A comparison of denoising methods for one dimensional time series

Torsten Köhler, Dirk Lorenz [*]

Abstract

This report presents the results of a comparison of denoising methods for one dimensional time series. The comparison has been carried out within the DFG Priority Program 1114 "Mathematical methods for time series analysis and digital image processing".

The aim of this report is to present an extensive comparison of some basic denoising methods and some more elaborated methods. We apply the different methods to a set of noisy test time series and measure the performance with different error measures.

One surprising result is, that in some cases which are assumed to be difficult, the most easy methods (namely a simple moving average) yield the best results.

1 Introduction

In practice many methods in signal and image processing are applied without evaluating the suitability of the specific underlying problems. Up to now, even for very simple test examples there are only very limited results on evaluation and comparison of different methods.

This report presents a comparison of some very basic and some more advanced denoising methods for one dimensional time series.

If one wants to compare the performance of different denoising methods one has to decide many things:

- A proper set of test functions and noisy versions of these test functions has to be generated.
- The denoising methods for the comparison has to be chosen reasonably
- Meaningful and appropriate quality measures has to be defined.
- If the methods depend on parameters, the question whether the results shall be optimized and how this should be done has to be answered.

This report is organized as follows: In the next section we describe the methodology which has been used for this comparison. We describe the test data, the methods for denoising, the used error measures and the procedure of optimization. The third section presents the results of our comparison, a discussion and an interpretation.

2 Description of Methodology

This section describes the methodology we have chosen for this comparison of methods. We are aware of the fact, that our choices are debatable and see this report as a contribution to the discussion on the comparison of methods in signal processing.

2.1 The test data

The test data consists of five different functions, each defined on the interval $[-10, 10]$ and sampled on grid of size $\Delta t = 10^{-3}$. This gives vectors of length $20,001$.

Each of the five test functions was disturbed by 80 different realizations of additive white noise which means Gaussian distributed noise with zero mean and variance σ generated by Matlab's `randn` function. We used three different noise levels, namely the variances $\sigma = .01,\ .05,\ .1$. In total we deal with $1,200$ different sampled functions, each consisting of $20,001$ data points.

The five test functions are:

1. A simple sinusoid:

$$f_1(t) = \sin(5t).$$

[*]Zentrum für Technomathematik, Universität Bremen. The work of the authors has been supported by DFG, SPP 1114-MA 1657/7.

2. A piecewise quadratic function, which has a discontinuity in the second derivative:

$$f_2(t) = \begin{cases} -\frac{t^2}{2} & \text{for } -10 \leq t < 0 \\ \frac{t^2}{2} & \text{for } 0 \leq t \leq 10 \end{cases}.$$

3. A piecewise constant function:

$$f_3(t) = \begin{cases} 5 & \text{for } -10 \leq t < -2 \\ -3 & \text{for } -2 \leq t \leq 7 \\ 8 & \text{for } 7 < t \leq 10 \end{cases}.$$

4. A mixture of the first three functions:

$$f_4(t) = \begin{cases} \sin(5t) & -10 \leq t \leq -3.33 \\ \sin(-3.33 * 5) & -3.33 < t \leq -1.11 \\ 2 & -1.11 < t \leq 1.11 \\ -\frac{3.33^2}{2} & 1.11 < t \leq 3.33 \\ -\frac{(t-6.66)^2}{2} & 3.33 < t \leq 6.66 \\ \frac{(t-6.66)^2}{2} & 6.66 < t \leq 10 \end{cases}.$$

5. A time series, generated by an invertible and casual ARMA model of order $(7, 15)$ (see [3], for example, for an introduction to ARMA and related models). The time series was generated by taking a vector z consisting of white noise of variance 0.02 and length 20301 and defining the time series x as

$$x_i = \sum_{j=1}^{7} a_j x_{i-j} + \sum_{k=0}^{15} b_j z_{i-j}$$

for $i = 16, \ldots, 20301$ and the values $a = [1, -.24, .13, .3, -.2, .3 - .3]$ and $b = [1,\ 1.2,\ 1.2,\ 1.2,\ 1.2,\ 1.2,\ 1.2,\ 1.2,\ 1.2,\ 1.2,\ 1.2,\ 1.2,\ 1.2,\ 1.2,\ 1.2,\ .9]$. Then, the time series we used was

$$(f_5)_i = x_{i+300}.$$

Plots of the test functions can be found in Figure 1. This set of test functions is a compromise between size of the set and the various different features one can imagine for functions. We restricted ourselves to only five functions to keep the number of computations manageable. But on the other hand, these five functions cover a wide range of desired features.

2.2 The compared methods

The choice of methods to be compared is probably the most debatable point in this comparison. We decided to use five different methods for denoising. Some very basic methods, which have been among the first denoising methods in signal processing and some more advanced methods. The three basic and rather simple methods are:

Moving average filter: The moving average can be seen as the convolution with a plateau function. This is probably the oldest and simplest denoising method (see [6] or [5] for example).We included this filter in a very simple form: We only considered the average over five neighbor values of the discrete time series:

$$(f^{\text{den}})_i = \frac{f_{i-2} + f_{i-1} + f_i + f_{i+1} + f_{i+2}}{5}.$$

In other words: We used a finite impulse response filter with filter mask 1/5[1 1 1 1 1]. Of course, the length of the filter has a great impact on the denoising results and on the quality of the denoising. But we included this really simple version so that it becomes clear how it keeps up with more elaborated filters. The data have been padded by symmetry for values at the boundary.

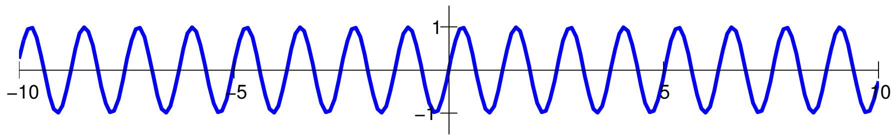
Exponential smoothing filter: The exponential smoothing filter is an infinite impulse response filter depending a real parameter $a \in [0, 1]$ (see [6]):

$$(f^{\text{den}})_1 = f_1 \quad \text{and}$$
$$(f^{\text{den}})_i = af_i + (1-a)(f^{\text{den}})_{i-1} \quad \text{for} \quad i > 1.$$

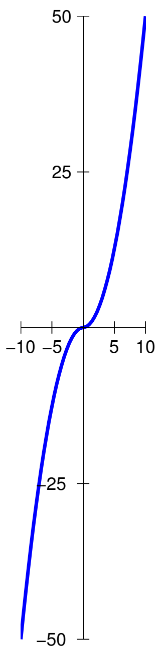
For $a = 1$ one gets no denoising ($f^{\text{den}} = f$) and for $a = 0$ a constant time series ($(f^{\text{den}})_i \equiv f_1$).

