Overview and main functionalities of **Fraclab**

Jacques Lévy Véhel 22 June 1998

This section presents an overview of the features available in **Fraclab**. A general presentation is made, followed by a brief explanation of the functionalities associated with each menu.

Contents

1 Overview

Fraclab is general purpose signal processing toolbox based on fractal and multifractal methods. It allows to perform many basic tasks in signal processing, including estimation, detection, regularization, denoising, modeling, segmentation and synthesis. Let us stress that **Fraclab** is not intended to process "fractal" signals (whatever meaning is given to this word), but rather to apply fractal tools to the study of irregular but otherwise arbitrary signals : just as e.g. gradient-based algorithms are often successfully applied for image segmentation even when there are no mathematical or physical reasons for the original signal to possess an ordinary derivative, a fractal analysis may yield useful insights for non "fractal" data. Of course, it does not in general give relevant indications when the signal is mainly regular or smooth, and reveals its interest only if there is enough singularity in the data.

A comparison with classical signal processing may be in order to make things clearer. In many cases, one assumes that the meaningful information is regular in essence, and that the irregular aspect of the observed data is due to noise coming from various sources: captor, thermal, coding, etc. A most useful tool is then filtering, using for instance Fourier analysis, in order to get rid of the noise. This approach has of course proven extremely valuable in many applications.

However, there are cases where the irregular part of the observed data contains useful information that cannot be recovered if only the smooth part is kept. It can even be the case that most or all of the relevant information is carried in the singular structure of the observation. Let us give some examples. It is well known that some useful information about a heart condition is contained in the "fractal dimension" (more precisely the correlation dimension, a feature related with the irregularity of the signal) of the ECG. The lower this dimension, the worse the condition of the heart. Although it is possible to assess the heart condition using classical methods, a regularity analysis seems to be a good alternative in this case. A second example is the case of radar images. These are difficult to process because of the presence of a specific noise, the speckle ("chatoiement" in French). However, speckle is not pure noise, but rather a genuine part of the signal, caused by the interferometric nature of radar images. In this respect, it contains information which is essential about the imaged region. Although removing the speckle can be useful for purposes of e.g. segmentation, analyzing it is a necessary task for other applications, as for instance classification, simply because the smoothed signal does not contain the necessary information. From a broader point of view, one may even argue that, though many image processing techniques aim at getting rid of irregularities in the data, the segmentation of simple, non noisy optical images should more logically be based on singularity analysis: one is indeed mostly interested in singularities, since edges are basically discontinuities in the grey levels. In that respect, the classical approaches, based on smoothing, do not appear as natural as is usually assumed.

Many tools in **Fraclab** are thus designed to measure different kinds of irregularity, and use these measures to perform signal processing. The regularity is analyzed either from a global point of view, or from a local one. In the first case, **Fraclab** allows to compute various fractional dimensions. In the second case, the Holder exponent is used. The exploitation of this local singularity information for signal processing can be performed in two different ways in **Fraclab** :

- By keeping all the information, which basically means that, starting from the signal s(t), one builds a new function a(t) which gives the Holder exponent of s at t. This is useful either when the singularity function a(t) is simpler than the original signal, or for purposes of e.g. detection or denoising.
- By using a global description of the singularity : while the use of fractional dimensions, such as the box or regularization dimension will be sufficient if the signal is "fractally homogeneous", in more general situations, a finer analysis is needed: Multifractal analysis aims at extracting higher level information from the singularity function associated with the signal, in cases where a(t) is as complex as, or more complex than s(t) (this happens for instance for self-affine functions), or if keeping trace of the singularity information at each point is not relevant (this is the case for instance in issues of classification): In these situations, one computes a multifractal spectrum, which yields a global characterization of the singularity structure. Usually, statistical or geometrical descriptions are used, leading to various multifractal spectra. These multifractal spectra are for instance useful in classification problems or in image segmentation.

2 Description of Fraclab Functionalities

2.1 Introduction

Fraclab can be approached from three different perspectives : *synthesis of fractal signals*, *fractal analysis*, and *signal processing*. This separation is artificial in a sense, since the tools associated with these three streams overlap greatly, but it is conceptually helpful. Most functionalities can be accessed either from a fractal analysis or from a signal processing point of view, and this help file will reflect this situation.

In order to make **Fraclab** user-friendly, a graphic interface, called **fltool**, is provided with this version. We describe briefly in the next sections the general organization of **fltool**, as well as the main features of the synthesis, analysis and signal processing tools, as they appear in the menus of **fltool**.

2.2 The main window of fltool

Once you launch **fltool**, the main window appears. It is divided into four zones :

- UPPER PART : the pop-up menus

  The *pop-up menus* allows to perform the various processings available in **Fraclab**. These are briefly described in sections 2.3 below and detailed in the corresponding parts of this help.

