Computers and Geosciences 112 (2018) 121–132

Contents lists available at ScienceDirect

Computers and Geosciences

journal homepage: www.elsevier.com/locate/cageo

Research paper

# Accelerating Sequential Gaussian Simulation with a constant path

Raphaël Nussbaumer [a,*], Grégoire Mariethoz [b], Mathieu Gravey [b], Erwan Gloaguen [c], Klaus Holliger [a]

[a] Institute of Earth Sciences, University of Lausanne, Lausanne, Switzerland
[b] Institute of Earth Surface Dynamics, University of Lausanne, Lausanne, Switzerland
[c] Centre Eau Terre Environnement, Institut National de la Recherche Scientifique, Québec, Canada

ARTICLE INFO

*Keywords:*
Simulation path
Sequential simulation
Sequential Gaussian Simulation
Constant path
Multi-grid approach
Parallelization

ABSTRACT

Sequential Gaussian Simulation (SGS) is a stochastic simulation technique commonly employed for generating realizations of Gaussian random fields. Arguably, the main limitation of this technique is the high computational cost associated with determining the kriging weights. This problem is compounded by the fact that often many realizations are required to allow for an adequate uncertainty assessment. A seemingly simple way to address this problem is to keep the same simulation path for all realizations. This results in identical neighbourhood configurations and hence the kriging weights only need to be determined once and can then be re-used in all subsequent realizations. This approach is generally not recommended because it is expected to result in correlation between the realizations. Here, we challenge this common preconception and make the case for the use of a constant path approach in SGS by systematically evaluating the associated benefits and limitations. We present a detailed implementation, particularly regarding parallelization and memory requirements. Extensive numerical tests demonstrate that using a constant path allows for substantial computational gains with very limited loss of simulation accuracy. This is especially the case for a constant multi-grid path. The computational savings can be used to increase the neighbourhood size, thus allowing for a better reproduction of the spatial statistics. The outcome of this study is a recommendation for an optimal implementation of SGS that maximizes accurate reproduction of the covariance structure as well as computational efficiency.

## 1. Introduction

Sequential Gaussian Simulation (SGS) is a popular method for generating stochastic values on a grid under the constraints of a statistical model and, possibly, some initially known values, herein referred to as hard data. SGS has been extensively used by practitioners because of its intuitive theoretical basis, its simple numerical implementation, and its high flexibility (e.g. Gómez-Hernández and Journel, 1993; Pebesma and Wesseling, 1998). Arguably, the major drawback of SGS is its computational cost. The exact estimation of kriging relies on taking into account all conditioning nodes, which results in large linear systems that need to be solved. For a square matrix of size $n$, common linear solvers have a computational complexity of $O(n^3)$ (Trefethen and Bau, 1997), which means that the computational effort is proportional to the cube of the matrix dimension. Therefore, the sequential simulation of a grid with $N$ nodes represents an $O(N^4)$-type problem (Dimitrakopoulos and Luo, 2004).

Various attempts have been undertaken to reduce the associated computational cost. The most widespread approach is the so-called limited or moving neighbourhood, that is, the approximation of the kriging estimate by using only a limited number of conditioning points referred to as the neighbours (e.g. Isaaks and Srivastava, 1989; Deutsch and Journel, 1992; Goovaerts, 1997). This reduces the computational complexity of SGS to $O(k^3N)$, where $k$ denotes the number of neighbours. This approach is rooted in the observation that neighbours which are located far away from the simulated point receive small or even vanishing weights. This effect originates from the rapid decrease of correlation with distance inherent to most covariance functions and is enhanced by the presence of intermediate neighbours screening the influence of those behind (e.g. Chilès and Delfiner, 1999). However, the omission of neighbours has shown to bias the simulation covariance matrix (Emery and Peláez, 2011; Nussbaumer et al., 2017), which in turn results in artifacts in the realizations (e.g. Meyer, 2004). Recent works on reducing such detrimental effects, while limiting the neighbourhood size and optimizing the computational efficiency, include those of Gribov and Krivoruchko (2004), Rivoirard and Romary (2011) and Dimitrakopoulos and Luo (2004).

