Fast Synthesis of Persistent Fractional Brownian Motion

PEDRO R. M. INÁCIO, Instituto de Telecomunicações and University of Beira Interior
MÁRIO M. FREIRE, Instituto de Telecomunicações and University of Beira Interior
MANUELA PEREIRA, Instituto de Telecomunicações and University of Beira Interior
PAULO P. MONTEIRO, Nokia Siemens Networks Portugal S. A. and Instituto de Telecomunicações

Due to the relevance of self-similarity analysis in several research areas, there is an increased interest in methods to generate realisations of self-similar processes, namely in the ones capable of simulating long-range dependence. This paper describes a new algorithm to approximate persistent fractional Brownian motions with a predefined Hurst parameter. The algorithm presents a computational complexity of $O(n)$ and generates sequences with $n$ ($n \in \mathbb{N}$) values with a small multiple of $\log_2(n)$ variables. Because it operates in a sequential manner, the algorithm is suitable for simulations demanding real-time operation. A network traffic simulator is presented as one of its possible applications.

Categories and Subject Descriptors: F.2.0 [Analysis of Algorithms and Problem Complexity]: Numerical Algorithms and Problems—General; G.3 [Probability and Statistics]: Correlation and regression analysis; Probabilistic algorithms (including Monte Carlo); Stochastic processes; Time series analysis; I.6.8 [Simulation and Modeling]: Types of Simulation—Continuous; Monte Carlo

General Terms: Algorithms, Performance, Theory, Verification

Additional Key Words and Phrases: Computer Based Simulation, Fractional Brownian Motion, Fractional Gaussian Noise, Generation Algorithm, Hurst Parameter, Long-Range Dependence, Persistence, Self-Similarity

ACM Reference Format:


1. INTRODUCTION

While the synthesis of $n$ ($n \in \mathbb{N}$) completely uncorrelated values with the Gaussian distribution may be achieved using $O(n)$ complexity procedures [Thomas et al. 2007], the same may not necessarily hold true when a specific autocorrelation structure (e.g. long-range dependence) is supposed to be embedded in the synthesised series. The interest in methods to generate realisations of self-similar processes with a Gaussian distribution, namely in the ones capable of simulating long-range dependence, has...
increased over the last few years. This interest was enhanced not only by the discovery of the self-similar nature of the traffic in computer networks [Norros 1992; Leland et al. 1993; 1994; Park and Willinger 2000], but also by the importance that self-similarity analysis gained in other areas of research, as e.g. biomedical sciences [Kenkel and Walker 1996] or economics [Mandelbrot et al. 1997]. For example, a self-similar process called fractional Brownian motion (fBm) was proposed by Norros [1992] as a possible model for network traffic (in aggregation points), and by Mandelbrot et al. [1997] for modelling stock market related processes.

Self-similarity is embedded in many natural [Mandelbrot 1982] or artificial sequences but their simulation is rather complex, mostly because the observations of such processes are not generally independent. The dependency is usually given by the Hurst parameter, also referred to as the Hurst exponent. Values of the Hurst parameter ranging between 0.5 (exclusively) and 1 characterize long-range dependent or persistent processes, also referred to as processes with long memory. The process is said to exhibit anti-persistent behaviour if the Hurst parameter takes values between 0 and 0.5 (exclusively). The process is memoryless when the Hurst parameter is equal to 0.5 [Mandelbrot et al. 1997; Sun et al. 2008]. Persistent processes are often more difficult to simulate, because the range (and strength of) the dependencies increases with the Hurst parameter (each observation is conditioned by the entire past of the series).

This paper is focused on the means to synthesise approximate realisations of long-range dependent fBms. By construction, the sequences produced by the herein proposed generator follow a Gaussian distribution and approximate a stationary discrete-time stochastic process known as fractional Gaussian noise (fGn), which may be used to directly simulate observations with an approximate structure of an fBm. Typically, these generators accept the expected Hurst parameter value as their only input, and it is common to evaluate their precision by comparing that input value with the one calculated for the simulated series [Dieker 2004]. Herein, such comparison is made by resorting to several Hurst parameter estimators, which explore different statistical properties of the sequences, so as to assess their compliance with the scaling laws of long-range dependent processes. The computational requirements of the generator are also subject to analysis, not only because they were on the basis to the development of the algorithm, but also because they constitute one of the most decisive factors for the adoption of an algorithm as a simulation primitive.

The remaining part of this document is structured as follows. Section 2 mentions briefly the existing algorithms used for generation of long-range dependent sequences, more specifically for approximation of fBm. It discusses the most interesting features of this type of generators and it includes also the formal definition of self-similarity, Hurst parameter, normalised fBm and fGn. Section 3 elaborates on the fundamental mathematical concepts on which the proposed algorithm is based. A study to the accuracy and computational requirements of the algorithm is included in Section 4. Section 5 describes a possible application of the algorithm as a network traffic simulator. The most important conclusions are presented in Section 6.

2. BASIC CONCEPTS AND BRIEF DISCUSSION ON METHODS FOR GENERATION OF FRACTIONAL BROWNIAN MOTION

This section contains mathematical definitions of the concept of self-similarity, Hurst parameter and fBm, followed by a brief discussion of some of the most interesting properties of the methods used to generate long-range dependent processes, namely fBm.
2.1. Basic Concepts

2.1.1. Self-Similarity and Hurst parameter. Let \( \{X(t)\}_{t \in \mathbb{R}_{\geq 0}} \) be a continuous-time stochastic process defined for \( t \in \mathbb{R}_{\geq 0} \), and \( H \) in (1) be the Hurst parameter. The process is said to be self-similar, with Hurst parameter \( H \), if for any real number \( a > 0 \), the distribution of \( \{X(at)\}_{t \in \mathbb{R}_{\geq 0}} \) is equal to the distribution of \( \{a^H X(t)\}_{t \in \mathbb{R}_{\geq 0}} \) (the symbol \( \overset{d}{=} \), in the following expression, denotes equality in distribution):

\[
X(t) \overset{d}{=} a^{-H} X(at), \forall a > 0. \tag{1}
\]

2.1.2. First Order Differences Process. The previous definition applies to continuous-time stochastic processes but, most of the times, it is sufficient and more useful to define self-similarity for the class of discrete-time stochastic processes with stationary increments. Consider the first order differences process \( \{Y(t)\}_{t \in \mathbb{N}} \), given by (2), and the respective aggregated series \( \{Y^{(m)}(i)\}_{i \in \mathbb{N}} \) defined by (3) (note the positive integer number \( m \), that defines a particular aggregation scale or block each time it is instantiated):

\[
Y(t) = X(t+1) - X(t); \tag{2}
\]

\[
Y^{(m)}(i) = \frac{Y(mi) + \ldots + Y((m+1)i - 1)}{m}, m \in \mathbb{N}. \tag{3}
\]

A discrete-time stochastic process \( \{X(t)\}_{t \in \mathbb{N}} \) is said to be self-similar, with Hurst parameter \( H \), if the (finite dimensional) distribution of \( \{Y(t)\}_{t \in \mathbb{N}} \) is equal to the one of \( \{m^{1-H} Y^{(m)}(i)\}_{i \in \mathbb{N}} \), for any \( m \in \mathbb{N} \), i.e.

\[
Y(t) \overset{d}{=} m^{1-H} Y^{(m)}(i), \forall m \in \mathbb{N}. \tag{4}
\]

2.1.3. Normalized Fractional Brownian Motion. The most general definition of fBm is due to Mandelbrot and Van Ness [1968] and is written in terms of a stochastic integral with respect to Brownian motion. However, for the purposes of the subsequent explanation, it is sufficient to define the simplified form of the referred stochastic process termed normalized fBm.