- UPPER MIDDLE PART : the Variables and Details windows

  The basic elements one manipulates in **Fraclab** are *structures*: The synthesis and the analysis tools all produce and process structures. A structure is a composite piece of information which may comprise matrices of different size and other more complex elements. The upper middle part of the **fltool** graphic interface is composed of two windows : the *Variables* window, which will display the name of the structures generated in the course of using **Fraclab**. The highlighted name corresponds to the current active structure (i.e. the one on which the processings are made). The *Details* window shows the building blocks of the structure currently highlighted in the Variables window. To select a block,

i.e. a sub-element of the structure (which may for instance be a matrix), just highlight it. This is useful for instance for purposes of visualization or if one wants to extract a particular building block from a composite structure.

- LOWER MIDDLE PART : files manipulation

  Scan Workspace allows you to transfer files from your matlab environment (workspace) into the **fltool** environment, while *Load* allows you to transfer files from your file system into the **fltool** environment. Files formats that are currently recognized include mat-files, 1D ASCII signals and 2D tif images. *Save* allows you to save structures from **fltool** into your file system. Finally, *Clear* removes structures from both the **fltool** and the matlab environment.

- LOWER PART : Miscellaneous tools

  View: This important menu allows to perform various display-related operations which are detailed below.

  Help lists most of the routines available in **Fraclab**. To get a detailed help on a particular function, double click on it.

  Quit is what you guess.

  Finally, you may want to check the *Message* zone often, as error messages are usually displayed here. In many occasions, a beep is heard when a new message is sent. The *Erase* button lets you clear the message zone.

2.3 Pop-up Menus Description

There are four kinds of pop-up menus : the first one allows to perform **synthesis** of "fractal" signals. The second one deals with the **fractal and multifractal analysis** of signals and includes : *Fractional Dimensions*, *1D Exponents Estimation*, *1D signals Multifractal Spectra*, and *Stable Motion*. The third group is related to **Signal Processing** and is composed of *Segmentation* and *Denoising*. Finally, **miscellaneous tools** are available, namely *Time Frequency and Time Scale analysis* and *Misc*, which allows to perform various editing tasks. Let us describe briefly what can be done with these various menus. More detailed explanations are given in the appropriate sections of this help.

2.3.1 Synthesis

Two types of signals can be generated : measures (i.e. an array of non negative data that add to one) or functions. Measures are interesting in particular when one needs to take into account the resolution in an explicit way. For both measures and signals, either deterministic or stochastic data may be generated. By and large, this menu allows to synthesize a substantial subset of all classical fractal models described in the literature : 1D and 2D fBm-s, mBm-s, (generalized) Weierstrass functions, stable motions, Wavelet based 1/f process, multifractal measures, ...

2.3.2 Fractal and Multifractal Analysis

The most basic parameters that can be computed are of course fractional dimensions. In the current implementation of **Fraclab**, only the box and regularization dimension are available. When one needs a local characterization of the signal, Hölder exponents are more pertinent, and the menu *1D Exponents Estimation* allows to estimate both pointwise and local exponents. In addition, a long range exponent may be computed, as well as 2-microlocal exponents. *1D signals Multifractal Spectra* offers various estimations procedures. Note that the computation of Multifractal Spectra for 2D data is possible using the *Segmentation* menu. Finally, *Stable Motion* allows to test the stability of a given process and to estimate the relevant parameters.

2.3.3 Signal Processing

Segmentation allows to segment both 1D signals and images. In the former case, a modeling based on a generalization of IFS, called weakly self affine functions, is used. Images are segmented into edges or regions of given regularity through multifractal analysis. *Denoising* allows to regularize and denoise 1D or 2D data using various methods.

2.3.4 Miscellaneous tools

TF-TS allows to compute various time frequency representations of a signal, while *misc* offers basic structure manipulations such as sums, extractions, ...

3 The View menu

Clicking on the *View* button opens a new window, which serves as a control center for all the displays (graphs + images) you might want to have. This window is composed of four sub-windows : *Figure*, *Image mode*, *Tools*, and finally a originally blank region which will contain the list of all opened figures along with the data displayed in each figure. This list allows you to select which figure or signal is currently active by clicking on it. Only the active element will be affected by the various commands available in the other sub-windows of the *View* menu.

It is important to understand the difference between a *View* and the data displayed in it. A *View* corresponds to a Matlab figure that will display one or several graphs or images. The case of several graphs corresponds to the use of the *sub-plot* command in Matlab. Since, in the case of multiple plots, you might want to apply a different processing to each sub-plot, you need to tell **Fraclab** which sub-plot is the current one. This is why the list at the bottom of the *View* window will show the name of all the opened views (i.e. Matlab figures) numbered in the chronological order of their appearance, and, for each of these views, the name of the signals that are shown in this figure as plots or sub-plots. This will allow you to select either a whole figure or one of its sub-plot. For instance, some viewing options such as *hold*, *rotate* or *zoom* are available only when a particular signal is highlighted, and are grayed out when the whole view is highlighted. Note finally that clicking in the list either on a view or on one of the signals that it contains will bring the corresponding window to the foreground, a useful feature if you have many opened windows or/and if you do not remember which signal is which.

