AN EFFECTIVE METHOD FOR SIMULATING GAUSSIAN RANDOM FIELDS

Grace Chan

KEY WORDS: Circulant matrix, fast Fourier transform, fractal dimension, sample path roughness, stationary increments, stationary random fields, Toeplitz matrix.

1 INTRODUCTION

Both stationary and non-stationary Gaussian random fields have been used to model real life phenomena such as surface roughness. Often data consist of observed surface height over regular grid. In order to study the properties of certain estimators of an unknown population parameter such as fractal dimension numerically, one needs to generate a large number of "long" realizations which resemble the real data. Figures 1 and 2 show examples of such realizations. Three types of scalar-valued Gaussian random processes are displayed in Figure 1:

(a) top panel: a stationary process with uniform roughness along its sample path,

(b) middle panel: a fractional Brownian motion is an example of a non-stationary process with uniform roughness along its sample path, and

(c) bottom panel: a multifractional Brownian motion is an example of a non-stationary process with variable roughness along its sample path.

Figure 2 shows a grey scale image of a scalar-valued stationary Gaussian random field with uniform roughness along all directions.

Besides modeling surface roughness, Gaussian random process can be used to model correlated time series. These multiple time series may be generated as vector-valued Gaussian random process with specified covariance structure. Figure 3 shows three vector-valued Gaussian random processes:

(a) top panel: two *rough* correlated scalar-valued Gaussian random processes with identical roughness,

(b) middle panel: two *smooth* correlated scalar-valued Gaussian random processes with identical roughness, and

(c) bottom panel: two *mixed* correlated scalar-valued Gaussian random processes with different roughness.

In this paper, we will describe a convenient method for generating these random processes and fields effectively. In Section 2, we will focus on stationary Gaussian Random Process and fields. Non-stationary Gaussian random process and how one may estimate its roughness will be discussed in Section 3. We will conclude this paper with information on CPU timing of all algorithms described in this paper in Section 4. Fortran programs implementing all simulation algorithms are available from `http://www.stat.uiowa.edu/~grchan`.

Before we conclude this section, let us introduce some notations that are used throughout this paper:

$$X = \{\boldsymbol{X}(\boldsymbol{t}) = (X_1(\boldsymbol{t}), \cdots, X_p(\boldsymbol{t}))' \in \mathbb{R}^p; \boldsymbol{t} \in \mathbb{R}^d\}$$

denotes a vector-valued Gaussian random field when $d, p \geq 1$,

$$\boldsymbol{t} = (t[1], \cdots, t[d])' \text{ with } t[\ell] = j[\ell] \text{ or } \frac{j[\ell]}{n[\ell]}$$

for $0 \leq j[\ell] \leq n[\ell] - 1, \ \ell = 1, \ldots, d$ denotes the regular grid on d directions with $n[\ell]$ equally-spaced observed points in the ℓth direction. For convenience, we will write

$$\left(\frac{j[1]}{n[1]}, \ldots, \frac{j[d]}{n[d]}\right)' \quad \text{as} \quad \frac{\boldsymbol{j}}{\boldsymbol{n}}$$

and likewise for products, addition and subtraction of two vectors in $\mathbb{R}^d$.

$$\gamma_{i,j}(\boldsymbol{k}, \boldsymbol{h}) = \operatorname{cov}\{X_i(\boldsymbol{k}), X_j(\boldsymbol{h})\},$$

for $i, j = 1, \ldots, p, \ \boldsymbol{k}, \boldsymbol{h} \in \mathbb{R}^d$, denote the covariance between $X_i(\boldsymbol{k})$ and $X_j(\boldsymbol{h})$. Without loss of generality, we will assume all random processes and fields have zero mean. For stationary random fields $\gamma_{i,j}(\boldsymbol{k}, \boldsymbol{h}) = \gamma_{i,j}(|\boldsymbol{k} - \boldsymbol{h}|)$.

