The **1D signals Multifractal Spectra** pop-up menu

Jacques Lévy Véhel 12 January 2001

This text presents a brief explanation of the functionalities of the **1D signals Multifractal Spectra** pop-up menu.

Contents

1 Overview

This menu allows you estimate to various multifractal spectra for 1D signals. Multifractal spectra for images may also be computed in **Fraclab**: Choose the **Segmentation** menu, and then **Image multifractal segmentation**. We refer to the help corresponding to this window for further details.

There exists three main multifractal spectra: The Hausdorff, large deviation, and Legendre spectra. Basically, any of the three spectra provides an information as to which singularities occur in your signal, and which

are dominant: a spectrum is a 1D curve, where abscissa represents the Hölder exponents actually present in your signal, and ordinates are related to "the amount" of points where you will encounter a given singularity. For instance, if the spectrum f has just one maximum at exponent a, with f(a) = 1, then if you pick a point at random in the signal, it will almost surely have exponent a. If b is such that f(b) = 0, then there is a very sparse set of points which have exponent b. An exponent c which does not occur has f(c) = -infinity.

This is the rough picture. There are essential differences between the three spectra, that we cannot detail in this help. We just briefly mention their main features.

The Hausdorff spectrum gives geometrical information pertaining to the dimension of sets of points in a signal having a given Hölder exponent. This is the most precise spectrum from a mathematical point of view, but is also unfortunately the most difficult to estimate. We hope to incorporate an estimator of this spectrum for 1D signals in forthcoming versions of **Fraclab**. Note that in the **Image multifractal segmentation** sub-menu of the **Segmentation** window, such a estimator is already proposed.

The second spectrum is the large deviation spectrum, denoted fg. This spectrum yields a statistical information, related to the probability of finding a point with a given Hölder exponent in the signal. More precisely, fg measures how this probability behaves under changes of resolution.

The third spectrum is the Legendre spectrum, denoted fl. It is just a concave approximation to the large deviation spectrum. Its main interest is that it usually yields much more robust estimates, though at the expense of a loss of information.

The **1D signals Multifractal Spectra** allows you to compute fl and fg for 1D signals which are either measures (i.e. arrays of non negative data adding up to 1) or functions.

2 The Functions sub-menu

In this part, you will compute the spectra for arbitrary 1D functions. You may choose to compute the **Legendre spectrum**, which is easier to estimate, or the **Large deviation Spectrum**, which contains more information.

2.1 The Legendre spectrum

There are three algorithms: one based on the discrete wavelet transform (DWT), one on the continuous wavelet transform (CWT), and one on box counting.

2.1.1 The DWT based Legendre spectrum

If you don't want to set any parameters, just verify that the signal you want to process is selected at the time you launch the **DWT based** sub-menu, and hit **Compute**. You should get an output called *dwt_sig_LegSpec#*, which is the estimated the Legendre spectrum.

If you want more control, hit **Advanced compute**. Note that the signal which will be analyzed is the one highlighted when you press **Advanced compute**. Thus, if, for instance, you hit first **Compute** and then **Advanced compute**, you'll get an error. This is because after you press **Compute**, the current signal

becomes *dwt_sig_LegSpec#*, which is not a 1D signal. When you hit **Advanced compute**, you get a new window, titled **DWT Based Legendre Spectrum**. Choose as usual your **Input Signal** using the **Refresh** button. You may then choose the analyzing **Wavelet** from a choice of various Daubechies and Coiflet wavelets. Specify next the number of **Octaves**, i.e. on how many scale levels you wish to perform the analysis.

