is capable of producing better results.

In order to ilustrate some qualitative results, Figure 3 shows a comparison between the linear Wiener filter and cubic NoLAW with third-order local windows in the smoothing of the shampoo time series. A Python implementation of the cubic NoLAW filter can be found at https://github.com/alexandrelevada/NoLAW.

6 Conclusions

In several machine learning applications, such as forecasting, regression and classification, time series smoothing is a fundamental pre-processing stage. In the last years, several deep learning methods have been proposed to tackle this problem, however, one of the main limitations with this approach is the computational burden required to train neural networks. Moreover, often, these

Table 1 Mean Absolute Percentage Error for time series smoothing.

Series	G	SES	HW	W1	W2	W3	NLW1	NLW2	NLW3
1	10.3181	13.6454	13.7350	7.6301	9.0543	10.1111	**7.5400**	8.8482	9.8020
2	1.8979	1.7853	1.7931	2.3313	1.8395	1.6995	2.2776	1.7838	**1.6169**
3	2.2512	1.8269	**1.8130**	3.0934	2.3481	1.9854	3.1133	2.2735	1.8810
4	6.8547	10.9287	10.9640	4.7340	5.1351	6.3006	**4.3037**	4.8612	5.8511
5	7.4732	11.3757	11.7976	5.4395	6.5691	7.3764	**5.3926**	6.4685	7.4994
6	17.2463	29.1607	29.1483	11.8693	15.0614	16.9807	**11.7878**	14.6040	16.4893
7	19.1383	24.6551	21.4708	19.2973	16.9165	17.5210	19.0884	16.6981	**15.2144**
8	4.0256	2.9049	2.8987	1.6814	1.9719	2.3450	**1.6124**	1.7315	2.3119
9	230.8878	251.1410	243.5270	247.1642	227.7846	219.5018	246.6997	223.2538	**208.1222**
10	22.0678	23.3302	23.3266	29.2221	23.2052	20.1507	29.0026	22.8043	**19.5576**
11	15.3532	24.8858	24.7842	11.4650	10.3309	10.3345	11.4022	10.2152	**9.6259**
12	26.5453	36.1334	36.1407	20.4645	21.5667	22.7302	**19.3382**	21.2576	22.2610
13	0.1730	0.2176	0.2203	0.1702	0.1488	0.1407	0.1664	0.1393	**0.1328**
14	228.9690	**103.1003**	103.1677	232.3131	202.3450	189.6312	223.0098	189.7999	159.9785
15	30.8565	41.8250	41.7624	17.2239	20.8449	23.4032	**16.0770**	20.4464	22.9741
16	217.4137	**40.9974**	41.2691	178.9524	152.7275	139.6670	170.4522	142.4578	125.1805
Median	16.2998	23.9926	22.3987	11.6672	12.6961	13.6576	**11.5950**	12.4096	12.5082

Table 2 Root Mean Squared Error for time series smoothing.

Series	G	SES	HW	W1	W2	W3	NLW1	NLW2	NLW3
1	5.3567	7.1030	7.0908	3.8140	4.5439	5.0781	**3.7847**	4.4531	4.9568
2	3.4568	2.0317	2.0328	2.7488	2.1543	1.9784	2.7102	2.1102	**1.9054**
3	0.1088	0.0776	**0.0774**	0.1285	0.0999	0.0843	0.1306	0.0973	0.0808
4	154.0298	232.7076	233.4265	97.0117	107.4777	129.3289	**91.6395**	101.5780	123.4322
5	12.6565	19.5423	19.7683	9.1450	10.7101	12.2835	**9.0875**	10.5474	12.2934
6	1.7117	2.6683	2.6683	1.2829	1.5512	1.7152	**1.2717**	1.5227	1.6603
7	70.5460	84.9639	75.1557	57.2539	53.3233	57.5866	56.6669	52.9985	**52.0938**
8	33.8448	12.4571	12.3926	7.2836	8.4515	10.4320	**6.8902**	7.5117	10.1709
9	12.3192	17.6787	17.8127	11.4367	10.9925	11.0672	11.4333	10.8846	**10.7527**
10	1.4290	1.6825	1.6824	1.9388	1.5794	1.4094	1.9243	1.5642	**1.4055**
11	2.4415	3.7462	3.7463	2.5237	2.3226	**2.3104**	2.4988	2.3152	2.3119
12	0.0807	0.1040	0.1040	0.0610	0.0638	0.0657	**0.0573**	0.0626	0.0639
13	33.9054	37.6314	37.6330	8.5687	7.5222	7.1229	8.3875	7.0568	**6.7731**
14	0.8009	1.0000	1.0000	0.5554	0.6715	0.7304	**0.5500**	0.6598	0.7107
15	0.1025	0.1285	0.1285	0.0781	0.0878	0.0919	**0.0718**	0.0856	0.0892
16	0.8327	1.0386	1.0385	0.2686	0.2714	0.2860	**0.2596**	0.2614	0.2741
Median	2.9492	3.2073	3.2073	2.6363	2.2385	2.1444	2.6045	2.2127	**2.1087**