A normalized fBm is a stochastic process \( \{B_H(t)\}_{t \in \mathbb{R}_{\geq 0}} \) that verifies conditions (5) to (9) [Norros 1994], where \( \mathbb{E}(B_H(t)) \) and \( \mathbb{V}(B_H(t)) \) denote respectively the expected value and the variance of the variable to which they are applied to (which, in this particular case, is the normalized fBm):

\[
B_H(t) \text{ is a Gaussian variable for } t \geq 0; \tag{5}
\]

\[
B_H(t) \text{ has stationary increments; } \tag{6}
\]

\[
B_H(0) = 0; \tag{7}
\]

\[
\mathbb{E}(B_H(t)) = 0, \text{ for } t \geq 0; \tag{8}
\]

\[
\mathbb{V}(B_H(t)) = t^{2H}, \text{ for } t \geq 0. \tag{9}
\]

Notice that the previous set of expressions define \( \{B_H(t)\}_{t \in \mathbb{R}_{\geq 0}} \) for the continuous-time case, but their particularization to the discrete-time class of processes is straightforward. The first order differences process of an fBm is the previously mentioned fGn (see Section 1) and is denoted herein by \( \{G_H(t)\}_{t \in \mathbb{N}} \), where \( H \) is the Hurst parameter. By construction, \( \{G_H(t)\}_{t \in \mathbb{N}} \) is a Gaussian process with expected value equal to 0 and variance equal to 1 (recall that according to (2), \( G_H(t) = B_H(t+1) - B_H(t), t \in \mathbb{N} \)).

2.2. Brief Discussion on Methods for Generation of Fractional Brownian Motion

Due to the importance that the simulation of self-similarity has gained along the years, several algorithms for the generation of sequences with the aforementioned property
have been proposed. In his master thesis, Dieker [2004] divides these algorithms in exact and approximate methods, depending on whether their outputs are theoretically perfect realisations of fBms, or approximations that depend on some convergence criterion.

The exact methods, as the ones known as the Hosking, the Davies and Hart and the Cholesky methods, elaborate directly on the autocorrelation matrix of self-similar processes. The difference between these three algorithms lies mostly in the way of performing the matrix decomposition. Their computational complexity is at least of $O(n \log(n))$, since each time a value is produced, the number of relations that have to be taken into account increases ($n \in \mathbb{N}$ denotes the number of values to be generated).

On the other hand, the approximate methods draw on simplifications to the initial assumptions of self-similarity (e.g. the Conditionalized Random Midpoint Displacement [Norros et al. 1999], the Stochastic Representation Method or the one proposed by Rambaldi and Pinazza [1994]), on the approximation of coefficients representing the self-similar processes in the spectral or wavelets domain (e.g. the Paxson Method [Paxson 1997] or the generator based on Wavelets [Jeong et al. 1999]), or on theorems that guarantee that a suitable transformation of an infinite number of processes converges to a process with the desired properties (e.g. the Method of Aggregation of Processes [Taqqu et al. 1997] or the one proposed by Enriquez [2004]). Thus, the approximate methods present often an inversely proportional relation between their performance and the quality of the generated sequences, but some of them have computational complexities of $O(n)$. While the exact methods have linear memory requirements, the approximate methods demand no more than the space for a fixed number of values, or for a small multiple of $\log(n)$.

The possibility to generate values in a sequential manner, or the capability to operate with no a priori knowledge of the length of the series to be synthesised, comprise particularly interesting features for this type of procedures, mostly because simulations with unknown running time and demanding real-time operation are becoming more common. The algorithm proposed in this paper belongs to the class of approximate methods and it was developed with those specific concerns in mind. It has a computational complexity of $O(n)$ and its memory requirements are directly proportional to $\log_2(n)$. Some of its basic concepts are similar to the ones used in the method of Enriquez [2004]. For example, both proposals start from the specification of the means to construct correlated random walks using persistence probabilities but, as it is going to be opportunely mentioned afterwards (see Section 3), the definition of the persistence probabilities is different in the two algorithms. On the other hand, the algorithm proposed herein was designed with the intention of assuring exact self-similar properties for non-overlapping blocks with size $m = 2^k$, $k \in \mathbb{N}$ only, while the method of Enriquez [2004] assures that the properties of the sequences it produces converge to the ones of an fBm, for any $m \in \mathbb{N}$. However, the precision of the latter counterbalances its performance.

3. FRACTIONAL BROWNIAN MOTION SEQUENTIAL GENERATION ALGORITHM

This section is divided in three main parts. The first part describes an algorithm to synthesise steps of correlated random walks with persistence. The second part discusses how the reasoning developed for the random walks can be generalised in order to enable the efficient generation of observations of a Gaussian process. A brief discussion about the autocorrelation structure of the sequences produced with the proposed means may be found in the final part of this section.
3.1. An Algorithm to Generate Persistent Random Walks

The algorithm described in this subsection elaborates on the means to generate the first order differences process \( \{Y(t)\}_{t \in \mathbb{N}} \) of a correlated random walk \( \{R(t)\}_{t \in \mathbb{N}} \) with constant size steps and a specific dependencies scheme. Without loss of generality and for the sake of the explanation, it is assumed that \( \{Y(t)\}_{t \in \mathbb{N}} \) can only take the integer values 1 or −1, as formalised by

\[
Y(0) = -1 \text{ or } Y(0) = 1 \text{ with probability } 0.5 \text{ and } \forall t \in \mathbb{N}, Y(t) = -1 \text{ or } Y(t) = 1. \tag{10}\tag{11}
\]

Notice that (10) and (11) specify the starting point of \( \{Y(t)\}_{t \in \mathbb{N}} \) and its space of occurrences (respectively), but they do not define its autocorrelation structure. The complete definition of this process results from the combination of (10) and (11) with (13), (14) and (24), which will be discussed afterwards. At a given moment \( t \in \mathbb{N} \), the position of the walk can be constructed by taking the sum of all the values of the first order differences process until that moment:

\[
R(t) = \sum_{i=0}^{t} Y(i). \tag{12}
\]

The importance of the aforementioned processes within the scope of this paper lies on the central limit theorem and on the fact that the relations for a process like \( \{Y(t)\}_{t \in \mathbb{N}} \) are conceptually easier to handle. The sum and normalization of several independent realisations of \( \{Y(t)\}_{t \in \mathbb{N}} \) approximates a Gaussian process which inherits the long-range dependence structure discussed along this section.

In here, the correlations between two given steps of the walk can be written in terms of what we have called the persistence probabilities, in resemblance to what was done by Enriquez [2004] (though their definition is different). It is assumed that a persistence probability exists for each aggregation scale which is a power of 2, and that those probabilities influence the progression of the walk in the sense given by (13). We denote them by \( p_s \), where \( s = 2^{k+1}, k \in \mathbb{N} \). Their definition is the conjunction of the following two conditions, where (14) assures that a given future observation of \( \{Y(t)\}_{t \in \mathbb{N}} \) can only be positively affected by an observation of the past, while avoiding the trivial situation where the sequence is constant (i.e. \( p_s = 1 \)):

\[
p_s = P \left( Y \left( t + \frac{s}{2} \right) = Y(t) \right), \quad t \in s\mathbb{N}, \text{ where } s = 2^{k+1}, k \in \mathbb{N} \text{ and } 0.5 \leq p_s < 1. \tag{13}\tag{14}
\]

Expressions (10), (11) and (13) imply that \( Y(t) \overset{d}{=} -Y(t) \), i.e., the probabilistic description of \( \{Y(t)\}_{t \in \mathbb{N}} \) is the same of \( \{-Y(t)\}_{t \in \mathbb{N}} \). Therefore, \( \{Y(t)\}_{t \in \mathbb{N}} \) follows the definition of symmetric stochastic processes given in [Schmitz 2003] with expected value equal to 0 (i.e. the unconditional marginal probabilities \( P(Y(t) = 1) \) and \( P(Y(t) = -1) \) are equal to 1/2). The fact that \( \mathbb{E}(Y(t)) = 0 \) is used below.