3.1 The Figure sub-window

view in new : opens a new figure which will display the highlighted data in the *Variables* zone of the main window.

view : displays the highlighted variable in the current figure, i.e. the one which is highlighted in the figures list. Depending on whether the *hold* button in the *Tools* sub-window is selected or not, pressing *view* will replace the current plot or will be placed on top of it.

close all : Usually, you will probably find it more convenient to keep the *View* window opened at all time. However, pressing this button will close all figures, control panels and toolbar.

3.2 The Image mode sub-window

This menu is active only when an image is selected in the list, and is grayed out when either a *view* is selected (because **Fraclab** needs to know which image must be processed and not merely which view is the current one) or a 1D signal is currently active (indeed, the processings in this sub-window make no sense for 1D signals).

image control opens the image control panel (note that *image control* is grayed out when the image control panel is already displayed). This panel first recalls which data is being processed in the *plot* line.

The *mode* button then allows to choose how the data should be displayed : as an image, a contour plot, a mesh, a surface, or as a superposition of 1D signals which might be its lines or columns. In addition, a particular line/column might be selected for display.

colormap selects a color map for the display.

dynamic allows to toggle between a linear and a logarithmic dynamic for both the numerical and graphical display of the data.

min level, *max level* : these boxes allows you to enter the interval in which the data must fall in order to be displayed. Pixels with grey levels outside this range will be clipped.

value governs the way the data are **numerically** displayed, for instance in the *min level* and *max level* boxes, or in the *get point* box : *normalized* will forces the output to be between 0 and 1, while *true* causes the real values to be displayed

display toggles between a *normalized* and a *true* display for the **grey levels** in the image.

binary displays all the pixels with grey level between *min level* and *max level* in white and all others in black.

reverse revert the order of the color map. This is particularly useful in conjunction with *binary* to toggle between "in" and "out" pixels.

get point : pressing this button will bring the image figure to the foreground and display a cross that you may center at any point. Clicking will cause **Fraclab** to display the coordinates of the chosen point along with its grey level in the box to the right. Whether the displayed grey level will be the true one, a normalized one, or a binary one depends on the choices described above. Note that it is possible to zoom in the image

before activating *get point* to gain more precision. As said in the *Message* line of the main window, press Enter in the image to get out of this mode, or click again on *get point*.

axis : toggles the aspect ratio between the true axis of the image and the Matlab default.

x axis - y axis : enables to load a new array to use as axes, when the data are displayed as anything but an image.

Note that the chosen modifications do not result in an immediate update of the display. Rather, you need to press *Apply* to see the effects of your choices.

The second button in *Image mode* is *superpose*. This allows to lay an image on the top of another one with a control of the "transparency". This works like a kind of *hold* facility for images, with additional functionalities. The main purpose of this is to compare an original image with e.g. its segmented version. More generally, you'll find this feature helpful when you need to compare two images one of which (at least) is "sparse" (like is a image of contours). Note however that you may perfectly use it for any kinds of images, for instance to compare consecutive images in a sequence. In practice, this is what you do: first display an image and select it in the views list. The *superpose* button should now be clickable. Select another image in the **Variables** list of the main window and press **view**. A new window titled **Superpose Parameters** appears, that allows to choose two parameters : the *lut ratio* decides how many entries in the current color map will be available for each image, while the *threshold* parameter controls the "amount" of each image that will actually be displayed. Let us detail this a little bit: The *lut ratio* may be varied between 0.5 and 1, i.e. the first image will used between one half and the whole of all available colors (or grey levels). Of course, when *lut ratio* = 1, you will not see the second image. Assume to simplify that your color map has 100 entries and that *lut ratio* = 0.8. Then prior to displaying, the first image will have its grey levels linearly re-mapped between 0 and 80, and the second one between 80 and 100. Second, if *threshold* is 0, only the second image will be displayed, while if *threshold* is 1 - *lut ratio* (this is the maximum possible value), you'll see only the first image. Setting *threshold* between these extremes allows you to balance the relative strength of the two images. Choose values for the *lut ratio* and the *threshold* and press **Apply** to see the effects of your selections. Experiment with other values until you are happy with the result. You may then **Close** the **Superpose Parameters** window.

Two known bugs of the *superpose* mode are the following: when you press **Apply**, the scalings of the axes are changed to the default matlab aspect ratio. Second, if you de-select *superpose*, the **Superpose Parameters** window does not close. This would not be a problem, except that it will yield an error next time you will press the **Apply** button. Thus, when you de-select *superpose*, remember to close also the **Superpose Parameters** window.

3.3 The Tools sub-window

hold causes the current plot to be held, so that subsequent graphs are displayed on top of it.

rotate interactively rotates the view of a plot.

zoom allows to zoom on the current plot or image by selecting a region with the mouse.

Vsplit performs a vertical split of the figure in as many sub-plots as are specified in the box to the right. *Hsplit* does the same for horizontal sub-plots. Thus, to have a figure with m horizontal sub-plots and n vertical sub-plots, enter m in the *Hsplit* box, n in the *Vsplit* box, select one by one the desired graphs in the

Variables window and press each time *view* : the data will be displayed sequentially in the corresponding sub-plots.