In the lower part of the window, you may specify three numerical values: **Qmin**, **Qmax**, and # **of Q's**: As its name indicates, fl is computed as the Legendre transform of some auxiliary function T(q). This function T(q) is estimated for q ranging in [**Qmin**, **Qmax**], and # **of Q's** values are computed in this interval. You may check the **Specify Regression Range**: In this case, as usual, a graphic window will appear after you hit **Compute**: You'll see a bunch of curves, one curve per value of q. For each q, T(q) is estimated as the slope of the the best linear fit of the corresponding curve. Thus, you will want to select, with the cross-shaped cursor, a range of scales (in abscissa), where an approximately linear behaviour holds for all or most of the curves. Once you have selected such a range, two new graphs will appear to the right: On the lower one, you'll see the estimate of T(q), while the upper one will be the Legendre spectrum fl. You may experiment with other ranges until you are satisfied with the result. Press then **return** on your keyboard to finish. If you uncheck **Specify Regression Range**, **Full Regression Range** will appear instead, and the estimate will be performed on the whole range of scales as defined by the number of **Octaves**. As in other windows of the same kind, you may also choose the **type of regression** from the usual choice.

In the advanced mode, you get two outputs: *dwt_sig#* is the discrete wavelet transform, while *dwt_sig#_LegSpec0* is the Legendre spectrum.

2.1.2 The CWT based Legendre spectrum

This is almost the same procedure as above, with only slightly different options due to the difference between the discrete and continuous wavelet transforms.

First, if you don't want to set any parameters, just verify that the signal you want to process is selected, then launch the **CWT based** sub-menu, and hit **Compute**. You should get an output called *sig_LegSpec#*, which is the estimated the Legendre spectrum.

If you want more control, hit **Advanced compute**. Note that the signal which will be analyzed is the one highlighted when you press **Advanced compute**. Thus, if, for instance, you hit first **Compute** and then **Advanced compute**, you'll get an error. This is because after you press **Compute**, the current signal becomes *sig_LegSpec#*, which is not a 1D signal. When you hit **Advanced compute**, you get a new window, titled **CWT Based Legendre Spectrum**. Choose as usual your **Input Signal** using the **Refresh** button.

In the middle part of the window, you'll set the parameters pertaining to the computation of the CWT: **fmin** and **fmax** let you choose the minimum and maximum frequencies of analysis. The default values are the ones yielding maximal span compatible with the size of the signal. You may change the extreme frequencies either by typing values under **fmin** and **fmax**, or by using the predefined values on the menus to the right. The **Voices** parameters governs the number of intermediate frequencies at which the continuous wavelet coefficients will be computed. Be warned that giving an excessive number of voices may result in large computing times for long signals. Checking the **Mirror** item will deal the border effects by mirroring the signal at its extremities. Otherwise, zero-padding is used. Finally, you may choose the **Size** and **Type** of

your analyzing wavelet: available wavelets are the **Mexican Hat**, and the real and analytic **Morlet** wavelet. The size may be any positive number (this parameter is not available for the Mexican hat). Once all the parameters that define the wavelet transform are chosen, hit **Compute WT**. The output signal is a matrix of size "number of voices" x "size of the original signal". It is called *cwt_signal#*, if "signal" is the name of your data, and where # is as usual an incremental parameter. It should appear in the **Input CWT** box just below the **Compute WT** button. You may want to view the continuous wavelet transform using the **View** menu. Note that **Fraclab** recognizes wavelet transforms, and display them differently form regular images. In particular, it uses a fixed aspect ratio (this is useful for instance if the number of voices is much smaller than the size of the signal), and the "jet" color-map, which often allows to highlight the important structures. If you want to view the transform as a normal image, or make other changes in the appearance, use the functionalities of the **View** menu described in the **Overview** help file.

The **Refresh** button to the left of the **Input CWT** box lets you load a wavelet transform which would already be present in the **Variables** list of the main window. This avoids computing several times the same transform. Once you are happy with your transform, move to the lower part of the window, which performs the actual computation of the spectrum.