Linear Fourier smoothing: The Fourier filtering methods are based on the well known Fourier transform which decomposes a signal into its frequency components. By suppressing the high frequency components one can achieve a denoising effect. This is called low-pass filtering. We used a perfect low-pass filter depending on the cut-off frequency as a parameter. In continuous time this looks like

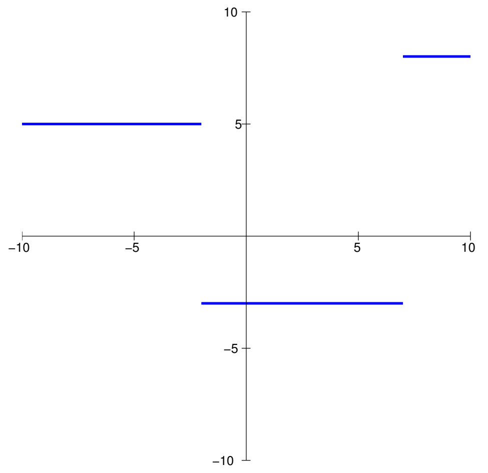
$$f^{\text{den}} = \mathcal{F}^{-1}(\chi_{[-\lambda,\lambda]}\mathcal{F}f)$$



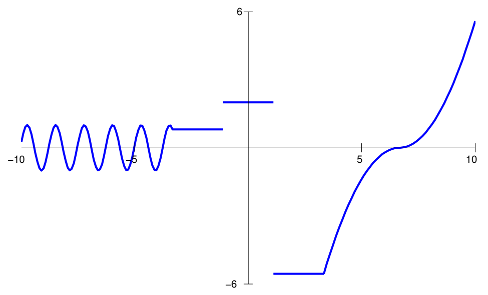
a) The test function f_1, a simple sinusoid.



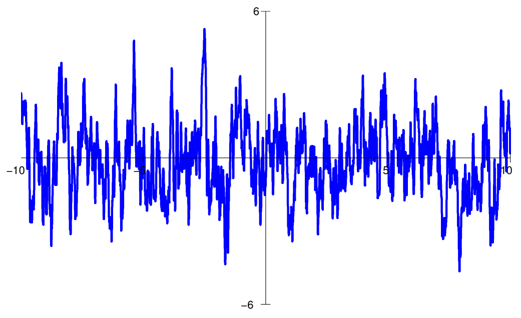
b) The test function f_2, a piecewise quadratic function.



c) The destination f_3, a step function,



d) The test function f_4, a mixture of the first three.



e) The test function f_5, generated by an ARMA model.

Figure 1: The five test functions

where $\mathcal{F}$ denotes the Fourier transform and χ_A is the characteristic function of the set A. The parameter λ is the cut off frequency. We used the standard FFT algorithm to perform the filtering. (Again, see [6] or [5] for example.)

In addition to these very basic methods, two more elaborated filters have been used:

Nonlinear wavelet shrinkage: The wavelet shrinkage methods are based on the discrete wavelet transform. The idea in this methods is, that on the one hand the wavelet transform yields in many cases a sparse representation of signals and one the other hand, white noise is transformed into white noise again. The noise reduction is done, by reducing the magnitude of the coefficients in a nonlinear way. See [8, 2] and the references therein for detailed descriptions of wavelet shrinkage. The implementation of this methods was done with the help of the wavelet toolbox of Matlab [10], namely the routine `wden`.

Simple nonlinear noise reduction: This nonlinear noise reduction method are built on assumptions from the theory of nonlinear time series analysis. They work in two steps: an embedding in a higher dimensional space, performed by the time-delay-embedding methods and a local averaging in the embedding space. The implementation was performed with the software package TISEAN 2.1 and the used routine was `nrlazy`. For an introduction to nonlinear time series analysis see [7], for example.

2.3 The quality measures

The choice of a quality measure is far from clear. Many different measures can be used and different measures show very different behavior.

Since in this investigation we know how the desired result of the denoising should look like, we can compare the denoising results with the original functions. in particular, we measure the distances $\|f - f^{\text{den}}\|$ for four different measures:

The L^1 norm: The discretized version of the L^1 norm is

$$\left\|f - f^{\text{den}}\right\|_{L^1} = \Delta t \sum_i \left|f_i - (f^{\text{den}})_i\right|.$$

The L^2 norm: The discrete L^2 norm looks like

$$\left\|f - f^{\text{den}}\right\|_{L^2} = \left(\Delta t \sum_i \left|f_i - (f^{\text{den}})_i\right|^2\right)^{\frac{1}{2}}.$$

The L^∞ norm: The L^∞ norm is just the magnitude of the largest value:

$$\left\|f - f^{\text{den}}\right\|_{L^\infty} = \max_i \left|f_i - (f^{\text{den}})_i\right|.$$

The *symmetrical visual error measure*: The "Symmetrical Visual Error Measure" has been proposed by Marron and Tsybakov [9]. It aims in measuring the distance of graphs in a graphical way. The definition is a little bit more complex. First we define the asymmetrical "visual error" which is given by

$$\text{VE}_2(f, f^{\text{den}}) = \left(\int_a^b d((t, f(t)), G_{f^{\text{den}}})^2 dt\right)^{1/2}$$

where $G_{f^{\text{den}}}$ is the graph of the function f^{den}, and $d((x,y), A)$ is the minimal euclidean distance from the point $(x,y) \in \mathbb{R}^2$ to a set $A \subset \mathbb{R}^2$. The "symmetrized version" is

$$\text{SE}_2(f, f^{\text{den}}) = \left(\text{VE}_2(f, f^{\text{den}})^2 + \text{VE}_2(f^{\text{den}}, f)^2\right)^{1/2}.$$