Table 3 Mean Absolute Error for time series smoothing.

Series	G	SES	HW	W1	W2	W3	NLW1	NLW2	NLW3
1	4.2612	5.6011	5.5929	3.1264	3.6960	4.1094	**3.0882**	3.6062	3.9821
2	1.7475	1.6288	1.6361	2.1122	1.6724	1.5446	2.0638	1.6235	**1.4673**
3	0.0737	0.0609	**0.0605**	0.1007	0.0767	0.0649	0.1013	0.0742	0.0617
4	113.2654	180.5409	180.8641	76.0800	84.8070	104.5605	**69.9654**	81.1116	97.6829
5	9.9764	15.3823	15.7527	7.0442	8.6282	9.6807	**6.9990**	8.4364	9.7424
6	1.3430	2.0934	2.0934	1.0245	1.2360	1.3613	**1.0172**	1.2064	1.3063
7	54.0411	68.8028	59.7352	47.1898	42.6532	46.1253	47.2269	41.6054	**39.9267**
8	13.9705	10.1387	10.1098	5.8944	6.8578	8.1356	**5.6586**	6.0689	8.0264
9	9.4396	13.1153	13.2172	9.0339	8.6406	8.6997	9.0322	8.5814	**8.4091**
10	1.1316	1.3049	1.3048	1.5019	1.2153	1.0854	1.4864	1.1989	**1.0773**
11	1.8885	2.8418	2.8424	1.9964	1.8312	1.8241	1.9767	**1.8216**	1.8232
12	0.0608	0.0773	0.0773	0.0482	0.0503	0.0519	**0.0454**	0.0493	0.0502
13	7.0175	8.8592	8.9655	6.8249	5.9660	5.6434	6.6733	5.5860	**5.3275**
14	0.6488	0.8263	0.8263	0.4514	0.5480	0.5977	**0.4456**	0.5353	0.5795
15	0.0785	0.0965	0.0965	0.0623	0.0697	0.0727	**0.0567**	0.0674	0.0699
16	0.3074	0.1725	0.1728	0.1979	0.1792	0.1715	0.1894	0.1689	**0.1570**
Median	1.8180	2.4676	2.4679	2.0543	1.7518	1.6844	2.0202	1.7225	**1.6452**

Table 4 Median Absolute Error for time series smoothing.