In the lower part of the window, you may specify three numerical values: **Qmin**, **Qmax**, and # **of Q's**: As its name indicates, fl is computed as the Legendre transform of some auxiliary function T(q). This function T(q) is estimated for q ranging in [**Qmin**, **Qmax**], and # **of Q's** values are computed in this interval. You may check the **Specify Regression Range**: In this case, as usual, a graphic window will appear after you hit **Compute**: You'll see a bunch of curves, one curve per value of q. For each q, T(q) is estimated as the slope of the the best linear fit of the corresponding curve. Thus, you will want to select, with the cross-shaped cursor, a range of scales (in abscissa), where an approximately linear behaviour holds for all or most of the curves. Once you have selected such a range, two new graphs will appear to the right: On the lower one, you'll see the estimate of T(q), while the upper one will be the Legendre spectrum fl. You may experiment with other ranges until you are satisfied with the result. Press then **return** on your keyboard to finish. If you uncheck **Specify Regression Range**, **Full Regression Range** will appear instead, and the estimate will be performed on the whole range of scales as defined by the number of **Voices**. As in other windows of the same kind, you may also choose the **type of regression** from the usual choice. Finally, in many cases, more relevant estimates are obtained if one chooses, at each scale, the largest coefficients in given neighbourhoods, instead of a mean value. This is the default in **Fraclab**, as is indicated by the fact that the **Local Maxima** box is checked. If you unmark this box by pressing the button to the left of **Yes** (which subsequently becomes **No**), then the program will use mean values.

As above, in the advanced mode, you get two outputs: *cwt_sig#* is the continuous wavelet transform, while *cwt_sig_LegSpec#* is the Legendre spectrum.

Note the following weird naming convention: When you perform an advanced compute, the spectrum here is called *cwt_sig_#LegSpec#*, consistent with the case of the DWT method. The first # refers to the number of times you have computed the wavelet transform, while the second refers to the spectrum number. That is, if you compute several spectra with the same wavelet transform, only the second # is incremented. However, in the basic CWT method, the name of the spectrum is *sig_LegSpec#*, while in the basic DWT one it is *dwt_sig_LegSpec#*. We hope this does not introduce to much confusion.

2.1.3 The Box method for the Legendre spectrum estimation

As in the wavelet-based methods, you may decide to use the default parameters by choosing the menu **Basic parameters**. However, in this case, you don't get a new window as above: Rather, the output, called *fcfl1d_sig#* is computed right away and sent directly to your **Variables** list. The **Advanced parameters** choice is not included in the current release of **Fraclab**.

2.2 The Large deviation Spectrum

The large deviation spectrum fg is computed in **Fraclab** using pure time-domain algorithms. If you choose **Basic parameters**, the estimated spectrum, called *fcfg1d_sig#*, is computed right away and directly sent to your **Variables** list. You may have an idea of the default parameters used by the algorithm by viewing the output: You'll notice in the caption the numerical values for the various parameters.

Choosing **Advanced parameters** instead, you get a new window, titled **Large deviation spectrum estimation**. On the first line, the type of the **Input data**, i.e. **measure**, **function** or **cwt** is recalled. You may change (at your own risks - changing manually the type will usually result in an error) this type, by checking the appropriate box. As usual, you may **Refresh** the signal and view its **name** and **size**.

The second part of the window deals with the **Coarse grain Hölder exponents estimation**. Coarse grain Hölder exponents are the basic bricks for estimating fg. They just measure the scaling behaviour of the data at finite resolution (rather than at infinite resolution in the case of usual Hölder exponents). More precisely, for each resolution n, we consider the 2^n dyadic intervals that partition (0,1) (recall that, by convention, all the signals are assumed to be supported on (0,1)). For each interval, we compute a coarse grain Hölder exponent: This is just the logarithm of some quantity measuring the variation of the signal in the interval (this measure of variation is discussed below), divided by the logarithm of the size of the interval. Thus, we have one coarse grain exponent for each interval at each resolution. The minimum and maximum resolutions you wish to consider for computing these coarse grain exponents are set by the parameters **min size** and **max size**. Warning: The sliders that control these values have a somewhat strange behaviour, and you may want to enter the desired values directly from your keyboard. The number of resolutions you want to consider between **min size** and **max size** is set by the parameter # **of scales**. Finally, you may decide how your # **of scales** resolutions are distributed between **min size** and **max size**: This is the **progress.** parameter. **linear** means equispacing, **logarithmic** means log-uniform spacing, while **decimated** implies that your **min size** and **max size** are powers of 2: If you choose **decimated**, then the three parameters **min size**, **max size** and # **of scales** are no longer independent: **Fraclab** will force them so that log(**max size**) - log(**min size**) +1 = # **of scales**. Once you have decided your discretization of resolutions, you may choose which quantity to investigate: In other words, you need to decide how precisely you measure the variation of your signal in each interval, in order to determine its scaling behaviour. The default choice is the **oscillation**, i.e. the maximum minus the minimum of the signal in the dyadic interval. A second choice is the **lpnorm**, in which case you may specify the **power** p using the box on the right (the default is p = 2). Finally, you may go for the **linfty norm**, i.e. set p = infinity.