We use the discretized versions:

$$\text{VE}_2^{\text{discr}}(f, f^{\text{den}}) = \left(\Delta t \sum_i d((t_i, f_i), G^{\text{discr}}_{f^{\text{den}}})^2\right)^{1/2}$$

resp.

$$\text{SE}_2^{\text{discr}}(f, f^{\text{den}}) = \left(\text{VE}_2^{\text{discr}}(f, f^{\text{den}})^2 + \text{VE}_2^{\text{discr}}(f^{\text{den}}, f)^2\right)^{1/2}.$$

The different measures have very different interpretations: The L^1 norm measure the "area between the true and the denoised signal", whereas the L^2 norm measures the energy of the difference. The L^∞ norm is only sensitive to the largest deviation of the signals. The symmetrical visual error measure measures the distance of the graphs in a more graphical sense. For more information on these error measures see [9].

2.4 Choice of parameters and optimization

As four of the considered methods depend on parameters, one has to find appropriate selection rules to obtain optimal results. The choice of the parameters is crucial for the performance of the methods and is thereof subject of extensive research.

The first question is: What do we want to optimize? Since we have four different measures of quality one could optimize every method for every quality measure. We decided to use only the L^2 norm for optimization, i. e. we optimized the parameters such that the difference between the clean signal and the denoised signal has minimal L^2 norm. It turned out, that the parameters has been almost or totally the same for the 80 different realization of the noisy versions of one signal, so that we used the same parameters for all these realizations.

The L^2 norm plays a special role because many denoising algorithms use this norm for the estimation of the denoising parameters. Thus, we decided only to use this norm for the optimization. The other norms are presented to give an impression where the strengths and weaknesses of the different methods are.

Now we explain in detail, how we managed the different denoising methods.

As explained above, we decided to use the moving average filter without any dependency on parameters. The other two simple routines (the exponential smoothing and the FFT filtering) only depend on one parameter (the weighting parameter a or the cut off frequency λ, respectively). For these two cases we searched for an optimal value of the parameter just by testing out the parameters on a discrete set and taking the optimal value.

The wavelet shrinkage is based on many parameters: The wavelet which is used, the level of detail, the shrinkage function, or the threshold on every level. Since there is a lot of literature on how to choose these parameters, we decided to take one of the strategies proposed in literature. Namely we have chosen the level dependent soft shrinkage where the threshold is chosen by the principle of Stein's Unbiased Risk which is performed by the Matlab routine `wden` (see the manual of the Wavelet Toolbox of Matlab [10] and the references therein). Thus, the only parameter which remains to be chosen is the used wavelet. To keep the complexity manageable we only considered the family of Daubechies wavelets db$n$ for $n \in \{1, \ldots, 8\}$ and the "coiflets" coif$n$ for $n \in \{1, \ldots, 5\}$ (again, see the manual of the Wavelet Toolbox of Matlab [10] for more detail).

In order to consider techniques from nonlinear time series analysis, we tested the simple noise reduction routine `nrlazy` from the software package TISEAN. This routine depends on the embedding dimension, the number of iterations, the size of the neighborhood and the delay time (see the TISEAN Manual [4] for more information). We optimized the performance by taking these parameters from a finite set of combinations. In particular we restricted the embedding dimension to values smaller than 8, the number of iterations smaller than three, the neighborhood size smaller than one and the delay time smaller than 1000.

3 Results and interpretations

This section presents the results of this comparison of methods. The Tables 1 to 5 present the results of the denoising in three different charts for each signal (one for each noise level). Each table shows the mean value of the results for the eighty different noisy signals together with the two-σ-neighborhood. To make the results easier to read, the best and the worst values for each column are highlighted: the best value is printed in boldface and the worst one is printed italic.

To illustrate how the results for the different methods look like, we present some graphical results of the denoising in the Figures 2 to 6. We show one picture for every test function and every method. All illustrations are based on the highest noise level .1 and the optimal parameters. To make the illustrations more meaningful, we do not show the whole signals but only smaller parts. The noisy signals are represented by dots, the denoising results by solid lines.

Before we go into a more detailed interpretation of the results, we would like to stress that there is no “over all winner” of this comparison which outperforms every other method at all noise levels and for all test functions. Actually this was neither the objective of study nor what we expected.It turned out that there are different methods which perform best for different noise levels for the same test function (e. g. for test function 1, the sinusoid). The same observation is valid for different error measures: The optimal value depends on the respective error measure. The only general conclusion is, that the exponential filter is not suitable for any setting. All other methods have their strengths and weaknesses.

3.1 Observations for signal 1

The best methods for this simple sinusoid are the wavelet shrinkage and the FFT filter. One could expect, that the FFT methods were best adapted to this problem. But especially the L^∞ error is high for the FFT filter. This is because the period of the sinusoid does not fit the length of the interval, such that the signal has a jump at the boundary if it is considered to be periodic (like the FFT does) and this produces the well known Gibbs effect.

One observes that for the highest noise level the cut off frequency is exactly the frequency of the test function, i. e. all the noise with a higher frequency has been removed.

See Figure 2 and Table 1 for illustration of the denoising results resp. the presentation of the error measures.

3.2 Observations for signal 2

For this piecewise quadratic signal the worst method is obviously the FFT filter. A closer look shows, that the best cut off frequency for this signal is alway 500 which means, that no denoising has taken place at all. The reason for this is the same as for signal 1: the heavy Gibbs effect at the boundary where the signal jumps from $+50$ to -50.

The best method is again the wavelet shrinkage and the optimal wavelet is the `coif2` wavelet which is not surprising since the coiflets are designed such that both the scaling function and the wavelet have vanishing moments and thus are adapted to polynomials [8].

See Figure 3 and Table 2 for illustration of the denoising results resp. the presentation of the error measures.

3.3 Observations for signal 3

Without any doubt the TISEAN routine is best for this piecewise constant signal. Even the L^∞ error is remarkable small for a signal which has jumps of height 11. Again the FFT filter does not improve the noisy signal at all for noise level .01 and it is almost the same for the other noise levels and the exponential filter.