Let us add the following before proceeding further. The algorithm proposed herein guarantees exact self-similarity properties for non-overlapping blocks whose size is a power of 2. For other block sizes, self-similarity cannot be assured theoretically, but simulation results suggest that the property is preserved, though not as strictly as for the aforementioned case. Moreover, assume that the maximum number of aggregation scales the procedure is concerned with is represented by \( N_p \), sometimes referred to as the number of precision scales, and that the expression scales of type \( x \times y^{N_p} \) stands for all of the aggregation scales whose sizes are given by \( m = x \times y^{k} \), where \( x \) and \( y \)
are positive integer numbers larger than 1, and $k = 0, 1, \ldots, N_p - 1$ (for the definition of aggregation scale, refer to Section 2.1.2).

The reasoning below is based on the second definition of self-similarity (equation (4)). From that definition, (15) can be derived:

$$V(Y(t)) = m^{2-2H}V(Y^{(m)}(i)).$$ (15)

Since (11), (13) and (14) hold, it is easy to conclude that $V(Y(t)) = 1$ and that, therefore, $m^{2H-2} = V(Y^{(m)}(i)).$ Under the circumstances, the first condition $\{Y^{(t)}\}_{t \in N}$ (or its aggregation) must comply with is $2^{2H-2} = V(Y^{(2)}(i)).$ Since $E(Y(t)) = 0,$ the subsequent equations are valid:

$$2^{2H-2} = V(Y^{(2)}(i)) = E(Y^{(2)}(i))^2.$$ (16)

Given the definition of $\{Y^{(2)}(i)\}_{i \in N},$ condition (11) and equation (16), the reasoning depicted by the following expression is easy to follow:

$$2^{2H-2} = E(Y^{(2)}(i))^2 = p_2 \times 1 + (1 - p_2) \times 0 \Leftrightarrow p_2 = 2^{2H-2},$$ (17)

where $p_2$ is the probability of $Y(t + 1)$ to be equal to its predecessor, $Y(t),$ when $t$ is a multiple of 2, i.e.

$$p_2 = P(Y(t) = Y(t + 1)), t \in 2N.$$ (18)

Probability $p_2$ assures that the empirical variance of the aggregated series for $m = 2$ converges to $2^{2H-2},$ as the number of generated points tends to infinity (law of large numbers). In other words, the convergence of the empirical Hurst parameter to the predefined value is guaranteed by the way $p_2$ is formulated, at least for the aggregation scale of 2.

If the scale of 4 is now taken into consideration we can notice that, at each four consecutive observations of $\{Y^{(t)}\}_{t \in N},$ there is only one for which no relation was presented until here (e.g. $Y(1)$ depends on $Y(0),$ $Y(3)$ depends on $Y(2),$ but the dependence of $Y(2)$ is still not defined). In here, we assume that such observation is dependent of the first of the two preceding ones (i.e. for $t \in 4N,$ the value of $Y(t + 2)$ depends of the value of $Y(t),$) as depicted in Figure 1 and formalized by

$$p_4 = P(Y(t) = Y(t + 2)), t \in 4N.$$ (19)

To get the persistence probability $p_4,$ one can then follow the reasoning depicted by

$$4^{2H-2} = V(Y^{(4)}(i'))^{i=2^{i'}}V\left(\frac{2}{4}Y^{(2)}(i) + \frac{2}{4}Y^{(2)}(i + 1)\right)$$
$$= \frac{1}{4}V(Y^{(2)}(i) + Y^{(2)}(i + 1)) = \frac{1}{2}V(Y^{(2)}(i)) + \frac{1}{2}C(Y^{(2)}(i), Y^{(2)}(i + 1))$$
$$= \frac{1}{2}2^{2H-2} + \frac{1}{2}C(Y^{(2)}(i), Y^{(2)}(i + 1)).$$ (20)

Notice that $C(Y^{(2)}(i), Y^{(2)}(i + 1)),$ in the previous expression, denotes the covariance between $Y^{(2)}(i)$ and $Y^{(2)}(i + 1)$ for $i \in 2N$ which, in this case, can be expressed in terms
of \( p_4 \) as follows:

\[
C(Y^{(2)}(i), Y^{(2)}(i + 1)) = E(Y^{(2)}(i) \times Y^{(2)}(i + 1)) = \frac{1}{2} 4^{2H-2} + \frac{1}{2} (2p_4 - 1) \left( E(Y^{(2)}_+(i)) \right)^2, \tag{21}
\]

where \( \{Y^{(s)}(i)\}_{i \in \mathbb{N}} \) denotes the series obtained from \( \{Y^{(s)}(i)\}_{i \in \mathbb{N}} \) via the multiplication of the value of each aggregated block by the first observation of the block, i.e. \( Y^{(s)}_+(i) = Y(i) \times Y^{(s)}(i) \).

Equation (20) can now be written as (22) or, equivalently, as (23):

\[
4^{2H-2} = \frac{1}{2} 2^{2H-2} + \frac{1}{2} (2p_4 - 1) \left( E(Y^{(2)}_+(i)) \right)^2, \tag{22}
\]

\[
\Leftrightarrow p_4 = \frac{2 \times 4^{2H-2} - 2^{H-2}}{2(2^{2H-2})^2} + \frac{1}{2}. \tag{23}
\]

If an analogous reasoning is applied to all the other scales of type \( 2 \times 2^{N_p} \), a closed form expression (formula (24)) can be derived for all persistence probabilities defined in (13) (consider observing Figure 1 for a conceptual representation of the relations between observations of the walk at different time steps):

\[
p_s = \frac{2 \times s^{2H-2} - (s/2)^{2H-2}}{2 \left( E(Y^{(s/2)}_+(i)) \right)^2} + \frac{1}{2}. \tag{24}
\]

The value of \( E(Y^{(2)}_+(i)) \), used to assess \( p_4 \) in (23), is given by \( E(Y^{(s)}_+(i)) = 2^{2H-2} \) and the successive values of \( E(Y^{(s/2)}_+(i)) \) can be obtained recursively using expression (26):

\[
E(Y^{(s/2)}_+(i)) = 2^{2H-2} \text{ for } s = 4, \tag{25}
\]

\[
E(Y^{(s/2)}_+(i')) = \frac{1}{p_{s/2} E(Y^{(s/4)}_+(i))}, \text{ for } s > 4 \text{ (i.e. } s = 2^{k+1}, k = 2, 3, ...). \tag{26}
\]
The last result can be justified in the following manner:

\[
E \left( Y_+^{(s/2)}(i') \right) = 2 \sum_{i=0}^{N-1} \frac{E \left( Y_+^{(s/4)}(i) \right) + E \left( Y_+^{(s/4)}(i+1) \right)}{s/2} = \frac{1}{2} E \left( Y_+^{(s/4)}(i) \right) + \frac{1}{2} E \left( Y_+^{(s/4)}(i+1) \right) = \frac{1}{2} E \left( Y_+^{(s/4)}(i) \right) + \frac{1}{2} (2p_{s/2} - 1) E \left( Y_+^{(s/4)}(i) \right) = p_{s/2} E \left( Y_+^{(s/4)}(i) \right).
\]

From a naive perspective, one can simply say that \( p_s \) is the probability of a non-overlapping block (with size \( s \)) to maintain the sign of its predecessor. The main differences between the approach described in this document and the one taken by Enriquez [2004] lie on the scheme and length of the dependencies between steps of the walks. The way the persistence probabilities are defined herein (expression (24)) makes it clear that the dependence from distant steps increases with the aggregation scale considered, while in [Enriquez 2004], each step depends directly on the previous one. To increase the length of the dependencies, the generator of Enriquez [2004] relies on the aggregation of the resulting Gaussian process in the time domain. In here, a long-range dependencies structure is immediately provided, though it only applies for a specific type of scales.