Pressing *axes* opens the *axes control panel* which allows to set up various parameters : the *Scale type* chooses between linear, semi-logarithmic or bi-logarithmic plots, the *X range* and *Y range* decides which parts of the data are to be displayed. For 1D signals, various *Aspect* parameters can be specified :

mark changes the symbol used to draw the computed points, *line* selects the symbol used to draw lines between the computed points, *color* defines the color of the plot, *width* controls the width of the line. Finally, the boxes *red* - *green* - *blue* allows to finely adjust the color of the line.

print opens the print control panel that allows to save the figure or print it with various options.

close figure closes the selected view and all its sub-plots.

4 General conventions and remarks

We gather in this section a number of basic facts about the general behaviour of **Fraclab**.

In many instances, you will have to choose the *size* of the signal you want to generate or process. Most of the time, you can either use predefined values (often, these will be powers of 2), or simply type any positive integer in the appropriate zone.

When you process the signal called *sig* with routine *rout*, the output signal (if any) will be called *rout_sig#*, where # is a number initially set to 0 and that increments each time you launch the same procedure on the same input signal.

Most windows at the final level (i.e. the ones that will actually launch computations) display three buttons in their lower part : **Compute** will launch the requested computation, **Help** will open a window displaying the technical help associated with the **Fraclab** routine involved, and **Close**.

When you choose a sub-menu that will take a signal or structure as an input, **Fraclab** assumes most of the times that the signal you want to process is the one that is highlighted when you call the sub-menu window. Thus, sometimes you will get an error if you have not been careful and checked that the current signal is in the right format for the routine. Assume for instance that you synthesize a Weierstrass function. You will get two outputs: *Wei#* is a 1D signal, while *GraphWei#* is a graph, i.e. a structure composed of 5 elements. However, when the synthesis is completed, the current variable is *GraphWei#*. Thus, if you launch for instance **GIFS based estimation** in the **Pointwise Hölder Exponent** sub-menu of the **1D Exponents Estimation**, you'll get a beep and an error message. This is because **GIFS based estimation** only knows how to process 1D signals and cannot deal with complicated structures like *GraphWei#*. Thus, you should select in the **Variables** list *Wei#* before you launch **GIFS based estimation**. In any case, it is always a good idea to check the **Message** zone of the main window if you suspect something fishy happened or if you don't get what you expected. Do not forget to **Erase** a message once you have read it.

Finally, note that in this help file, **Fraclab** commands, sub-windows names and parameters appear in **boldface**, while output signals are *italicized*.

5 Known bugs

In some occasions, the whole matlab session becomes very unstable. This may happen when too many errors have appeared, or if you have worked a long time and used a lot of memory that matlab has not been able to free up, etc... If strange things start to happen like you cannot even synthesize a simple signal with **Fraclab** or launch a simple command in matlab, it is advised that you simply quit the matlab session and start afresh.

When you perform an invalid operation in a given window, the cursor turns into a watch when you point inside this window, and remains so even if you subsequently launch valid computations, until you close the window in which this happened.

For some reasons, Matlab sometimes flips or rotates the image around before displaying it. Thus it may happen, when you try to view the output of e.g. a denoising of an original image, that the result seems weird. You just have to remember that the data may have been rotated.

6 Homework

In many of the help files associated with the various menus of **Fraclab**, you'll find a "homework" section that describes an example of application of the corresponding tools. This is intended to help you get started with **Fraclab** and to show the possibilities of a fractal approach to signal processing.

In this general help, we highlight three examples taken from the following menus: **1D exponents estimation**, **Denoising** and **Segmentation**. See the corresponding helps for more details.

6.1 Analysis of a stock market log

The stock market is a fascinating area for fractal analysis. Many authors have argued that models based on stochastic processes exhibiting long dependence and/or infinite variance are relevant in this area. **Fraclab** can compute both long range dependence exponents and various parameters that characterize processes without a second moment. However, the main focus in **Fraclab** is on the measure of local regularity: independently of any assumption of long dependence and/or infinite variance, it is certainly true that stock market logs are very irregular. Moreover, this irregularity is a function of time, and we expect that, for instance, at "quiet" periods, the market should evolve smoothly, while krachs translate into sudden changes in the regularity.

A nice illustration of the above intuition is provided by the analysis of the Nikkei225 index during the period 01/01/80 to 05/11/2000. The log consists in 5313 daily values corresponding to that period. Load first these data into **Fraclab**: Press the **Load** button in the main window. A new window appears, showing the files of your current directory. Change directory to the **DATA** directory that comes with the **Fraclab** release. Choose the file called *nikkei225.txt* by clicking on it. Its name is then displayed at the top of the window, in the **Name:** box. Since this file is plain text, click on the button to the right of **Load as:**, and select the item **ASCII**. Then press **Load**, and **Close** the loading window. The *nikkei225* file should appear in your **Variables** list of the main window, under the name *fnikkei225*. View this signal: Open the **View** window by pressing on the **View** button. In the **View** window, click on **View in new**. This will open a window

displaying the stock market log. Like most data of this type, this signal is quite erratic. Other obvious features include a steady increase at the beginning of the log, and strong discontinuities around the points 1780, 2040, 2650, 2760 or 3200. Let us see if we can highlight these and other significant events with a local regularity analysis.