The wavelet filter shows fairly good results. It is a well known fact that simple wavelet shrinkage has difficulties to recover jumps due to its dependence on translations [1].

The results of the denoising are presented in Figure 4 and Table 3.

3.4 Observations for signal 4

For this mixture of the first three signals the result is a mixture of the first three results: Wavelet shrinkage and the TISEAN package perform well, for the L^∞ norm the TISEAN package gives better results for the L^1 and L^2 norm wavelet shrinkage yields smaller values.

Here the graphical distance measure SE_2 shows an interesting behavior of the TISEAN denoising method: The TISEAN routine yields nearly twice as high values of the L^2 error than the wavelet filter, but for the SE_2 measure the results are roughly comparable. This shows, that the TISEAN routine produces results which ”look like the true function” but does not have a small distance to the true function in the usual error measures.

See Figure 5 and Table 4 for illustration of the denoising results resp. the presentation of the error measures.

3.5 Observations for signal 5

This time series, generated by an ARMA model, shows a kind of surprising results. Although this signal has a complicated structure and is adapted

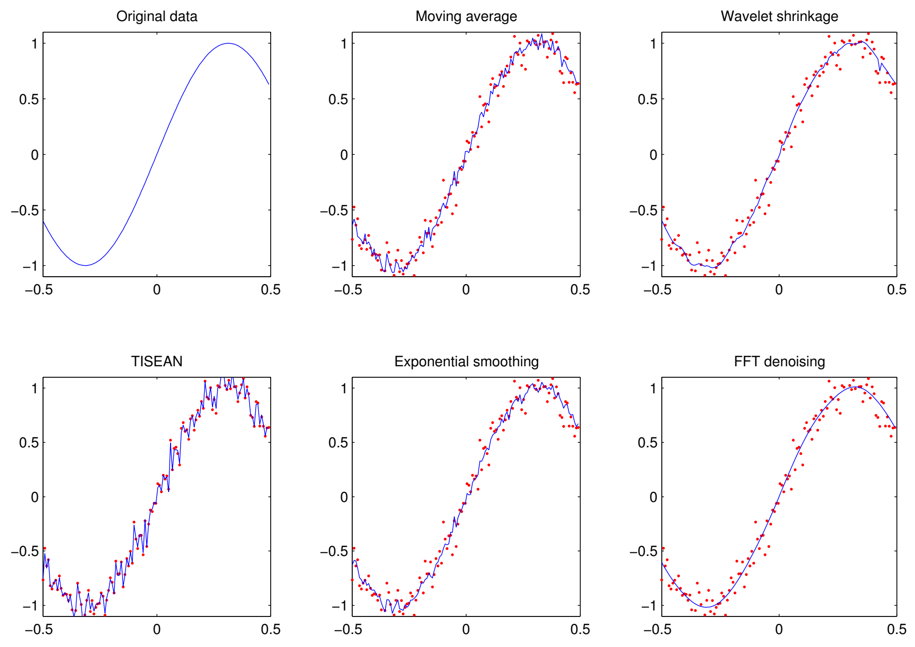


Figure 2: Denoising results for signal 1, the sinusoid. The noisy signal is represented by dots, the denoised signal by solid line.

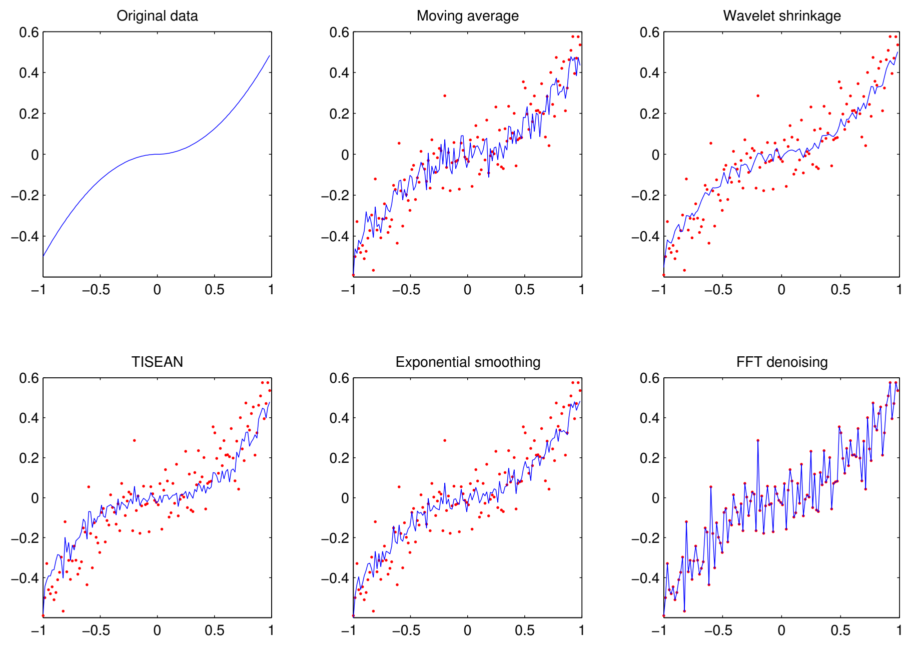


Figure 3: Denoising results for signal 2, the piecewise quadratic polynomial. The noisy signal is represented by dots, the denoised signal by solid line.