An alternative to reducing the size of the kriging covariance matrix is

* Corresponding author.
*E-mail address:* raphael.nussbaumer@unil.ch (R. Nussbaumer).

https://doi.org/10.1016/j.cageo.2017.12.006
Received 2 May 2017; Received in revised form 5 September 2017; Accepted 14 December 2017
Available online 16 December 2017
0098-3004/

to approximate it. Barry and Kelley Pace (1997) formulate covariance-based kriging, which leads to the inversion of sparse symmetric matrices. Sparse matrix solvers considerably improve the computational performance, but this approach is limited to simulations based on covariance functions with a finite range. Furrer et al. (2006) and Memarsadeghi and Mount (2007) further increase the sparsity of the matrix by tapering the covariance for large lag-distances. Related approaches comprise the approximate iterative method (Billings et al., 2002), the low rank approximation (Kammann and Wand, 2003), the Sherman-Morrison-Woodbury formula (Sakata et al., 2004), and fast summation methods (Memarsadeghi et al., 2008; Srinivasan et al., 2008).

Another approach is to only consider simulations whose covariance function is from a limited set of easily solvable covariance models. Omre et al. (1993) propose the screening sequential simulation, which provides exact simulations for covariance models with the Markov property, such as, for example, the exponential model in 1D. Hartman and Hössjer (2008) approximate the simulated Gaussian field with a set of Gaussian Markov random fields (Rue and Tjelmeland, 2002), which can be simulated exactly and efficiently. Finally, Cressie and Johannesson (2008) consider covariance models composed of a fixed number of basic non-stationary functions. This technique is also referred to as fixed-rank kriging. A related approach is the predictive processes method (e.g. Banerjee et al., 2008).

A more general technique to cope with the high computation costs of SGS is parallelization, which reduces the computation time by splitting the work among several cores (Vargas et al., 2007; Mariethoz, 2010; Nunes and Almeida, 2010; Rasera et al., 2015). It is important to note that parallelization does not reduce the computational burden, but merely spreads it over several cores, and hence is just a useful complement to the other techniques.

The approach explored in this study aims at decreasing the overall computational cost by taking advantage of the large number of realizations typically needed in geostatistical applications. Indeed, an uncertainty assessment can only be performed with an ensemble of realizations spanning the variability of outcomes. When the simulation path, that is, the order in which the nodes are simulated, is kept identical among multiple realizations, the neighbourhood configurations of each simulated node are also identical throughout these realizations. Because the kriging weights are computed solely with the relative distances between nodes, a constant neighbourhood configuration produces the same kriging weights. Therefore, these weights only need to be computed once and then can be re-used for all realizations. This reduces the computational effort of each additional realization to simple matrix multiplications.

While some works outline the advantages of using a constant path (e.g. Verly, 1993), the overwhelming majority still discourages its use, because of the risk to draw correlated realizations, and rather advocate a randomized path to explore the solution space more homogenously (e.g. Deutsch and Journel, 1992; Goovaerts, 1997). Conversely, Cáceres et al. (2010); Boisvert and Deutsch (2011) reported that using a constant path in SGS does not result in a significant reduction of the space of uncertainty for neither first- nor second- order statistics, while allowing for compelling reductions in computational cost. However, both studies are based on empirical evidence and hence the generic validity of their findings remains to be verified.

In the present work, we seek to provide a thorough understanding of the implications of changing the simulation path in order to assess the constant path method. The paper is organized as follows. We begin by presenting a methodological description of randomized path simulations (section 2), followed by the implementation of a constant path method (section 3) and the quantification of the associated computational gains (section 4). Finally, we discuss some limitations of the covariance matrix evaluation (section 5).

## 2. Theory of randomized paths simulations

In order to understand the implications of generating stochastic realizations based on the same simulation path, the links between the random function (RF) $Z$, the realizations $z$, and the path $p_i$ need to be explored in some detail.

### 2.1. Definition of a random function

In probability theory, a random variable (RV) denoted $X$ is a deterministic function mapping the set of possible outcomes $\Omega$ of a random phenomenon to their values, usually a real number $\mathbb{R}$,

$$X : \Omega \rightarrow \mathbb{R} \tag{1}$$

$$\omega \mapsto x.$$

In the definition of a RV, $\Omega$ has to be a probability space, which implies that each possible outcome $\omega$ has a well-defined probability. Thus, the probability $P(X \leq x_T)$ is defined by the set of events $\{\omega \in \Omega : X(\omega) \leq x_T\}$.