The generator proposed herein is not suitable for the generation of correlated random walks exhibiting anti-persistent behaviour because, depending on the aggregation scale and on the Hurst parameter, it is not possible to express the persistence probabilities as numbers in the interval \([0, 1]\).

3.2. Approximating Fractional Brownian Motion

Now that the persistent random walks and the persistence probabilities have been defined, a procedure for approximating an fBm can be easily described. Assume that for each \( j \in \mathbb{N} \), \( \{ Y_{H,j}(t) \}_{t \in \mathbb{N}} \) is the first order differences process of a correlated random walk as defined in the previous subsection for a fixed Hurst parameter \( H \). Based on the central limit theorem and given (28), one can write (29), where \( \text{d-lim} \) denotes limit in distribution and \( G(\mu, \sigma^2) \) represents a normally distributed variable with mean \( \mu \) and variance \( \sigma^2 \):

\[
G^N_H(t) = \frac{\sum_{j=0}^{N} (Y_{H,j}(t) - 0)}{\sqrt{N}};
\]

\[
G^*_H(t) = \text{d-lim}_{N \to \infty} G^N_H(t) = G(0, 1).
\]

It is a simple exercise to prove that the normalized process inherits the long-range dependencies structure of the individual correlated walks (under the condition that the dependencies structure is equal for all processes \( \{ Y_{H,j}(t) \}_{t \in \mathbb{N}} \)). Since the central limit theorem assures the convergence of \( \{ G^N_H(t) \}_{t \in \mathbb{N}} \) to a Gaussian process for large values of \( N \), \( \{ G^*_H(t) \}_{t \in \mathbb{N}} \) approximates an fGn and, therefore, \( \{ B^*_H(t) \}_{t \in \mathbb{N}} \) (see (30)) approximates an fBm with pre-defined Hurst parameter \( H \):

\[
B^*_H(t) = \sum_{i=0}^{t} G^*_H(i).
\]

Condition (28) immediately provides a way to approximate an fBm. However, the convergence to a Gaussian variable depends on the number of different and indepen-
dent instances of random walks: the larger $N$ is, the more accurate is the approximation. This fact raises a problem in terms of the computational complexity of the procedure, since the simulation of a large number of random walks would ultimately render it useless.

To overcome this issue, it suffices to notice that, for any finite number $N$ and for each $t \in \mathbb{N}$, $G^N_H(t)$ is the normalized sum of $N_+(t)$ $1$s and $N_-(t)$ $-1$s, where $N_+(t) + N_-(t) = N$. For $t \in s\mathbb{N}$ and due to the properties of the correlated random walks, the number of processes $\{Y_{H,j}(t)\}_{j \in \mathbb{N}}$ with equal observations for $t$ and $t + s/2$, is governed by a binomial distribution with success probability $p_s$. Because of that, the conditional variances of $N_+(t+s/2)$ and $N_-(t+s/2)$, given $N_+(t)$ and $N_-(t)$, are both equal to $p_s(1-p_s)N$, and their expected value can be defined as in (31), where $\mathbb{E}(N_s(t+s/2) | N_+(t))$ denotes the conditional expected value of $N_s(t+s/2)$ given $N_+(t)$:

$$\mathbb{E}(N_+(t+s/2) | N_+(t)) = (2p_s - 1)N_+(t) + (1-p_s)N \quad \text{and}$$

$$\mathbb{E}(N_-(t+s/2) | N_-(t)) = (2p_s - 1)N_-(t) + (1-p_s)N,$$  \text{for} \ t \in s\mathbb{N}, \ s = 2^{k+1}, \ k \in \mathbb{N}. \quad (31)

The construction of the dependencies structure can now be generalized for Gaussian processes. Recall that $G^N_H(t) = (2N_+(t) - N)/\sqrt{N}$ and that the binomial distribution converges to a Gaussian distribution as $N \rightarrow \infty$. Given $G^H_H(t)$ and for $t \in s\mathbb{N}$, the value of $G^N_H(t+s/2)$ is an occurrence of a Gaussian variable with mean equal to $(2p_s - 1)G^H_H(t)$ and variance equal to $4p_s(1-p_s)$:

$$G^N_H(t+s/2) = G((2p_s - 1)G^H_H(t), 4p_s(1-p_s)),$$ \text{for} \ t \in s\mathbb{N}, \ s = 2^{k+1}, \ k \in \mathbb{N}. \quad (32)

Figure 2 represents an effort to describe the previous sentences and equations graphically, substantiating the statements with a numerical example. It also illustrates how the dependencies scheme devised for the correlated random walks can be embedded in a sequence of values with a Gaussian distribution, without having to simulate a large number of independent processes $\{Y_{H,j}(t)\}_{j \in \mathbb{N}}$.

The only thing left to be discussed here is how to get the range of the dependency (and the respective persistence probability) of the value that the algorithm is about to generate via (32). To find that relation (i.e. to find the value of $s$ that has to be applied for each $t$), one just needs to find the first positive integer $k$ that satisfies $t \mod 2^{k+1} = 2^k$ in the sub-algorithm (33). The persistence probability is then given by $p_s^{(s)}$ and the extent of the dependency is given by $2^{k}/2$ (i.e. the actual step $Y(t)$ depends on step $Y(t-2^{k}/2)$). The algorithm herein entitled as fractional Brownian motion Sequential Generation Algorithm (fBm-SGA) can be, at this point, completely defined as follows:

**fractional Brownian motion Sequential Generation Algorithm**

If $t = 0$, $G^H_H(t) = G(0,1)$;

if $t > 0$, find the smallest number $k \in \{0, 1, \ldots, N_p - 1\}$ that satisfies $t \mod 2^{k+1} = 2^k$ and calculate $s = 2^{k+1}$, \quad (33)

if $s = 2$, $p_s = 2^{2H-2}$;

if $s > 2$, $p_s = \frac{2 \times s^{2H-2} - (s/2)^{2H-2}}{2 \left( \mathbb{E}(Y_{+}^{(s/2)}(i)) \right)^2} + \frac{1}{2}$,

obtain $\mathbb{E}(Y_{+}^{(s/2)}(i))$ by using (25) and the recursive formula (26):

$$G^N_H(t) = G((2p_s - 1)G^H_H(t-s/2), 4p_s(1-p_s)).$$

ACM Transactions on Modeling and Computer Simulation, Vol. x, No. x, Article 1, Publication date: September 2011.
Recall that \( N_p \) is the maximum number of aggregation blocks (whose size is a power of 2) handled by the algorithm. This means that, in practical applications, this number should either be set to a sufficiently large value (refer to the end of Section 4.4 for more on this subject) or an additional clause should be added to the first line of the algorithm, so as to reinitialise it every time \( t \) is a multiple of \( 2N_p \) (i.e. every time \( t \mod 2N_p = 0 \)).

### 3.3. Approximate Autocorrelation Structure of the fBm-SGA

The autocorrelation structure is not easily drawn from the explanation of the algorithm. By observing the relations depicted in Figure 1, one may conclude that deriving a close form expression for the autocorrelation of the values generated by the fBm-SGA constitutes a defying task, mostly because the extent of the dependencies changes according to the index of the generated values. For example, the relations between consecutive values change from direct and simple, to indirect and potentially complex dependencies, as the index shifts from even to odd numbers.

Nonetheless, to provide this description with an idea of the autocorrelation structure created by the proposed generator, 100 sequences with \( 10^6 \) points were computationally synthesized for each one of the Hurst parameter values indicated in Figure 3, and the autocorrelation of those sequences was respectively calculated for lags \( l = 2^k \), \( k \in \mathbb{N} \).