Financial analysts do not work directly on the prices, but on their logarithms, so we'll first type *lnikkei = log(fnikkei225);* in the matlab window, and import *lnikkei* into **Fraclab**. To do this, press the **Scan Workspace** button in the main window. In the new windows that appears, titled **Import Data from MATLAB Workspace**, locate the signal *lnikkei*, select it by clicking on it, and hit **Import**, then **Close** this window. *lnikkei* will appear in the **Variables** list of the main window, under the same name.

We will now estimate the local regularity of *lnikkei*: Click on **1D Exponents Estimation** and choose **Local Hölder Exponent** then **oscillation based method**. In the window that appears, check that the **Input data** is *lnikkei*. Otherwise, select *lnikkei* by clicking on it in the **Variables** list of the main window, and hit **Refresh** in the **Local Hölder Exponent** window. Set the parameters as follows: **Nmin = 1**, **Nmax = 8**, **Neighbourhood size = 16**, and regression type = **Least Square** (see the help file corresponding to this menu for details on the meaning of these parameters). Hit **Compute**, and wait for less than a minute. The output signal appears in the **Variables** list of the main window, and is called *pht_lnikkei0*. View this signal, by pressing **View in new** in the **View** menu (check that *pht_lnikkei0* is selected before doing so). As you see, most values of the local Hölder exponent are between 0 and 1, with a few peaks above 1 and up to more than 6. Recall that a Hölder exponent between 0 and 1 means that the signal is continuous but not differentiable at the considered point. In addition, the lower the exponent, the more irregular the signal. Looking at the original signal, it appears obvious that the log is almost nowhere smooth, which is consistent with the values of *pht_lnikkei0*. What is more interesting is that important events in the log have a specific signature in *pht_lnikkei0*: periods where "things happen" are characterized by sudden increase in regularity, which passes above 1, followed by very small values, e.g. below 0.2, which correspond to low regularity. Let us take some examples. The most prominent feature of *pht_lnikkei0* is the peak at abscissa 2018 with amplitude larger than 6. Note also that the points with the lowest values in regularity of the whole log are located just after this peak: The Hölder exponent is around 0.2 at abscissa roughly between 2020 and 2050, and 0.05 at abscissa between 2075 and 2100. Both values are well below the mean of *pht_lnikkei0*, which is 0.4 (its variance of is 0.036). As a matter of fact, only 10 percent of the points of the signal have an exponent smaller than 0.2. Now the famous October 19 1987 krach corresponds to abscissa 2036, right in the middle on the first low regularity period after the peak. The days with smallest regularity in the whole log are thus logically located in the weeks following the krach, and one can assess precisely which days were more erratic. However, if you go back to original *fnikkei225* signal, things are not so clear: although the krach is easily seen as a downward discontinuity at abscissa 2036, the area around this point does not appear to be more "special" than, for instance, the last part of the log (you may zoom on the different areas for easier visualization).

Consider now another region which contains many points with low Hölder exponents with a few isolated very regular points (i.e. with exponent larger than 1). Look at the area between abscissa 4450 and 4800: This roughly corresponds to the "Asian crisis" period, which approximately took place between January 1997 and June 1998 (there are no exact dates for the beginning and end of the crisis. Some authors place the beginning of the crisis mid-1997, and the end by late 1999, or even later). On the graph of the original log of the Nikkei225, you can see that this period is quite erratic, with some discontinuities and pseudo-cycles (this behaviour arguably seems to extend between points 3500 and maybe the end of the trace). Looking

now at *pht_lnikkei0*, we notice that there are two peaks with exponents larger than one in the considered period (there is an additional such point around abscissa 4300, which, however, is not followed by points with low values of regularity -e.g. smaller than 0.15-, but is preceded by such points, between abscissa 4255 and 4285). The first peak is around 4455, and is followed by irregular points between 4465 and 4475. The second is around 4730. This region, between abscissa 4450 and 4800, has a large proportion of irregular points: 12 percent of its points have exponent smaller than 0.15. This is three times the proportion observed in the whole log. In addition, this area is the one with highest density of points with exponent smaller than 0.15 (we exclude in these calculations the first and last points of the log, because of border effects). A nice way of seeing this is to zoom on the graph of *pht_lnikkei0* to display only the ordinates between, say, 0.05 and 0.2. This can easily be done using the **axes control** facility in the **View** menu of **Fraclab** by selecting the appropriate **Y range** (don't forget to hit **Apply** so that your settings take effect).

Although the discussion above is overly simplistic, it shows that strong perturbations in this particular financial log generally correspond to regions with very low values of the local regularity, with most of the times the presence of a single or a couple of extremely regular points. Such a behaviour has been observed in a large number of other logs. You may care to try the same kind of analysis on your own signals. Chances are that "interesting" regions, or points, will have a specific signature in the regularity graph: The evolution of the Hölder exponents brings an information which is, in some situations, perhaps more intrinsic than the amplitude of the original signal.