Series	G	SES	HW	W1	W2	W3	NLW1	NLW2	NLW3
1	3.7837	4.8130	4.8777	2.7930	3.2371	3.5926	**2.7043**	2.9548	3.3374
2	1.3528	1.3903	1.3647	1.6512	1.3590	1.2349	1.5929	1.2974	**1.1399**
3	0.0599	0.0511	0.0508	0.0823	0.0611	0.0515	0.0831	0.0589	**0.0493**
4	89.1531	140.1907	140.0319	64.8827	69.4424	89.8057	**52.2370**	74.5794	85.2024
5	8.2099	12.6944	13.2143	**5.3906**	7.6022	8.5821	5.5055	7.1130	8.2335
6	1.0952	1.7152	1.7148	0.8785	1.0557	1.1260	**0.8731**	1.0032	1.0756
7	41.2141	62.9749	57.0766	42.7589	36.1135	39.6063	44.7423	**33.0170**	33.7548
8	4.8837	8.0523	8.0549	5.0498	6.2324	6.0570	**5.0163**	5.0426	6.2778
9	7.7403	9.9284	9.9841	7.5487	7.1491	7.1553	7.5253	6.9623	**6.7605**
10	0.9487	1.0663	1.0659	1.2170	0.9758	0.8759	1.1972	0.9569	**0.8573**
11	1.5573	2.3147	2.3126	1.6551	1.5205	1.5130	1.6506	**1.5116**	1.5123
12	0.0481	0.0598	0.0598	0.0406	0.0423	0.0431	**0.0385**	0.0411	0.0418
13	5.1528	5.9746	5.9946	5.7053	4.9700	4.7300	5.5983	4.6443	**4.4104**
14	0.5618	0.7746	0.7740	0.3911	0.4813	0.5304	**0.3826**	0.4642	0.5096
15	0.0616	0.0761	0.0762	0.0523	0.0572	0.0606	**0.0469**	0.0557	0.0582
optdigits	0.1466	**0.0394**	0.0401	0.1542	0.1332	0.1233	0.1468	0.1245	0.1084
Median	1.4551	2.0150	2.0137	1.6532	1.4398	1.3739	1.6217	1.4045	**1.3261**

Table 5 Peak Signal-to-Noise Ratio for time series smoothing.

Series	G	SES	HW	W1	W2	W3	NLW1	NLW2	NLW3
1	22.6886	20.2377	20.2526	25.6389	24.1179	23.1525	**25.7059**	24.2933	23.3624
2	30.7759	35.3924	35.3873	32.7666	34.8831	35.6232	32.8894	35.0629	**35.9496**
3	31.8988	34.8393	**34.8607**	30.4589	32.6465	34.1169	30.3133	32.8689	34.4879
4	24.7259	21.1418	21.1150	28.7415	27.8517	26.2441	**29.2364**	28.3420	26.6495
5	24.7149	20.9416	20.8418	27.5374	26.1653	24.9747	**27.5923**	26.2983	24.9677
6	23.7304	19.8743	19.8744	26.2350	24.5856	23.7127	**26.3114**	24.7470	23.9953
7	19.7062	18.0910	19.1565	21.5196	22.1374	21.4693	21.6091	22.1904	**22.3400**
8	21.3496	30.0311	30.0762	34.6926	33.4007	31.5721	**35.1748**	34.4246	31.7923
9	26.2782	23.1408	23.0752	26.9238	27.2679	27.2090	26.9264	27.3536	**27.4595**
10	30.1558	28.7371	28.7376	27.5058	29.2864	30.2759	27.5711	29.3709	**30.3001**
11	32.9066	29.1880	29.1877	32.6190	33.3403	**33.3862**	32.7051	33.3679	33.3805
12	19.6113	17.4017	17.4016	22.0337	21.6490	21.3940	**22.5760**	21.8193	21.6304
13	47.2417	46.3360	46.3356	59.1887	60.3202	60.7939	59.3744	60.8749	**61.2313**
14	13.2091	11.2811	11.2811	16.3881	14.7398	14.0096	**16.4730**	14.8925	14.2474
15	10.1556	8.1911	8.1916	12.5218	11.5027	11.1056	**13.2523**	11.7243	11.3678
16	25.6725	23.7536	23.7542	35.5004	35.4116	34.9540	**35.7963**	35.7362	35.3250
Median	24.7204	22.1413	22.0951	27.5216	27.5598	26.7266	27.5817	**27.8478**	27.0545

models require a large amount of samples for convergence to suitable results, which is not always possible.

The adaptive Wiener filter computes the optimal linear estimative under the hypothesis of additive independent Gaussian noise. Hence, the motivating question for this paper was: how to compute an optimal non-linear estimative in a computationally efficient way? Our answer is: by using the NoLAW filter. Computational experiments showed that the proposed method can obtain better results than other techniques with similar computational cost. As any statistical method, it has positive and negative aspects. One limitation of the NoLAW filter is that, due to the MSE loss function in the least squares problem, it may not be robust to the presence of outliers and other kinds of noise in data.