Once you're done with your settings, hit **Compute exponents**. The output, called *fch1d_sig#*, will appear in the Variables list. It is a matrix of size (# **of scales**) x (length of the original signal). Each line in the matrix corresponds to the vector of coarse grain exponents at one of the considered resolution. Since at resolution n there are only 2^n points, and in order to deal with lines of constant sizes without resorting to

zero-padding, each coarse grain exponent at resolution n is repeated the needed number of times.

The lower part of the window lets you perform the actual computation of the large deviation spectrum. The computation is based on techniques used in density estimation, and uses a kernel. You first need to tell **Fraclab** how the **density** of the coarse grain exponents will be estimated. In the current version of **Fraclab**, only the **continuous** choice is available. This simply means that a fully non parametric method is used. We hope to include the other choices, namely **discrete**, **wavelet** and **parametric** in future releases. The next parameter is the size of the kernel. You control this by selecting, in front of the item **adaptation**, one of **maxdev**, **diagonal**, **double kernel** and **manual**. Only **maxdev** and **manual** are implemented in the current version of **Fraclab**. Choosing **maxdev** will make **Fraclab** use an optimal size computed from some empirical statistical criterion. If you rather want to go for **manual**, then you may enter a numerical value in the box to the right, or use the slider (these box and slider are grayed out when you choose **maxdev**, since the choice is automated in this case). The default 0.1 is often a good starting point. Larger values decrease variance at the expense of bias, resulting in smoother estimates, and vice-versa. You finally choose the shape of the kernel, by selecting in front of **kernel** one of **Gaussian**, **boxcar**, **epanechnikov**, **mollifier** and **triangle** (consult any book on density estimation to know more about these kernels).

Note that this routine will not estimate one but several spectra: One for each of the # **of scales** resolutions. Thus, the output signal, called *fcfg1d_sig#*, is a structure that contains # **of scales** graphs, each being a spectrum estimated at a given resolution. High resolution (i.e.close to **min size**) spectra have higher accuracy, while low resolution ones are more robust. In general, you should use a small number of different resolutions, i.e. around 4 or 5, ranging from high to moderate resolutions (i.e. **min size** = 1 or 2 and **max size** of the order of 10 to 100). You know that you have obtained a meaningful estimate if all the spectra coincide approximately. More precisely, you may trust that, for a given exponent a, the large deviation spectrum of your signal at point a has been well estimated if the values of most or all of your graphs almost coincide above abscissa a. You'll find that, in many cases, better agreements are obtained for large values of the spectra (large *ordinates*), and also for values of the exponent smaller than the mode of the spectra. On the contrary, values to the extreme right (large *exponents*) often show more discrepancy. Also, in many cases, the spectrum computed with the highest resolution significantly departs from the others. If all others agree reasonably, it is safe to discard the highest resolution one and use as an estimate the common value of the others.

3 The Measures sub-menu

Everything in this sub-menu is very similar to the one for functions. We highlight below the main differences.

3.1 The Legendre spectrum

This sub-menu has exactly the same features as in the **Functions** menu.