Before leaving this signal, let us compute its "long range dependence" exponent. More precisely, **Fraclab** allows you to compute an exponent that describes the power law behaviour of the frequency spectrum of the increments of the signal around the origin (i.e. at low frequencies), of course assuming that such a power law holds. If this is the case and if the exponent is larger than 1/2, one says that the data display long range dependence (LRD), in the sense that the correlations decay "slowly" when the time lag increases. The LRD exponent estimator in **Fraclab** is a wavelet-based one. Select first *lnikkei* in the **Variables** list, then go to **1D Exponents Estimation** and choose **LRD Exponent**. In the window that appears, choose **Advanced Compute**. A new window pops up, titled **Long Range Dependence Parameter**. Check that the **Input Signal** is *lnikkei*, and modify the **Voices** parameter from its default 128 to 64, just to speed up a little bit the computations. Then hit **Compute WT**. This will launch the computation of the continuous wavelet transform of *lnikkei*, using the complex Morlet wavelet as an analyzing wavelet, and with the various parameters specified in the window (see the help corresponding to this area of **Fraclab** for more). When the computation is over, you should see a new structure in the **Variables** list, called *cwt_lnikkei0*. This is the continuous wavelet transform of *lnikkei*, that you may care to visualize in the usual way: hit **View in new** in the **View** menu (if the **View** menu is not opened, hit **View** in the main window). *cwt_lnikkei0* should also appear in the box facing **Input CWT** in the lower part of the **Long Range Dependence Parameter** window. Now hit **Compute** at the bottom of this window. A new window appears, showing a graph where abscissa represent the logarithms of the scales in the wavelet transform, and ordinates are estimates of the logarithms of the energy in the signal at the corresponding scale. The red line is the least square regression line corresponding to the displayed circles. You'll see that the linear fit is poor when the whole graph is considered, as it is here. Since we are interested in LRD, we should however restrict our attention to large scales. Using the black cross that appears when you point inside the graphic window, select the region between abscissa 3 and 7: put the pointer at any point above abscissa 3, click, then put the pointer at any point above abscissa 7, and click again. The red line should now fit approximately the part of the graph above these abscissa. The **Estimated Global Scaling Exponent** displayed above the

graph should be around 0.56. You may try to compute the least square regression line above other parts of the graph by repeating the same steps. When you're finished, hit **Return** on your keyboard, as indicated on the lower right part of the graphic window. This will make the cross disappear and will display the last estimated value of the exponent at the bottom of the **Long Range Dependence Parameter** window, in the box facing **Scaling Exponent**. To make the graphic window go away, close it manually in the usual way (i.e. not with the help of the **View** menu). According to this estimate, thus, our financial log exhibit a slight long range dependence, because the exponent is a bit above 0.5.

6.2 Synthetic Aperture Radar image denoising

SAR images are generally difficult to read and to analyze because they contain a large amount of a specific noise, called speckle. Dozens of methods have been proposed to enhance their quality. Some use precise knowledge about, e.g., the statistics of the noise, while other are rather generic. The fractal denoising method is based on the following simple observation : consider an image I, and its noisy version J. Pick a particular location (x,y) at random in the image. Then, chances are that the local regularity of I around (x,y) will be larger than the one of J. Of course, this statement is rather imprecise if we do not define how we measure regularity. We will however content ourselves here with the intuitive fact that adding noise decreases the local regularity at all points. Denoising can then be performed by increasing in a controlled way the local regularity. This is exactly how the fractal method works.

To see a practical example of this, first load a SAR image into **Fraclab** by following these steps: Press the **Load** button in the main window. A new window appears, showing the files of your current directory. Change directory to the **DATA** directory that comes with the **Fraclab** release. Choose the file called *sar.tif* by clicking on it. Its name is then displayed at the top of the window, in the **Name:** box. Check that the **Load as:** box displays the item **image**, and press **Load**. Then **Close** the loading window. The *sar* image should appear in your **Variables** list of the main window, under the name *im2d_0* (or *im2d_1*, etc...). View this image: Open the **View** window by pressing on the **View** button. In the **View** window, click on **View in new**. This will open a window displaying the SAR image. As you'll see, this image appears very noisy, and does not seem to hold any useful information. However, this is not quite true, as this scene does contain a river flowing from North to South. Our aim here is to perform a pre-processing that will enhance the image so that it will be possible to detect automatically the river. Such a procedure is used by the IRD, a French agency, which, in this particular application, is interested in monitoring water resources in this region of Africa.

Go to **Denoising**, and choose **Multifractal Pumping**. In the new window that appears, check that the name appearing in the **Analyzed signal** box is *im2d_0*. Then choose a **Spectrum shift value**, either by using the sliders or by entering directly a value. A value of 1.5 will give you an interesting result at this stage. Press **Compute**. The processing is fast (probably less than a second). You should see a new signal in the **Variables** list, called *den_im2d_00*. Display this image by clicking **View in new** in the **View** menu. If everything went right, you should be able to distinguish some structures on the processed image. Most prominently, the river now appears, flowing from the top of the image and assuming roughly an inverted "Y" shape. Other values of the **Spectrum shift value** around 1.5 may give more visually pleasing results.