2 STATIONARY GAUSSIAN RANDOM FIELDS

In this section we will describe an effective simulation method for stationary Gaussian random process and field on regular grid. In Subsection 2.1, we give the main idea behind this method and an algorithm to simulate scalar-valued stationary Gaussian random process. Some modifications to the algorithm are needed to simulate scalar-valued and vector-valued stationary Gaussian random fields. Required modifications will be given in Subsections 2.2 and 2.3 respectively.

2.1 Scalar-valued Stationary Gaussian Random Processes

Now $d = 1$, $p = 1$, suppose we need realization of length n on the unit interval $[0, 1)$: $X = \left\{X(0), X\left(\frac{1}{n}\right), \ldots, X\left(\frac{n-1}{n}\right)\right\}$, with covariance function specified by γ. Hence its covariance matrix

$$\Sigma = \begin{bmatrix} \gamma(0) & \gamma\left(\frac{1}{n}\right) & \cdots & \gamma\left(\frac{n-1}{n}\right) \\ \gamma\left(\frac{1}{n}\right) & \gamma(0) & \cdots & \gamma\left(\frac{n-2}{n}\right) \\ \vdots & \vdots & & \vdots \\ \gamma\left(\frac{n-1}{n}\right) & \gamma\left(\frac{n-2}{n}\right) & \cdots & \gamma(0) \end{bmatrix}$$

is an $n \times n$ symmetric nonnegative definite Toeplitz matrix which can be embedded into an $m \times m$ ($m \geq 2(n-1)$) symmetric circulant matrix

$$C = \begin{bmatrix} c_0 & c_1 & \cdots & c_{m-1} \\ c_{m-1} & c_0 & \cdots & c_{m-2} \\ \vdots & \vdots & & \vdots \\ c_1 & c_2 & \cdots & c_0 \end{bmatrix}$$

with

$$c_j \equiv \begin{cases} \gamma\left(\frac{j}{n}\right) & 0 \leq j \leq \frac{m}{2} \\ \gamma\left(\frac{m-j}{n}\right) & \frac{m}{2} < j \leq m-1. \end{cases}$$

By the above construction, all $n \times n$ submatrices of C down the diagonal is equal to Σ. Hence if C is nonnegative definite, it is a covariance matrix of a stationary random process Y at m equally spaced points on a circle and any n consecutive elements from it form a realization with the specified covariance structure. Note that $C = Q\Lambda Q^*$ where Λ is a diagonal matrix with eigenvalues of C down the diagonal and Q is an $m \times m$ unitary matrix with columns being eigenvectors of C. Also note that eigenvalues of C is the FFT of any row from C, eigenvectors of C do not depend on C, and $Q\boldsymbol{a}$ is the FFT of $\boldsymbol{a}$. These three points implies that $Y \equiv Q\Lambda^{1/2}Q^*Z$, $Z \sim N(0, I)$ can be generated effectively with two applications of FFT if C is nonnegative definite, i.e. if its minimum eigenvalue is greater than or equal to zero.

For relatively small n (< 1000), the Cholesky factorization method generates realization with the specified covariance structure. But in the study of surface roughness and in many other geostatistic areas, observed realizations have length of the order of several thousands and in studying the asymptotic properties of estimators, one needs realization with length as long as possible. Hence we need other method to generate these long realizations. Cressie (1993) and Ripley (1987) described many different methods to generate long realizations. But all those methods only generate long realization with covariance structure *approximate* the specified covariance structure unless the specified covariance structure is very special. Wood and Chan (1994) introduced a circulant embedding method to generate long realization with the specified covariance structure for a wide range of covariance structure. Often a symmetric nonnegative definite circulant matrix C can be found with minimal length, i.e. $m = 2(n-1)$. In some cases, one may need to increase m in order to get a nonnegative definite circulant matrix C. Then m depends on n and the covariance function γ. Choosing m to be a power of 2 can increase the efficiency of this method. Next we present an algorithm to generate X using this circulant embedding idea.