For instance, a RV describing the sum of two rolled dice $n_1$ and $n_2$ is defined as the function mapping every possible outcome $(n_1, n_2)$ to the measure $n_1 + n_2$

$$X(\{n_1, n_2\}) = n_1 + n_2. \tag{2}$$

With this formalism, the probability of the sum of two dice being 5 is defined as

$$\begin{aligned} P(X = 5) &:= P(\{n_1, n_2 \in \{1,2,3,4,5,6\} : n_1 + n_2 = 5\}) \\ &= P(\{1,4\},\{2,3\},\{3,2\},\{4,1\}) = 4/6^2 = 1/9. \end{aligned} \tag{3}$$

A realization $x^{(l)}$ is the value observed from a RV $X$ given a specific outcome of the random phenomenon, also called random variate, $\omega_l$

$$x^{(l)} = X(\omega_l). \tag{4}$$

### 2.2. Sequential Gaussian Simulation

GS is an algorithm whose purpose is to produce realizations $z^{(l)}(\boldsymbol{u})$ of a *regionalized multi-Gaussian random function (RF)* $Z(\boldsymbol{u})$.

1. A RF is a collection of indexed RV. If the indexation is multi-dimensional, the collection is usually referred to as random field instead.
2. A RF is called regionalized (Matheron, 1965) if it is distributed in a continuous space domain $D \subset \mathbb{R}^n$,

$$Z = \{Z(\boldsymbol{u}),\ \boldsymbol{u} \in D\}, \tag{5}$$

where $\boldsymbol{u}$ represents a space coordinate vector.

3. A RF is multi-Gaussian if any finite collection of its components has a multi-variate normal distribution. While this constraint is restrictive, it allows for the RF to be fully determined by its first- and second-order moments, that is, the mean $\boldsymbol{\mu}_Z$ and the covariance matrix $\boldsymbol{C}_Z$

$$Z \sim \mathcal{N}(\boldsymbol{\mu}_Z, \boldsymbol{C}_Z). \tag{6}$$

SGS takes advantage of this multi-Gaussian property to produce realizations of $Z$. It iteratively visits each node of the grid, computes the kriging estimate and variance error $\sigma_E$ based on previously simulated nodes and samples a value from the corresponding conditional probability distribution. A newly simulated node thus becomes a conditioning node for the next one to be simulated. Mathematically, this can be summarized as

$$Z(\boldsymbol{u}_i) = \sum_{j=1}^{i-1} \lambda_j(\boldsymbol{u}_i) Z(\boldsymbol{u}_j) + \sigma_E(\boldsymbol{u}_i) U(\boldsymbol{u}_i), \quad \forall i = 1, \ldots, n, \tag{7}$$

where $U$ is a standard Gaussian vector used for randomly sampling the conditional distribution and $\lambda_j$ are the kriging weights which define the influence of the conditioning nodes.

In the framework of probability theory, the underlying random phenomenon of $Z(\boldsymbol{u})$ is the standard normal variable $U$ (equation (7)). Indeed, given a specific vector $U_l$, SGS always produces the same realization $z^{(l)}$

$$z^{(l)} = Z(U_l). \tag{8}$$

### 2.3. Algorithm-driven random function (ADRF)

For computational reasons, SGS is commonly used with a limited neighbourhood. As a result, the realizations are altered and the actual simulated RF deviates from the original RF $Z$. In such a case, SGS is sensitive to both the simulation path and the neighbourhood search strategy.

As it is a common situation in geostatistics to have algorithms deviating from the original RF, Boucher (2007) introduced the formalism of an algorithm-driven random function (ADRF): an RF defined by an algorithm which is parametrized by a random variate, or seed number, and a set of parameters. From now on, the term simulated RF will be used to refer to the ADRF simulated by the SGS algorithm. For simplicity, the neighbourhood search strategy is assumed to be defined as the $n$ closest neighbours of the simulated node and is therefore constant for a neighbourhood configuration.

In this context, SGS is a technique which produces realizations of the simulated RF $Z_{p_i}$ defined by

$$z_{p_i}^{(l)} = Z(U_l; p_i) = Z_{p_i}(U_l), \tag{9}$$

where $U_l$ is the random variate of the underlying random process and $p_i$ the simulation path used as a parameter in the simulated RF.