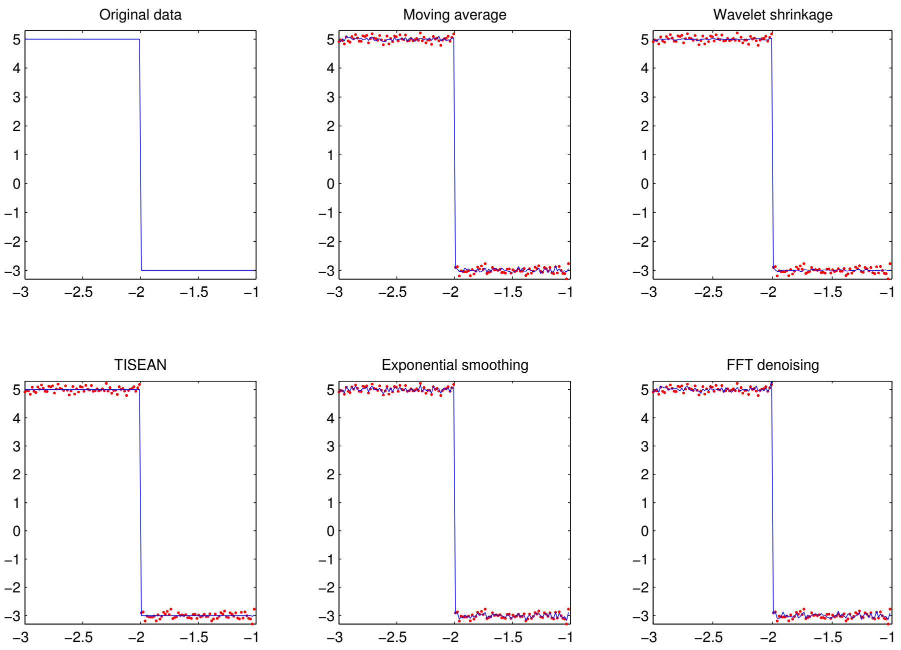


Figure 4: Denoising results for signal 3, the step function. The noisy signal is represented by dots, the denoised signal by solid line.

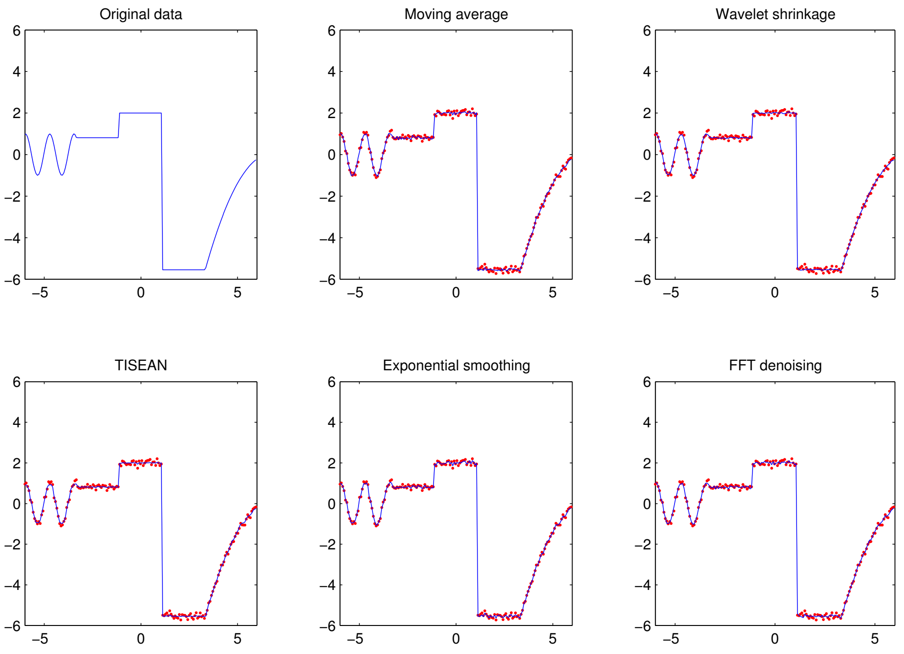


Figure 5: Denoising results for signal 4, the mixture of the first three functions. The noisy signal is represented by dots, the denoised signal by solid line.

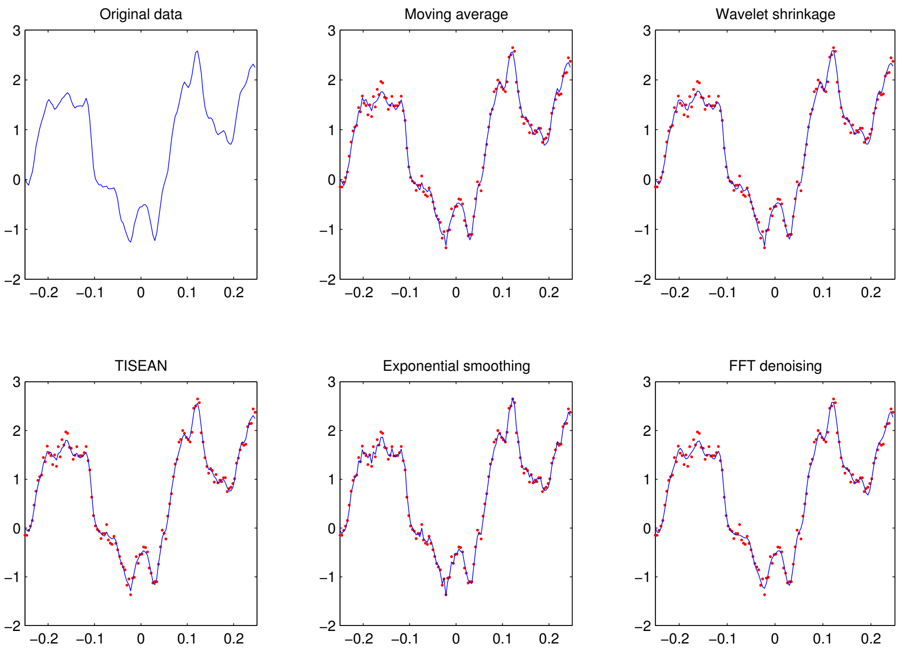


Figure 6: Denoising results for signal 5, the time series generated by an ARMA-model. The noisy signal is represented by dots, the denoised signal by solid line.

to nonlinear time series, the easiest methods perform best: the FFT filter and sometimes even the moving average yields good results.

One the other hand one should notice that the results for the different methods are closer together than for the other signals Thus the results does not have a great significance.

See Figure 6 and Table 5 for illustration of the denoising results resp. the presentation of the error measures.