Algorithm:

1. Ask user to specify covariance structure.
2. Ask user to specify the length of the required process, n.
3. Compute the smallest embedding length, $m = 2^g \geq 2(n-1)$.
4. Define the first row of a circulant matrix C as described above.
5. Compute eigenvalues of C using FFT on the first row of C.
6. If the minimum eigenvalue of C is less than zero, then increase g by 1, set $m = 2^g$, and return to step 4. Otherwise, compute square roots of all eigenvalues.
7. Simulate Q^*Z directly where Z is an $m \times 1$ vector with m independent standard normal random numbers.
8. Define $\boldsymbol{a}$ as $\Lambda^{1/2}Q^*Z$.
9. Define Y as the FFT of $\boldsymbol{a}$.
10. Extract n consecutive elements of Y to form the require realization X.

Note that step 1 is usually done in the programming stage. The specified covariance structure is usually programmed as either subroutine or function. Steps 2 – 3 only need to be done once. Steps 4 – 6 may be repeated if the minimal highly composite m does not produce a nonnegative matrix. The upper limit of m depends on the power of the computer. If a nonnegative definite matrix cannot be found subject to computer limitation, Wood and Chan (1994) described a modified algorithm to generate realization with distribution approximate the required one. We also described how one can determine the difference between the two distributions. Steps 7 – 10 will be repeated depending on the number of independent realizations required. Detail of simulating Q^*Z directly can be found in Wood and Chan (1994).

Figure 1(a) presents a single realization of length 10000 based on the following covariance function:

$$\gamma(t) \ = \ \exp\{-c|t|^\alpha\}, \quad c>0, \alpha \in (0,2],$$

with $c = 100$ and $\alpha = 1$. Adler (1981) showed that the roughness of these sample path is defined by its fractal dimension,

$$D \ = \ 2 - \frac{\alpha}{2} \ \in \ [1,2).$$

Hence the fractal dimension of Figure 1(a) is 1.5.

Detail on Fortran program implementing the above algorithm is given in Chan and Wood (1997).

2.2 Scalar-valued Stationary Gaussian Random Fields

When $d \geq 1$, $p = 1$, X is a scalar-valued stationary Gaussian random fields with "size" $\boldsymbol{n} = (n[1], \ldots, n[d])'$ and length $\bar{n} = \prod_{\ell=1}^d n[\ell]$. We can follow the above algorithm with three modifications: choice of $\boldsymbol{m} = (m[1], \ldots, m[d])$, dimension of FFT, and extraction of X.

Writing X as an $\bar{n} \times 1$ vector. Its covariance matrix Σ is an $\bar{n} \times \bar{n}$ symmetric nonnegative definite block Toeplitz matrix (and Toeplitz within each block) which can be embedded into an $\bar{m} \times \bar{m}$ ($\bar{m} = \prod_{\ell=1}^d m[\ell]$ and $m[\ell] \geq 2(n[\ell]-1)$) symmetric block circulant matrix (and circulant within each block) C with

$$c_{\boldsymbol{j}} = \gamma\left(\frac{\tilde{\boldsymbol{j}}}{\boldsymbol{n}}\right), \quad \boldsymbol{j} \in \mathbb{Z}^d, \quad 0 \leq |j[\ell]| \leq m[\ell]-1$$

where for $\ell = 1, \ldots, d$

$$\tilde{j}[\ell] \equiv \begin{cases} j[\ell] & 0 \leq |j[\ell]| \leq m[\ell]/2 \\ j[\ell]-m[\ell] & m[\ell]/2 < j[\ell] \leq m[\ell]-1 \\ j[\ell]+m[\ell] & \text{elsewhere.} \end{cases}$$