The several autocorrelation values were organized in the 3D chart included on the left side of Figure 3, while the exact values are depicted on the right. (The exact values of the autocorrelation function \( \gamma_H(l) \) of an fGn, with Hurst parameter \( H \), may be calculated using \( \gamma_H(l) = \frac{1}{2}(|l - 1|^{2H} - 2|l|^{2H} + |l + 1|^{2H}) \) [Dieker 2004].)

The similar shape of the surfaces shows that the desired property is indeed printed on the generated sequences, but the noticeable lack of correlation for smaller lags unfolds the effects of enforcing the relations between non-intersecting blocks (aggregation scales) only. The design of the presented procedure was initially inspired by the way the estimators of the Hurst parameter are typically implemented and, as so, deviations of the autocorrelation values were expected. While the dependencies between the pairs with indexes \( 2^k+1N \) and \( 2^k+1N + 2^k \), \( k \in \mathbb{N} \) are perfect in terms of self-similarity, the same does not hold for the remaining pairs, in which case the autocorrelations are always smaller than the expected ones. For instance, for the lag of 1, the weakest autocorrelation case happens between the values with index \( 2^k+1 \), \( k \in \mathbb{N} \) and their predecessors, whose value is estimated to be of \( \prod_{i=1}^{k+2}(2p_{2i} - 1) \) (this value tends rapidly to 0, as the number of factors in the product increases). Reinforcing the dependencies between each output of the fBm-SGA and a larger number of past values of the generated sequence, without jeopardizing the efficiency of the algorithm, is the matter of future research work.
Fig. 3. Analysis of the autocorrelation of the sequences generated with the fBm-SGA: the chart on the left side plots the values of the autocorrelation obtained using the fBm-SGA, for different lags and Hurst parameter values; while the chart on the right depicts the theoretical values for the same parameter combinations, for comparison purposes.

4. QUALITY ASSESSMENT, COMPUTATIONAL REQUIREMENTS AND OTHER PRACTICAL ASPECTS OF THE FBM-SGA

In this section, some examples of sequences with the long-range dependencies scheme created by the fBm-SGA are graphically represented and briefly analysed. After that, an empirical study about the accuracy of the generator is included. The third and fourth subsections are dedicated to the discussion of the computational requirements of the algorithm.

4.1. Graphical Examples of the Outputs of the fBm-SGA

The three charts included in Figure 4 depict three approximate fBms with different Hurst parameter values. The graphical representation is obtained by plotting 1000 points of the sequences generated with the fBm-SGA against their indexes. From a careful observation, it is possible to differentiate the lines by their roughness, which reflects a typical behaviour for self-similar processes. The weaker the dependence on the past is, the more irregular the line in the graphical representation should look like. One can also notice that the spatial span of the charts (the y-axis range) increases with the value of the Hurst parameter, in accordance with (9).

The fBm-SGA creates the approximate fBms in Figure 4 by generating the individual (discrete) steps of the motion. In the charts of Figure 5, the steps of the previously depicted sequences are plotted against the order by which they were generated. Long-range dependence is reflected in these charts as well. For instance, the frequency by which the values change from positive to negative, and vice versa, decreases for higher values of the Hurst parameter. This behaviour can be explained by drawing on the perspective provided by the persistence probabilities, in which the probability for a step of the motion to be equal to the previous ones increases with the Hurst parameter. Consequently, the moving average of the first order differences process is affected (biased) by the value of Hurst parameter for a relatively small number of steps.

4.2. Quality Assessment via Hurst Parameter Estimation

The accuracy of the fBm-SGA is evaluated here by analysing the differences between the expected Hurst parameter, which is an input of the algorithm, and the value estimated from the generated sequences. With the purpose of estimating the Hurst pa-
Fig. 4. Graphical representation of approximate fBms generated with the fBm-SGA and exhibiting persistent behaviour with pre-determined Hurst parameter: a) $H = 0.6$, b) $H = 0.7$ and c) $H = 0.8$.

Fig. 5. Graphical representation of approximate fGns generated with the fBm-SGA, with pre-determined Hurst parameter: a) $H = 0.6$, b) $H = 0.7$ and c) $H = 0.8$.

rameter of the latter, the fBm-SGA and the three following estimation algorithms were implemented in Java programming language: the Variance Time (VT) and the Rescaled Statistics (RS) methods, as described in [Dieker 2004]; and the Detrended Fluctuation Analysis (DFA) method, as described e.g. in [Peng et al. 1995; Goldberger et al. 2003]. All these methods elaborate on the analysis of statistical properties of non-intersecting aggregation blocks, which is a typical mode of operation for these kind of estimators.
Though these methods are not regarded to be as robust as e.g. the estimator proposed by [Veitch and Abry 1999], they are suitable for the analysis of computationally generated stationary sequences.

Two different kinds of tests were conducted for each one of the implemented estimators. Initially, the accuracy was analysed for the scales of type $2 \times 2^{N_p}$, for which the algorithm is supposed to be exact. Next, to prove that the synthesized sequences maintain the self-similarity properties, at least in the approximated sense for other types of scales, the accuracy was also analysed for scales of type $3 \times 2^{N_p}$. A total of 100 simulations were performed for each one of these two possibilities, and the results were statistically compiled in the form of a pair (sample average and variance) and included below. All the generated sequences had $10^6$ points and the number of precision scales was set to 18 for all simulations.

The accuracy of the fBm-SGA was analysed until the hundredth of the value of the Hurst parameter, because it soon became noticeable that no valuable conclusions could be taken beyond that degree of precision. One of the aspects that can explain this fact is that the convergence of the estimates to the predefined value depends on the length of the sequence. Hurst parameter values ranging from 0.5 to 0.99 (with increments equal to 0.01) were tested through simulation. Table I and Table II contain a subset of the numerical results obtained for the scales of type $2 \times 2^{N_p}$ (Table I) and for the scales of type $3 \times 2^{N_p}$ (Table II) for the three above-mentioned estimators. All the values were averaged and rounded off to the hundredth. The variances were rounded and written in scientific notation (with two decimal places). The graphical representations included in Figure 6 compress a larger set of results. In the two charts, the average value of the Hurst parameter estimates is plotted against the expected one, for each estimation method. The line with equation $y = x$ was also plotted in the charts as a comparison reference.

For the three estimation methods indicated above, the estimation of the Hurst parameter involves the calculation of specific statistics for several aggregations of the empirical series. The logarithms of these statistics are usually plotted against the logarithms of the aggregation scales in the so-called log-log plots. The estimate is then obtained from the slope of the line that best fits the coordinates of the plots (e.g., for DFA, the slope $\beta$ of the line is the estimate of Hurst parameter, while for VT, the estimate is $1 + \beta/2$). From careful analysis of many RS and DFA log-log plots, we have concluded that the implementations of the respective methods exhibit a bias in the smallest aggregation scales taken into consideration. It was therefore decided not to include the aggregation scales of 2, 3, 4 and 6 in the linear regression of the RS method, and the ones of 2 and 3 in the fitting procedure of the DFA. Moreover, the rescaled range statistic used in the RS method seems to be more susceptible to the number of samples available during the analysis than the other two methods. For that reason, RS was set to operate with the 15 intermediate points of the log-log plots only.

The presented results show that the fBm-SGA is accurate in terms of expected Hurst parameter. On average, the estimated values do not differ from the expected ones more than 0.02 for any of the methods used. The DFA produced the most favourable assessment of the precision of the algorithm. It returned values that are equal to the expected one (on average) 94% of the times for the scales of type $2 \times 2^{N_p}$, and 82% of the times for the scales of type $3 \times 2^{N_p}$. Because of that, the lines with equation $y = x$ and the estimated values for the aforementioned method are almost indistinguishable in both charts. The same happens with the VT method for scales of type $2 \times 2^{N_p}$. The VT estimator is based on the same condition that motivated the development of the proposed algorithm (condition (15)). Therefore, this was the method for which the best performance was expected, and the results included in the second column of Table I corroborate this belief, supporting the validity of the reasoning behind the formulation.
Table I. Target and estimated Hurst parameter values for scales of type $2 \times 2^{N_p}$.