Signal number 1, noise level 0.01

	L^1	L^2	L^∞	SE$_2$
Noisy data	0.160 ± 0.002	0.045 ± 0.001	0.042 ± 0.005	0.025 ± 0.000
MA filter	0.071 ± 0.001	0.020 ± 0.000	0.018 ± 0.003	0.014 ± 0.000
Wavelet filter	**0.030 ± 0.002**	**0.008 ± 0.001**	**0.015 ± 0.008**	**0.008 ±0.000**
TISEAN	0.060 ± 0.002	0.017 ± 0.001	0.018 ± 0.003	0.012 ± 0.000
Exp. Smoothing	*0.109 ± 0.002*	*0.030 ± 0.000*	0.027 ± 0.004	*0.017 ± 0.000*
FFT filter	0.058 ± 0.002	0.022 ± 0.000	*0.232 ± 0.005*	0.016 ± 0.000

Parameters:

Wavelet-Filter: `db8`-wavelet

TISEAN: embedding dimension 8, one iteration, neighbourhood size 0.1, delay time 314

Exponential smoothing: a = 0.5

FFT denoising: cut frequency 60.0

Signal number 1, noise level 0.05

	L^1	L^2	L^∞	SE$_2$
Noisy data	0.797 ± 0.009	0.223 ± 0.002	0.211 ± 0.037	0.102 ± 0.002
MA filter	*0.357 ± 0.007*	*0.100 ± 0.002*	0.093 ± 0.012	*0.049 ± 0.001*
Wavelet filter	0.144 ± 0.010	**0.041 ± 0.003**	**0.049 ± 0.018**	**0.025 ± 0.002**
TISEAN	0.248 ± 0.009	0.075 ± 0.002	0.094 ± 0.019	0.027 ± 0.001
Exp. Smoothing	0.347 ± 0.008	0.097 ± 0.002	0.089 ± 0.024	0.045 ± 0.001
FFT filter	**0.125 ± 0.009**	0.049 ± 0.002	*0.260 ± 0.009*	0.035 ± 0.001

Parameters:

Wavelet-Filter: `db8`-wavelet

TISEAN: embedding dimension 8, one iteration, neighbourhood size 0.2, delay time 314

Exponential smoothing: a = 0.3

FFT denoising: cut frequency 10.0

Signal number 1, noise level 0.10

	L^1	L^2	L^∞	SE$_2$
Noisy data	1.595 ± 0.019	0.447 ± 0.005	0.418 ± 0.054	0.202 ± 0.004
MA filter	0.714 ± 0.013	0.200 ± 0.004	0.185 ± 0.029	0.093 ± 0.003
Wavelet filter	0.289 ± 0.017	0.082 ± 0.005	**0.097 ±0.049**	**0.047 ± 0.003**
TISEAN	*1.443 ± 0.026*	*0.412 ± 0.007*	*0.418 ± 0.054*	*0.184 ± 0.004*
Exp. Smoothing	0.580 ± 0.013	0.162 ± 0.004	0.153 ± 0.054	0.073 ± 0.002
FFT filter	**0.183 ± 0.015**	**0.069 ± 0.003**	0.267 ± 0.013	**0.047 ± 0.002**

Parameters:

Wavelet-Filter: `db8`-wavelet

TISEAN: embedding dimension 8, one iteration, neighbourhood size 0.1, delay time 314

Exponential smoothing: a = 0.2

FFT denoising: cut frequency 5.0

Table 1: The denoising results for function 1, the sinusoid.

Signal number 2, noise level 0.01

	L^1	L^2	L^∞	SE_2
Noisy data	0.160 ± 0.002	0.045 ± 0.001	0.042 ± 0.005	0.025 ± 0.000
MA filter	0.071 ± 0.001	0.020 ± 0.000	**0.019 ± 0.004**	0.015 ± 0.000
Wavelet filter	**0.029 ± 0.002**	**0.009 ± 0.001**	0.024 ± 0.010	**0.010 ± 0.000**
TISEAN	0.095 ± 0.002	0.029 ± 0.001	*0.042 ± 0.009*	*0.026 ± 0.001*
Exp. Smoothing	0.122 ± 0.002	0.034 ± 0.000	0.031 ± 0.004	0.020 ± 0.000
FFT filter	*0.162 ± 0.002*	*0.045 ± 0.001*	*0.042 ± 0.006*	0.025 ± 0.000

Parameters:

Wavelet-Filter: `coif2`-wavelet

TISEAN: embedding dimension 7, one iteration, neighbourhood size 0.1, delay time 1

Exponential smoothing: a = 0.6

FFT denoising: cut frequency 500.0

Signal number 2, noise level 0.05

	L^1	L^2	L^∞	SE_2
Noisy data	0.797 ± 0.009	0.223 ± 0.002	0.211 ± 0.037	0.091 ± 0.002
MA filter	0.357 ± 0.007	0.100 ± 0.002	0.093 ± 0.012	0.046 ± 0.001
Wavelet filter	**0.143 ± 0.010**	**0.041 ± 0.003**	**0.060 ± 0.026**	**0.024± 0.002**
TISEAN	0.346 ± 0.011	0.097 ± 0.003	0.105 ± 0.036	0.045 ± 0.003
Exp. Smoothing	0.401 ± 0.008	0.112 ± 0.002	0.100 ± 0.020	0.045 ± 0.001
FFT filter	*0.798 ± 0.009*	*0.224 ± 0.002*	*0.212 ± 0.036*	*0.091 ± 0.002*

Parameters:

Wavelet-Filter: `coif2`-wavelet

TISEAN: embedding dimension 8, one iteration, neighbourhood size 0.2, delay time 1

Exponential smoothing: a = 0.3

FFT denoising: cut frequency 500.0

Signal number 2, noise level 0.10

	L^1	L^2	L^∞	SE_2
Noisy data	1.595 ± 0.019	0.447 ± 0.005	0.418 ± 0.054	0.177 ± 0.003
MA filter	0.714 ± 0.013	0.200 ± 0.004	0.185 ± 0.029	0.084 ± 0.002
Wavelet filter	**0.287 ± 0.017**	**0.082 ± 0.005**	**0.118 ± 0.060**	**0.041 ± 0.003**
TISEAN	0.637 ± 0.021	0.180 ± 0.006	0.178 ± 0.035	0.081 ± 0.005
Exp. Smoothing	0.651 ± 0.018	0.181 ± 0.005	0.164 ± 0.049	0.068 ± 0.002
FFT filter	*1.596 ± 0.019*	*0.447 ± 0.005*	*0.418 ± 0.054*	*0.177 ± 0.003*

Parameters:

Wavelet-Filter: `coif2`-wavelet

TISEAN: embedding dimension 8, one iteration, neighbourhood size 0.4, delay time 1

Exponential smoothing: a = 0.2

FFT denoising: cut frequency 500.0

Table 2: The denoising results for function 2, the piecewise quadratic polynomial.