For reflective symmetric covariance function: $\gamma(\boldsymbol{t}) \ = \ \gamma(-\boldsymbol{t})$, Σ may not be symmetric within each block so as C. In this case, $m[\ell]$'s have to be an odd number whenever

$$\begin{aligned} & \gamma\{(t[1], \ldots, t[\ell], \ldots, t[d])'\} \\ \neq \ & \gamma\{(t[1], \ldots, -t[\ell], \ldots, t[d])'\}, \end{aligned}$$

say $m[\ell] = 3^g \geq 2(n[\ell]-1)$. One can still choose even number for $m[\ell]$ but now $m[\ell] = 2^g \geq 2n[\ell]$ and special care is needed in defining $c_{\boldsymbol{j}}$ whenever $j[\ell] = \pm m[\ell]/2$ for some ℓ. Wood and Chan (1994) suggested one way to define these elements.

For either isotropic symmetric covariance function:

$$\gamma(\boldsymbol{t}) \ = \ \gamma(\boldsymbol{s}), \quad |\boldsymbol{t}| = |\boldsymbol{s}|$$

or elliptical symmetric covariance function:

$$\gamma(\boldsymbol{t}) \ = \ \gamma(\boldsymbol{s}), \quad |t[\ell]| = |s[\ell]|, \ \ell = 1, \ldots, d,$$

Σ is symmetric within each block so as C. In this case, we can choose $m[\ell] \geq 2(n[\ell]-1)$ as even number.

The easiest way to extract X from Y is to define X as follow:

$$X(\boldsymbol{j}) \ \equiv \ Y(\boldsymbol{j}), \quad 0 \leq j[\ell] \leq n[\ell]-1.$$

Figure 2 present a single realization with $d = 2$ and "size" = (100, 100) based on the following covariance function:

$$\gamma(\boldsymbol{t}) \ = \ \exp\{-c\|\boldsymbol{t}\|^\alpha\}, \ c > 0, \ \alpha \in (0,2],$$

with $c = 100$ and $\alpha = 1.9$. Adler (1981) showed that the roughness of these surfaces is defined by its fractal dimension,

$$D \ = \ 3 - \frac{\alpha}{2} \ \in \ [2,3).$$

Hence the fractal dimension of Figure 2 is 2.05.

Detail on Fortran program implementing the above modified algorithm for $d = 2$ is given in Chan and Wood (1997).

2.3 Vector-valued Stationary Gaussian Random Fields

When $d \geq 1$, $p \geq 1$, the covariance matrix Σ is a $p\bar{n} \times p\bar{n}$ symmetric block Toeplitz matrix which can be embedded into a $p\bar{m} \times p\bar{m}$ symmetric block circulant matrix C. For general covariance function γ, there is no special structure within each block except

$$\Gamma\left(\frac{\boldsymbol{j}}{\boldsymbol{n}}\right) \ = \ \begin{pmatrix} \gamma_{11}\left(\frac{\boldsymbol{j}}{\boldsymbol{n}}\right) & \cdots & \gamma_{1p}\left(\frac{\boldsymbol{j}}{\boldsymbol{n}}\right) \\ \vdots & & \vdots \\ \gamma_{p1}\left(\frac{\boldsymbol{j}}{\boldsymbol{n}}\right) & \cdots & \gamma_{pp}\left(\frac{\boldsymbol{j}}{\boldsymbol{n}}\right) \end{pmatrix}$$

$$= \left\{\Gamma\left(\frac{-\boldsymbol{j}}{\boldsymbol{n}}\right)\right\}'.$$

We will refer to Γ as the matrix-valued covariance function. The m's are chosen as in Subsection 2.2.