<table>
<thead>
<tr>
<th>Hurst</th>
<th>Variance</th>
<th>Detrended</th>
<th>Rescaled</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time</td>
<td>Fluctuation Analysis</td>
<td>Statistics</td>
</tr>
<tr>
<td>0.50</td>
<td>0.50(1.77E-05)</td>
<td>0.50(1.86E-04)</td>
<td>0.50(1.02E-04)</td>
</tr>
<tr>
<td>0.55</td>
<td>0.55(1.96E-05)</td>
<td>0.55(2.18E-04)</td>
<td>0.55(9.44E-05)</td>
</tr>
<tr>
<td>0.60</td>
<td>0.60(1.89E-05)</td>
<td>0.60(3.16E-04)</td>
<td>0.60(1.25E-04)</td>
</tr>
<tr>
<td>0.65</td>
<td>0.65(1.58E-05)</td>
<td>0.65(3.12E-04)</td>
<td>0.64(9.44E-05)</td>
</tr>
<tr>
<td>0.70</td>
<td>0.70(1.18E-05)</td>
<td>0.69(3.21E-04)</td>
<td>0.68(1.16E-04)</td>
</tr>
<tr>
<td>0.75</td>
<td>0.75(1.55E-05)</td>
<td>0.75(3.43E-04)</td>
<td>0.73(1.37E-04)</td>
</tr>
<tr>
<td>0.80</td>
<td>0.80(3.12E-05)</td>
<td>0.80(5.96E-04)</td>
<td>0.77(1.68E-04)</td>
</tr>
<tr>
<td>0.85</td>
<td>0.85(3.83E-05)</td>
<td>0.85(6.31E-04)</td>
<td>0.82(3.82E-04)</td>
</tr>
<tr>
<td>0.90</td>
<td>0.90(1.56E-04)</td>
<td>0.90(1.18E-03)</td>
<td>0.87(5.54E-04)</td>
</tr>
<tr>
<td>0.95</td>
<td>0.94(3.99E-04)</td>
<td>0.95(2.32E-03)</td>
<td>0.93(1.12E-03)</td>
</tr>
</tbody>
</table>

Each cell contains the average and the variance (in brackets) of the results of 100 simulations.

Table II. Target and estimated Hurst parameter values for scales of type $3 \times 2^{N_p}$.

<table>
<thead>
<tr>
<th>Hurst</th>
<th>Variance</th>
<th>Detrended</th>
<th>Rescaled</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time</td>
<td>Fluctuation Analysis</td>
<td>Statistics</td>
</tr>
<tr>
<td>0.50</td>
<td>0.50(2.24E-05)</td>
<td>0.50(1.24E-04)</td>
<td>0.51(1.29E-04)</td>
</tr>
<tr>
<td>0.55</td>
<td>0.55(2.97E-05)</td>
<td>0.54(1.81E-04)</td>
<td>0.55(1.31E-04)</td>
</tr>
<tr>
<td>0.60</td>
<td>0.60(3.78E-05)</td>
<td>0.60(2.29E-04)</td>
<td>0.60(1.75E-04)</td>
</tr>
<tr>
<td>0.65</td>
<td>0.64(2.03E-05)</td>
<td>0.64(1.71E-04)</td>
<td>0.64(1.64E-04)</td>
</tr>
<tr>
<td>0.70</td>
<td>0.69(2.05E-05)</td>
<td>0.69(2.02E-04)</td>
<td>0.69(2.17E-04)</td>
</tr>
<tr>
<td>0.75</td>
<td>0.74(1.86E-05)</td>
<td>0.74(3.23E-04)</td>
<td>0.73(2.42E-04)</td>
</tr>
<tr>
<td>0.80</td>
<td>0.79(3.81E-05)</td>
<td>0.79(4.96E-04)</td>
<td>0.78(3.67E-04)</td>
</tr>
<tr>
<td>0.85</td>
<td>0.84(6.70E-05)</td>
<td>0.85(5.78E-04)</td>
<td>0.84(5.77E-04)</td>
</tr>
<tr>
<td>0.90</td>
<td>0.88(2.17E-04)</td>
<td>0.90(9.02E-04)</td>
<td>0.86(9.43E-04)</td>
</tr>
<tr>
<td>0.95</td>
<td>0.93(5.64E-04)</td>
<td>0.95(1.80E-03)</td>
<td>0.91(1.18E-03)</td>
</tr>
</tbody>
</table>

Each cell contains the average and the variance (in brackets) of the results of 100 simulations.

of the persistence probabilities. On the other hand, the RS method returned the worst results for both types of scales. Nevertheless, despite showing an error larger than 0.03 for Hurst parameter values larger than 0.85, it can be seen that the estimates follow the trend of the values obtained by the other estimators, and that for smaller values of the Hurst parameter, the same error is again not larger than 0.001, on average.

The variance of the estimated values increases with the value of the Hurst parameter, for all methods, suggesting that the statistical properties of the generated sequences converge more slowly to the ones of a long-range dependent process as the dependencies between the outputted values become stronger. The difference between the expected and the estimated self-similarity degree for scales of type $3 \times 2^{N_p}$ also evolves in the same manner. For this type of scales, all methods underestimate the value of the Hurst parameter, mostly due to the fact that the approach taken here does not make any direct effort to enforce the relations between values or blocks belonging to scales of type different from $2 \times 2^{N_p}$.

4.3. Computational Complexity

The fBm-SGA has a computational complexity of $O(n)$. This can be proven by focusing on the formal description of the algorithm (final part of Section 3.2) and by using the mathematical properties of the Gabriel’s staircase series [Weisstein 2008], namely the one concerning the limit of the partial sums.

First of all, notice that the values of the $N_p$ persistence probabilities do not need to be recomputed at each generation step. They may be calculated (using a $O(N_p)$ complexity procedure) during the initialization phase of the algorithm or when they are first
Fast Synthesis of Persistent Fractional Brownian Motion

Fig. 6. Comparison between the expected and the estimated Hurst parameter values. The results depicted in the left side were obtained by initializing the estimators with both root and scale factor parameters set to 2 (i.e. the estimators were testing scales of type \(2 \times 2^N_p\)). The chart of the right side concerns the analysis conducted for scales of type \(3 \times 2^N_p\).

needed, and stored in the memory for later usage. Moreover, according to the definition of the fBm-SGA, \(G^*_p(t)\) is always drawn from a Gaussian variable with expected value equal to \((2p_s - 1)G^*_p(t - s/2)\) and variance equal to \(4p_s(1 - p_s)\). Such operation, i.e. generation of pseudo random numbers following the Gaussian distribution has an \(O(n)\) computational complexity [Thomas et al. 2007], and the remaining operations do not require more than fetching a past value from the memory and a finite number of multiplications. Therefore, the only step that could actually embody a computational burden is the one where the extent of the dependence is assessed (represented by (33)). After careful analysis of the referred procedure, it is possible to see that 1/2 of the times, \(k\) is 1; 1/4 of the times, \(k\) is 2; 1/8 of the times, \(k\) increments until 3; and that, in the general case, \(k\) increments to the integer \(L\), \(1/2^k\) of the times. This means that the average number of operations required by such procedure is a multiple of \(C\), defined as follows:

\[
C = \sum_{k=1}^{N_p} \frac{k}{2^k}. \tag{34}
\]

As it is going to be explained afterwards, \(N_p\) does not need to be larger than the integer part of \(\log_2(n)\) (where \(n\) is the size of the sequence to be generated) which, for any practical instantiation of the algorithm, constitutes a very strict and comfortable limit. However, even if such fact was to be disregarded, one could still argue that the sum in the right side of expression (34) tends to the Gabriel’s staircase series as \(N_p\) tends to infinity, and that the partial sums of this series converge monotonously to 2 [Weisstein 2008].

