SIMULATION AND IDENTIFICATION OF THE FRACTIONAL BROWNIAN MOTION: A BIBLIOGRAPHICAL AND COMPARATIVE STUDY

COEURJOLLY Jean-François [1]

Date: June 7, 2000

Abstract : We present a non exhaustive bibliographical and comparative study of the problem of simulation and identification of the fractional Brownian motion. The discussed implementation is realized within the software S-plus 3.4. A few simulations illustrate this work. Furthermore, we propose a test based on the asymptotic behavior of a self-similarity parameter's estimator to explore the quality of different generators. This procedure, easily computable, allows us to extract an efficient method of simulation. In the Appendix are described the S-plus scripts related to simulation and identification methods of the fBm.

Keywords: Fractional Brownian motion, simulation, identification of a parametric model, quality of a generator, S-plus.

[1] LMC-IMAG, BP 53 - 38041 Grenoble cedex 9 - FRANCE, Jean-Francois.Coeurjolly@imag.fr. To whom correspondence should be addressed.

1 Introduction

Many naturally occurring phenomena can be effectively modeled using self-similar processes. For such processes observations that are far apart (in time or space) are correlated too strongly indicating the presence of a long-range dependence. As a result self-similar processes have been used to successfully model data exhibiting long-range dependence and arising in a variety of different scientific fields, including hydrology, see *e.g.* [6], geophysics [12], biology [8], telecommunication networks [28] and economics [24]. The empirical presence of long-memory in such series is found in a local version of the power spectrum which behaves, as $|\lambda|^{1-2H}$, as $\lambda \to 0$, where $H \in]1/2, 1[$ is the long-memory parameter. Among the simplest models that display long-range dependence, one can consider the fractional Brownian motion (fBm), introduced by Kolmogorov in a theoretical context [19], and by Mandelbrot and his co-workers [21] for its statistical applications. The fractional Brownian motion (in short fBm), denoted by $\{B_{H,C}(t), t \geq 0\}$, with parameters $(H, C) \in (]0, 1[\times\mathbb{R}_+^*)$, is the process defined as the fractional integration of a Gaussian pure white noise, or equivalently by the stochastic integral:

$$B_H(t) = C\ V_H^{1/2} \int_{\mathbb{R}} f_t(s) dB(s) \qquad (1)$$
$$\text{with} \quad f_t(s) = \frac{1}{\Gamma(H+1/2)} \left\{ |t-s|^{H-\frac{1}{2}} \mathbb{1}_{]-\infty,t]}(s) - |s|^{H-1/2} \mathbb{1}_{]-\infty,0]}(s) \right\},$$

with $B_H(0) = 0$ and $V_H = \Gamma(2H+1)\sin(\pi H)$. Due to the non stationarity of the fBm and the presence of long-memory, simulation and identification of a fBm is a delicate task. A vast literature has been published on these subjects. A good survey can be found in Beran [6] where historical and statistical aspects are considered. We refer to Adler and *al.* [4] or Taqqu and *al.* [27] for an empirical study on estimation methods.

Through a bibliographical study, we intend to draw up a non exhaustive list of methods for simulating a fBm, and for estimating the self-similarity parameter H. Firstly in Section 2, we recall fundamental properties of fBm: covariance and autocovariance functions, spectral density, Hausdorff dimension. We then describe five simulation methods in Section 3: the method of Mandelbrot and *al.* [21], that of Sellan and *al.* [2], the Choleski method, the Levinson one [24] and finaly the method of Wood and Chan [29]. Section 4 discusses several methods for estimating the self-similarity parameter: spectral methods, maximum likelihood, time-scale methods and temporal methods. Section 5 presents a few simulation results with Boxplots, thus illustrating numerically the notions of Sections 3 and 4. Section 6 explores the quality of pseudo-random generators of fBm. A similar study has been undertaken by Jennane and *al.* [17]: three testing procedures defined independently of the model's identification were considered there. Our approach is slightly different: we provide a theoretical test based on the asymptotic behavior of a parametric estimator of H which allows us to point out a good simulation method of the fBm. Finally we display in the Appendix the S-plus scripts implementing the methods considered in Sections 3 and 4.

2 Properties of fractional Brownian motion

As an alternative way to (1), the fBm can be defined as the unique mean-zero Gaussian process, null at the origin, with stationary and self-similar increments, such that

$$\mathbb{E}\,(\,B_{H,C}(t) - B_{H,C}(s)\,)^2 \;=\; C^2\,|t-s|^{2H}\;, \quad \forall\, s,t \in \mathbb{R}^+\;. \tag{2}$$

Hereafter, we shall call standard fractional Brownian motion, the fBm with scale parameter $C \equiv 1$. Let us briefly review some fundamental results about the fBm. From the self-similarity property, we deduce the covariance and the autocovariance functions, given by

$$\begin{aligned}
\Gamma(t,s) &= \mathrm{Cov}\big(\,B_H(t),\,B_H(s)\,\big) \;=\; \frac{C^2}{2}\,\big(\,|t|^{2H} \;+\; |s|^{2H} \;-|t-s|^{2H}\,\big) &(3)\\
\gamma(t-s) &= \mathrm{Cov}\big(\,B_H(t+1)-B_H(t),\,B_H(s+1)-B_H(s)\,\big)\\
&= \frac{C^2}{2}\big(\,|t-s-1|^{2H} \;-\; 2\,|t-s|^{2H} \;+\; |t-s+1|^{2H}\,\big). &(4)
\end{aligned}$$

In the particular case $H = 1/2$, the fBm is identical to the Brownian motion; consequently $\gamma(k) = 0$, for $|k| \geq 1$. When $H \neq 1/2$, an asymptotic expansion, as $|k| \to +\infty$ exhibits an hyperbolic decrease of $\gamma(\cdot)$:

$$\gamma(k) \sim\; C^2\; H(2H-1)\;|k|^{2H-2}\;, \qquad \text{as } |k| \to +\infty\;. \tag{5}$$

This asymptotic behavior clearly shows that a path deviating from its mean, will have tendency to deviate more when $H > 1/2$, or to return closer to the mean when $H < 1/2$.

The increments process of the fBm is called the fractional Gaussian noise (in short fGn). The fGn constitutes a stationary time-series, and admits a spectral density, defined as the Fourier transform of $\gamma(.)$, explicitly given by

$$f(\lambda) \;=\; 2\,c_\lambda\,(1-\cos\lambda)\,\sum_{j\in\mathbb{Z}} |\,2\pi j + \lambda)\,|^{-1-2H}, \qquad \forall\,\lambda \in [0,2\pi]\;, \tag{6}$$

with $c_\lambda = \frac{C^2}{2\pi}\sin(\pi H)\Gamma(2H+1)$. A Taylor expansion of f near 0 shows that the spectral signature of the fGn is $|\lambda|^{1-2H}$, indicating a pole at zero for $H > 1/2$, a characteristic fact of long-memory processes.

Concerning the paths' regularity, the fBm similarly to the Brownian motion, has continuous and almost surely non differentiable sample paths. The fractal approach refines the difference. Indeed, the Hausdorff dimension of a fBm with parameter $H \in]0,1[$ is almost surely equal to $2-H$, which implies that for $H < 1/2$, the paths are more irregular than those of the Brownian motion and conversely for $H > 1/2$. Figure Fig.1 illustrates this remark.

Finally, let us mention a property of continuity in H, in the sense of Kolmogorov, for the fBm proved by Peltier and Lévy-Véhel [22]:

$$\forall\;\; [a,b] \subset]0,1[\text{ and } K>0, \quad \lim_{h\to 0}\;\sup_{\substack{a\leq H,H'\leq b\\ |H-H'|<h}}\;\sup_{t\in[0,K]}\; |B_{H,C}(t) - B_{H',C}(t)| \;=\; 0\;. \tag{7}$$

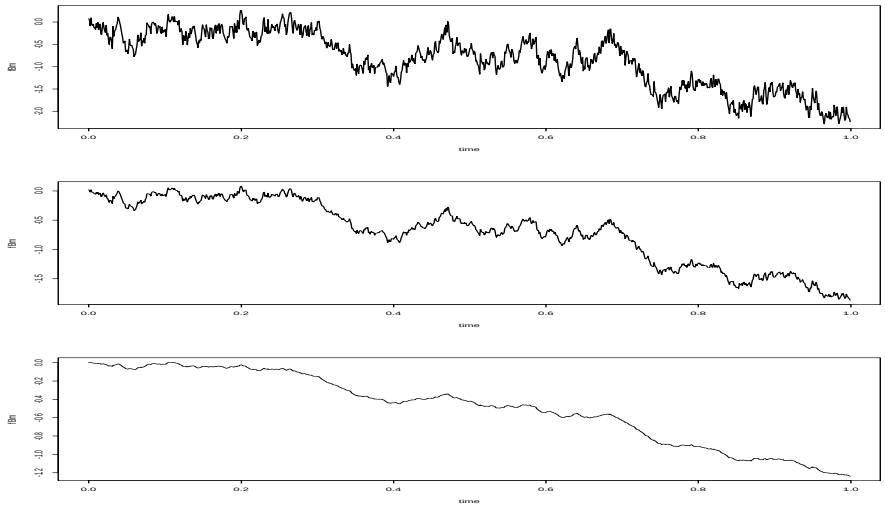

Figure 1: Samples of a fractional Brownian motion on [0,1], simulated by Wood-Chan's method for values $H = 0.3, 0.5, 0.8$ from top to bottom.

3 Simulating the fractional Brownian motion

3.1 Statement of the problem

In this Section, our aim is to describe few methods for simulating a fBm. We adopt the following framework: simulation of a sample of a standard fractional Brownian motion ($C = 1$), of length N at times $i/N,\ i = 0, \ldots, N-1$.

Two approaches are distinguished: the first one consists in using only the properties of the fBm. It gives rise to three methods: the first one is based on a stochastic representation of the fBm [21], the second one consists in extracting the square root of the covariance matrix of the fBm, and the last one consists in the fractional integration of a Gaussian white noise, decomposed on a multiresolution analysis and relies upon a method of Sellan and *al* [2].

The second approach is in fact an indirect way. The idea is to generate a fGn, $\widetilde{X}$, at times i/N, for $i = 0, \ldots, N-1$ and then to define a sample of a fBm via the cumulated sums of $\widetilde{X}$, that is to say to define $\widetilde{B}_H(i/N) = \sum_{k=0}^{i} \widetilde{X}(i/N)$ for $i = 1, \ldots, N-1$ and $\widetilde{B}_H(0) = 0$. The interest to use the increments process rests in its stationarity. The covariance matrix of the fGn at times $i/N,\ i = 0, \ldots, N-1$ is a Toeplitz matrix. The two methods presented (method of Levinson [24], method of Wood-Chan [29]) consist in computing in an exact way the square root of a Toeplitz matrix.

3.2 Stochastic representation of fBm

By considering the fBm's representation of Mandelbrot and Van Ness [21], the first natural idea to simulate a fBm consists in discretizing the stochastic integral (1). We have to approximate the integral

(1) by a Riemann sum truncated to a bound $-a_N \to -\infty$. For $t = 1, \ldots, N-1$, one gets

$$\widetilde{B}_H\left(\frac{t}{N}\right) =$$

$$\frac{V_H^{1/2}}{\Gamma(H+1/2)}\frac{1}{N^H}\left\{\sum_{k=-a_N}^{0}\left[(t-k)^{H-1/2}-(-k)^{H-1/2}\right]B_1(k)+\sum_{k=0}^{t}(t-k)^{H-1/2}B_2(k)\right\},$$

where B_1 (resp. B_2) is a vector of a_N+1 (resp. N) zero-mean standard Gaussian variables independent, and independent of B_2. The choice of a_N results from a compromise between the desired precision and the number of temporal points. In practice, and for the illustration that follows, we have chosen $a_N = N^{1.5}$. This approach is purely historic, and owing to several approximations is not a good way to generate a fBm.

3.3 Method of Sellan, Meyer and Abry

This method has been established by Sellan, Meyer and Abry [2]. Since the fBm is derived by fractional integration of a Gaussian pure white noise, the idea is to start from the decomposition of a pure white noise onto a multiresolution analysis (in short MRA), see *e.g.* Daubechies [9] for generalities on wavelets and MRA:

$$w(t) = \sum_{k\in\mathbb{Z}}\lambda(k)\,\phi_0(t-k) + \sum_{j\geq 0}\sum_{k\in\mathbb{Z}}\gamma_j(k)\,\psi_{j,k}(t)\,, \tag{8}$$

where $\lambda(k)$ and $\gamma_j(k)$ are standard independent Gaussian variables. ϕ_0 is the scaling function and $\{\psi_{j,k}\}_{j\geq 0,k\in\mathbb{Z}}$ the wavelets associated to the MRA. Applying the operator of fractional integration, denoted by D^{-s} (with $s = H+1/2$), to (8) leads to

$$B_H(t) = \sum_{k\in\mathbb{Z}}\lambda(k)\,(D^{-s}\phi_0)(t-k) + \sum_{j\geq 0}\sum_{k\in\mathbb{Z}}\gamma_j(k)\,(D^{-s}\psi_{j,k})(t)\,. \tag{9}$$

The following result, due to Sellan [26], describes explicitly how to integrate fractionally a MRA, and the necessity to introduce biorthogonal wavelets.

