The **Denoising** pop-up menu

Jacques Levy Vehel 12 January 2001

This text presents a brief explanation of the functionalities of the **Denoising** pop-up menu.

Contents

1 Overview

The Denoising menu includes various routines that allow to both denoise and regularize 1D and 2D data. Except for the well-known wavelet shrinkage, all other methods are based on a manipulation of the Hölder regularity of the input signal. The basic idea is as follows: it is intuitively clear that any signal that has at least some amount of regularity will undergo a decrease of its Hölder exponents when noise is added. Denoting X the original signal, B the noise, and Y the observations, we shall have that, in general, the *estimated* exponents of Y are "in between" those of X and B (this is not true of the theoretical exponents). A plausible denoising procedure is then to look for a signal Z which would minimize the risk subject to the constraint that its regularity is close the one of X. Usually, of course, the regularity of X is not known. We will thus be content with imposing equalities of the form: regularity of Z = regularity of Y + shift, or: regularity of Z = (regularity of Y)(1 + shift), where "shift" is some positive parameter. The same approach makes senses in a regularization framework: one then seeks a signal Z close to Y and with prescribed regularity. For various reasons, it is much easier to consider regularity in the sense of local Hölder exponents than pointwise ones. To allow for simple algorithms, the method is wavelet-based, i.e. we estimate regularity with the help of wavelet coefficients. These are modified and the updated coefficients are used to reconstruct the smoothed

signal (an alternative algorithm based on the use of genetic algorithms will hopefully be implemented in future releases of **Fraclab**).

In the current implementation of the denoising/regularization algorithms, you may deal with both 1D and 2D signals, in a transparent way: Just input your signal, and **Fraclab** will recognize its type.

2 The Multifractal Pumping sub-menu

The most obvious way to increase the local Hölder regularity by an amount d is simply to multiply all the wavelet coefficients at scale j by 2^(-dj). This roughly amounts to performing a fractional integration of order d (indeed, the local Hölder exponent is related to a notion of local fractional derivative). This sub-menu does exactly this. You just have to select the **Analyzed signal** and to adjust a **Spectrum shift value**, either by entering a value or by using the arrows. The spectrum shift value is the parameter d mentioned above (the justification for this denomination is that all exponents are increased by d, thus the whole multifractal spectrum is shifted to the right by an amount of d), and hit **Compute**. The output will be called *den_signal#*, and will be a regularized version of your original (1D or 2D) signal. It is interesting to note that you can input negative values for d, so that you may decrease the regularity of your signal. Also, this procedure is fully reversible, except for numerical round-offs. Try a large positive value for d (e.g. 4). The result will be a very blurred signal, which seems to contain almost no information. Since the blurred signal is the currently selected one in the **Variables** list, hit **Refresh** so that the denoising algorithm knows that you now want to process another signal. Select -d as a spectrum shift. You will recover your original signal.

Note finally that typical values for the shift in this case are around 0.5.

3 The Multifractal Denoising (L2 norm) sub-menu

In this menu, one makes the hypothesis that B is an additive white Gaussian noise. Then, we seek Z that minimizes the risk subject to the constraint that its regularity is that of Y plus a shift. One can show that this yields an asymptotically minimax estimator.

First specify your **Analyzed signal**, and the desired **Hölder exponent shift**. The algorithm needs to know the standard deviation of the noise. It is often a good idea to experiment with different values. To do this, check the **Specify** button, and enter a value of your choice for **Standard Deviation**. Alternatively, you may let the system estimate the power of the noise for you: Uncheck **Specify** so that **Automatic** appears instead. A new box called **Estimated Standard Deviation** pops up, in which the estimated value will be displayed when you hit **Compute**. The output signal is called *mden_signal#*. Typical values for the shift in this case are around 2.

4 The Multifractal Regularization (L2 norm) sub-menu

The framework is now that of regularization, i.e. we make no assumption whatsoever about a possible noise. We just seek a signal Z, which is close in the L2 sense to the observations Y (your input signal), and with

regularity equal to that of Y plus a shift. Just specify your **Analyzed signal**, and the desired **Hölder exponent shift**. Then hit **Compute** to get the regularized signal, called *mreg_signal#*. Typical values for the shift in this case are again around 2.

5 The Multifractal Regularization (pseudo Kullback norm) sub-menu

This is exactly the same as above, except this time we minimize the distance between Z and Y using the Kullback norm (or something close to it). The regularized signal is called *mreglog_signal#*. The advantages of this distance are that the computations are much simpler (the numerical minimization is replaced by an analytical one), and that it allows for further generalizations. For instance, the forthcoming version of **Fraclab** will include a multiplicative transform of the exponents instead of a shift, in the case where this distance is use. Typical values for the shift in this case are around 1.

6 The Wavelet Shrinkage sub-menu

This is the very well-known wavelet based denoising method (see reference (1)): The essential idea is that "meaningful" signals have their energy concentrated in few significant wavelet coefficients, while white noise, to the contrary, has coefficients which all behave the same in the statistical sense. The denoising is then performed by fixing a threshold and setting to 0 all coefficients which are below this threshold. Coefficients above it are shrinked towards 0, i.e. the value of the threshold is subtracted to them (if they are positive, with obvious modification for negative coefficients). With a right choice of the threshold, this procedure has been proved to be asymptotically rate-minimax for additive noise.

Again, specify your **Analyzed signal**, and the desired **Threshold Factor**. Then hit **Compute** to get the regularized signal, called *den_signal#*. Note that typical values for the threshold are around 0.1 or below.

7 Homework

An example of a multifractal denoising of a SAR image has been described in the homework section of the general help **Overview and main functionalities**.

8 References

(1) D.L. Donoho, *De-noising by soft-thresholding*, IEEE Trans. Inf. Theory 41, No. 3, 613–627 (1994).

(2) J. Lévy Véhel, B. Guiheneuf, *Multifractal image denoising*, Scandinavian Conference on Image Analysis, Finland, 1997.

(3) J. Lévy Véhel, *Signal enhancement based on Hölder regularity analysis*, Inria technical report, 1999.