To test the performance of the fBm-SGA, a Java implementation of the algorithm was run several times to produce different sized sequences repeatedly. These tests were performed in a non-dedicated, though controlled, system with the following specifications: a 2.8 GHz Pentium IV processor with 504 Mb of effective RAM, running Microsoft
Windows XP service pack 2. There were no other applications running on the computer while this simulation was being performed. As before, the presented conclusions are the result of the statistical treatment of a total of 26000 simulations, conducted for 260 different configurations.

The charts in Figure 7 were included here for two main reasons: besides emphasizing the impact that the number of precision scales has on the efficiency of the algorithm, they also disclose the linear relation between the number of points generated and the time spent to generate them. The above-mentioned convergence of the number of operations to a finite value is also depicted on the right side of the figure, where the several lines join together as the associated number of precision scales increases. The same chart also shows that the generation of an approximate fBm with $10^6$ samples (and 18 precision scales) takes less than 0.3 seconds. In other words, the Java implementation of the fBm-SGA used in the scope of this work operates at an average rate of 3500 values per millisecond, on the aforementioned simulation device (see the chart on the left side of Figure 7).

![Figure 7](image)

**Fig. 7.** Computational complexity of the fBm-SGA: the chart on the left side of the figure plots the average number of generated points per millisecond, while the chart on the right side plots the average time spent to generate an approximate fBm against the total number of points generated and different numbers of precision scales supported.

### 4.4. Memory Requirements

A fully functional implementation of the fBm-SGA does not require more than the storage space for the representation of $\left\lfloor 2 \log_2(n) + k \right\rfloor$ real numbers, where $n$ is the total number of values to be generated, $k$ is the number of implementation specific auxiliary variables and $\lfloor x \rfloor$ denotes the floor function. This number accounts for the amount of past observations that need to be stored during execution of the algorithm (which are, at most, $\lfloor \log_2(n) \rfloor$) and for the values of the persistence probabilities for the different scales of type $2 \times 2^{N_p}$ (whose number is again not larger than $\lfloor \log_2(n) \rfloor$).

The reduced memory prerequisites of fBm-SGA can be explained by the way the dependencies are defined. There is a persistence probability for each scale of type $2 \times 2^{N_p}$ and, as so, the simulation of series with $n$ samples does not require $N_p$ to be bigger than the integer part of $\log_2(n)$. Moreover, only a very limited set of previously generated values are used in the synthesis of the current and future observations of the approximate process, and their validity expires once they have been used to generate
a certain number of values. It is thus possible to reuse memory blocks during runtime efficiently.

Observing Figure 1, it is possible to notice that the very first observation of the process is used to generate the values with indexes 1, 2, 4, 8, 16, ... and that the ones with odd indexes are created resorting always to their predecessors. On the other hand, the observation at the time step $4 = 2^2$, for example, is only used for the synthesis of two values; the observations at time step $8 = 2^3$ is used for the synthesis of three values, and so on. In an optimized codification of the fBm-SGA, the past points are stored in an array (say $pp[1]$), whose first value ($pp[1]$) is used to synthesize observations of $G_H^*(t + 2^{i-1})$ with $t \in 2^i \mathbb{N}$ (odd values), and it is replaced at each $2^i$ points; $pp[2]$ is used to synthesize observations of $G_H^*(t + 2^{i-1})$ for time steps $t \in 2^2 \mathbb{N}$, and it is updated at each $2^2$ values; $pp[3]$ is used to synthesize observations of $G_H^*(t + 2^{i-1})$ for time steps $t \in 2^3 \mathbb{N}$, and it is updated once per each $2^3$ values, and so on. In general, $pp[i]$ must contain the value used for the synthesis of $G_H^*(t + 2^{i-1})$ with $t \in 2^i \mathbb{N}$ and it can be replaced at each $2^i$ steps. Based on this reasoning, it may be concluded that the maximum length of such array is the integer part of $\log_2(n)$.

As a final remark, it should be mentioned that in the case where the length of the series to be generated is not known a priori, the size of the arrays for the persistence probability values and for the storage of the past observations (i.e. the number of precision levels $N_p$) may be set to a sufficiently large value, so as to cover feasible simulation times. For example, setting $N_p = 100$ in the implementation used in the scope of this work would be sufficient for synthesizing series with $2^{100}$ values, but the generator would not occupy more than 900 B in the memory of the computational system.

5. SIMULATION OF SELF-SIMILAR NETWORK TRAFFIC IN A PACKET-BY-PACKET MANNER

One of the main applications of fBm (or fGn) generators is the simulation of traffic in network aggregation nodes, where the bit count per time unit process is known to exhibit self-similar properties. In this section, we describe a possible means to simulate traffic traces in a packet-by-packet manner, by formalising the generation of the sequences of packet sizes and inter-packet gaps. Though the sequential generation of long-range dependent series is not an essential prerequisite of the proposed model, the combination of the latter with an $O(n)$ complexity algorithm, capable of producing values on-demand, may embody an attractive solution for situations demanding real-time operation.

5.1. Formalization

The following network parameters were taken into consideration when designing this simulator: the intended self-similarity degree of the network trace (given by the Hurst parameter $H$); the maximum bandwidth $B$ at which the network system is able to operate; the intended load $L$ (a fraction of the total available bandwidth); the minimum inter-packet gap $I_{min}$; and a known (e.g. empirical) probability distribution for the sizes of the packets arriving to a given network aggregation point.

In here, it is assumed that a trace consists of the sequences of packet sizes and inter-packet gaps. Below, $P(t)$ is used to represent the size of packet $t$ and $I(t)$ is used to denote the time gaps between the end of packet $t$ and the beginning of packet $t + 1$ (i.e. the inter-packet gaps). Assume that, for each $t \in \mathbb{N}$, $P(t)$ is a random variable with a known empirical distribution defined for the sample space of all positive integer numbers between $P_{min}$ and $P_{max}$. This means that no special measures are to be taken for the generation of the packet sizes and that these values may or may not exhibit self-similar properties. In other words, only the inter-packet gaps will certainly exhibit self-similar properties.
Given the packet size distribution, one can calculate the average packet size value \( \mathbb{E}(P(t)) \). Fix a time unit \( \tau \) and suppose that the maximum bandwidth \( B \) supported by the device is already in bits per time unit. Assume that the inter-packet gaps are expressed in bit time, i.e. the time it takes to transmit a bit. At a given moment in time, the average value of the bandwidth being used is given by \( B_U \) and the average number of packets that fill that bandwidth is given by

\[
\text{n}_{\text{packets}} = \frac{B_U}{\mathbb{E}(P(t))}.
\]  

(35)

As in the considered model each packet size is followed by an inter-packet gap, there will exist an average of \( n_{\text{packets}} \) inter-packet gaps per time unit. It is assumed herein that \( I(t) \) is a shifted and scaled (ultimately because there are no negative inter-packet gaps) fGn process, with Hurst parameter \( H \). The mean value and variance of this process must fulfill certain criteria. Since the used part of the bandwidth is filled with the packet sizes, the unused part must be filled with the gaps between the data units. To assure that, at least on average, the amount of traffic that arrives to the given aggregation point in the time unit considered is given by \( B_U \), the average inter-packet gap (denoted by \( \mu_I \)) must be defined as in (36):

\[
\mu_I = \frac{B_{NU}}{n_{\text{packets}}}, \text{ where } B_{NU} = B - B_U.
\]  

(36)

The previous condition also assures that, on average, the inter-packet gaps are positive numbers, but it is not sufficient to guarantee that all of the observations of \( \{I(t)\}_{t \in \mathbb{N}} \) are so, nor that the minimum inter-packet gap is seldom retrieved by the generation procedure. This problem can be solved (and, in some way, controlled) if the standard deviation of the process (denoted by \( \sigma_I \)) is defined by one of the following expressions:

\[
\sigma_I = \frac{\mu_I - I_{\text{min}}}{2}, \text{ or}
\]

\[
\sigma_I = \frac{\mu_I - I_{\text{min}}}{3}.
\]  

(37)

(38)

To better understand what expressions (36), (37) and (38) mean, consider observing Figure 8. The choice of the \( \sigma_I \) value ((37) or (38)) provides a way to choose how many values of the simulated Gaussian process are going to be smaller than \( I_{\text{min}} \) (\( \sim 0.1\% \) or \( \sim 2.2\% \) of the whole population). Ideally, one would think that this value (i.e. \( I_{\text{min}} \)) should only be retrieved by the generation algorithm with infinitesimal probability (except when we are simulating heavy load conditions), but empirical analysis shows that the \( I_{\text{min}} \) is regularly measured, even under light load conditions, since it separates the packets of bursts sent from remote nodes.