### 2.4. Covariance matrix for error quantification

The errors due to the limited neighbourhood in $Z_{p_i}$ can be fully characterized through the mismatch $\boldsymbol{\varepsilon}$ between the covariance matrix of the simulated RF $\boldsymbol{C}_{Z_{p_i}}$ and the covariance matrix of the target RV $\boldsymbol{C}_Z$ (Emery and Peláez, 2011), hereafter, referred to as the simulated covariance matrix and the model covariance matrix, respectively,

$$\boldsymbol{\varepsilon} = \boldsymbol{C}_{Z_{p_i}} - \boldsymbol{C}_Z \tag{10}$$

The model covariance matrix is computed from the covariance function of the spatial model

$$\boldsymbol{C}_Z(\alpha,\beta) = C(\boldsymbol{u}_\alpha - \boldsymbol{u}_\beta). \tag{11}$$

The simulation covariance matrix $\boldsymbol{C}_{Z_{p_i}}$ can be theoretically evaluated based on the kriging weights and the variance errors. Indeed, Emery and Peláez (2011) showed that equation (7) can be reformulated to extract $U$

$$U = \begin{bmatrix} \frac{1}{\sigma_1} & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ -\frac{\lambda_1^{n-1}}{\sigma_{n-1}} & \cdots & \frac{1}{\sigma_{n-1}} & 0 \\ -\frac{\lambda_1^{n}}{\sigma_n} & \cdots & -\frac{\lambda_{n-1}^{n}}{\sigma_n} & \frac{1}{\sigma_n} \end{bmatrix} Z_{p_i} = \Lambda Z_{p_i}, \tag{12}$$

where $\Lambda$ is constructed during the simulation by storing each kriging weight and variance error. The simulation covariance matrix can be expressed exclusively with $\Lambda$

$$\begin{aligned} \boldsymbol{C}_{Z_{p_i}} &= E\left[\left(Z_p - E[Z_{p_i}]\right)\left(Z_p - E[Z_{p_i}]\right)^T\right] \\ &= E\left[Z_{p_i} Z_{p_i}^T\right] \\ &= E\left[\left(\Lambda^{-1}U\right)\left(\Lambda^{-1}U\right)^T\right] \\ &= \Lambda^{-1} E[UU^T]\left(\Lambda^{-1}\right)^T \\ &= \Lambda^{-1}\left(\Lambda^{-1}\right)^T. \end{aligned} \tag{13}$$

A key advantage of this evaluation of a simulated RF is that it does not rely on realization statistics, but purely on the actual simulated RF equations. The random variate is taken out of the equation, leaving only the deterministic part of the algorithm.

### 2.5. Simulated random function with a randomized path

The purpose of this study is to compare the error of a simulated RF with and without a randomized path, that is, if the simulation path is changed for each realization or kept constant. When the simulation path $p_i$ is randomly sampled among a set of paths $P$, it can be viewed as a random variate rather than a parameter. In this case, a newly simulated RF $Z_P$ can be defined

$$z_P^{(l,i)} = Z_P(U_l, p_i), \quad p_i \in P, \tag{14}$$

where both $U_l$ and $p_i$ are considered as random variates. Yet, it is important to note that, as each realization $z_P^{(l,i)}$ of $Z_P$ is sampled with a unique path $p_i$, it is also a realization of the simulated RF $Z_{p_i}$ where the path $p_i$ is viewed as a parameter rather than a random variate

$$Z_P(U_l, p_i) = z_P^{(l,i)} = z_{p_i}^{(l)} = Z_{p_i}(U_l). \tag{15}$$

The simulation covariance matrix $\boldsymbol{C}_{Z_P}$ can be computed based on $\boldsymbol{C}_{Z_{p_i}}$ using the discrete version of the covariance due to the finite number of realizations. When $n_k$ realizations $\{z_P^{(l,k)}, k = 1, \ldots, n_k\}$ are performed, the empirical covariance at two locations $\boldsymbol{u}_\alpha$ and $\boldsymbol{u}_\beta$ of the grid is given by

$$\begin{aligned} \boldsymbol{C}_{Z_P}(\alpha,\beta) &= \mathrm{Cov}\left[Z_P(\boldsymbol{u}_\alpha), Z_P(\boldsymbol{u}_\beta)\right] \\ &= \frac{1}{n_k}\sum_k \left\{z_P^{(l,k)}(\boldsymbol{u}_\alpha) - E[Z_P(\boldsymbol{u}_\alpha)]\right\}\left\{z_P^{(l,k)}(\boldsymbol{u}_\beta) - E\left[Z_P(\boldsymbol{u}_\beta)\right]\right\}. \end{aligned} \tag{16}$$