Theorem 1 *Let $V_0(\phi_0)$ be an orthogonal MRA of $L^2(\mathbb{R})$ with regularity $r \in \mathbb{N}^*$, ϕ_0 and ψ_0 representing respectively the scale function and the mother wavelet deduced from this analysis. Assume that $s \in]\frac{1}{2}, \frac{3}{2}[$, then*

$$V_0^{(s)} = \{\, f \in L^2(\mathbb{R}), D^s f \in V_0 \,\}\,, \quad \text{and} \quad V_0^{(-s)} = \overline{\{\, f \in L^2(\mathbb{R}), \overline{D^{-s}} f \in V_0 \,\}}\,,$$

define two biorthogonal MRAs, admitting for scale functions

$$\phi_0^{(s)} = U_s(\phi_0) \quad \text{for} \quad V_0^{(s)}(\phi_0)\,, \quad \text{and} \quad \phi_0^{(-s)} = \overline{U}_{-s}(\phi_0) \quad \text{for} \quad V_0^{(-s)}(\phi_0)\,,$$

where $g = U_s(f)$ has for Fourier transform $(i2\pi\nu)^{-s}(1-\exp(i2\pi\nu))^2\,\widehat{f}(\nu)$, and for mother wavelets:

$$\psi_0^{(s)} = 4^s\, D^{-s}(\psi_0)\,, \quad \text{and} \quad \psi_0^{(-s)} = 4^{-s}\,\overline{D}^s(\psi_0)\,.$$

$\overline{D^{-s}}$ denotes the conjugate operator of D^{-s}, and $\overline{E}$, for a set E, is the adherence of E. Under the conditions of Theorem 1, Sellan proves that there exists a Gaussian white noise with variance σ^2, allowing to construct an $ARIMA(0, s, 0)$, denoted by b_H, and for $j \in \mathbb{Z}^+$, a Gaussian discrete white noise, with variance $2^j\sigma^2$, denoted by $(\gamma_j(k))_{k\in\mathbb{Z}}$ such that the restriction of B_H to the interval $]0, T]$, $T > 0$ admits the following decomposition

$$\forall\, t \in]0, T], \qquad B_H(t) - b_0 \;=\; \sum_{k\in\mathbb{Z}} b_H(k)\; \phi_0^{(s)}(t-k) + \sum_{j\geq 0}\sum_{k\in\mathbb{Z}} 4^{-s}\; 2^{-js}\; \gamma_j(k)\; \psi_{j,k}^{(s)}(t)\;. \qquad (10)$$

The computing implementation is then realized in three steps:

1. Estimation of the filters related to $\phi_0^{(s)}$ and $\psi_0^{(s)}$: let $u(k)$ be the filter of the initial MRA related to ϕ_0, and $v(k)$ the associated quadrature mirror filter. The above cited authors show, by denoting $u^{(s)}$ and $v^{(s)}$ (resp. $u^{(-s)}$ and $v^{(-s)}$), the filters associated to $\phi_0^{(s)}$ and $\psi_0^{(s)}$ (resp. $\phi_0^{(-s)}$ and $\psi_0^{(-s)}$), the following relations:

$$\begin{array}{lcllcl} u^{(s)} & = & f^{(s)} * u, & F^{(s)}(z) & = & 2^{-s}(1+z^{-1})^s\;. \\ v^{(s)} & = & g^{(s)} * v, & G^{(s)}(z) & = & 2^{s}(1-z^{-1})^{-s}\;. \\ u^{(-s)} & = & f^{(-s)} * u, & F^{(-s)}(z) & = & 2^{s}(1+z)^{-s}\;. \\ v^{(-s)} & = & g^{(-s)} * v, & G^{(-s)}(z) & = & 2^{-s}(1-z)^{s}\;. \end{array}$$

   where $*$ denotes the convolution product and where $F^{(\epsilon s)}$ (resp. $G^{(\epsilon s)}$) denotes the z-transform of $f^{(\epsilon s)}$ (resp. $g^{(\epsilon s)}$), $\epsilon = \pm 1$. However, a numerical problem appears: for $s = H + 1/2$, the functions $f^{(s)}$ and $g^{(s)}$ have, in general, an infinite support and the series (10) diverges. To avoid this problem, the following approximations are proposed:

$$\begin{array}{lcllcl} u^{(s)} & = & u * f^{(1)} * tf^{(d)}\;, & u^{(-s)} & = & -\delta_{-1} * (\widetilde{v}^{(s)})^{\vee}\;, \\ v^{(s)} & = & v * g^{(1)} * tg^{(d)}\;, & v^{(-s)} & = & \delta_1 * (\widetilde{u}^{(s)})^{\vee}\;, \end{array}$$

   where $d = s - 1$ and where $tf^{(d)}$ and $tg^{(d)}$ are versions of $f^{(d)}$ and $g^{(d)}$ truncated up to an order chosen a priori.

2. Simulation of b_H: the process $ARIMA(0, s, 0)$ results from the convolution of a Gaussian white noise and a filter with impulsive response defined by:

$$\alpha_k^{(s)} \;=\; \sum_{p=0}^{k} (-1)^p \binom{-s}{p} \quad \text{with} \binom{-s}{p} = \frac{\Gamma(-s+1)}{\Gamma(p+1)\Gamma(-s-k+1)}$$

   Since the same numerical difficulty noted previously appears, the following approximation is proposed: $b_H = \gamma_j * \alpha^{(1)} * t\alpha^{(d)}$, where $t\alpha^{(d)}$ is a version of $\alpha^{(d)}$ truncated up to an order chosen a priori.

3. One then truncates the series (10) at some resolution J to get,

$$B_H^J(t) \;=\; \sum_{k\in\mathbb{Z}} b_H(k)\; \phi_{-J,k}^{(s)}(t) \;+\; \sum_{j=0}^{J}\sum_{k\in\mathbb{Z}} \gamma_j(k)\; 4^{-s}\; 2^{-js}\; \psi_{j,k}^{(s)}(t)\;. \qquad (11)$$

One can now use the pyramidal algorithm of Mallat adapted to biorthogonal wavelets, see Daubechies [9]: decomposition with the help of filters $u^{(-s)}$ and $v^{(-s)}$, and synthesis with the help of $u^{(s)}$ and $v^{(s)}$. The diagram in Fig.2 illustrates these operations. To remain coherent, getting a sample of length N through a resolution J needs to generate a fBm on a duration $T = 2^{-J}N$; the self-similarity property is used to get a sample at times i/N, for $i = 0, \ldots, N-1$. In practice, we have, as Abry and Sellan recommend, chosen a resolution of 6 or 7, and used the filters of a Daubechies wavelet of order 20, for its regularity properties.

Caption

$\triangleright$ $[\uparrow 2]$: dilatation operator defined by $[\uparrow 2]x_k = x_{2k}$.

$\triangleright$ b_H : $ARIMA(0, s, 0)$ simulated.

$\triangleright$ $g_0, \ldots, g_{J-1}$: simulated standard Gaussian variables.

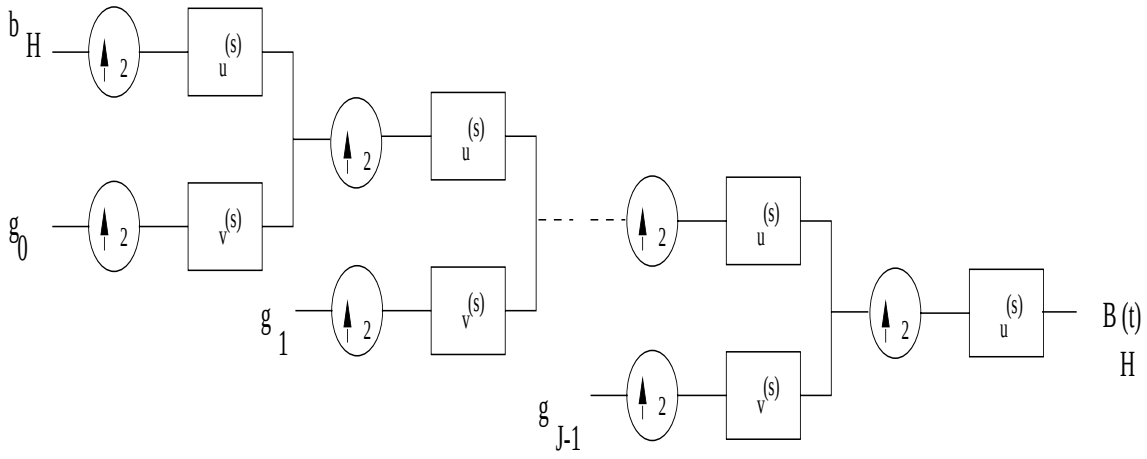

Figure 2: Diagram for the simulation of a fBm by wavelet synthesis.

3.4 Method of Choleski

Let Γ be the covariance matrix of a standard fBm, discretized at times i/N, for $i = 0, \ldots, N-1$. From (3),

$$(\Gamma)_{i,j} \;=\; \Gamma\left(\frac{i}{N}, \frac{j}{N}\right), \quad \text{for } i, j = 0, \ldots, N-1.$$

Define Γ' as the matrix Γ deprived of its first row and its first column. Since Γ' is a symmetric definite positive matrix, it admits a Choleski decomposition $\Gamma' = LL^t$, where L is a lower triangular matrix.

Thus simulating a sample of a fBm at times i/N for $i = 1, \ldots, N-1$ is equivalent to generate a vector Z of $(N-1)$ standard independent Gaussian variables and apply the product LZ. Indeed, LZ is a centered Gaussian vector and $\mathbb{E}\big((LZ)(LZ)^t\big) = \Gamma'$. Define $\widetilde{B} = \big(0, (LZ)^t\big)^t$, $\widetilde{B}$ is a sample of a fBm at times i/N for $i = 0, \ldots, N-1$. This method is the only one exact in theory, but due to a computational complexity of order $\mathcal{O}(N^3)$ and to the fact that Γ' is extremely ill conditioned, it is of interest to derive methods that are less computational demanding.

3.5 Method of Levinson

Let G be the autocovariance matrix of a standard fBm, discretized at times i/N, for $i = 0, \ldots, N-1$. From (4),

$$(G)_{i,j} \;=\; \gamma(j-i), \quad \text{for } i, j = 0, \ldots, N-1.$$

To avoid the computation of the Choleski decomposition of G (which would lead to a method identical to the one presented in Section 3.4), it suffices to remark that G is a Toeplitz matrix. The Toeplitz nature means that the first row of G suffices to reconstruct G, which leads to advanced algorithms to extract the square root of G. The following one can be found in [24].
At step 1 define:

$$\longrightarrow\; k_1 \;=\; -\gamma(\tfrac{1}{N})\,,\; \sigma_1^2 \;=\; 1\,,$$

$$\longrightarrow\; L^1 \;=\; \big(\,1,\; \gamma\left(\tfrac{1}{N}\right),\; \ldots,\; \gamma\left(\tfrac{N-1}{N}\right)\big)^t \;\text{ and }\; \widehat{L}^1 \;=\; \big(\,0,\; \gamma\left(\tfrac{1}{N}\right),\; \ldots,\; \gamma\left(\tfrac{N-1}{N}\right)\big)^t\,.$$

Then define vectors $L^j \;=\; \sigma_j^2 \left(\,0,\; \ldots,\; 0,\; 1,\; \ell_2^j,\; \ldots,\; \ell_{N-j+1}^j\,\right)^t$ and
$\widehat{L}^j \;=\; \sigma_j^2 \left(\,0,\; \ldots,\; 0,\; 0,\; \hat{\ell}_2^j,\; \ldots,\; \hat{\ell}_{N-j+1}^j\,\right)^t$.
At step $j+1$, one has:

$$\longrightarrow\; k_{j+1} \;=\; \tfrac{-\hat{\ell}_1^j}{\sigma_j^2} \quad \sigma_{j+1}^2 \;=\; \sigma_j^2(1-k_{j+1}^2)\,,$$

$$\longrightarrow \begin{pmatrix} \widehat{L}^{j+1} \\ L^{j+1} \end{pmatrix} = \begin{pmatrix} I_N & k_{j+1}\, Z \\ k_{j+1}\, I_N & Z \end{pmatrix} \begin{pmatrix} \widehat{L}^j \\ L^j \end{pmatrix}, \quad \text{where } Z = \begin{pmatrix} 0 & & & \\ 1 & 0 & & \\ & \ddots & \ddots & \\ & & 1 & 0 \end{pmatrix}.$$

Let us denote by D the matrix $D = diag(\sigma_1, \ldots, \sigma_N)$ and let $L = (L^1, \ldots, L^N)\; D^{-1}$. One can check that $LL^t = G$. Denoting by Z a vector of N standard independent Gaussian variables, the vector (LZ) defines a sample of a fGn at times i/N, for $i = 0, \ldots, N-1$. The cumulated sums of this sample define a sample of a fBm, $\widetilde{B}_H$ at the desired times (setting moreover $\widetilde{B}_H(0) = 0$). This method generates exactly a fGn with a computation cost $\mathcal{O}(N^2 \log(N))$ but still remains particularly slow within S-plus, as soon as $N > 1000$.

3.6 Method of Wood-Chan

Initially proposed by Davis and Harte [10], this method, available for any stationary Gaussian process, has been recently improved by Wood and Chan [29]. In order to extract the square root of the autocovariance matrix G, the idea is to embed G in a circulant matrix C, of size $m = 2^g$, $g \in \mathbb{N}^*$, and then to generate a vector $Y = (Y_0, \ldots, Y_{m-1})^t \rightsquigarrow \mathcal{N}(0, C)$, and thanks to an appropriate construction of C, to generate $(Y_0, \ldots, Y_{N-1})^t \rightsquigarrow \mathcal{N}(0, G)$. Let C be the matrix defined by:

$$C = \begin{pmatrix} c_0 & c_1 & \ldots & c_{m-1} \\ c_{m-1} & c_0 & \ldots & c_{m-2} \\ \vdots & \vdots & \ddots & \vdots \\ c_1 & c_2 & \ldots & c_0 \end{pmatrix} \quad \text{where} \quad c_j = \begin{cases} \gamma(\frac{j}{N}) & \text{if} \quad 0 \leq j < \frac{m}{2} \\ \gamma(\frac{m-j}{N}) & \text{if} \quad \frac{m}{2} < j < m-1 \, . \end{cases}$$

By construction, C is symmetric and circulant. One chooses m the first power of two, for questions of algorithmic rapidity, $m \geq 2(N-1)$ such that C is definite positive. The authors suggest an approximation when this condition can not be fulfilled. For a fBm, this condition is satisfied for the value $m = 2 * 2^\nu$, where 2^ν is the first power of two superior to N. Then, in order to diagonalize C, one uses a result of Brockwell and Davis: C can be decomposed as $C = Q \Lambda Q^*$, where Λ is the diagonal matrix of eigenvectors of C, and Q is the unitary matrix defined by

$$(Q)_{j,k} = m^{-1/2} \exp\left(-2i\pi \frac{jk}{m}\right) , \qquad \text{for} \quad j,k = 0, \ldots, m-1 \, .$$

Because Q is unitary, if $Y = Q\Lambda^{1/2}Q^*Z$ with $Z \rightsquigarrow \mathcal{N}(0, I_m)$, one has $Y \rightsquigarrow \mathcal{N}(0, C)$. Thus the simulation procedure of a fBm's sample at times i/N, for $i = 0, \ldots, N-1$ reduces itself to the three following steps:

1. Estimation of the eigenvalues of C: a matrix calculus shows that

$$\lambda_k = \sum_{j=0}^{m-1} c_j \exp\left(-2i\pi \frac{jk}{m}\right) = \sum_{j=0}^{m-1} c_j \exp\left(2i\pi \frac{jk}{m}\right) , \qquad \text{for } k = 0, \ldots, m-1 \, .$$

   This estimation may be done using the Fast Fourier Transform (direct or inverse).

2. Fast simulation of Q^*Z: by decomposing Q^*Z into real and imaginary parts, simulating of $W = Q^*Z$ amounts to the two following substeps:

   $\longrightarrow$ generate U, V two independent normal variables, $\mathcal{N}(0,1)$, and write $W_0 = U$ and $W_{\frac{m}{2}} = V$.

   $\longrightarrow$ for $1 \leq j < \frac{m}{2}$, generate U_j, V_j two independent normal variables, $\mathcal{N}(0,1)$ and write

$$\begin{aligned} W_j &= \frac{1}{\sqrt{2}}(U_j + iV_j) \, , \\ W_{m-j} &= \frac{1}{\sqrt{2}}(U_j - iV_j) \, . \end{aligned}$$

3. Reconstruction of X: the last step consists in calculating

$$X\left(\frac{k}{N}\right) \;=\; \frac{1}{\sqrt{m}} \sum_{j=0}^{m-1} \sqrt{\lambda_j}\; W_j \;\exp\left(-2i\pi\,\frac{jk}{m}\right), \qquad \text{for } \; k = 0, \ldots, m-1 \; .$$

using again the FFT. We get a sample of a fBm, denoted $\widetilde{B}_H$ by evaluating cumulated sums of the vector $\left\{X(0), X(\frac{1}{N}), \ldots, X(\frac{N-1}{N})\right\}$ and setting moreover $\widetilde{B}_H(0) = 0$.