Future works may include the development of the Robust NoLAW filter, in which the sample averages are replaced by the medians, making the filter not optimal in a mean squared error sense (L2-norm), but instead in a different loss function that uses the absolute value of the differences (L1-norm). Another possibility consists in the development of the Non-Local NoLAW filter by employing a non-local strategy in the estimation of the filter parameters, through the computation of a similarity measure between the current local window and local windows from different past samples. Finally, we also intend to extend the NoLAW filter to work with multivariate time series using the multivariate linear Wiener filter as the base case.

Acknowledgments. This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

Conflict of interest. The author declares no conflicts of interest.

Data availability. The author declares that all data and code used in the computational experiments is available at https://github.com/alexandrelevada/NoLAW.

References

[1] Brown, R. G., *Smoothing, forecasting and prediction of discrete time series*, Courier Corporation, 2004.

[2] Shrestha, M. B., Bhatta, G. R., Selecting appropriate methodological framework for time series data analysis, The Journal of Finance and Data Science, v. 4, no. 2, pp. 71-89, 2018.

[3] Enders, W., *Applied Econometric Time Series*, 4th ed., John Wiley & Sons, 2014.

[4] Weiss, C. H., *An introduction to discrete-valued time series*, John Wiley & Sons, 2018.

[5] Rhif, M., Abbes, A. B., Farah, I. R., Martinez, B., Sang, Y. Wavelet Transform Application for/in Non-Stationary Time-Series Analysis: A Review, Applied Sciences, v. 9, no. 7, p. 1345, 2019.

[6] Broomhead, D. S., Huke, J. P., Muldoon, M. R. Linear filters and non-linear systems. Journal of the Royal Statistical Society: Series B (Methodological), v. 54, n. 2, p. 373-382, 1992.

[7] Wiener, N. *Extrapolation, Interpolation, and Smoothing of Stationary Time Series: with Engineering Applications*, MIT Press, 1964.

[8] Kalman, R.E. A new approach to linear filtering and prediction problems. Journal of Basic Engineering, v. 82, no. 1, pp. 35–45, 1960.

[9] Kuan D.T., Sawchuk A.A., Strand T.C., Chavel P. Adaptive noise smoothing filter for images with signal-dependent noise. IEEE Transactions on Pattern Analysis and Machine Intelligence, v. 7, no. 2, pp. 165-77, 1985.

[10] Holt, C. C. Forecasting Trends and Seasonal by Exponentially Weighted Averages. Office of Naval Research Memorandum, v. 52, 1957.

[11] Winters, P. R. Forecasting Sales by Exponentially Weighted Moving Averages. Management Science, v. 6, no. 3, pp. 324–342, 1960.

[12] Peksinski, J., Mikołajczak, G. Estimation of the noise variance in time series using a median filter. Computer Applications in Electrical Engineering, v. 12, pp. 316-323, 2014.

[13] Wibawa, A.P., Utama, A.B.P., Elmunsyah, H., Pujianto, U., Dwiyanto, F. A., Hernandez, L. . Time-series analysis with smoothed Convolutional Neural Network. Journal of Big Data v. 9, no. 44, 2022.

[14] Livieris, I.E., Stavroyiannis, S., Iliadis, L., Pintelas, P. Smoothing and stationarity enforcement framework for deep learning time-series forecasting. Neural Computating and Applications, v. 33, pp. 14021–14035, 2021.

[15] Xuerong, L., Wei, S., Shouyang, W. Text-based crude oil price forecasting: A deep learning approach. International Journal of Forecasting, v. 35, no. 4, pp. 1548–1560, 2019.

[16] Tealab, A. Time series forecasting using artificial neural networks methodologies: A systematic review, Future Computing and Informatics Journal, v. 3, no. 2, pp. 334-340, 2018.

[17] Dupond, S. A thorough review on the current advance of neural network structures. Annual Reviews in Control, v. 14, pp. 200–230, 2019.