Now we need to define the first block row (p rows) of C as above except all γ's are replaced by Γ. Hence $C = (Q \otimes I)\mathrm{diag}\{A(\boldsymbol{j})\}(Q^* \otimes I)$ where Q is an $\bar{m} \times \bar{m}$ unitary matrix, $\otimes$ denotes the Kronecker product of two matrices, I is a $p \times p$ identity matrix, and $A(\boldsymbol{j})$ are $\bar{m}$ $p \times p$ Hermitian matrices such that $A(\boldsymbol{j}) = R\Lambda(\boldsymbol{j})R^*$ where R is a $p \times p$ unitary matrix, and $\Lambda(\boldsymbol{j})$ are diagonal matrices with eigenvalues of $A(\boldsymbol{j})$ down their diagonal. Thus we have to modify steps 5, $7-9$:

step 5 We compute $A(\boldsymbol{j})$ with $p(p+1)/2$ applications of d-dimensional FFT and then their eigenvalues.

step 7 We simulate $(Q^* \otimes I)Z$ directly, where Z is a $p\bar{m} \times 1$ vector of independent standard normal random numbers.

step 8 We define $\bar{m}$ $\boldsymbol{a}(\boldsymbol{j})$ as $R\Lambda(\boldsymbol{j})^{1/2}R^*(Q^* \otimes I)Z$.

step 9 Compute Y with p applications of d-dimensional FFT.

Finally we will extract X from Y as in Subsection 2.2.

Figure 3 presents three independent realizations with $d = 1$, $p = 2$ and length 10000 based on the following matrix-valued covariance function:

$$\Gamma(t) = (1 + \theta^2 p - 2\theta)^{-1} G \mathrm{diag}\{\gamma_1(t), \gamma_2(t)\} G$$

where

$$\begin{aligned} G &= (I - \theta \mathbf{1}\mathbf{1}') \\ I &= p \times p \text{ identity matrix} \\ \mathbf{1} &= (1 \ldots 1)', \quad \text{a } p \times 1 \text{ vector} \\ \gamma_j &= \exp\{-c|t|^{\alpha_j}\}, \end{aligned}$$

$\theta \in \mathbb{R}$, $c > 0$, $\alpha_1, \alpha_2 \in (0, 2]$ with $c = 100$, $\theta = 0.05$, $(\alpha_1, \alpha_2) = (0.1,\ 0.1)$, $(1.9,\ 1.9)$ and $(1.9,\ 0.1)$.

Detail for this case can be found in Chan and Wood (1999).

3 NON-STATIONARY GAUSSIAN RANDOM PROCESSES

In many applications, it may be more appropriate to model real data as non-stationary Gaussian process. In this section we will describe how one can extend the above algorithms to generate non-stationary process with stationary increment effectively using the circulant embedding idea. In Subsection 3.1, we will consider non-stationary Gaussian process with uniform roughness. And then we will discuss how to generate non-stationary Gaussian process with non-uniform roughness. Detail for both cases can be found in Chan and Wood (1998).

3.1 Fractional Brownian motions

The standard Brownian motion is a well-known example of non-stationary Gaussian process with stationary increment and covariance function

$$\gamma(s, t) = \frac{1}{2}\{|s| + |t| - |s - t|\}.$$

Falconer (1990) showed that the sample path of standard Brownian motion has fractal dimension 1.5. A length 10000 standard Brownian motion is presented in Figure 1(b). One can generalize standard Brownian motion to fractional Brownian motion (FBM) with covariance function

$$\gamma(s, t) = \frac{1}{2}\{|s|^{2H} + |t|^{2H} - |s - t|^{2H}\},$$

where $H \in (0, 1)$. Note that standard Brownian motion is one particular example of fractional Brownian motion with $H = 0.5$. Falconer (1990) also showed that the sample path of fractional Brownian motion has fractal dimension $D = 2 - H$.

One can use the same algorithm described in Subsection 2.1 to generate the stationary increments of fractional Brownian motion. Since fractional Brownian motion always starts at zero, the cumulative sum of its increments will be the required fractional Brownian motion.