The method of Wood and Chan is exact for simulating a fGn, has a complexity of $N\log(N)$, and in a practical point of view is fast even for large values of N.

3.7 About the approximation of a fBm via the cumulated sums of a fGn

Let $\widetilde{B}_H$ denote the vector defined by:

$$\begin{aligned} \widetilde{B}_H(0) &= 0 \\ \widetilde{B}_H(i/N) &= \sum_{k=0}^{i} \widetilde{X}(k/N), \quad i = 1, \ldots, N-1 \; , \end{aligned}$$

where $\widetilde{X}$ denotes a sample of a fGn generated at times $i/N, i = 0, \ldots, N-1$. Recall that, $\mathbb{E}\left(\widetilde{X}(i/N)\widetilde{X}(j/N)\right) = \gamma\left((j-i)/N\right),\ i, j = 0, \ldots, N-1$. So, it is easy to see that

$$\mathbb{E}\left(\widetilde{B}_H\,(i/N)\,\widetilde{B}_H\,(j/N)\right) \;=\; \sum_{k=0}^{i}\sum_{l=0}^{j} \gamma\left((j-i)/N\right), \quad i, j = 1, \ldots, N-1.$$

Thus, one can discuss the approximation of a sample of a fBm via the cumulated sums of a sample of a fGn by estimating the relative error done on the second order structure, via the function E defined on $\{0, \ldots, N-1\}^2$ by

$$E(i,j) \;=\; \begin{cases} 0 & \text{if } i \text{ or } j = 0 \\ \left|\left(\Gamma(i,j) - \sum_{k=0}^{i}\sum_{l=0}^{j}\gamma(j-i)\right)/\,\Gamma(i,j)\right| & \text{otherwise.} \end{cases}$$

Figure Fig.3 displays the function E computed for different values of H. It is clear that such an approximation involves negligible mistakes on the covariance structure.

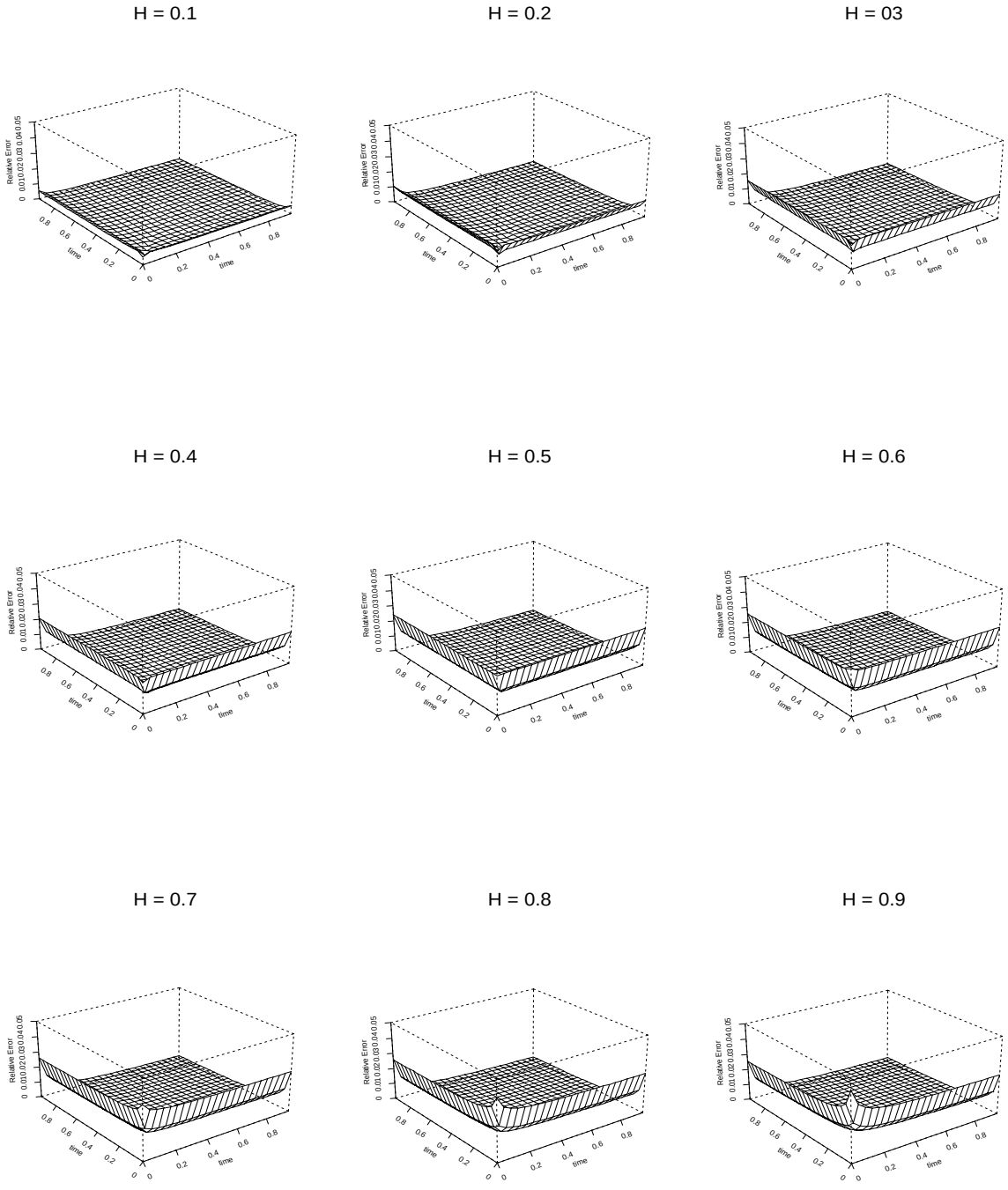

Figure 3: Relative Error on the covariance structure of a fBm approximated via the cumulated sums of a fGn, for different values of H.

4 Identification of the fractional Brownian motion

4.1 Statement of the problem

The irregularity's analysis of data modelled by a fBm, the study of its spectral behavior, and any forecasting problem based on fBm imply the necessity to estimate the Hurst parameter. In this Section, we briefly describe the main parametric methods to estimate the self-similarity parameter. We distinguish four approaches:

1. Spectral methods: log-periodogram, a variant of Lobato and Robinson's method.
2. Maximum likelihood: Whittle's estimator.
3. Time-scale methods: wavelet decomposition of the fBm.
4. Temporal methods: number of level crossings, discrete variations.

This list is not exhaustive (the R/S method or correlogram's approach has not been considered here, see *e.g.* Beran [6]) but presents the different ways of tackling the identification that have been discussed in the literature recently. Later on, we address the same problem when observing a sample $(B_{H,C})$ of a non-standard fBm of length N at times i/N for $i = 0, \ldots, N-1$; (X) will denote the sample of the increments of $(B_{H,C})$. Unless otherwise stated, the scale coefficient C is supposed to be unknown.

4.2 Spectral methods

4.2.1 Log-periodogram

This approach consists in exploiting, on the one hand, the spectral signature of the fGn, $f(\lambda) \sim c_f|\lambda|^{1-2H}$, as $|\lambda| \to 0$, and on the other hand the fact that the periodogram defined by

$$I_N(\lambda) = \frac{1}{2\pi N} \left| \sum_{t=0}^{N-1} X(t)\, e^{-it\lambda} \right|^2, \qquad \text{for } \lambda = \lambda_{k,N} = \frac{2\pi k}{N},$$

is an asymptotical unbiased estimator of the spectral density. One immediately notices that

$$\log \mathbb{E}\big(I_N(\lambda)\big) \simeq \log c_f + (1-2H)\log(|\lambda|),$$

pointing out the linearity in H of $\log \mathbb{E}(I_N(\lambda))$ in a neighborhood of 0. Let $1 \le m_1 < m_2 \le N^* = [N - 1/2]$. Define $\widehat{\alpha}_N$ the estimator deduced from the linear regression of $\big\{\log(I_N(\lambda_k))\big\}_{m_1 \le k \le m_2}$ on $\big\{\log(\lambda_k)\big\}_{m_1 \le k \le m_2}$, one gets an estimator of the self-similarity parameter by the equation

$$\widehat{H}_N(m_1, m_2) = \frac{1}{2}(1 - \widehat{\alpha}_N).$$

From a theoretical point of view, if m_1 and m_2 satisfy $\sqrt{m_2}\log(m_2)/m_1 \; + \; m_1\log^2(N)/m_2 \; \to \; 0$, Geweke and Porter-Hudak [13] prove the asymptotic normality for $\widehat{H}_N(m_1, m_2)$:

$$\frac{\sqrt{m_2}}{\log(N)} \left(\widehat{H}_N(m_1, m_2) - H \right) \; \xrightarrow{d} \; \mathcal{N}\left(0, \frac{\pi^2}{6}\right) . \tag{12}$$

4.2.2 Variant of Lobato and Robinson

By denoting $F(\lambda) = \int_0^\lambda f(\theta)d\theta$, Lobato and Robinson [20] remark that there exists, in a neighborhood of 0, a log-linear relation between two values of $F(\lambda)$. Let $q \in]0, 1[$, one gets immediately, $\frac{F(q\lambda)}{F(\lambda)} \sim q^{2-2H}$, as $|\lambda| \to 0$. By estimating $F(\lambda_k)$ by

$$\widehat{F}(\lambda_k) \; = \; \frac{2\pi}{N} \sum_{j=1}^{[N\lambda_{k,N}/2\pi]} I(\lambda_j), \qquad \text{for } \; k = m_1, \ldots, m_2 \; ,$$

one deduces an estimator of H:

$$\widehat{H}_N(q, m_1, m_2) \; = \; 1 \; - \; \frac{1}{2 \; \log(q)} \; \log\left\{ \; \frac{\widehat{F}(q\lambda_{m_2})}{\widehat{F}(\lambda_{m_2})} \; \right\} .$$

For $1/2 < H < 3/4$, Lobato and Robinson exhibit the optimal value of q via simulations. Let us mention that if one chooses m_1 and m_2 as previously, one may obtain an asymptotic normality result similar to (12).

4.3 Maximum likelihood: Whittle's estimator

Applied to our framework, the method consists in maximizing the log-likelihood of the fGn with parameter $\theta = \left(\frac{\sigma_\epsilon^2}{2\pi}, \; H \right)$, where σ_ϵ^2 is the innovation variance, given by:

$$L_N(x, \theta) = -\frac{N}{2}\log(2\pi) - \frac{1}{2}\log|det(G)| - \frac{1}{2}x^t G^{-1} x \; ,$$

where $G = G(\theta)$ is the covariance matrix of the fGn sample. The computational implementation points out two numerical problems: firstly the estimations of G^{-1} and $\log|det(G)|$ are particularly slow and expensive, and secondly G is extremely ill-conditioned. To get around these problems an approximation of the likelihood has been proposed, see *e.g.* Beran [6], giving rise to a new estimator called Whittle's estimator and explicitly given by:

$$\widehat{H}_N \; = \; \arg\min_H \sum_{j=1}^{N^*} \frac{I_N(\lambda_{j,N})}{f(\; \lambda_{j,N}, (1, H) \;)} \; ,$$

where $f(., (1, H))$ denotes the spectral density with parameters $(1, H)$ of the fGn, and $I_N(\lambda)$ the empirical periodogram. It is well-known, see *e.g.* Beran [6], that $\widehat{H}_N$ tends almost surely to H, and verifies the following asymptotic normality result:

$$\sqrt{N} \left(\widehat{H}_N - H \right) \; \xrightarrow{d} \; \mathcal{N}\left(0, \frac{2}{D} \right) , \qquad \text{with} \quad D = \frac{1}{2\pi}\int_{-\pi}^{\pi} \left\{ \frac{\partial}{\partial H} \log f(x, \theta) \right\}^2 dx \; .$$

Despite these performances, the Whittle's estimator suffers from being slowly executable, biased for finite samples, and very sensitive to perturbation with an additive white noise.

4.4 Time-scale method: wavelet decomposition.

Let $\{\psi_{j,k}(t) = 2^{-j/2}\psi_0(2^{-j}t - k),\ j = 1, \ldots, J, k \in \mathbb{Z}\}$ be the family of wavelet basis functions, generated from the mother wavelet ψ_0, itself defined via a multiresolution analysis of L^2, see [9]. We denote $< B_{H,C}, \psi_{j,k} >$ the coefficients of the discrete wavelet transform. Two reasons can explain the use of wavelet decomposition for the fBm identification:

1. Self-similarity of the wavelet coefficients: the self-similarity of the fBm ensures that the variance of the wavelet coefficients can be written

$$\mathbb{E}\big(< B_{H,C}, \psi_{j,k} >^2 \big) = \frac{1}{2} 2^{j(2H+1)} \frac{C^2}{N^{2H}} \int_{\mathbb{R}^2} |u - v|^{2H} \psi(u)\psi(v) du dv = K_H\ 2^{j(2H+1)} ,$$

   where $< \cdot, \cdot >$ denotes the inner product of $L^2(\mathbb{R})$.

2. The wavelet transform decorrelates the sample of the fBm. Indeed, if M denotes the number of vanishing moments of the mother wavelet, Flandrin and *al* [1], for example, show that

$$\mathbb{E}\big(< B_{H,C},\ \psi_{j,k} > < B_{H,C},\ \psi_{j',k'} > \big) \ = \ \mathcal{O}\big(|2^j k - 2^{j'} k'|^{2(H-M)} \big) .$$

Similarly to spectral methods, one notices the linearity in H of $\log_2 \mathbb{E}(< B_{H,C}, \psi_{j,k} >^2)$,

$$\log_2\ \mathbb{E}\big(< B_{H,C}, \psi_{j,k} >^2 \big) \ = \ j(2H + 1) \ + \ \log_2(K_H) .$$

One estimates $\mathbb{E}(< B_{H,C}, \psi_{j,k} >^2)$ by the empirical moment of order 2,

$$\mu_j \ = \ 2^{-j} \sum_{k=0}^{2^j - 1} < B_{H,C}, \psi_{j,k} >^2 .$$

Then, an estimator of H is deduced from a linear regression of $\big\{ \log_2(\mu_j) \big\}_{j_1 \le j \le j_2}$ on $\{j\}_{j_1 \le j \le j_2}$, where $[j_1, j_2]$ represents the used resolutions. Abry and *al* [3] have recently improved this method. Let ξ be the noise vector defined by

$$\xi_j \ = \ \log_2(\mu_j) - (2H + 1)j - \log_2(K_H) , \quad \text{for } j = j_1, \ldots, j_2$$

the above cited authors show that

$$\mathbb{E}(\ \xi_j\ ) \ = \ \frac{\Psi(2^{j-1})}{\log 2} \ - \ (j - 1) \quad \text{and} \quad \mathrm{Var}(\ \xi_j\ ) \ = \ \frac{\zeta(2, 2^{j-1})}{(\log 2)^2} ,$$

where Ψ is the Digamma function defined by $\Psi(t) = \frac{\Gamma'(t)}{\Gamma(t)}$, and $\zeta(2, t)$ is the generalized Riemann Zeta function. A new estimator of H is then deduced from a linear regression of $\big\{ \log_2(\mu_j) - \mathbb{E}(\xi_j) \big\}_{j_1 \le j \le j_2}$ on $\{j\}_{j_1 \le j \le j_2}$, weighted by $\big\{ \mathrm{Var}(\xi_j) \big\}_{j_1 \le j \le j_2}$.