[18] Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory, Neural Computation. v. 9 no. 8, pp. 1735–1780, 1997.

[19] Myttenaere, A., Golden, B., Le Grand, B., Rossi, F. Mean Absolute Percentage Error for regression models, Neurocomputing, v. 192, pp. 38-48, 2016.

[20] Hyndman, R. J.; Koehler, A. B. Another look at measures of forecast accuracy, International Journal of Forecasting, v. 22, no. 4, pp. 679–688, 2006.

[21] Willmott, C. J., Matsuura, K. Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance, Climate Research, v. 30, pp. 79–82, 2005.

[22] Rousseeuw, P. J., Croux, C. Alternatives to the median absolute deviation. Journal of the American Statistical Association, v. 88, no. 424, pp. 1273–1283, 1993.

[23] Korhonen, J., You, J. Peak signal-to-noise ratio revisited: Is simple beautiful?, 4th International Workshop on Quality of Multimedia Experience (QoMEX), IEEE, pp. 37-38, 2012.

[24] Wilcoxon, F. Individual comparisons by ranking methods, Biometrics Bulletin, v. 1, no. 6, pp. 80–83, 1945.

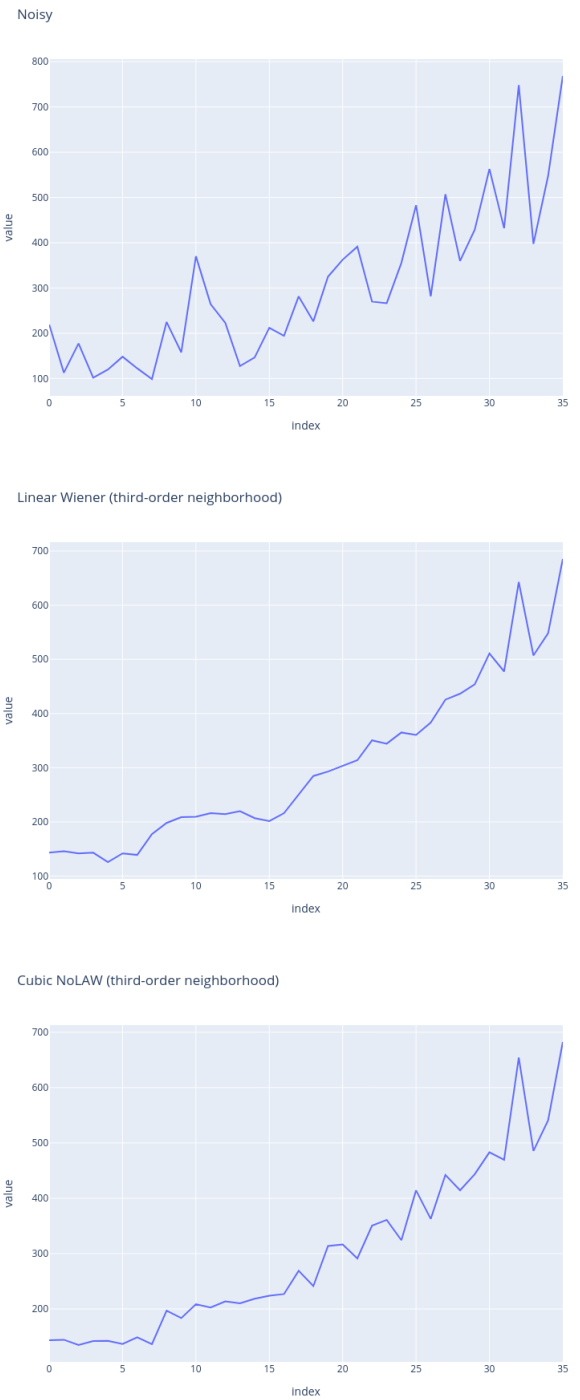

Fig. 3 Comparison between the the linear Wiener filter and the cubic NoLAW filter in the smoothing of the shampoo series. From top to bottom we have: a) noisy series; b) linear Wiener filter; c) cubic NoLAW filter.