3.2 Multifractional Brownian motions

Some surfaces may not be stationary and may not have uniform fractal dimension over the whole surface. For example data in image analysis and signal processing. One can generalize standard Brownian motion further to multifractional Brownian motion (MFBM) with fractal dimension $D(t) = 2 - H(t)$, $t \in \mathbb{R}$. Now both D and H are functions of t. The covariance between any two points in this sample path is given by

$$\gamma(s, t) = \frac{g_{H(s),H(t)}}{2}\{|s|^{Hsum} + |t|^{Hsum} - |s - t|^{Hsum}\},$$

where g depends on the *local* fractal dimensions at the two points of interest and $Hsum = H(s) +$

$H(t)$. One can treat the above γ as covariance between correlated fractional Brownian motions and generate their increments as vector-valued stationary Gaussian random processes. (See Subsection 2.3) When D is a continuous function of t, one chooses a finite number of fractal dimensions, say k, generates vector-valued stationary Gaussian random processes with these k fractal dimensions. Finally the require process at each time point is defined via some form of kriging. That is $X(t)$ is defined as a linear combination of observations from the realization of the vector-valued stationary Gaussian random processes with weight depending on the difference between $H(t)$ and those k fractal dimensions. One can also consider nearby observations in terms of time. The coefficients in the linear combination can be estimated based on a Mean Squared Error (MSE) criterion.

Figure 1(c) represents a multifractional Brownian motion of length 1000 with a continuous periodic function $H(t) = 0.5 + 0.49 \sin(4\pi t)$. This process is generated using eight correlated fractional Brownian motion of length 1000 with fractal dimensions $1/9, \ldots, 8/9$. The MSE are of the order of 10^{-5} unless $H(t)$ is outside the range of fractal dimensions, i.e. when $H(t) > 8/9$ or $H(t) < 1/9$.

To conclude this section, we will estimate the fractal dimension $D(t)$ of the multifractional Brownian motion showed in Figure 1(c) using a moving window increment-based estimator. Theoretical results are given in Istas and Lang (1997). Figure 4 shows the true fractal dimension (solid line) and a smoothed estimate fractal dimension (dotted line). As in the case of uniform fractal dimension, the increment-based estimators do well for rough sample path but not so good for smooth sample path. Nevertheless, the smoothed estimate fractal dimension reflect the variation of fractal dimension along the sample path.

4 Conclusion

We will conclude this paper by listing some CPU timings. Table 1 shows CPU timings in seconds on a HP Visualize C360 based on 50 independent runs in three main steps:

T_1: Time required to find nonnegative definite C and information on computing its square root. (This is done *once* for each covariance structure.)

T_2: Time required to generate one realization. (This will be repeated to generate more than one realization of the same covariance structure.)

T_3: Time required to compute the non-stationary process. (This is only for non-stationary process and will be repeated to generate more than one realization.)

The largest standard deviation on timing is of order 10^{-2} second. According to these figures, to generate 1000 scalar-valued stationary Gaussian random processes of length 100000 will take approximately $1.514 + 1000 \times 1.465 = 24.5$minutes, and to generate 1000 multifractional Brownian motions of length 1000 will take approximately $1.413 + 1000 \times (0.228 + 0.325) = 9.2$minutes.

REFERENCES

Adler, R., (1981), *The Geometry of Random Fields,* John Wiley.

Chan, G. and Wood, A.T.A., (1997), "An Algorithm for Simulating Stationary Gaussian Random Fields," *Applied Statistics,* 46, 171–181.

Chan, G. and Wood, A.T.A., (1998), "Simulation of Multifactional Brownian Motion," *Proceedings in Computational Statistics 1998,* Edited by R. Payne and P. Green. 233–238.

Chan, G. and Wood, A.T.A., (1999), "Simulation of Stationary Gaussian Vector Fields," *Statistics and Computing,* to appear.

Cressie, N.A.C., (1993), *Statistics for Spatial Data,* John Wiley.