In practice, the S-plus library `Wavethresh()` has been used to estimate the wavelet coefficients of the fBm.

4.5 Temporal methods

4.5.1 Statistic related to level 0 crossings of fGn

The smooth characteristic of the fGn covariance function ensures the convergence of the local time (see Azaïs [5]). Feuerverger and *al* [11] are based on this remark to estimate the Hölder exponent of a non differentiable Gaussian process, by counting the number of crossings of a given level. We present here a simplified version of this estimator. Let us define the mean number of 0 crossings of the fGn's sample (X) by the statistic

$$S_N \;=\; \frac{1}{N-1}\sum_{i=0}^{N-2} \mathbb{1}\!\left(X(i/N)X(i+1/N) < 0 \right) ,$$

where $\mathbb{1}(.)$ denotes the indicator function. The ergodicity of the increments and an integral calculus show that: $S_N \xrightarrow{a.s.} \theta/\pi$,

$$\text{where} \qquad \theta = \begin{cases} \arctan((1-r^2)^{1/2}/r) & \text{if } r > 0 \\ \pi/2 + \arctan(-r/(1-r^2)^{1/2}) & \text{otherwise,} \end{cases} \qquad \text{and} \quad r \;=\; 2^{2H-1} - 1 \; .$$

Assuming that one knows $\epsilon = \mathrm{sgn}(H - \frac{1}{2})$, one deduces immediately the following estimator

$$\widehat{H}_N \;=\; \frac{1}{2}\left\{ 1 + \log_2\left(1 + \epsilon |\cos(\pi S_N)| \right) \right\} .$$

Let us mention that for $0 < H < 3/4$, $\widehat{H}_N$ is asymptotically Gaussian with a rate of convergence $1/\sqrt{N}$, see Ho and Sun [15].

4.5.2 Discrete variations of the fBm

This method is relatively recent: the first results are due to Istas and Lang [16], on the one hand, and Kent and *al* [18], on the other hand. It relies on a specific filtering of the sample of a fBm that is designed to destroy long-range dependence of observations. In [7], we have generalized these results: the convergence of the k-th absolute moment of discrete variations, defined by

$$S_N(k,a) \;=\; \frac{1}{N-\ell}\sum_{i=\ell}^{N-1} \left| V^a(i/N) \right|^k , \qquad \text{for } k > 0$$

is studied. The parameter a denotes a filter of length $\ell+1$ and of order $p \geq 1$, verifying $\sum_{q=0}^{\ell} a_q \, q^r = 0$, for $r = 0, \ldots, p-1$, and (V^a) denotes the vector derived from filtering the vector $(B_{H,C})$ with the vector a,

$$V^a\left(\frac{i}{N}\right) \;=\; \sum_{q=0}^{\ell} a_q \; B_{H,C}\left(\frac{i-q}{N}\right) , \qquad\qquad \forall\, i \in \{\ell, \ldots, N-1\} \; .$$

We provide two classes of estimators of H: the first one, assuming that the scale coefficient is known, is convergent at a rate $1/\sqrt{N}\log(N)$, for $0 < H < 1$; the second one, without any assumption on C

is convergent at a rate $1/\sqrt{N}$.

Scale parameter known:

Without loss of generality, one can assume that $C = 1$. The definition of our estimators proceeds from ergodicity and self-similarity of fBm increments. Let π_H^a be the covariance function of (V^a). >From properties of a, one has

$$\pi_H^a(j) \;=\; \mathbb{E}\Big(\; V^a\Big(\frac{i}{N}\Big)V^a\Big(\frac{i+j}{N}\Big)\;\Big) \;=\; -\frac{1}{2}\,\frac{1}{N^{2H}}\sum_{q,r=0}^{\ell} a_q a_r\;|j+q-r|^{2H}\;. \tag{13}$$

Moreover by denoting $g_{k,a,N}(t) \;=\; \pi_t^a(0)^{\frac{k}{2}}\;E_k$, where $E_k = \mathbb{E}(|Y|^k)$ and $Y \rightsquigarrow \mathcal{N}(0,1)$, one shows that $\mathbb{E}(S_N(k,a)) \;=\; g_{k,a,N}(H)$ which allows us to define a first class of estimators based on a non-linear regression

$$\widehat{H}_N(k,a) \;=\; g_{k,a,N}^{-1}(\;S_N(k,a)\;)\;.$$

We prove ([7]) that if one chooses $p > H + 1/4$, this class is well-defined, converges almost surely to H, and verifies a central limit theorem for $0 < H < 1$, with a rate of convergence $1/\sqrt{N}\log(N)$. If $p = 1$ (resp. $p \geq 2$) the results are available for $0 < H < 3/4$ (resp. for $0 < H < 1$). It is also proved in [7] that the asymptotic constant of the estimators' variance is minimal for $k = 2$. In that case, one obtains the simple asymptotic result

$$\sqrt{N}\log(N)\Big(\;\widehat{H}_N(2,a)\;-\;H\;\Big)\;\xrightarrow{\;d\;}\;\mathcal{N}\left(\;0,\frac{1}{2}\;\sum_{i\in\mathbb{Z}}\rho_H^a(i)^2\;\right)\;,\quad\text{where}\;\;\rho_H^a(i)=\frac{\pi_H^a(i)}{\pi_H^a(0)}\;.$$

This asymptotic behavior will be further used to explore the quality of different simulation methods (see Section 6).

Scale parameter unknown:

Through spectral methods or wavelet methods, we have noticed that the use of a log-linear regression allowed us to exhibit estimators that are independent of C. We may apply a similar reasoning in this framework. Let us define the sequence of filters $(a^m)_{1\leq m\leq M}$ by

$$a_i^m \;=\left\{\begin{array}{ll} a_j & \text{if } i = jm \\ 0 & \text{otherwise}\end{array}\right.\;,\qquad \text{for } i = 0,\ldots,m\ell+1\;.$$

One immediately sees that

$$\mathbb{E}(S_N(k,a^m)) = \;m^{Hk}\;\mathbb{E}(S_N(k,a))\;.$$

By estimating $\mathbb{E}(S_N(k,a^m))$ by $S_N(k,a^m)$, an estimator of H can be deduced from a simple linear regression of $\big\{\log S_N(k,a^m)\big\}_{1\leq m\leq M}$ on $\big\{k\log(m)\big\}_{1\leq m\leq M}$. This procedure allows us to get a class of estimators denoted by $\widetilde{H}_N^{ols}(k,a,M)$ for which we have proved ([7]) the almost sure convergence and the asymptotic normality (with a rate of convergence in $1/\sqrt{N}$) if $p > H + 1/4$. Let us mention that

we have also considered a linear regression weighted by the diagonal weight matrix, of size $M \times M$, $\widehat{G}_k$ defined for $m = 1, \ldots, M$ by

$$\widehat{g}_{mm} \;=\; \sum_{j\geq 1} (c_{2j}^k)^2 \; (2j)! \sum_{i\in\mathbb{Z}} \rho_{\widetilde{H}^{ols}}^{a^m}(i)^{2j} \;, \quad \text{with } c_{2j}^k \;=\; \frac{1}{(2j)!} \prod_{q=0}^{j-1} (k-2q).$$

The consistency, the asymptotic normality, and the optimality for $k = 2$ hold again.

5 Examples of simulations

We intend to illustrate the implemented methods through simulations. For a sample size $N = 1000$, we generate 50 paths of standard fractional Brownian motion with parameter $H = 0.9$, discretized uniformly on $[0, 1[$, via methods of Mandelbrot, Sellan and *al*, Choleski, Levinson and Wood-Chan. The S-plus functions related to these methods are respectively `mvnFBM()`, `waveFBM()`, `cholFBM()`, `levFBM()`, `circFBM()`, see Section 7. For each path, we compute the estimators of H, via the methods explained in Section 4, and for which scripts are presented hereafter. Some Boxplots, Fig.4, Fig.5 and Fig.6, illustrate the results. A high value of the Hurst parameter was used to exhibit several weak points for the various simulation and estimation methods, discussed in previous Sections.

6 Quality of generators

In this section, our aim is to explore the quality of different simulation methods. Such a study has been already undertaken by Jennane and *al* [17]. They consider various simulators of the fBm, and give explicitly three procedures for testing the normality, the stationarity and the self-similarity of fGn. The two first procedures do not allow an exploration of the quality of generators: indeed Tables 3 and 4 of [17] show that four simulators among the five considered pass their tests successfully. Due to these omnibus tests, we have not envisaged to test the normality and the stationarity. And instead of using the proposed procedure to check the increments self-similarity, we orientate ourself towards another approach based on the theoretical asymptotic behavior of a *parametric* estimation method. Recall that if $\widehat{H}_N(2, a)$ denotes the estimator obtained by discrete variations of *standard* fractional Brownian motion (method explained p.16), we have the asymptotic result: if a denotes a filter of order $p > H + 1/4$ then

$$\sqrt{N}\log(N)\left(\,\widehat{H}_N(2,a) - H\,\right) \;\xrightarrow{d}\; \mathcal{N}(0, \frac{1}{2}\sum_{i\in\mathbb{Z}} \rho_H^a(i)^2)\;.$$

where ρ_H^a is given by (13). Let us describe how we use this property to extract an efficient method of simulation. For different values of the Hurst parameter, and for two filters (one is related to the increments, $a = Inc1 = (1, -1)$, the other is related to a Daubechies wavelet of order 4, $a = Db4 = (0.4829629, -0.8365163, 0.2241439, 0.1294095)$):

1. Simulate 200 paths of a standard fractional Brownian motion for each simulation method.

2. For $i = 1, \ldots, 200$:

   - Evaluate the estimator of the self-similarity parameter by discrete variations and denote it by $\widehat{H}_i(a)$.
   - Build a confidence interval of level $\alpha = 0.05$ related to the Gaussian distribution

   $$I(\sigma^2, H, a) \;=\; \left]H - u_\alpha \frac{\sigma}{\sqrt{N}\log(N)} \;;\; H + u_\alpha \frac{\sigma}{\sqrt{N}\log(N)}\right[\,,$$

   where $u_\alpha = \Phi^{-1}(\alpha/2) \approx 1.96$, and σ^2 is the asymptotic constant depending on H and the filter used, and given by Tab.1.

Hurst parameter	H=0.1	H=0.3	H=0.5	H=0.7	H=0.9
$a = Inc1$	0.6820765	0.5625909	0.5	0.7854074	$\times$
$a = Db4$	0.7790751	0.7396438	0.6388889	0.5922214	0.5661291

Table 1: Estimates of the asymptotic constant $\sigma^2 = \frac{1}{2}\sum_{i\in\mathbb{Z}} \rho_H^a(i)^2$ for different values of H and for filters $Inc1$ and $Db4$

3. Estimate the percentage test success by the statistic:

$$S(H, a) \;=\; \frac{1}{200}\sum_{i=1}^{200} \mathbb{1}\left(\widehat{H}_i(a) \in I(\sigma^2, H, a)\right) .$$

The results are presented in Tab.2. For $H > 3/4$, the test has not been evaluated for the filter $Inc1$ since the asymptotic behavior for $\widehat{H}_N(2, Inc1)$ is available only for $0 < H < 3/4$. One can notice that the methods of Mandelbrot and *al* and Levinson are not efficient for low and high values of the Hurst parameter. Concerning the method of Choleski and the one developed by Wood and Chan, the results are quite satisfactory since the level 95% is almost always reached. It appears also clearly that the estimators deduced from samples simulated by the method of Sellan and *al* are strongly biased. This study points out that the most stable simulation method are the Choleski's method and the one based on circulant matrix.

As a general conclusion, since Subsection 3.7 illustrates the excellent approximation of a sample of a fBm via the cumulated sums of a fGn, we advice to use the method of Wood and Chan [29], for its rapidity (even for large N) for simulating a fBm.

Hurst parameter		H=0.1	H=0.3	H=0.5	H=0.7	H=0.9
`mvnFBM()`	$Inc1$	72.0 %	89.0 %	97.0 %	83.0 %	×
	$Db4$	92.0 %	93.0 %	96.0 %	3.0 %	0.0 %
`waveFBM()`	$Inc1$	69 %	2.5 %	0 %	0 %	×
	$Db4$	87.0 %	22.0 %	20.0 %	7.5 %	0.0 %
`cholFBM()`	$Inc1$	97.5 %	94.5 %	93.0 %	91.0 %	×
	$Db4$	93.0 %	96.5 %	94.5 %	% 97.5 %	93.0 %
`levFBM()`	$Inc1$	50.5 %	79.5 %	97.0 %	63.5 %	×
	$Db4$	85.0 %	93.5 %	99.0 %	89.0 %	0.0 %
`circFBM()`	$Inc1$	94 %	93.5 %	92.5 %	92.0 %	×
	$Db4$	94.5 %	96.5 %	96.5 %	97.5 %	100 %

Table 2: Results of percentage test success for various methods of simulation.

7 S-plus scripts

In this Section, we present the S-plus scripts implementing the simulation and identification methods for the fBm, i.e. the methods described in Section 3 and 4. The Table Tab.7 summarizes this study: for each method its S-plus function is associated, with the arguments it needs, and eventually the subroutines related to. We advice the user to fix precisely the arguments of each function: for example, to generate a standard fBm of length N with parameter $H = 0.8$, by the Wood-Chan's method, one writes `circFBM(n=1000, H=0.8)`. By default `plotfBm=1`, the resulting path will be drawn. The function `waveST()` estimating H by a wavelet method has been written using the functions from the S-plus library `Wavethresh()`, available on the web at the following address `http://www.stat.cmu.edu/S/`. This library allows only the decomposition of a signal of length a power of 2. For a signal of any length N, one mirrors the data to the smallest power of two larger than N.