Emery and Peláez (2011) showed that the expected value of each simulated RF $Z_{p_i}$ is exactly reproduced for unconditional simulations and for conditional simulations provided that all hard data are present in every kriging system. We thus have

$$E[Z_P(\boldsymbol{u})] = E\left[Z_{p_i}(\boldsymbol{u})\right] = E[Z(\boldsymbol{u})], \quad \forall p_i \in P. \tag{17}$$

Furthermore, as outlined by equation (15), each realization $z_P^{(k,l)}$ has only one path associated with it and can be replaced by $z_{p_k}^{(l)}$.

Combining these two aspects in equation (16) simplifies the covariance matrix of the simulated RF $Z_P$ to an arithmetic average of the covariance matrices of the simulated RFs $Z_{p_k}$ where $p_k \in P$

$$\begin{aligned} \boldsymbol{C}_{Z_P}(\alpha,\beta) &= \frac{1}{n_k}\sum_k \left\{z_{p_k}^{(k)}(\boldsymbol{u}_\alpha) - E\left[Z_{p_k}(\boldsymbol{u}_\alpha)\right]\right\}\left\{z_{p_k}^{(k)}(\boldsymbol{u}_\beta) - E\left[Z_{p_k}(\boldsymbol{u}_\beta)\right]\right\} \\ &= \frac{1}{n_k}\sum_k \boldsymbol{C}_{Z_{p_k}}(\alpha,\beta). \end{aligned} \tag{18}$$

Let $\varepsilon_{\alpha,\beta}^{p_i}$ denote the covariance error of the simulated RF $Z_{p_i}$ between a pair of points $\boldsymbol{u}_\alpha$, $\boldsymbol{u}_\beta$. Using equation (10), we then have

$$\varepsilon_{\alpha,\beta}^{p_i} = \boldsymbol{C}_{Z_{p_i}}(\alpha,\beta) - \boldsymbol{C}_Z(\boldsymbol{u}_\alpha, \boldsymbol{u}_\beta), \tag{19}$$

and hence, using equation (18),

$$\varepsilon_{\alpha,\beta}^{P} = \frac{1}{n_P}\sum_i^{n_P} \boldsymbol{C}_{Z_{p_i}}(\alpha,\beta) - \boldsymbol{C}_Z(\boldsymbol{u}_\alpha, \boldsymbol{u}_\beta) = \frac{1}{n_P}\sum_i^{n_P} \varepsilon_{\alpha,\beta}^{p_i}. \tag{20}$$

Therefore, the errors of a simulated RF with a randomized path $Z_P$ are equal to the average of the errors of all simulated RFs $Z_{p_i}$, such that $p_i \in P$. This is exact for an infinite number $n_P$ of paths in $P$ and is reasonably well approximated if a large number of paths is used. This average operator has two main effects: (1) the expected values of the errors are identical but (2) their variances are reduced proportionally to the number of paths

used.

## 3. Numerical implementation and computational savings

### 3.1. Pseudo-code

The simplest modification of the classical SGS (Algorithm 1) allowing for a constant path consists in moving the loop over the realizations into the loop over the grid nodes, following the computation of the weights (Algorithm 2). This simple permutation of loops illustrates why the simulation path $P$ has to remain the same for all realizations.