3.2 The Large Deviation Spectrum sub-menu

Here, only the basic parameter menu is available: You cannot, in the current implementation of **Fraclab**, set the advanced parameters for the estimation of fg in the case of measures. You may however "cheat" and

consider your measure as a function. You can then use the menu corresponding to the estimation of the large deviation spectrum for functions. There is no real harm in doing this, except that you do not take advantage of the special structure of measures.

3.3 The Tricot Spectrum sub-menu

This is not implemented in the current implementation of **Fraclab**.

3.4 The Haussdorff Spectrum sub-menu

This is not implemented in the current implementation of **Fraclab**.

4 Concluding remarks

As a general rule, Legendre spectra will yield more robust results, but will wipe out any departure from concavity in the spectrum. It is often a good idea to compute both fg and fl. If fl is approximately the concave hull of fg, then it is reasonable to assume that the estimates of both fg and fl are relevant. Also, you'll find that, in most cases, pure time-domains estimates of the spectra give better results. An interesting test in that view is to synthesize a multifractal measure, using the **measure** sub-menu of the **synthesis** menu with the default parameters (this will yield a trinomial measure at resolution 7). Try then the estimation of both fg and fl using all methods above, i.e. DWT-based, CWT-based and box method for fl, and the kernel method for fg. In all cases, use the default or **basic** parameters. You'll see how the box-method estimate of fl agrees reasonably well with the kernel estimate of fg, as it should. Also, the theoretical spectrum (which **Fraclab** may compute at the same time it synthesizes the measure) agrees quite well with these estimates of fl and fg. Both wavelet estimators are quite off.

5 Homework

We'll start with a simple test on the paradigm of multifractals, i.e. a multinomial measure. Synthesize first a multinomial measure: Go to **Synthesis**/**Measures**. We'll generate a measure with the default parameters, i.e. a 1D trinomial deterministic measure at resolution 7. This results in 3^7=2187 intervals. The reason why we generate a trinomial measure and not a binomial one is that many algorithms use (sometimes implicitly) at some point a dyadic partition of space, and it would be cheating to match the structure of the data to that of the algorithm. Since there is a closed formed formula for the multifractal spectrum of such measures, we'll check the box to the right of **Compute theoretical spectrum**, so that **Yes** appears. Recall finally that, in this case, all three spectra, i.e. the Hausdorff, large deviation and Legendre spectra, coincide. Hit **Compute**, and view the two generated signals: *mu_n0* is the trinomial measure, and *theof0* the associated multifractal spectrum. Let us now try to estimate the spectrum from the data *mu_n0*. We'll try first the enhanced box method implemented in **Fraclab**: With *mu_n0* selected, go to **1D signals Multifractal Spectra**/**Measures**/**Legendre Spectrum**/**Box Method**/**Basic parameters**. The estimated spectrum *mdfl1d_mu_n00* appears in the **Variables** list. View it, and compare it with the theoretical one *theof0*

(you can superpose them using the **hold** facility of the **View** menu). As you can see, the match is almost perfect. As a second test, we'll try the discrete wavelet based method. With *mu_n0* selected, go to **1D signals Multifractal Spectra/Measures/Legendre Spectrum/DWT based**. Hit **Compute** to get the estimated spectrum *dwt_mu_n00_LegSpec0*. You'll notice that although not completely off, this estimate is quite wrong on its left part. The fact that it is not smooth but rather displays angles is not a serious problem: It is just that the discretization in the q parameter is too coarse. To get rid of this artifact, let us try the **Advanced compute** method (recall to select *mu_n0* before clicking on **Advanced compute**, otherwise you'll get an error **Error : Input must be a Matrix or Vector !** in the **Message** zone of the main window). Change **Qmin** to -10, **Qmax** to 10, and # **of Q's** to 30, and hit **Compute**. You get the usual window that allows you to select a range where approximate linearity holds. Note that, although the data are strictly multifractal, the scaling does not appear so good on the graph, i.e. there is no region where a linear behaviour is observed. If however you select the whole range, you'll get a spectrum which is not too bad: Hit return to validate your choice and view the resulting spectrum *dwt_mu_n01_LegSpec0*. This one is smoother, but it is still wrong on the left part. This discrepancy as well as the non linear behaviour noticed above are probably due to the fact that we are analyzing a triadic structure with a dyadic wavelet. As you may care to try, changing the wavelet will not improve the result. Let us now see what happens with a continuous wavelet transform based algorithm. Select again *mu_n0* and go to **1D signals Multifractal Spectra/Measures/Legendre Spectrum/CWT based**. Hit **Compute** to get the estimated spectrum *mu_n0_LegSpec0*. On viewing the result, you'll notice that it is completely false: It has no decreasing part. Let us now try the **Advanced compute** menu (recall to select *mu_n0* before clicking on **Advanced compute**). Use the default parameters or change them as you like, and hit **Compute WT**, then **Compute**. In the new window that appears, you'll see why we get only an increasing part for the spectrum: This comes from the fact that the curves on the left do not cross as they should.