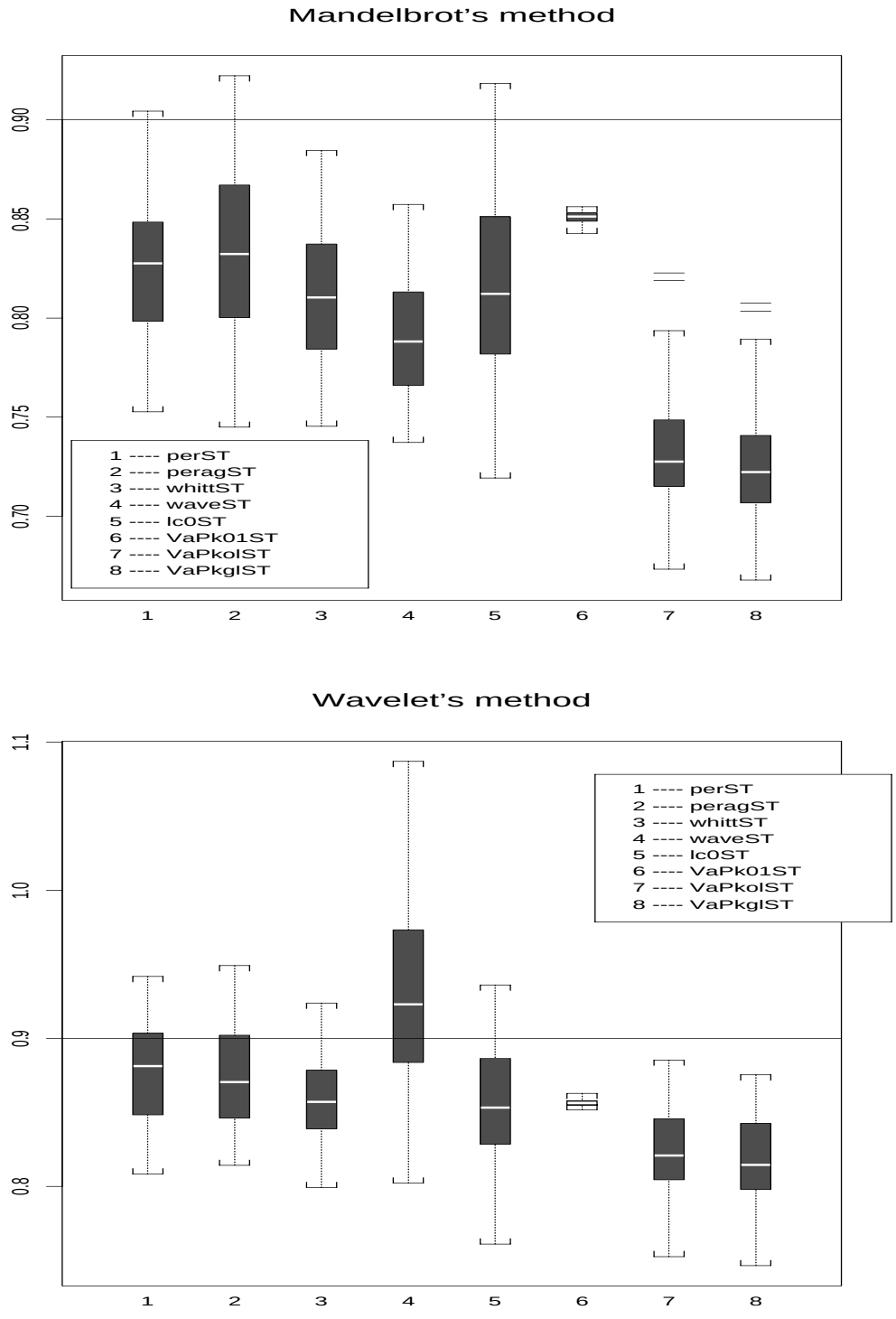

Figure 4: Boxplots of estimators of H for 50 paths simulated respectively by Mandelbrot's method and by wavelet synthesis.

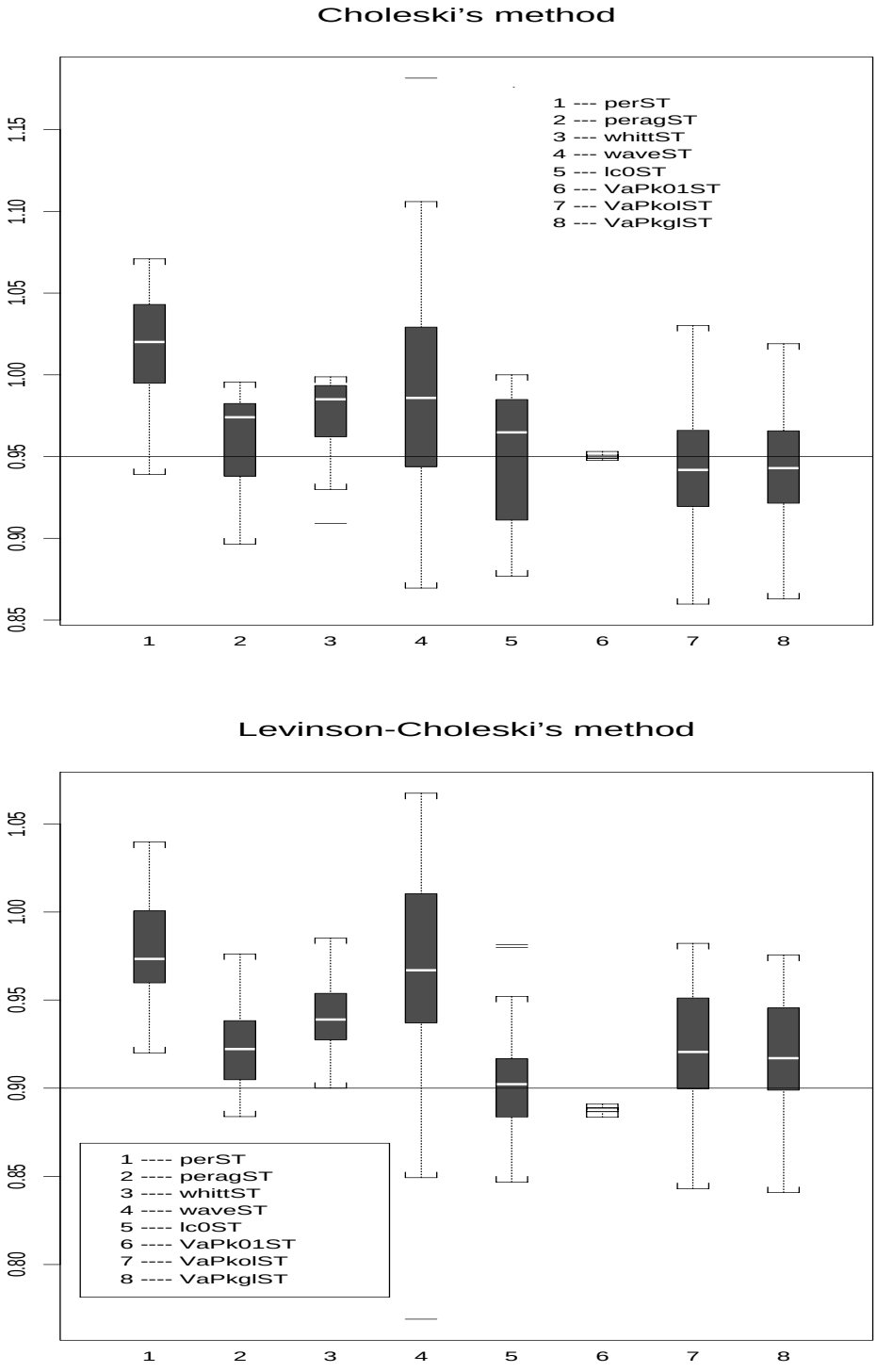

Figure 5: Boxplots of estimators of H for 50 paths simulated respectively by Choleski's and Levinson's method.

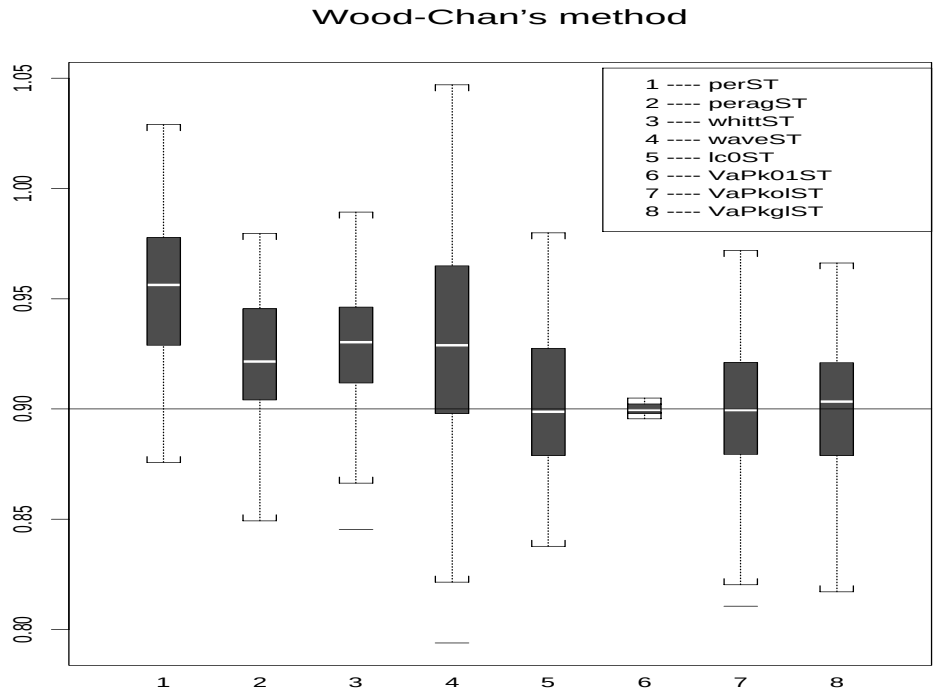

Figure 6: Boxplots of estimators of H for 50 paths simulated by Wood-Chan's method.

	S-plus name	Object
Generators of fBm	`mvnFBM(n= ,H= ,plotfBm= )`	Mandelbrot's method
	`waveFBM(n= ,H= ,J= ,plotfBm= )` subroutine: `convol(x,y)`	wavelet synthesis: fractional integration of a MRA
	`cholFBM(n= ,H= ,plotfBm= )`	Choleski decomposition of the covariance matrix.
	`levFBM(n= ,H= ,plotfBm= )`	Levinson's algorithm for Toeplitz matrices
	`circFBM(n= ,H= ,plotfBm= )`	method of circulant matrix
Estimators of H	`perST(fBm= ,m1= ,m2= ,llplot= )`	log-periodogram
	`peraggST(fBm= ,q= ,m1= ,m2= )`	variant of Lobato and Robinson
	`whittST(fBm= ,Hprel= )` subroutine: `spdFGN(Htry= ,n= )`	Whittle's estimator
	`waveST(fBm= ,j1= ,j2= ,llplot= )`	wavelet decomposition
	`lc0ST(fBm= ,sign= )`	level 0 crossings of fGn
	`VaPkstST(fBm= ,k= ,a= ,Hprel= )` subroutine: `piaH(a= ,H= ,i= )`	k-th absolute empirical moment of discrete variations of standard fBm
	`VaPkolST(fBm= ,k= ,a= ,M= ,llplot= )` subroutine: `piaH(a= ,H= ,i= )`	k-th absolute empirical moment of discrete variations of fBm: ordinary least squares
	`VaPkglST(fBm= ,k= ,a= ,M= ,llplot= )` subroutine: `piaH(a= ,H= ,i= )` `rhoadil(a,H,j,m1,m2)`	k-th absolute empirical moment of discrete variations of fBm: generalized least squares

Table 3: Summary of synthesis and analysis methods implemented over the software S-plus for fractional Brownian motion.

A Generators of the fBm

A.1 Mandelbrot's method

```
mvnFBM  <-  function(n, H, plotfBm)
{
## ------------------------------------------------------------------------
## Input        :               n : length of the desired sample
##                              H : self-similarity parameter
##                         plotfBm : =1 ----> plot a path of fBm
##
## Output       :         simulation of a standard fractional Brownian motion
##                        at times { 0, 1/n,..., n-1/n }
##                        by numerical approximation of stochastic integral
##
## Example      :    mvnFBM(n=500,H=0.6)
##
## ------------------------------------------------------------------------
## Coeurjolly      06/2000
## ------------------------------------------------------------------------
##
## Reference    :   Mandelbrot and Van Ness, Fractional brownian motions,
##                  fractional noises and applications, SIAM Review, 10,
##                  n.4, 1968.
## ------------------------------------------------------------------------
##
##
      if(missing(n)) n <- 500
      if(missing(H)) H <- 0.6
      if(missing(plotfBm)) plotfBm <- 1   ##
## --------------
## Initialization
## --------------
      dB1 <- rnorm(n)
      borne <- trunc(n^1.5)
      dB2 <- rnorm(borne)
      fBm <- rep(0, n)
      CH <- sqrt(gamma(2*H+1)*sin(pi*H))/gamma(H+1/2)  ##
## ------------
## Main program
## ------------
      ind1 <- (1:n)^(H - 1/2)
      ind2 <- (1:(borne + n))^(H - 1/2)
      for(i in (2:n)) {
           I1 <- dB1[(i - 1):1] * ind1[1:(i - 1)]
           I2 <- (ind2[(i + 1):(i + borne)] - ind2[1:borne]) * dB2
           fBm[i] <- sum(I1) + sum(I2)
```

```
      }
      fBm <- fBm * n^( - H) * CH
      fBm[1] <- 0 ##
## -----------
## plot of fBm
## -----------
      if(plotfBm == 1) {
           par(mfrow = c(1, 1))
           time <- (0:(n - 1))/n
           Nchar <- as.character(n)
           Nleg <- paste(c("N= ", Nchar), collapse = " ")
           Hchar <- as.character(round(H, 3))
           Hleg <- paste(c(", H=", Hchar), collapse = "")
           NHleg <- paste(c(Nleg, Hleg), collapse = "")
           leg <- paste(c("Path of a fractional Brownian motion
                  ----- parameters", NHleg), collapse = " : ")
           plot(time, fBm, type = "l", main = leg)
     }
     drop(fBm)
}
```