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

Istas, J. and Lang, G., (1997), "Quadratic Variations and Estimation of the Local Hölder Index of a Gaussian Process," *Annales de l'Institut Henri Poincaré,* 33, 407–436.

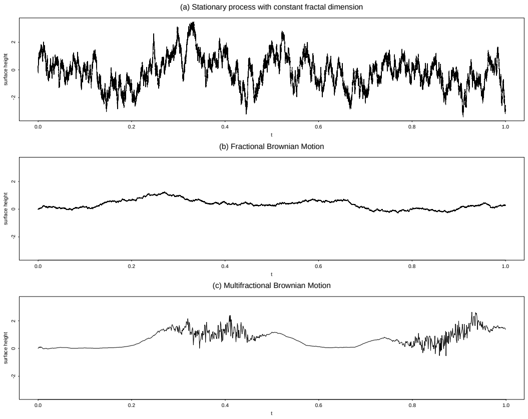
Ripley, B. D., (1987), *Stochastic Simulation,* John Wiley.

Wood, A.T.A. and Chan, G., (1994), "Simulation of Stationary Gaussian Processes in $[0, 1]^d$," *Journal of Computational and Graphical Statistics,* 3, 409–432.

Table 1: CPU Timings in Seconds on a HP Visualize C360

Type of random field	“length”	parameters	g		CPU timing		
			initial	final	T_1	T_2	T_3
STATIONARY		c, α					
scalar, process	100000	100, 1	18	18	1.514	1.465	
scalar, field	100×100	100, 1	8, 8	8, 8	0.211	0.189	
vector, process	10000×2	100, 1	15	15	0.580	0.370	
NON-STATIONARY		H					
FBM	100000	0.5	18	18	1.719	1.467	0.134
MFBM	1000 (8)	periodic	12	12	1.413	0.228	0.325

Figure 1: Realizations of scalar-valued Gaussian Random Process



(a) Stationary process with constant fractal dimension

(b) Fractional Brownian Motion

(c) Multifractional Brownian Motion

Figure 2: Realization of scalar-valued Gaussian Random Field

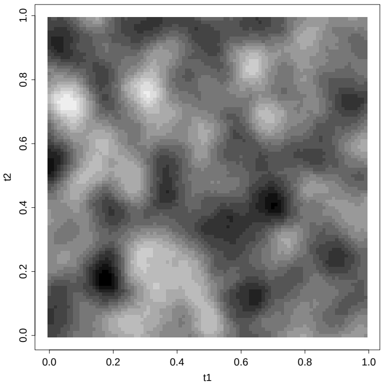
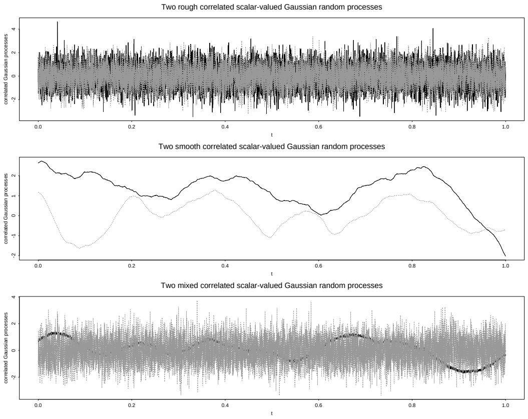


Figure 3: Realizations of vector-valued Gaussian Random Process

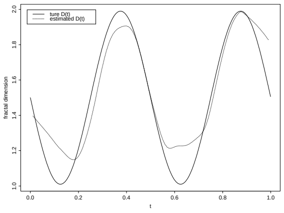


Two rough correlated scalar-valued Gaussian random processes

Two smooth correlated scalar-valued Gaussian random processes

Two mixed correlated scalar-valued Gaussian random processes

Figure 4: True and estimate fractal dimension of the multifactional Brownian motion showed in Figure 1(c)



ture D(t)
estimated D(t)
fractal dimension
t