Let us now try to estimate the spectrum of a Weierstrass function. Synthesize a deterministic Weierstrass function with the default parameters except you require that 1024 samples are generated. The theoretical Legendre and large deviation spectra coincide in this case: it is a half line with slope -1 starting from the point (H,1). If however one computes the spectrum with a wavelet with a sufficient number of vanishing moments, the spectra reduce to the point (H,1). Let us first compute the Legendre spectrum. Since this is a function and not a measure, go to **1D signals Multifractal Spectra/Functions/Legendre Spectrum/DWT based**. Hot **Compute**. The result is correct, as you will check by viewing *dwt_Wei0_LegSpec0*, although there is some lack of precision that you can correct by using the **Advanced compute** features. Set **Qmin** to -10, **Qmax** to 10, and # **of Q's** to 30, and hit **Compute**. You get the usual window that allows you to select a range where approximate linearity holds. Again, the scaling is not so good, although it should be from a theoretical point of view. Select the whole range and hit return to get the spectrum *dwt_Wei00_LegSpec0*. View it and notice how we have gained precision in the location of the maximum, which is quite close to 0.5. Try now a wavelet with a high number of vanishing moments, e.g. the Daubechies 20. The graphs do not look more linear, but, selecting again the whole range will give you an almost perfect result, i.e. a spectrum reduced to a single point with abscissa 0.5 and ordinate which is a bit underestimated around 0.85. Note finally that with a Coiflet 6, you do get a large region (between log scale 0 and 9) where approximate linearity holds, with corresponding spectrum comparable to the one obtained with the Daubechies wavelet. As you may care to check, the spectra obtained with the CWT method are not good, for the same reasons as the ones exposed in the case of the multinomial measure. The spectrum obtained using **1D signals Multifractal Spectra/Functions/Legendre Spectrum/Box Method/Basic parameters** is not too far from the theoretical one, with however a noticeable bias on the maximum (0.6 instead of 0.5). Let us

now estimate the large deviation spectrum. Go to **1D signals Multifractal Spectra/Functions/Large deviation Spectrum/Basic parameters**. You get the estimate called *fcfg1d_ Wei00*: this is almost the same spectrum as the one obtained using the Legendre approach in the box method implementation.

Our last test on a synthetic signal is with an IFS: synthesize an IFS in **Synthesis/Functions/Deterministic/IFS** with the default parameters. We shall analyze the signal called *ifs_ ord_ 0*. This IFS has a (Legendre and large deviation) spectrum which assumes the same shape as the one of a deterministic multinomial measure. It has the following features: the Hölder exponent ranges in the interval (0.0959, 1.4650) with a mode at 0.7306. Let us see if we can recover this shape and values with our estimators. As you may care to check, the CWT method and its variants do not perform good in this case either. To test the DWT method, go to **1D signals Multifractal Spectra/Functions/Legendre Spectrum/DWT based** and hit **Advanced compute**. Choose **Daubechies 12** as a wavelet, and set **Qmin** to -10, **Qmax** to 10, **# of Q's** to 30 and hit **Compute**. Select the whole range and hit return. The obtained spectrum is not too bad, with a minimum at 0.01, a maximum at 1.65, and a mode at 0.69. These values depend however heavily on the choice of the analyzing wavelet. We will now see how the Box method behaves. Select *ifs_ ord_ 0* and go to **1D signals Multifractal Spectra/Functions/Legendre Spectrum/Box Method/Basic parameters**. The computation will unfortunately take a while. However, the result is quite good: You get the correct shape, with a minimum at 0.26, a maximum at 1.43 and a mode at 0.72. In contrast, the computation of the estimate of the large deviation spectrum, obtained through **1D signals Multifractal Spectra/Functions/Large deviation Spectrum/Basic parameters**, is very fast: You'll verify that you get roughly the same estimate as with the Legendre spectrum with the box method. The important fact here is that, contrarily to the Legendre one, the estimated large deviation spectrum does not have to be concave. The fact that we do obtain a perfectly concave spectrum is thus already an important piece of information.