A.2 Wavelet synthesis: method of Sellan, Meyer and Abry

```
waveFBM <- function(n, H, J, plotfBm)
{
## ------------------------------------------------------------------------
## Input        :              n : length of the desired sample
##                             H : self-similarity parameter
##                             J : resolution
##                        plotfBm : =1 ----> plot a path of fBm
##
## Output       :         simulation of a standard fractional Brownian motion
##                        at times { 0, 1/n,..., n-1/n }
##                        by wavelet synthesis
##
## Example      :    waveFBM(n=500, H=0.6, J=7)
##
## See :  convol    ------> convolution of two vectors
##
## ------------------------------------------------------------------------
## Coeurjolly      06/2000
##
## Reference    :  Abry P. and Sellan F., The wavelet-based synthesis
##                 for fractional Brownian motion, Applied and computational
##                 harmonic analysis, 1996 + Matlab scripts from P. Abry
## ------------------------------------------------------------------------
##
```

```
##
      if(missing(n)) n <- 500
      if(missing(H)) H <- 0.6
      if(missing(J)) J <- 7
      if(missing(plotfBm)) plotfBm <- 1   ##
## ------------------------------
## Daubechies filter of length 20
## ------------------------------
      Db20 <- c(0.026670057901, 0.188176800078)
      Db20 <- c(Db20, 0.52720118932, 0.688459039454)
      Db20 <- c(Db20, 0.281172343661, -0.249846424327)
      Db20 <- c(Db20, -0.195946274377, 0.127369340336)
      Db20 <- c(Db20, 0.093057364604, -0.071394147166)
      Db20 <- c(Db20, -0.029457536822, 0.033212674059)
      Db20 <- c(Db20, 0.003606553567, -0.010733175483)
      Db20 <- c(Db20, 0.001395351747, 0.001992405295)
      Db20 <- c(Db20, -0.000685856695, -0.000116466855)
      Db20 <- c(Db20, 9.358867e-05, -1.3264203e-05)
      secu <- 2 * length(Db20)   ##
## --------------------------------
## quadrature mirror filters of Db20
## --------------------------------
      Db20qmf <- (-1)^(0:19) * Db20
      Db20qmf <- Db20qmf[20:1]
      nqmf <- -18 ##
## ------------------------------------------------------------------------
## truncated fractional coefficients appearing in fractional integration
## of the multiresolution analysis
## ------------------------------------------------------------------------
      prec <- 0.006
      hmoy <- c(1, 1)
      s <- H + 1/2
      d <- H - 1/2
      if(H == 1/2) {
            ckbeta <- c(1, 0)
            ckalpha <- c(1, 0)
      }
      else { ## truncature at order prec
            ckalpha <- 1
            ckbeta <- 1
            ka <- 2
            kb <- 2
            while(abs(ckalpha[ka - 1]) > prec) {
                  g <- gamma(1 + d)/gamma(ka)/gamma(d + 2 - ka)
                  if(is.na(g)) g <- 0
                  ckalpha <- c(ckalpha, g)
                  ka <- ka + 1
            }
```

```
            while(abs(ckbeta[kb - 1]) > prec) {
                  g <- gamma(kb - 1 + d)/gamma(kb)/gamma(d)
                  if(is.na(g)) g <- 0
                  ckbeta <- c(ckbeta, g)
                  kb <- kb + 1
            }
      } ##
## -------------------------
## number of starting points
## -------------------------
      nbmax <- max(length(ckbeta), length(ckalpha))
      nb0 <- n/(2^(J)) + 2 * secu ##
## ------------
## sequence fs1
## ------------
      fs1 <- convol(ckalpha, Db20)
      fs1 <- convol(fs1, hmoy)
      fs1 <- 2^( - s) * fs1
      fs1 <- fs1 * sqrt(2) ##
## ------------
## sequence gs1
##-------------
      gs12 <- convol(ckbeta, Db20qmf)
      gs1 <- cumsum(gs12)
      gs1 <- 2^(s) * gs1
      gs1 <- gs1 * sqrt(2) ##
## --------------
## initialization
## --------------
      nb1 <- nb0 + nbmax
      bb <- rnorm(nb1)
      b1 <- convol(bb, ckbeta)
      bh <- cumsum(b1)
      bh <- bh[c(nbmax:(nb0 + nbmax - 1))]
      appro <- bh
      tappro <- length(appro) ##
## ---------------
## useful function
## ---------------
      dilatation <- function(vect)
      { ## dilates one time vector vect
            ldil <- 2 * length(vect) - 1
            dil <- rep(0, ldil)
            dil[seq(1, ldil, by = 2)] <- vect
            drop(dil)
      }
##
## -------------------
```

```
## synthesis algorithm
## -------------------
      for(j in 0:(J - 1)) {
            appro <- dilatation(appro)
            appro <- convol(appro, fs1)
            appro <- appro[1:(2 * tappro)]
            detail <- rnorm(tappro) * 2^(j/2) * 4^( - s) * 2^( - j * s)
            detail <- dilatation(detail)
            detail <- convol(detail, gs1)
            detail <- detail[( - nqmf + 1):( - nqmf + 2 * tappro)]
            appro <- appro + detail
            tappro <- length(appro)
      }
      debut <- (tappro - n)/2
      fBm <- appro[c((debut + 1):(debut + n))]
      fBm <- fBm - fBm[1]
      fGn <- c(fBm[1], diff(fBm))
      fGn <- fGn * 2^(J * H) * n^( - H) ## path on [0,1]
      fBm <- cumsum(fGn)
      fBm[1]<- 0    ##
## -----------
## plot of fBm
## -----------
      if(plotfBm == 1) {
            par(mfrow = c(1, 1))
            time <- (0:(n - 1))/n
            Nchar <- as.character(n)
            Nleg <- paste(c("N= ", Nchar), collapse = " ")
            Hchar <- as.character(round(H, 3))
            Hleg <- paste(c(", H=", Hchar), collapse = "")
            NHleg <- paste(c(Nleg, Hleg), collapse = "")
            leg <- paste(c("Path of a fractional Brownian motion
                     ----- parameters",NHleg), collapse = " : ")
            plot(time, fBm, type = "l", main = leg)
      }
      drop(fBm)
}


convol <- function(x, y)
{
## ------------------------------------------------------------------------
## Input : x,y : vectors
##
## Output : convolution of vectors x and y
##
## ------------------------------------------------------------------------
## Coeurjolly  06/2000
```

```
## ------------------------------------------------------------------------
##
##
      if(missing(x) | missing(y)) break
      else {
            a <- c(x, rep(0, (length(y) - 1)))
            b <- c(y, rep(0, (length(x) - 1)))
            a <- fft(a, inverse = F)
            b <- fft(b, inverse = F)
            conv <- a * b
            conv <- Re(fft(conv, inverse = T))
            conv <- conv/length(conv)
            drop(conv)
      }
}
```

A.3 Choleski's method

```
cholFBM  <-  function(n, H, plotfBm)
{
## ------------------------------------------------------------------------
## Input        :              n : length of the desired sample
##                             H : self-similarity parameter
##                        plotfBm : =1 ----> plot a path of fBm
##
## Output       :         simulation of a standard fractional Brownian motion
##                        at times { 0, 1/n,..., n-1/n }
##                        by Choleski's decomposition of the covariance
##                        matrix of the fBm
##
## Example      :    cholFBM(n=500,H=0.6)
##
## ------------------------------------------------------------------------
## Coeurjolly      06/2000
## ------------------------------------------------------------------------
##
##
      if(missing(n)) n <- 500
      if(missing(H)) H <- 0.6
      if(missing(plotfBm)) plotfBm <- 1    ##
## ---------------------------------------
## Construction of covariance matrix of fBm
## ---------------------------------------
      H2 <- 2 * H
      matcov <- matrix(0, n - 1, n - 1)
      for(i in (1:(n - 1))) {
            j <- i:(n - 1)
            r <- 0.5 * (abs(i)^H2 + abs(j)^H2 - abs(j - i)^H2)
```

```
            r <- r/n^H2
            matcov[i, j] <- r
            matcov[j, i] <- matcov[i, j]
      }
      L <- chol(matcov)
      Z <- rnorm(n - 1)
      fBm <- t(L) %*% Z
      fBm <- c(0, fBm) ##
## -----------
## plot of fBm
## -----------
           if(plotfBm == 1) {
               par(mfrow = c(1, 1))
               time <- (0:(n - 1))/n
               Nchar <- as.character(n)
               Nleg <- paste(c("N= ", Nchar), collapse = " ")
               Hchar <- as.character(round(H, 3))
               Hleg <- paste(c(", H=", Hchar), collapse = "")
               NHleg <- paste(c(Nleg, Hleg), collapse = "")
               leg <- paste(c("Path of a fractional Brownian motion
                        ----- parameters",NHleg), collapse = " : ")
               plot(time, fBm, type = "l", main = leg)
           }
           drop(fBm)
      }
}
```

A.4 Levinson's method

```
levFBM <-  function(n, H, plotfBm)
{
## ------------------------------------------------------------------------
## Input        :              n : length of the desired sample
##                             H : self-similarity parameter
##                             plotfBm :  =1 ---> plot a path of fBm
##
## Output       :         simulation of a standard fractional Brownian motion
##                        at times { 0, 1/n,..., n-1/n }
##                        by Levinson's method
##
## Example      :    levFBM(n=500,H=0.6)
##
## ------------------------------------------------------------------------
## Coeurjolly   06/2000
##
## Reference    :  Peltier R.F., Processus stochastiques fractals avec
```

```
##                 applications en finance, these de doctorat, p.42, 28.12.1997
## ------------------------------------------------------------------------
##
##
      if(missing(n)) n <- 500
      if(missing(H)) H <- 0.7
      if(missing(plotfBm)) plotfBm <- 1 ##
## ------------------
## covariances of fGn
## ------------------
      k <- 0:(n - 1)
      H2 <- 2 * H
      r <- (abs((k - 1)/n)^H2 - 2 * (k/n)^H2 + ((k + 1)/n)^H2)/2 ##
## ---------------------------
## Initialization of algorithm
## ---------------------------
      y <- rnorm(n)
      fGn <- rep(0, n)
      v1 <- r
      v2 <- c(0, r[c(2:n)], 0)
      k <-  - v2[2]
      aa <- sqrt(r[1])  ##
## --------------------
## Levinson's algorithm
## --------------------
      for(j in (2:n)) {
            aa <- aa * sqrt(1 - k * k)
            v <- k * v2[c(j:n)] + v1[c((j - 1):(n - 1))]
            v2[c(j:n)] <- v2[c(j:n)] + k * v1[c((j - 1):(n - 1))]
            v1[c(j:n)] <- v
            bb <- y[j]/aa
            fGn[c(j:n)] <- fGn[c(j:n)] + bb * v1[c(j:n)]
            k <-  - v2[j + 1]/(aa * aa)
      }
      fBm <- cumsum(fGn)
      fBm[1] <- 0 ##
## -----------
## plot of fBm
## -----------
     if(plotfBm == 1) {
            par(mfrow = c(1, 1))
            time <- (0:(n - 1))/n
            Nchar <- as.character(n)
            Nleg <- paste(c("N= ", Nchar), collapse = " ")
            Hchar <- as.character(round(H, 3))
            Hleg <- paste(c(", H=", Hchar), collapse = "")
            NHleg <- paste(c(Nleg, Hleg), collapse = "")
            leg <- paste(c("Path of a fractional Brownian motion
```

```
                    ----- parameters",NHleg), collapse = " : ")
            plot(time, fBm, type = "l", main = leg)
    }
    drop(fBm)
}
```

A.5 Wood-Chan's method: circulant matrix

```
circFBM <- function(n, H, plotfBm)
{
## ------------------------------------------------------------------------
## Input        :              n : length of the desired sample
##                             H : self-similarity parameter
##                        plotfBm : =1 ---> plot a path of fBm
##
## Output       :         simulation of a standard fractional Brownian motion
##                        at times { 0, 1/n,..., n-1/n }
##                        by Wood-Chan's method
##
## Example      :    circFBM(n=500,H=0.6)
##
## ------------------------------------------------------------------------
## Coeurjolly      06/2000
##
## Reference   : Wood A. and Chan G., Simulation of stationnary Gaussian
##               processes, Journal of computational and grahical statistics,
##               vol.3, 1994.
## ------------------------------------------------------------------------
##
##
      if(missing(n)) n <- 500
      if(missing(H)) H <- 0.6
      if(missing(plotfBm)) plotfBm <- 1 ##
## ------------------------------------------------------------------
## first line of the circulant matrix, C, built via covariances of fGn
## ------------------------------------------------------------------
      lineC <- function(n, H, m){
             k <- 0:(m - 1)
             H2 <- 2 * H
             v <- (abs((k - 1)/n)^H2 - 2 * (k/n)^H2 + ((k + 1)/n)^H2)/2
             ind <- c(0:(m/2 - 1), m/2, (m/2 - 1):1)
             v <- v[ind + 1]
             drop(v)
      }
##
## --------------------
```

```
## next power of two > n
## ---------------------
      m <- 2
      repeat {
              m <- 2 * m
              if(m >= (n - 1))
      break
      }
      stockm <- m ##
## ---------------------------------------------
## research of the power of two (<2^18) such that
## C is definite positive
## ---------------------------------------------
      repeat {
               m <- 2 * m
               eigenvalC <- lineC(n, H, m)
               eigenvalC <- fft(c(eigenvalC), inverse = F)
               if((all(eigenvalC > 0)) | (m > 2^17))
                    break
      }
      if(m > 2^17) {
               cat("----> exact method, impossible!!", fill = T)
               cat("----> can't find m such that C is definite positive", fill=
                    break
      }
      else {
##
## ---------------------------------------------
## simulation of W=(Q)^t Z, where Z leads N(0,I_m)
## and (Q)_{jk} = m^(-1/2) exp(-2i pi jk/m)
## ---------------------------------------------
               ar <- rnorm(m/2 + 1)
               ai <- rnorm(m/2 + 1)
               ar[1] <- sqrt(2) * ar[1]
               ar[(m/2 + 1)] <- sqrt(2) * ar[(m/2 + 1)]
               ai[1] <- 0
               ai[(m/2 + 1)] <- 0
               ar <- c(ar[c(1:(m/2 + 1))], ar[c((m/2):2)])
               aic <-  - ai
               ai <- c(ai[c(1:(m/2 + 1))], aic[c((m/2):2)])
               W <- complex(real = ar, imaginary = ai) ##
## -------------------------
## reconstruction of the fGn
## -------------------------
               W <- (sqrt(eigenvalC)) * W
               fGn <- fft(W, inverse = F)
               fGn <- (1/(sqrt(2 * m))) * fGn
               fGn <- Re(fGn[c(1:n)])
```

```
               fBm <- cumsum(fGn)
               fBm[1] <- 0 ##
## -----------
## plot of fBm
## -----------
               if(plotfBm == 1) {
                   par(mfrow = c(1, 1))
                   time <- (0:(n - 1))/n
                   Nchar <- as.character(n)
                   Nleg <- paste(c("N= ", Nchar), collapse = " ")
                   Hchar <- as.character(round(H, 3))
                   Hleg <- paste(c(", H=", Hchar), collapse = "")
                   NHleg <- paste(c(Nleg, Hleg), collapse = "")
                   leg <- paste(c("Path of a fractional Brownian motion
                            ----- parameters",NHleg), collapse = " : ")
                   plot(time, fBm, type = "l", main = leg)
               }
               drop(fBm)
               }
}
```

B Estimators of the self-similarity parameter

B.1 Log-periodogram

```
perST <-  function(fBm, m1, m2, llplot)
{
## ------------------------------------------------------------------------
## Input        :              fbm : data modelled by a fractional Brownian
##                                   motion.
##                          m1...m2 : range of frequencies used
##                           llplot : =1 ---> a log-log-plot is done
##
## Output       :        Estimation of the self-similarity parameter, H,
##                       using a simple linear regression of log-periodogram
##                       on log-frequencies
##
## Example      :    perST( circFBM(n=500, m1=1, m2=249, H=0.6) )
##
## ------------------------------------------------------------------------
## Coeurjolly      06/2000
##
## Reference    :  Beran J., Statistics for long memory processes,
##                 Chapman and Hall, 1994.
## ------------------------------------------------------------------------
##
##
      if(missing(fBm)) fBm <- circFBM(n = 500, H = 0.6)
      if(missing(m1))  m1 <- 1
      if(missing(m2))  m2 <- trunc((length(fBm) - 1)/2)
      if(missing(llplot)) llplot <- 1
      fGn <- c(fBm[1], diff(fBm))
      n <- length(fGn) ##
## ---------------------------------
## periodogram and Fourier frequecies
## ---------------------------------
      I.lambda <- (Mod(fft(fGn, inverse = F)))^2/(2 * pi * n)
      I.lambda <- I.lambda[m1:m2]
      lambda <- (2 * pi * (m1:m2))/n ##
## ------------------------------------------------
## regression of log-periodogram on log-frequencies
## ------------------------------------------------
      Reg <- lsfit(log(lambda), log(I.lambda), intercept = T)
      Hest <- 0.5 * (1 - Reg$coef[2]) ##
## ------------
## log-log-plot
## ------------
      if(llplot == 1) {
```

```
            Hchar <- as.character(round(Hest, 4))
            Hleg <- paste(c("Hest", Hchar), collapse = "  =  ")
            par(mfrow = c(1, 1))
            plot(log(lambda), log(I.lambda), main =
                "Regression of log-periodogram on log-frequencies")
            abline(Reg)
            legend(min(log(lambda)), max(log(I.lambda)), c("", Hleg, ""))
    }
    drop(Hest)
}
```