Here are some other tests worth trying. A characteristic feature of the **Multifractal Pumping** is that it is invertible. A striking illustration is to denoise the SAR image with a large **Spectrum shift value**, say 5. You obtain as an output the "enhanced" image *den_im2d_01*, say. View *den_im2d_01*, and notice

that it is very blurred, and thus seems to contain even less information than the original data. Now, with *den_im2d_01* selected in the **Variables** list, hit **Refresh** in the denoising window, so that **Fraclab** knows that you want to process this new signal. Enter -5 as **Spectrum shift value** (that is, instead of denoising, you "increase the noise"). View the output, called *den_den_im2d_010*. You'll see that this last image exactly coincides with *im2d_0*.

A final test is to compare this **Multifractal Pumping** with the classical wavelet shrinkage method. Wavelet shrinkage is a denoising procedure that gives excellent results for data corrupted with independent additive noise, provided the original signal has some minimum regularity. In our case, the noise is non additive and strongly correlated with the signal, which, furthermore, has no *a priori* regularity. Thus, we do not expect this method to behave well here. Go to **Denoising**, and open the **Wavelet shrinkage** window. Check that the name appearing in the **Analyzed signal** box is *im2d_0*. Otherwise, select the signal *im2d_0* in the **Variables** list and hit **Refresh** in the **Wavelet shrinkage** window. Choose a **threshold value** and hit **Compute**. No matter which value you choose for the threshold, the output signal never appears really "denoised".

6.3 Optical image segmentation

This is intended to show how multifractal analysis may be used for edge detection. Very roughly speaking, the multifractal analysis of a signal or an image consists in two steps: One first compute the Hölder exponents of each point in the signal. Second, one groups all points with the same exponent, say t, to form a set E(t). The multifractal spectrum is the function that associates to each t the "dimension" of the set E(t). In other words, multifractal analysis computes, for each singularity exponent, the "size" of the set of points in the image where this exponent is found.

To apply multifractal analysis to edge detection, we thus start by characterizing the local regularity of the image around each point by its Hölder exponent. Edge points are usually irregular points, so we expect them to have low Hölder exponent. This is true however when one measures the local regularity in the "usual way", i.e. by comparing the grey levels in a given zone with the size of this zone. In this experiment, we use a different measure of regularity: We associate to each region in the image the maximum of its grey levels, and we record the regularity of this quantity. More precisely, we do the following: Around each point in the image, we center windows of increasing size. We "measure" the content of each window by the maximum of the grey levels in the window. The regularity exponent is then obtained by evaluating the scaling law between the logarithms of the maxima and those of the window sizes. It is easy to see that smooth regions will now have a low regularity exponent, while textured zones have a large exponent. For instance, in a zone with constant grey levels, the maximum will not depend on the window size, thus the scaling exponent is 0 (see the references given in the help file of the **Segmentation** menu for more).

Let us try this on an optical image. Load first the image called *door.tif* into **Fraclab** by following these steps: Press the **Load** button in the main window. A new window appears, showing the files of your current directory. Change directory to the **DATA** directory that comes with the **Fraclab** release. Choose the file called *door.tif* by clicking on it. Its name is then displayed at the top of the window, in the **Name:** box. Check that the **Load as:** box displays the item **image**, and press **Load**. Then **Close** the loading window. The *door* image should appear in your **Variables** list of the main window, under the name *im2d_0* (or *im2d_1*, etc...). View this image: Open the **View** window by pressing on the **View** button. In the **View** window, click on **View in new**. This will open a window displaying the *door* image. This is an image of a

Japanese door (more precisely, a *toryi*).

Click on the **Segmentation** pop-up menu and select **Image multifractal segmentation**. In the new window that appears, click on **Refresh** on the first line, in front of **Analyzed** (check before that *im2d_0* was selected in the **Variables** list of the main window). *im2d_0* should appear on the first line, meaning the it will be the analyzed image. Ignore the three lines below, and move the **Pointwise Hölder exponent** zone. Note that the **max** capacity is selected: This corresponds to the fact that we will be measuring the content of a region by its maximum grey level, as explained above. You will now change the **max size** parameter from 5, its default, to 3. Do this by selecting 3 in the pop-up list that appears when you click on 5. Next, hit **Compute Hoelder**. After a few seconds, a new signal should appear in your **Variables** list, called *hld2dCoef_im2d_00*. This is the image of the Hölder exponents. You can view this image by clicking on **View in new** in the **View** menu. Notice that the image of the Hölder exponents gives a nice representation of the main edges of *im2d_0*. As explained above, pixels with low regularity, as are edges, appear as bright points, while smooth regions have low grey levels.