Signal number 3, noise level 0.01

	L^1	L^2	L^∞	SE_2
Noisy data	0.160 ± 0.002	0.045 ± 0.001	0.042 ± 0.005	0.046 ± 0.001
MA filter	0.094 ± 0.001	*0.273 ± 0.000*	*4.403 ± 0.005*	*0.273 ± 0.000*
Wavelet filter	0.030 ± 0.002	0.009 ± 0.001	0.054 ± 0.020	0.012 ± 0.001
TISEAN	**0.002 ± 0.002**	**0.001 ± 0.000**	**0.000 ± 0.000**	**0.001 ± 0.001**
Exp. Smoothing	*0.160 ± 0.002*	0.045 ± 0.001	0.042 ± 0.005	0.046 ± 0.001
FFT filter	*0.160 ± 0.002*	0.045 ± 0.001	0.042 ± 0.005	0.046 ± 0.001

Parameters:

Wavelet-Filter: `db1`-wavelet

TISEAN: embedding dimension 1, one iteration, neighbourhood size 0.1, delay time 1

Exponential smoothing: a = 1.0

FFT denoising: cut frequency 500.0

Signal number 3, noise level 0.05

	L^1	L^2	L^∞	SE_2
Noisy data	0.797 ± 0.009	0.223 ± 0.002	0.211 ± 0.037	0.225 ± 0.002
MA filter	0.379 ± 0.007	*0.290 ± 0.001*	*4.418 ± 0.027*	*0.290 ± 0.001*
Wavelet filter	0.147 ± 0.011	0.043 ± 0.003	0.274 ± 0.100	0.053 ± 0.003
TISEAN	**0.010 ± 0.009**	**0.004 ± 0.002**	**0.068 ± 0.040**	**0.005 ± 0.003**
Exp. Smoothing	0.656 ± 0.008	0.202 ± 0.002	2.194 ± 0.086	0.204 ± 0.002
FFT filter	*0.796 ± 0.009*	0.223 ± 0.002	0.224 ± 0.043	0.225 ± 0.002

Parameters:

Wavelet-Filter: `db1`-wavelet

TISEAN: embedding dimension 5, one iteration, neighbourhood size 0.4, delay time 1

Exponential smoothing: a = 0.8

FFT denoising: cut frequency 490.0

Signal number 3, noise level 0.10

	L^1	L^2	L^∞	SE_2
Noisy data	1.595 ± 0.020	0.447 ± 0.005	0.418 ± 0.054	0.449 ± 0.005
MA filter	0.736 ± 0.014	0.337 ± 0.003	*4.429 ±0.049*	0.339 ± 0.003
Wavelet filter	0.295 ± 0.018	0.087 ± 0.005	0.539 ± 0.197	0.101 ± 0.005
TISEAN	**0.036 ± 0.014**	**0.017 ± 0.002**	**0.098 ± 0.041**	**0.018 ± 0.003**
Exp. Smoothing	*1.057 ± 0.014*	*0.347 ± 0.004*	4.401 ± 0.113	*0.349 ± 0.004*
FFT filter	0.967 ± 0.016	*0.347 ± 0.003*	3.669 ± 0.061	0.347 ± 0.003

Parameters:

Wavelet-Filter: `db1`-wavelet

TISEAN: embedding dimension 2, one iteration, neighbourhood size 0.4, delay time 1

Exponential smoothing: a = 0.6

FFT denoising: cut frequency 170.0

Table 3: The denoising results for function 3, the step function.

Signal number 4, noise level 0.01

	L^1	L^2	L^∞	SE_2
Noisy data	0.159 ± 0.002	0.045 ± 0.000	0.042 ± 0.007	0.035 ± 0.000
MA filter	0.082 ± 0.001	*0.154 ± 0.000*	*3.021 ± 0.006*	0.151 ± 0.000
Wavelet filter	**0.032 ± 0.002**	**0.010 ± 0.001**	0.054 ± 0.015	**0.011 ± 0.001**
TISEAN	0.048 ± 0.002	0.017 ± 0.001	0.049 ± 0.008	**0.011 ± 0.000**
Exp. Smoothing	*0.159 ± 0.002*	0.045 ± 0.000	**0.042 ± 0.007**	*0.035 ± 0.000*
FFT filter	*0.159 ± 0.002*	0.045 ± 0.000	**0.042 ± 0.007**	*0.035 ± 0.000*

Parameters:

Wavelet-Filter: `db4`-wavelet

TISEAN: embedding dimension 8, one iteration, neighbourhood size 0.1, delay time 313

Exponential smoothing: a = 1.0

FFT denoising: cut frequency 500.0

Signal number 4, noise level 0.05

	L^1	L^2	L^∞	SE_2
Noisy data	0.798 ± 0.010	0.224 ± 0.003	0.209 ± 0.027	0.165 ± 0.002
MA filter	0.367 ± 0.007	0.183 ± 0.001	*3.034 ± 0.030*	*0.168 ± 0.001*
Wavelet filter	**0.154 ± 0.008**	**0.048 ± 0.002**	0.264 ± 0.072	**0.046 ± 0.002**
TISEAN	0.274 ± 0.012	0.092 ± 0.004	**0.128 ± 0.020**	0.058 ± 0.003
Exp. Smoothing	*0.559 ± 0.007*	0.180 ± 0.002	2.645 ± 0.072	0.146 ± 0.002
FFT filter	0.583 ± 0.008	*0.197 ± 0.002*	1.915 ± 0.056	0.154 ± 0.002

Parameters:

Wavelet-Filter: `db4`-wavelet

TISEAN: embedding dimension 8, one iteration, neighbourhood size 0.2, delay time 1