In order to avoid inter-packet gaps smaller than \( I_{\text{min}} \), the simulated observations below that threshold should be discarded and the predefined value of \( I_{\text{min}} \) should be retrieved instead. Unfortunately, this last step is the same as projecting the simulated values of the fGn when they are smaller than \( I_{\text{min}} \), according to the following expression:

\[
\text{return } I_{\text{min}}, \text{ if } I(t) < I_{\text{min}}.
\]  

(39)

To avoid having a slight bias in terms of the target traffic load, one might want to define a symmetrical upper bound condition as well:

\[
\text{return } 2\mu_I - I_{\text{min}}, \text{ if } I(t) > 2\mu_I - I_{\text{min}}.
\]  

(40)
Nonetheless, by resorting to these projections, one is inadvertently decreasing the variance of the inter-packet gaps and bit count per time unit, and decreasing the fidelity to the pure Gaussian model.

Having defined the expected value and the variance of the inter-packet gaps, it is easy to write the law that, according to this model, transfers self-similar properties to the traffic trace (recall that, in the next equation, \( \{G_H(t)\}_{t \in \mathbb{N}} \) denotes an fGn with Hurst parameter \( H \)):

\[
I(t) = \sigma_I G_H(t) + \mu_I.
\]  

Notice that specifying \( \sigma_I \) as (37) or (38) comprises a strategy to maximize the usage of the fGn generator and it does not reflect directly any real network traffic phenomenon. Nonetheless, by choosing one of these two options, we make sure that most of the values generated using (41) do not need to be projected, optimizing the transference of the self-similar properties to the bit count per time unit.

Figure 8 provides the graphical representation of the idea behind the simulation apparatus. It depicts where the long-range dependent Gaussian variable is to be placed in order to assure the self-similar properties of the bit count per time unit. For the sake of simplicity (and without loss of generality), some of the previously described concepts were relaxed to produce the illustration in the referred figure. To be exact, it must be said that the illustration is valid for \( \tau = \frac{E(P(t))}{L} \) (i.e. the figure is valid for the time unit where, on average, fits one packet only).

5.2. Implementation Details and Pictorial Evidence of Self-Similarity

The network traffic simulator was implemented in Java programming language and tested in the system described in Section 4.3. To generate the packet sizes, a file containing a packet size distribution of an aggregation network point was downloaded from one of the sites of NLANR [NLANR 2005], and later read by the traffic generator. Each time a packet size was to be returned, the probability domain was emulated by a pseudo-random number between 0 and 1, and the respective packet size value located in the empirical distribution. The generation of the inter-packet gaps is done by combining the fBm-SGA with transformation (41).

The traffic generator was tested for several network parameters combinations (different loads and Hurst parameter values). For each scenario, the number of bits generated was aggregated for three different time scales (0.1, 1 and 10 seconds - see charts in Figure 9) and the self-similarity degree of the resulting series was analysed in detail.
Simulations showed that the model indeed embeds the long-range dependence structure in the byte count per time unit of the trace, though the estimated Hurst parameter values were slightly smaller than the intended ones. The deviations are mainly due to the restrictions imposed to the inter-packet gaps and to the fact an approximate method for simulation of the fGn was used. To compensate them, one may consider initialising the generator with a Hurst parameter value that is larger than the one expected for the trace. In the case of the fBm-SGA, empirical studies have shown that, up to the value of 0.85, the value that has to be added to the input parameter of the algorithm is of 0.04. The target load was also a subject of analysis but, since this particular aspect is mostly dependent on the stationarity of the outputs of the algorithm in the long-term, the difference between the loads of the simulated traces and the intended ones was insignificant.

The charts in Figure 9 are the graphical representation of the bit count per time unit process, obtained from one of the simulations to the scenario with the following combination of parameters: $H = 0.85$; $B = 1$ Gbps; $L = 0.5$; $P_{\text{min}} = 360$ bit; $P_{\text{max}} = 12000$ bit; $I_{\text{min}} = 96$ bit. It was on charts of this kind that Leland et al. [1993; 1994] first found evidences of self-similarity in network traffic, and they show for example that the variances of the aggregated processes do not decrease as fast as it would happen for uncorrelated (or anti-persistent) sequences.

The simulator does not replicate all the particularities of the traffic for a given network aggregation point. For example, it does not simulate accurately the phenomenon of packet trains. There are two reasons behind this fact: (i) the generation of the trace is being made at the point where self-similarity is to be assured, emphasizing this objective the most and forgetting each one of the individuals (remote nodes) that contribute for the generation of the trace and; (ii) the packet sizes are modelled as realisations of a random variable and do not necessarily obey to any rule of persistence.

Fig. 9. Pictorial evidence of self-similarity in the simulated network traffic: the aggregation scale increases from 0.1 to 10 seconds as we move from chart a) to chart c). All traces were generated using the fBm-SGA with a predefined Hurst parameter equal to 0.85 and load parameter equal to 0.5.
6. CONCLUSION
A new method to approximate fBms exhibiting persistent behaviour was proposed and explained with detail in this paper. Its accuracy, in terms of expected Hurst parameter of the generated series, was tested through simulation, and its computational requirements were analysed and quantified. Both memory and computational requirements of the proposed algorithm are extremely low. It was proven that the complexity of the fBm-SGA is independent of the number of points to be generated, and that the amount of memory required during runtime is negligible for any practical instantiation of the algorithm. Although the method was initially designed to assure long-range dependence properties for scales which are powers of 2 and for non-intersecting blocks, the simulations demonstrate that self-similarity is preserved for other scales also. For scales of type different than $2 \times 2^N$, the estimates of the Hurst parameter of the simulated series are typically smaller than the expected one, but the difference is small. Because the fBm-SGA is capable of synthesising sequences of values with an approximate long-range dependence structure in an efficient and sequential manner, it is suitable for simulations demanding real-time generation of self-similar sequences with predefined Hurst parameter and unknown running time.

A possible application of the proposed algorithm as a network traffic generator was also presented. The description includes the definition of important network parameters, but it does not take into account all the characteristics of the traffic in a given aggregation point, since that level of detail is out of the scope of this paper. Nevertheless, as defined, the proposed model can be used to generate extremely long traces on a packet-by-packet basis, without imposing severe burden to the machine running the simulation software. Just to give an idea of the performance of the simulator, a Java implementation was able to simulate network traffic at a data rate higher than 8 Gbps and would produce a 100 GB trace with less than 340 B of memory.

ACKNOWLEDGMENTS
Authors would like to thank José Miguel Santos for many valuable discussions on the subject, and to João Gomes, Branka Lakic and Bodhisattwa Gangopadhyay for their revision of the first version of this article. The authors are also thankful to all the anonymous reviewers for constructively criticizing this work.

REFERENCES


Received May 2008; revised September 2011; accepted —