Technically, however, *hld2dCoef_im2d_00* does not represent an edge extraction of the original image, because edge images are supposed to be binary images: edge points are displayed in white, while all other points are in black. In this easy example, it seems that a simple threshold could turn *hld2dCoef_im2d_00* into a binary image that would coincide more or less with the contours. We will follow another path here, by using the second part of multifractal analysis. We thus proceed to compute the multifractal spectrum of our image, i.e. the function that will give the dimension of the sets of pixels having a given exponent. There are several type of multifractal spectra, and we will use the **Hausdorff** one, which is the default in **Fraclab**. In the zone facing **max boxes**, replace the default 32 by 128. This will yield a more precise spectrum. Hit **Compute spectrum**. The new signal *hSpectrum_im2d_0_fd2d_alpha0* is added to the **Variables** list. View this signal by clicking on **View in new** in the **View** menu. This is a 1D graph: abscissa represent the various Hölder exponents present in the image, while ordinates display the associated dimensions. Thus, for instance, the dimension 2 corresponding to the exponent 0 means that a subset of pixels of dimension 2 have exponent 0. Since no other exponent has associated dimension 2, this means that "most" points (more precisely Lebesgue-almost all points) in the image have exponent 0. Recalling the 0 is the exponent of points in smooth regions, we recover the visual fact that, in the door image, most points lie in smooth regions. Notice the shape of the spectrum, which is roughly a decreasing segment starting from the point (0, 2) and ending at (x, 0) (here, x= 0.9). This shape is characteristic of optical non noisy images when one measure the exponents using the maxima of the grey levels. Here, we are interested in contours. Let us see how we can detect them using the spectrum: Since contours must form a set of dimension 1 in the image (because contours are smooth curves), and because we expect contours to be made of points which have roughly the same regularity, we expect that edge points should be approximately characterized by those exponents t such that the dimension of E(t) is 1. To verify this assumption, we will perform the segmentation of the image by putting all pixels with exponent t such that E(t) has dimension close to 1 to white, and all other points to black. To do this, go to the **Segmentation** part of the **Multifractal Image Segmentation** window, and set the **min dim.** to 0.9 and the **max dim.** to 1.1. Hit **Compute seg.**. The output image, called *seg_im2d_00* appears in the **Variables** list. View this image by clicking on **View in new** in the **View** menu. As you can see, we have obtained, by the very simple procedure above, a good approximation to the edges of our original image. Notice in particular that some fine details have been detected, such as the small triangular holes on the right part of the door, the delicate contour of the bush, or the small sphere and the defect in the water on the lower left part of the image. Also, the method has detected some features inside

the bush on the left part. These do correspond to some irregularities, that you can see by manipulating a little bit the grey levels of the original image.

An interesting additional feature of this approach is that one can extract relevant subsets of points other than the contours, again based on a dimension analysis. For instance, we expect that sets of irregular points that lie in strongly textured region should form a set, roughly homogeneous with respect to the Hölder exponent, and of dimension between 1 and 2: smooth contours are 1D curves, while smooth regions are 2D areas; textures form subsets which lie "in-between" those two extremes. Verify this by extracting now those points with associated dimension between, say, 1.3 and 1.7, i.e. set **min dim.** to 1.3 and **max dim.** to 1.7 (beware that, since **Fraclab** checks that **min dim.** is smaller than **max dim.**, you need to enter these values in the right order, otherwise **Fraclab** will return to its default values 0 and 2). Hit **Compute seg.** again, and view the output. You should see in white mostly points on the water and in the bush, with additional pixels on the door and the foreground mountains, where one can distinguish some textures. The sky and the background mountains, which only display smoothly varying grey levels, are mostly black, except the edge of the mountain to the right of the image. These sets of white points could rightfully be called "textured points". Finally, if you put **min dim.** to 1.9 and **max dim.** to 2, you will verify that you do get full 2D regions mainly composed of points in smooth regions. These three segmentations show that a multifractal analysis of the image allows to extract several kind of points using the information contained in the spectrum. Of course, much more refined methods based on the technique explained here can be applied. See the references indicated above.

7 Conclusion

Fraclab is an open toolbox: We distribute freely both the executables and source files. Our main objective is to help disseminate the use of fractal methods in signal processing. We also hope to include many more functionalities in **Fraclab** in a near future. Contributions of any kind are welcome: Comments, bugs reports, new algorithms, source codes, examples of applications, etc...

We hope you'll enjoy using **Fraclab**.

8 References

There are many good books on the various aspects of fractal analysis, not to mention the thousands of papers related to the subject. Here is a small selection.

(1) C. Bandt, S. Graf, M. Zähle, Eds., *Fractal Geometry and Stochastics*, Spinger Verlag, 1995.

(2) M. Barnsley, *Fractals Everywhere*, Morgan Kauffman, 1993.

(3) C. Evertsz, H.O. Peitgen, R.F. Voss, *Fractal Geometry and Analysis : The Mandelbrot Festschrift, Curacao 1995* World Scientific, 1996.

(4) K. J. Falconer, *Fractal Geometry - Mathematical Foundations and Applications*, John Wiley, 1990

(5) J. Lévy Véhel, E. Lutton, C. Tricot, Eds., *Fractals in Engineering*, Spinger Verlag, 1997.

(6) B. B. Mandelbrot, *The Fractal Geometry of Nature*, Freeman, 1982

(7) P. Massopust, *Fractal Functions, Fractal Surfaces and Wavelets*, Academic Press, 1995.

(8) H.O. Peitgen, D. Saupe, *The Science of Fractal Images*, Springer Verlag, 1988.

(9) C. Tricot, *Curves and Fractal Dimensions*, Springer Verlag, 1995