We end this section with the multifractal analysis of a real world signal, namely an Internet traffic log. It has been discovered (see reference (4)) that Internet traffic does exhibit, in certain conditions, a strong multifractal behaviour. Although the mechanisms behind this are not yet completely understood, this fact yields interesting information on the structure of Internet traffic and on its small scale behaviour. We shall analyze here the data called *traffic.txt* that you will find in the **DATA** directory that comes with the **Fraclab** release. 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. Choose the file called *traffic.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 *traffic.txt* file should appear in your **Variables** list of the main window, under the name *ftraffic*. View this signal, and notice how irregular it is. This is a high frequency log of a TCP trace. More precisely, the values in this graph corresponds to the number of packets that went through a gateway at the French CNET during successive very short periods of time. Since all ordinates must be positive, it is natural to treat these data as a measure. In that view, we need to normalize it, so that the "total mass" is one. To do this, type *n_ traffic = ftraffic/sum(ftraffic);* in the matlab window, and import *n_ traffic* in **Fraclab**: Hit **Scan Workspace** and select *n_ traffic* in the windows that appears. Click on **Import** and close that window. You may view *n_ traffic* to check that it is the same as *f_ traffic* except the scale in ordinate has been divided roughly by 10^8. Now go to **1D signals Multifractal Spectra/Measures/Large deviation Spectrum/Basic parameters**. You'll get the output *mcfg1d_ n_ traffic0*. View it. You'll see that the estimated spectrum is not concave (compare

with the ones obtained above on the synthetic signals). Indeed, there is a small but noticeable "bump" on the increasing part of the spectrum. As a consequence, this estimate looks like a superposition of two "basic" spectra, i.e. spectra like the ones of a multinomial measure. This can be taken as an indication that we have here (at least) two different phenomena. This view is further supported by a more refined analysis: Indeed, if one separates the above aggregated traffic into an incoming traffic and an outgoing one, then one gets that each of these two traffics has a concave spectrum. Multifractal analysis thus allows us in this case to separate the two kinds of traffic on the basis of their high frequency behaviour. We shall not pursue this here, and refer the interested to reference (4).

Finally, recall that another example of the computation and use of a multifractal spectrum on real data (segmentation of an optical image) has been described in the homework section of the help file **Overview and main functionalities of Fraclab**.

6 References

(1) J. Lévy Véhel, R. Vojak, *Multifractal Analysis of Choquet Capacities: Preliminary Results*, Advances in Applied Mathematics, Vol. 20, No. 1, pp. 1-43, January 1998.

(2) S. Jaffard, *Multifractal formalism for functions, Part 1: Results valid for all functions*, S.I.A.M. Journal of Mathematical Analysis Vol. 28 N. 4 p. 944-970 (1997).

(3) J. Lévy Véhel, *Numerical Computation of the Large Deviation Multifractal Spectrum*, CFIC, Rome, 1996.

(4) J. Lévy Véhel, R. Riedi, *Fractional Brownian motion and data traffic modeling: The other end of the spectrum*, Fractals in Engineering, Eds. J. Lévy Véhel, E. Lutton and C. Tricot, Springer Verlag, 1997.