B.2 Variant of Lobato and Robinson

```
peraggST <-  function(fBm, q, m1, m2)
{
## ------------------------------------------------------------------------
## Input        :              fbm : data modelled by a fractional Brownian
##                                    motion.
##                               q : scale parameter, 0<q<1.
##                          m1...m2 : range of frequencis.
##
## Output       :        Estimation of the self-similarity parameter, H,
##                       using aggregated periodogram
##
## Example      : peraggST( fBm=circFBM(n=500, H=0.6), q=0.27, m1=1, m2=249)
##
## ------------------------------------------------------------------------
## Coeurjolly      06/2000
##
## Reference    :   Lobato I., Robinson P.M., Averaged periodogram
##                  estimation of long memory, Journal of Econometrics,
##                  vol. 73, 1996.
## ------------------------------------------------------------------------
##
##
      if(missing(fBm)) fBm <- circFBM(n = 500, H = 0.6)
      if(missing(q))  q <- 0.27
      if(missing(m1)) m1 <- 1
      if(missing(m2)) m2 <- trunc((length(fBm) - 1)/2)
      fGn <- c(fBm[1], diff(fBm))
      n <- length(fGn) ##
## ------------------------------------------------------------------------
## periodogram at Fourier frequencies up to frequence indexed by m2 and q*m2
## ------------------------------------------------------------------------
      I.lambda <- (Mod(fft(fGn, inverse = F)))^2/(2 * pi * n)
      Iqm <- I.lambda[m1:(trunc(q * m2))]
      Im <- I.lambda[m1:m2]
      Hest <- 1 - log(sum(Iqm)/sum(Im))/(2 * log(q))
```

```
      drop(Hest)
}
```

B.3 Maximum likelihood: Whittle's estimator

```
whittST <- function(fBm, Hprel)
{
## ------------------------------------------------------------------------
## Input        :              fbm : data modelled by a fractional Brownian
##                                    motion.
##                            Hprel : preliminary estimation of H to initialize
##                                    minimization's algorithm
##
## Output       :        Estimation of the self-similarity parameter, H
##                       by maximization of approximated likeklihood :
##                       Whittle's estimator
##
## Example      :   whittST( fBm= circFBM(n=500,H=0.6), Hprel=0.6)
##
## See          :  spdFGN which calculates the normalized spectral density
##                 of the FGN
##
## ------------------------------------------------------------------------
## Coeurjolly      06/2000
##
## Reference    :  Beran J., Statistics for long memory processes
##                 Chapman and Hall, 1994.
## ------------------------------------------------------------------------
##
##
      if(missing(fBm)) fBm <- circfBm(n = 500, H = 0.6)
      if(missing(Hprel)) {
            Db4 <- c(0.4829629, -0.8365163, 0.22414386, 0.12940952)
            Hprel <- VaPkolST(fBm=fBm, k=2, a=Db4, M=5, 0)$Hols
      }
      fGn <- c(fBm[1], diff(fBm))
      n <- length(fGn)
      nstar <- trunc((n - 1)/2)
      perio <- (Mod(fft(fGn, inverse = F)))^2/(2 * pi * n)
      perio <- perio[1:nstar]
      global <- vector("list", 2)
      global[[1]] <- perio
      global[[2]] <- n
      assign("global", global, frame = 1)  ##
## ---------------------
## criterion to minimize
## ---------------------
```

```
      Q <- function(Htry) {
            perio <- global[[1]]
            n <- global[[2]]
            spd <- spdFGN(Htry, n)
            result <- (4 * pi)/n * sum(perio/spd)
            drop(result)
      }
      epsilon <- 1.0000000000000001e-05
      Hest <- nlminb(objective = Q, start = Hprel, lower = epsilon, upper = 1 -
      epsilon)$parameters
      drop(Hest)
}

spdFGN <- function(Htry, n)
{
## ------------------------------------------------------------------------
## Input :                      Htry : value of the self-similarity parameter
##                                     for which the spectral density of fGn
##                                     is computed
##                                n : length of time series
##
## Output :                      spd : spectral density of fGn computed at
##                                     Fourier frequencies, such that
##                                         integral( log(spd) ) = 0
##                                     (reduction of number of parameters)
##
## Example :     spdFGN( Htry=0.6, n=500 )
##
## ------------------------------------------------------------------------
## Coeurjolly 06/2000
##
## Reference :  Beran J., Statistics for long memory processes
##              Chapman and Hall, 1994.
## ------------------------------------------------------------------------
##
##
      if(missing(Htry)) Htry <- 0.6
      if(missing(n)) n <- 500
      alpha <- 2 * Htry + 1
      nstar <- trunc((n - 1)/2)
      clambda <- (sin(pi * Htry) * gamma(alpha))/pi ##
## ----------------------------------------
## computation of spd at Fourier frequencies
## ----------------------------------------
      j <- 2 * pi * ((-300):300)
      spd <- rep(0, nstar)
      for(k in (1:nstar)) {
            lambda <- (2 * pi * k)/n
```

```
            stocksum <- sum(abs(j + lambda)^( - alpha))
            spd[k] <- clambda * (1 - cos(lambda)) * stocksum
      }
## ----------------------
## renormalization of spd
## ----------------------
      theta <- exp(2/n * sum(log(spd)))
      spd <- spd/theta
      drop(spd)
}
```

B.4 Time-scale method: wavelet decomposition

```
waveST <- function(fBm, j1, j2, llplot)
{
## ------------------------------------------------------------------------
## Input        :              fbm : data modelled by a fractional Brownian
##                                   motion.
##                          j1...j2 : range of used resolutions level
##                           llplot : =1 ---> a log-log-plot is done
##
## Output       :        Estimation of the self-similarity parameter, H,
##                       using a simple linear regression of log2-scalogram
##                       on log2-scales.
##
## Example      :    waveST( circFBM(n=512, H=0.6), j1=2, j2=7 )
##
## Subroutine   :    wd (wavelet decomposition) included in the
##                   library Wavethresh available on the web to the
##                   following address: http://www.stat.cmu.edu/S/
##
## ------------------------------------------------------------------------
## Coeurjolly      06/2000
##
## Reference    : Abry P., Veitch D., Waveletet-based joint estimation
##                for the long-range dependence parameters, preprint 96.
## ------------------------------------------------------------------------
##
##
      n <- length(fBm)
      if(missing(fBm)) fBm <- circFBM(n = 512, H = 0.6)
      if((trunc(log(n)/log(2)) - log(n)/log(2)) != 0) {
##
## ----------------
## data duplication
## ----------------
              m <- 2
              repeat {
```

```
               m <- 2 * m
               if(m > n) break
              }
              fBm <- c(fBm, fBm[(n - 1):(2 * n - m)])
      }
      if(missing(j1)) j1 <- 2
      if(missing(j2)) j2 <- log(length(fBm))/log(2) - 3
      if(missing(llplot)) llplot <- 1
      n <- length(fBm)
      J <- log(n)/log(2) ##
## ------------------------------------------------
## wavelet decomposition of fGn and log-scalogram
## ------------------------------------------------
      fGnwd <- wd(c(fBm[1], diff(fBm)), 3, family = "DaubExPhase")
      scalog <- rep(0, j2 - j1 + 1)
      for(j in (j1:j2)) {  ## empirical variance of wavelet coefficients
                  scalog[j - j1 + 1] <- mean(accessD(fGnwd, j)^2)
      }
##
## ------------------------------------------------------
## epsilon_j = log(scalog[j])/log(2) - (2H+1)j - log2(K_H)
## estimations of E( epsilon_j) and Var( epsilon_j )
## ------------------------------------------------------
      moy.eps <- c(-0.83527463, -0.39006796, -0.18780535, -0.092044036,
                    -0.045553664, -0.022659505, -0.011300406, -0.0056428654,
                    -0.0028195982, -0.0014093405, -0.00070455559, -0.00035224913
                    -0.0001761174, -8.8056909e-05, -4.4028006e-05, -2.2013891e-0
     var.eps <- c(3.4237147, 1.3423458, 0.5907403, 0.27710725, 0.13423536,
                     0.066069658, 0.032776787, 0.016324379, 0.0081462477,
                     0.0040691462, 0.0020335796, 0.0010165415, 0.00050820866,
                     0.00025408878, 0.00012704048, 6.3519251e-05)
     resolution <- j1:j2
     moy.eps <- moy.eps[resolution]
     var.eps <- var.eps[resolution] ##
## ------------------------------------------------------------
## weighted linear regression of log2-scalogram on log2-scales
## ------------------------------------------------------------
     Reg <- lsfit(resolution, log(scalog)/log(2) - moy.eps, 1/var.eps)
     Hest <- 0.5 * (1 - Reg$coef[2]) ##
## ----------------------------------------
## bias brought by non-linearity of logarithm
## ----------------------------------------
     A <- resolution - mean(resolution)
     normA <- as.vector(t(A) %*% A)
     bias <- t(A) %*% moy.eps/normA ##
## ------------
## log-log-plot
## ------------
```

```
     if(llplot == 1) {
      Hchar <- as.character(round(Hest, 4))
      Hleg <- paste(c("Hest", Hchar), collapse = "  =  ")
      par(mfrow = c(1, 1))
      plot(resolution, log(scalog)/log(2) - moy.eps, main = "Regression of
                                          log2-scalogram on log2-scales")
      abline(Reg)
      if(Hest < 1/2) {
      yaxis <- max(log(scalog)/log(2) - moy.eps)
      }
      else yaxis <- (log(scalog[j2 - 2])/log(2) - moy.eps)
      legend(min(resolution), yaxis, c("", Hleg, ""))
     }
     drop(list(Hest = Hest, bias = bias))
}
```

B.5 Crossings of 0 of the fGn

```
lc0ST <- function(fBm, sign)
{
## ------------------------------------------------------------------------
## Input        :              fbm : data modelled by a fractional Brownian
##                                    motion.
##                            sign : value of sgn(H-1/2)
##
## Output       :        Estimation of the self-similarity parameter, H,
##                       using level 0 crossings of increments of
##                       fractional Brownian motion.
##
## Example      :   lc0ST( fBm= circFBM(n=500,H=0.6), sign=1)
##
## ------------------------------------------------------------------------
## Coeurjolly      06/2000
##
## Reference    :   Poggi J.M., Statistiques quadratiques pour le brownien
##                  fractionnaire, preprint universite Paris-sud Orsay,
##                  1995.
## ------------------------------------------------------------------------
##
##
      if(missing(fBm)) fBm <- circFBM(n = 500, H = 0.6)
      if(missing(sign)) { ## estimated sign
            Db4 <- c(0.4829629, -0.8365163, 0.22414386, 0.12940952)
            Hprel <- VaPkolST(fBm = fBm, k = 2, a = Db4, M = 5, 0)$Hols
            sign <- (Hprel - 0.5)/abs(Hprel - 0.5)
      }
      fGn <- c(fBm[1], diff(fBm))
      n <- length(fGn) ##
```

```
## ------------------------------------------------
## estimation of number of level 0 crossings of fGn
## ------------------------------------------------
      x1 <- fGn[ - n]
      x2 <- fGn[-1]
      prodx1x2 <- x1 * x2
      nlevcross <- length(prodx1x2[prodx1x2 < 0]) ##
## ---------------
## estimation of H
## ---------------
      theta <- (pi * nlevcross)/(n - 1)
      r <- sign * abs(cos(theta))
      Hest <- 0.5 * (1 + log(1 + r)/log(2))
      drop(Hest)
}
```

B.6 Discrete variations of the fBm: scale parameter known

```
VaPkstST <- function(fBm, k, a, Hprel)
{
## ------------------------------------------------------------------------
## Input        :              fbm : data modelled by a standard fractional
##                                   Brownian motion.
##                              k : power of discrete variations
##                              a : filter
##                          Hprel : preliminar estimation of Hto initialize
##                                   minimization's algorithm
##
## Output       :        Estimation of the self-similarity parameter, H,
##                       using the k-th absolute empirical moment of
##                       standard fractional Bronian motion
##
## Example      :    VaPkstST( fBm=circFBM(500,0.6), k=2, a=c(1,-1),
##                                Hprel=0.6 )
##
## See          :    piaH ----> calculates the covariance function of
##                                discrete variations of fBm.
## ------------------------------------------------------------------------
## Coeurjolly      06/2000
##
## Reference    :  Coeurjolly, Estimating the parameters of a fractional
##                 Brownian motion by discrete variations
##                 of its sample paths,submitted for publication, 1999
## ------------------------------------------------------------------------
##
##
      if(missing(fBm)) fBm <- circFBM(n = 500, H = 0.6)
```

```
      if(missing(k))   k <- 2
      if(missing(a))
            a <- c(0.4829629, -0.8365163, 0.22414386, 0.12940952)
      if(missing(Hprel)) { ## first estimation of H
            Db4 <- c(0.4829629, -0.8365163, 0.22414386, 0.12940952)
            Hprel <- VaPkolST(fBm = fBm, k = 2, a = Db4, M = 5, 0)$Hols
      }
##
## ------------------------------------------------------------
## estimation of the k-th absolute empirical moment of discrete
## variations of fBm
## ------------------------------------------------------------
      n <- length(fBm)
      Va <- filter(fBm, a, sides = 1)
      Va <- Va[ - (1:(length(a) - 1))]
      Ek <- (2^(k/2) * gamma(0.5 * k + 0.5))/gamma(0.5)
      SNka <- mean(abs(Va)^k)
      param <- vector("list", 5)
      param[[1]] <- n
      param[[2]] <- k
      param[[3]] <- a
      param[[4]] <- SNka
      param[[5]] <- Ek
      assign("param", param, frame = 1) ##
## ------------------------------------------
## minimization to get g_{k,a,n}^{-1} ( SNka )
## see Coeurjolly 99
## ------------------------------------------
      Q <- function(Hessai)
      {
            n <- param[[1]]
            k <- param[[2]]
            a <- param[[3]]
            SNka <- param[[4]]
            Ek <- param[[5]]
            pia0 <- piaH(a, Hessai, 0)
            res <- log(SNka) + Hessai * k * log(n) - log(pia0^(k/2) * Ek)
            res <- abs(res)
            drop(res)
      }
      epsilon <- 1.0000000000000001e-05
      Hest <- nlminb(objective = Q, start = Hprel, lower = epsilon, upper = 1 -
      epsilon)$parameters
      drop(Hest)
}

piaH <- function(a, H, i)
{
```

```
## ------------------------------------------------------------------------
## Input :                   a : filter
##                           H : self-similarity parameter
##                           i : indice for which the covariance is evaluated
##
## Output :         calculus of the covariance function of process of
##         discrete variations of fBm, given by
##         pi_aˆH (i) = -1/2 sum_{q=0}ˆ{l}{ a_q a_r |q-r+i|ˆ{2H} }
##
## Example :         piaH( a=c(1,-1), H=0.6, i=0 )
##
## ------------------------------------------------------------------------
## Coeurjolly 06/2000
## ------------------------------------------------------------------------
##
##
      if(missing(a)) a <- c(1, -1)
      if(missing(H)) H <- 0.6
      if(missing(i)) i <- 0
      l <- length(a) - 1
      d <- l + 1
      mat <- matrix(rep(0, d * d), ncol = d)
      for(q in (0:l)) {
                for(r in (0:l)) {
                          z <- a[q + 1] * a[r + 1] * abs(q - r + i)ˆ(2 * H)
                          mat[q + 1, r + 1] <- -0.5 * z
                }
      }
      piaH.i <- sum(apply(mat, 1, sum))
      drop(piaH.i)
}
```