**Algorithm 1.** Traditional SGS.

**Input:** Number of realizations $m$, grid size $n$, covariance structure $covar$, neighbourhood size $k$

**Output:** Realizations $R[m, n]$

```
R[m, n]                                          ▷ Initialize empty realizations
for j ← 1 to m do
    P[n] ← GeneratePath(n, seed)                 ▷ Randomly generates path
    for i ← 1 to n do
        id_i ← P(i)                              ▷ Get index of simulated node
        id_k ← Findneighbours(id_i, P(1 to i − 1), k)
        λ, σ_E ← ComputeWeight(id_i, id_k, covar)
        R(j, id_i) ← SimulateValue(λ, σ_E, R(j, id_k), seed)
    end
end
```

**Algorithm 2.** SGS with constant path.

**Input:** Number of realizations $m$, grid size $n$, covariance structure $covar$, neighbourhood size $k$

**Output:** Realizations $R[m, n]$

```
R[m, n]
P[n] ← GeneratePath(n, seed)                     ▷ Generates a single path
for i ← 1 to n do
    id_i ← P(i)
    id_k ← Findneighbours(id_i, P(1 to i − 1), k)
    λ, σ_E ← ComputeWeight(id_i, id_k, covar)
    for j ← 1 to m do
        R(j, id_i) ← SimulateValue(λ, σ_E, R(j, id_k), seed)
    end
end
```

### 3.2. Parallelization

Parallelization is easily implemented on the traditional SGS by sending each realization to a different core, which corresponds to parallelizing the loop of line 1 in Algorithm 1. In the new implementation, the realization loop can also be parallelized (line 2 in Algorithm 2). Yet, a more efficient implementation is described in Algorithm 3, where the weights are computed and stored prior to the simulation. An important advantage of separating the computation of the weights and the simulation of the nodes is the ability to compute the weights in parallel. In practice, this is implemented by searching for neighbours with a lower path index than that of the simulated node, instead of searching for all existing values in the realization. Note that this parallelization strategy is not as easily implemented in the traditional SGS algorithm because the simulation of the node is in the same loop and requires the value of the neighbours. Finally, a third option is to parallelize the whole algorithm so that each core computes a subset of realizations, each with a different constant path. This hybrid solution between randomized and constant path does not combine the computational benefits of parallelization and constant path, but improves realization accuracy as is later discussed in the paper.

**Algorithm 3**. Parallelized SGS with constant path.

**Input:** Number of realizations $m$, grid size $n$, covariance structure $covar$, neighbourhood size $k$

**Output:** Realization $R[m,n]$

1 $\Lambda[n,k]$ ▷ Initialize the kriging weights array

2 $\Sigma_E[n]$ ▷ Initialize the kriging variance error array

3 $ID_k[n,k]$ ▷ Initialize neighbours indices array

4 $P[n] \leftarrow$ `GeneratePath`$(n,$ `seed`$)$

5 **parfor** $i \leftarrow 1$ **to** $n$ **do**

6 $\quad id_i \leftarrow P(i)$

7 $\quad ID_k(i) \leftarrow$ `Findneighbours`$(id_i,\ P(1 \text{ to } i-1),\ k)$

8 $\quad \Lambda(i), \Sigma_E(i) \leftarrow$ `ComputeWeight`$(id_i,\ ID_k(i),\ covar)$

9 **end**

10 $R[m,n]$

11 **parfor** $j \leftarrow 1$ **to** $m$ **do**

12 $\quad$ **for** $i \leftarrow 1$ **to** $n$ **do**

13 $\quad\quad id_i \leftarrow P(i)$

14 $\quad\quad R(j,id_i) \leftarrow$ `SimulateValue`$(\Lambda(i), \Sigma_E(i), R(j, ID_k(i)),$ `seed`$)$

15 $\quad$ **end**

16 **end**

### 3.3. Memory requirements

Compared to the traditional approach (Algorithm 1), the serial implementation (Algorithm 2) does not require additional memory. However, the parallel implementation (Algorithm 3) has to store the indices of the neighbours $ID_k[n,k]$, the kriging weights $\Lambda[n,k]$, and the kriging variance errors $\Sigma_E[n]$, which results in a memory increase of $4nk + 8nk + 8n = (12k + 8)n$ bytes. In comparison, storing $m$ realizations requires $8nm$ bytes. Thus, reducing $m$ by $1.5k$ compensates the memory increase, which is not a problem for typical applications where $m \gg k$.