Exponential smoothing: a = 0.7

FFT denoising: cut frequency 250.0

Signal number 4, noise level 0.10

	L^1	L^2	L^∞	SE_2
Noisy data	1.595 ± 0.020	0.447 ± 0.005	0.418 ± 0.055	0.327 ± 0.005
MA filter	0.724 ± 0.014	0.252 ± 0.003	3.051 ± 0.063	0.211 ± 0.003
Wavelet filter	**0.306 ± 0.016**	**0.095 ± 0.005**	0.497 ± 0.139	**0.085 ± 0.005**
TISEAN	0.530 ± 0.019	0.165 ± 0.006	**0.215 ± 0.053**	0.098 ± 0.004
Exp. Smoothing	*0.930 ± 0.013*	*0.294 ± 0.004*	*3.777 ± 0.120*	*0.229 ± 0.003*
FFT filter	0.784 ± 0.017	0.287 ± 0.003	2.980 ± 0.065	0.225 ± 0.003

Parameters:

Wavelet-Filter: `db4`-wavelet

TISEAN: embedding dimension 8, one iteration, neighbourhood size 0.3, delay time 1

Exponential smoothing: a = 0.5

FFT denoising: cut frequency 110.0

Table 4: The denoising results for function 4, the mixture of the first three functions.

Signal number 5, noise level 0.01

	L^1	L^2	L^∞	SE_2
Noisy data	0.160 ± 0.002	0.045 ± 0.000	0.042 ± 0.006	0.049 ± 0.001
MA filter	*0.188 ± 0.001*	*0.053 ± 0.000*	0.062 ± 0.011	*0.056 ± 0.000*
Wavelet filter	0.127 ± 0.002	0.036 ± 0.000	**0.034 ± 0.005**	0.041 ± 0.000
TISEAN	0.160 ± 0.002	0.045 ± 0.001	0.047 ± 0.013	0.049 ± 0.000
Exp. Smoothing	0.160 ± 0.002	0.045 ± 0.000	0.042 ± 0.006	0.049 ± 0.001
FFT filter	**0.122 ± 0.001**	**0.035 ± 0.000**	*0.148 ± 0.009*	**0.040 ± 0.000**

Parameters:

Wavelet-Filter: `db4`-wavelet

TISEAN: embedding dimension 5, one iteration, neighbourhood size 0.1, delay time 1

Exponential smoothing: a = 1.0

FFT denoising: cut frequency 242.5

Signal number 5, noise level 0.05

	L^1	L^2	L^∞	SE_2
Noisy data	0.797 ± 0.009	0.223 ± 0.002	0.211 ± 0.036	0.108 ± 0.001
MA filter	0.397 ± 0.007	0.111 ± 0.002	**0.105 ± 0.017**	0.081 ± 0.001
Wavelet filter	0.397 ± 0.008	0.112 ± 0.002	0.107 ± 0.018	0.081 ± 0.001
TISEAN	0.403 ± 0.008	0.113 ± 0.002	0.118 ± 0.035	0.083 ± 0.001
Exp. Smoothing	*0.669 ± 0.008*	*0.187 ± 0.002*	0.175 ± 0.026	*0.102 ± 0.001*
FFT filter	**0.369 ± 0.006**	**0.104 ± 0.002**	*0.265 ± 0.021*	**0.079 ± 0.001**

Parameters:

Wavelet-Filter: `db4`-wavelet

TISEAN: embedding dimension 8, one iteration, neighbourhood size 0.2, delay time 1

Exponential smoothing: a = 0.8

FFT denoising: cut frequency 57.5

Signal number 5, noise level 0.10

	L^1	L^2	L^∞	SE_2
Noisy data	1.595 ± 0.020	0.447 ± 0.005	0.418 ± 0.054	0.139 ± 0.002
MA filter	0.734 ± 0.014	0.206 ± 0.004	0.189 ± 0.027	0.100 ± 0.002
Wavelet filter	0.658 ± 0.016	0.185 ± 0.004	**0.178 ± 0.034**	0.095 ± 0.002
TISEAN	0.711 ± 0.016	0.200 ± 0.005	0.194 ± 0.031	0.102 ± 0.002
Exp. Smoothing	*1.161 ± 0.015*	*0.325 ± 0.004*	*0.303 ± 0.039*	*0.117 ± 0.002*
FFT filter	**0.585 ± 0.015**	**0.165 ± 0.004**	0.279 ± 0.044	**0.093 ± 0.002**

Parameters:

Wavelet-Filter: `db4`-wavelet

TISEAN: embedding dimension 8, one iteration, neighbourhood size 0.4, delay time 1

Exponential smoothing: a = 0.6

FFT denoising: cut frequency 50.0

Table 5: The denoising results for function 5, the time series, generated by an ARMA-model.

References

[1] Ronald R. Coifman and David L. Donoho. Translation-invariant de-noising. In A. Antoniadis and G. Oppenheim, editors, *Wavelets and Statistics*, volume 103 of *Springer Lecture Notes in Statistics*, pages 125–150. Springer-Verlag, 1995.

[2] David L. Donoho. Denoising via soft thresholding. *IEEE Transactions on Information Theory*, 41(3):613–627, 1995.

[3] James D. Hamilton. *Time Series Analysis.* Princeton University Press, 1994.

[4] R. Hegger, H. Kantz, and T. Schreiber. Practical implementation of nonlinear time series methods: The TISEAN package. *CHAOS*, 9:413–435, 1999.

[5] Bernd Jähne. *Digitale Bildverarbeitung.* Springer Verlag, Berlin, fifth edition, 2002.

[6] Karl-Dirk Kammeyer and Kristian Kroschel. *Digitale Signalverarbeitung.* Teubner, Stuttgart, fifth edition, 2002.

[7] H. Kantz and R. A. Schreiber. *Nonlinear time series analysis.* Cambridge University Press, 1997.

[8] Stéphane Mallat. *A Wavelet Tour of Signal Processing.* Academic Press, San Diego, CA, USA, 1999.

[9] J. S. Marron and A. B. Tsybakov. Visual error criteria for qualitative smoothing. *Journal of the American Statistical Association*, 90(430):499–507, June 1995.

[10] Michel Misiti, Yves Misiti, Georges Oppenheim, and Jean-Michel Poggi. *Wavelet Toolbox User's Guide.* The MathWorks, Inc., third edition, 2004.