B.7 Discrete variations of the fBm: scale parameter unknown

B.7.1 Ordinary least squares (ols)

```
VaPkolST <- function(fBm, k, a, M, llplot)
{
## ------------------------------------------------------------------------
## Input        :              fbm : data modelled by a fractional
##                                    Brownian motion.
##                               k : power of discrete variations
##                               a : filter
##                               M : maximum number of dilatations of a
##                          llplot : =1 ---> a log-log-plot is done
##
## Output       :
##                Hest,  Cest : estimation of the self-similarity parameter, H
```

```
##                               and the scale coefficient by a simple linear
##                               regression of
##                               log( S_n(k,aˆm) ) on log( m ) for m=1,...,M
##                               where aˆm is the filter a, dilated m times and
##                               S_n(k,aˆm) is the k-th absolute empirical
##                               moment of discrete variations of fractional
##                               Brownian motion
##                         LN :     vector of log( S_n(k,aˆm) )
##                       bias :     bias brought by the non-linearity of the
##                                  logarithm function
##
## Example      :    VaPkolST( fBm=circFBM(500,0.6), k=2, a=c(1,-1),
##                               M=5, llplot=1 )
##
## See          :  piaH ---> calculates the covariance function of
##                             discrete variations of fBm
##
## ------------------------------------------------------------------------
## Coeurjolly      06/2000
##
## Reference    :  Coeurjolly, Estimating the parameters of a fractional
##                 Brownian motion by discrete variations
##                 of its sample paths,submitted for publication, 1999
##
## ------------------------------------------------------------------------
##
##
      if(missing(fBm)) fBm <- circFBM(n = 500, H = 0.6)
      if(missing(k))     k <- 2
      if(missing(a))
            a <- c(0.4829629, -0.8365163, 0.22414386, 0.12940952)
      if(missing(M)) M <- 5
      if(missing(llplot)) llplot <- 1
      N <- length(fBm)   ##
## ------------------------------
## dilatation m times of filter a
## ------------------------------
      dilatation <- function(a, m)
      {
            la <- length(a)
            am <- rep(0, m * la - 1)
            am[seq(1, m * la - 1, by = m)] <- a
            drop(am)
      }
## ---------------------------------------------
## estimation of H by a simple linear regression
## ---------------------------------------------
      la <- length(a)
```

```
      Vam <- rep(0, M)
      SNkam <- rep(0, M)
      Vam <- filter(fBm, a, sides = 1)
      Vam <- Vam[ - (1:(la - 1))]
      SNkam[1] <- mean(abs(Vam)^k)
      for(m in (2:M)) {
            am <- dilatation(a, m)
            Vam <- filter(fBm, am, sides = 1)
            lam <- m * la - 1
            Vam <- Vam[ - (1:(lam - 1))]
            SNkam[m] <- mean(abs(Vam)^k)
      }
      LN <- log(SNkam)
      m <- 1:M
      Reg <- lsfit(log(m), LN, intercept = T)
      Hols <- Reg$coef[2]/k ##
## ------------------------------------------------------
## calculus of the bias brought by non-linearity of log
## ------------------------------------------------------
      mean.eps <- rep(0, M)
      for(m in (1:M)) {  ## approximation by expansion of psi(z)-log(z)
            z <- 0.5 * (N - m * (la - 1))
            mean.eps[m] <- -1/2/z - 1/12/z^2 + 1/120/z^4 - 1/252/z^6
      }
      m <- 1:M
      A <- log(m) - mean(log(m))
      norm.A <- as.vector(t(A) %*% A)
      bias <-  - t(A) %*% mean.eps/2/norm.A ##
## ----------------------------------
## estimation of the scale coefficient
## ----------------------------------
      Ek <- (2^(k/2) * gamma(0.5 * k + 0.5))/gamma(0.5)
      pia0 <- piaH(a, Hols, 0)
      thetaols <- Reg$coef[1]
      Cols <- N^(Hols)/Ek^(1/k)/sqrt(pia0) * exp(thetaols/k) ##
## ------------
## log-log-plot
## ------------
       if(llplot == 1) {
            par(mfrow = c(1, 1))
            Hchar <- as.character(round(Hols, 4))
            Hleg <- paste(c("Hest", Hchar), collapse = "  =  ")
            plot(log(m), log(SNkam), main = "Regression of
                              log( SN(k,a^m) ) on log(m)")
            abline(Reg)
            legend(min(log(m)), max(log(SNkam)), c("", Hleg, ""))
      }
      drop(list(Hols = Hols, Cols = Cols, LN = LN, bias = bias,  ))
```

```
}
```

B.7.2 Generalized least squares (gls)

```
VaPkglST <- function(fBm, k, a, M, llplot)
{
## ------------------------------------------------------------------------
## Input        :              fbm : data modelled by a fractional
##                                    Brownian motion.
##                               k : power of discrete variations
##                               a : filter
##                               M : maximum number of dilatations of a
##                          llplot : =1 ---> a log-log-plot is done
##
## Output       :
##               Hest,  Cest : estimation of the self-similarity parameter, H
##                             and the scale coefficient by a weighted linear
##                             regression of
##                             log( S_n(k,aˆm) ) on log( m ) for m=1,...,M
##                             where aˆm is the filter a, dilated m times and
##                          S_n(k,aˆm) is the k-th absolute empirical moment
##                        of discrete variations of fractional Bronian motion
##
##
##
## Example      :    VaPkglST( fBm=circFBM(500,0.6), k=2, a=c(1,-1),
##                               M=5, llplot=1 )
##
## See          :    piaH  ---> calculates the covariance function of
##                             discrete variations of fBm for the filter a
##                   rhoadil -->  calculates the covariance function of
##                             discrete variations of fBm for filters aˆm and
##                             aˆn (filters dilated of a)
##                   VaPkolST
##
## ------------------------------------------------------------------------
## Coeurjolly      06/2000
##
## Reference    :  Coeurjolly, Estimating the parameters of a fractional
##                 Brownian motion by discrete variations
##                 of its sample paths,submitted for publication, 1999
##
## ------------------------------------------------------------------------
##
##
      if(missing(fBm)) fBm <- circ(n = 500, H = 0.6)
      if(missing(k)) k <- 2
```

```
      if(missing(a))
            a <- c(0.4829629, -0.8365163, 0.22414386, 0.12940952)
      if(missing(M)) M <- 5
      if(missing(llplot)) llplot <- 1
      N <- length(fBm)
      stock <- VaPkolST(fBm, k, a, M, 0) ##
## -------------------------
## Preliminar estimation of H
## -------------------------
      Hols <- stock$Hols
      LN <- stock$LN ##
## ----------------------------------------
## Coefficients of asymptotic variance deduced
## by Hermite's expansion
## ----------------------------------------
      coefVar <- function(j, k)
      {
            res <- 1
            for(q in (0:(j - 1))) {
                res <- res * (k - 2 * q)
            }
            res <- resˆ2/prod(1:(2 * j))
            drop(res)
      }
      if((k %% 2) == 0)  seuilj <- k %/% 2
      else   seuilj <- 50   ## approximation
      coef.Var <- rep(0, seuilj)
      for(j in (1:seuilj)) {
            coef.Var[j] <- coefVar(j, k)
      }
## ---------------------------
## asymptotic covariance matrix
## ---------------------------
      Gk <- matrix(0, M, M)
      for(m in (1:M)) {
           seuil <- 50
           rhoamam <- rep(0, 2 * seuil + 1)
           for(i in (( - seuil):seuil)) {
                    z <- rhoadil(a, Hols, i, m, m)
                    rhoamam[i + seuil + 1] <- z
           }
           for(j in (1:seuilj)) {
                    z <- coef.Var[j] * sum(rhoamamˆ(2 * j))
                    Gk[m, m] <- z
           }
      }
## ----------------------------------------
## normal equations: generalized least squares
```

```
## ------------------------------------------
      Gkinv <- solve(Gk)
      vm <- c(2 * log(1:M), rep(1, M))
      X <- matrix(vm, ncol = 2)
      matS <- t(X) %*% Gkinv %*% X
      Sinv <- solve(matS)
      matT <- t(X) %*% Gkinv %*% LN
      betaest <- Sinv %*% matT   ##
## ----------------------------
## estimations of Hgls and Cgls
## ----------------------------
      Hgls <- betaest[1]
      pia0 <- piaH(a, Hgls, 0)
      Cgls <- N^(Hgls)/sqrt(pia0) * exp(betaest[2]/2)
      drop(list(Hgls = Hgls, Cgls = Cgls))
}


rhoadil <- function(a, H, j, m1, m2)
{
## ------------------------------------------------------------------------
## Input  :                    a : filter
##                             H : self-similarity parameter
##                             j : indice for which the covariance is evaluated
##                            m1 : first number of dilatations of a
##                            m2 : second number of dilatations of a
##
## Output       :          Correlation function of
##                         discrete variations of fBm for filters a^m1 and
##                         a^m2 (filters dilated of a)
##
## ------------------------------------------------------------------------
## Coeurjolly      06/2000
## ------------------------------------------------------------------------
##
##
     l <- length(a) - 1
     d <- l + 1
     mat <- matrix(rep(0, d * d), ncol = d)
     mat0 <- matrix(rep(0, d * d), ncol = d)
     for(q in (0:l)) {
         for(r in (0:l)) {
                     mat[q + 1, r + 1] <- a[q+ 1] * a[r + 1] *
                              abs(j + m1 * q - m2 * r)^(2 * H)
                     mat0[q + 1, r + 1] <- a[q + 1] * a[r+ 1] *
                                  abs(m1 * q - m2 * r)^(2 * H)
         }
     }
```

```
    piam1am2.j <- sum(apply(mat, 1, sum))
    piam1am2.0 <- sum(apply(mat0, 1, sum))
    rhoam1am2.j <- piam1am2.j/piam1am2.0
    drop(rhoam1am2.j)
}
```

Acknowledgements

I would like to thank P. Abry for providing me the Matlab scripts of Sellan and *al* implementing their simulation method, and A. Antoniadis, my thesis advisor, for his helpful comments and suggestions.

References

[1] Abry P.,Gonçalvès P., Flandrin P. (1995), *Wavelets, spectrum analysis and 1/f processes*, Wavelets and Statistics, Lectures Note in Statistics, Vol.103, p.15-29.

[2] Abry P. and Sellan F. (1996), *The wavelet-based synthesis for fractional Brownian motion*, Applied and computational harmonic analysis, Vol.3, p.377-383.

[3] Abry P., Veitch D. (1998), *Wavelet Analysis of Long-Range-Dependent Traffic* , IEEE Transactions on Information Theory, Vol.44, No.1, p.2-15.

[4] Adler R.J., Feldman R.E. and Taqqu M. (1998), *A Practical Guide to Heavy Tails: Statistical Techniques and Applications*, Birkhäuser, Boston.

[5] Azaïs J.M. (1990), *Conditions for convergence of number of crossings to the local time. Application to stable processes with independent increments and to Gaussian processes*, Probability and mathematical statistics, Vol.11, p.19-36.

[6] Beran J. (1994), *Statistics for long memory processes*, Chapman and Hall, London.

[7] Coeurjolly J.F. (1999), *Estimating the parameters of a fractional Brownian motion by discrete variations of its sample paths*, Submitted for publication.

[8] Collins J.J., De Luca C.J. (1994), *Upright, correlated random walks: A statistical-biomechanics approach to the human postural control system*, Chaos, Vol.5, No1, p.57-63.

[9] Daubechies I. (1992), *Ten lectures on wavelets*, CBMS-NSF Regional conference series in applied mathematics.

[10] Davis R.B. and Harte D.S. (1987) *Tests for Hurst effect*, Biometrika, 74, 95-101.

[11] Feuerverger A., Hall P., Wood T.A. (1994), *Estimation of fractal index and fractal dimension of a Gaussian process by counting the number of level crossings*, Journal of times series analysis, Vol.15, No.6, p.587-606.

[12] Frisch, U. (1997), *Turbulence*, Cambridge University Press.

[13] Geweke J. and Porter-Hudak S. (1983), *The estimation and application of long-memory time-series models*, Journal of Time Series Analysis, Vol.4, p.221-238.

[14] Higuchi T. (1988), *Approach to an irregular time series on the basis of the fractal theory*, Physica D, Vol.31, p.277-283.

[15] Ho H.C., Sun T.C. (1987), *A central limit theorem for non-instantaneous filters of Gaussian stationary process*, Journal of Multiv. Anal., Vol.22, 144-155.

[16] Istas J. et Lang G. (1997), *Variations quadratiques et estimation de l'exposant de Hölder local d'un processus gaussien*, Ann. Inst. Henri Poincaré, Vol.33, p.407-436.

[17] Jennane R., Harba R., Jacquet G. (1996), *Estimation de la qualité des méthodes de synthèse du mouvement brownien fractionnaire*, Traitement du Signal, Vol.13, No4.

[18] Kent J.T. and Wood A.T.A. (1997), *Estimating the fractal dimension of a locally self-similar Gaussian process using increments*, Journal of the Royal Statistical Society, Series B, Vol 59, No.3, p.679-700.

[19] Kolmogorov A.N. (1940), *Winersche spiralen und einige andere interessante kurven in Hilbertschen Raum*, S.R. (Doklady) Acad. Sci. USSR (N.S.), Vol.26, p.115-118.

[20] Lobato I., Robinson P.M. (1996), *Averaged periodogram estimation of long memory*, Journal of Econometrics, Vol. 73, p.303-324.

[21] Mandelbrot B., Van Ness, Wallis J. (19689), *Fractional brownian motions, fractional noises and applications*, SIAM review Vol.10, p.422-437.

[22] Peltier R.F. et Lévy-Véhel J. (1994), *A new method for estimating the parameter of fractional Brownian motion*, Rapport de recherche de l'INRIA, n.2396, p.1-40.

[23] Peltier R.F. et Lévy-Véhel J. (1995), *Multifractional Brownian motion : definition and preliminary results*, Rapport de recherche de l'INRIA, n.2645.

[24] Peltier R.F. (1998) *Processus stochastiques fractals avec applications en finance*, Thèse de doctorat, Univ. Paris VI.

[25] Poggi J.M. (1995), *Statistiques quadratiques pour le brownien fractionnaire*, Univ. Paris-Sud Orsay, preprint.

[26] Sellan F. (1995), *Synthèse de mouvements browniens fractionnaires à l'aide de la transformation par ondelettes*, Académie des sciences de Paris, t.321, Sér.I, p.351-358.

[27] Taqqu M., Teverovsky V. (1995), *Estimators for long-range dependence: an empirical study*, Fractals, Vol.3, No.4, p.785-798.

[28] Taqqu M., Teverovsky V. and Willinger W. (1997), *Is network traffic self-similar or multifractal?*, Fractals, Vol.5, p.63-73.

[29] Wood A. and Chan G. (1994), *Simulation of stationary Gaussian processes in* $[0, 1]^d$, Journal of computational and graphical statistics, Vol.3, No.4, p.409-432.

[30] Becker, Chambers and Wilks (1988), *The new S-language,* Computer Science Series.

[31] Chambers J.M and Hastie T.J. (1992), *Statistical models in S*, Computer Science Series.

[32] Nason G.P. et Silverman B.W. (1994), *The Discrete Wavelet Transform in S*, Journal of computational and graphical statistics, Vol.3, p.163-191.