In simulations with large grids and/or large numbers of realizations, memory can still become an important issue. In the traditional implementation (Algorithm 1), this was handled by writing each realization or group of realizations on the disk when completed. In the parallelized implementation (Algorithm 3), the realizations can similarly be written on the disk in the realization loop (line 3). If the neighbourhood size $k$ is large, storing the weights $\Lambda$ and the neighbours indexes $ID_k$ can become a challenge. The solution for such situations is to write them during the first loop (line 3) and read them in the realization loop (line 3) but this can increase the computation time. Note that, in most typical situations, using a constant path remains more efficient because the computation of the weights is more expensive than writing and reading a file. Moreover, simulations with such settings are usually performed on high-performance computers, which have large amounts of memory

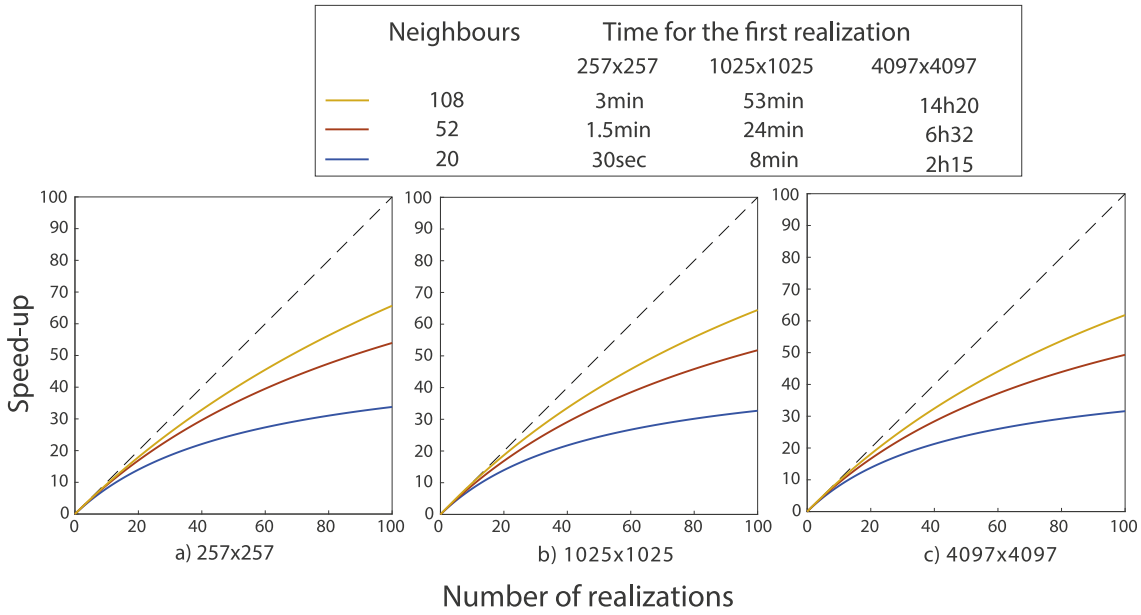


**Fig. 1.** Speed-up as a function of the number of realizations performed for 3 different grid sizes and neighbourhood sizes. The grid sizes were selected such that the multi-grid path is optimal and the neighbourhood size such that the resulting full neighbourhood is symmetric. The computation time for the first realization is also shown in the legend.

available and thus are able to store at least the weights and indices. Despite not storing the weights and indices, the serial implementation (Algorithm 2) cannot be efficiently used for simulations with a large grid and/or large numbers of realizations.

### 3.4. Computation time

With the constant path approach, the only operation left to be repeated for each realization is to iterate through all the nodes and simulate a value, which corresponds to a simple vector multiplication (line 3 in Algorithm 3). As such, with an ideal implementation, the gain in computation time corresponds to the effort of finding the neighbours (line 3 in Algorithm 3), and computing the kriging weights and variance errors (line 3 in Algorithm 3) for each additional realization $j = \{2, \ldots, m\}$. In the following paragraphs, these two computational efforts are described in more detail to better understand the conditions and, the extent of the benefits of using a constant path.

First, the kriging neighbours are found by posing the optimization problem known as $k$-Nearest Neighbour (k-NN). The optimal solution to this problem varies with the simulation parameters: (1) grid size, (2) neighbourhood size, (3) number and location of hard data, and (4) covariance model range. Because of this diversity of settings, the choice of an appropriate neighbourhood search strategy is of upmost importance and it often is the bottleneck with regard to computation time in SGS. The most common strategies in SGS are listed below. The "exhaustive search" sorts all $n$ available nodes by their distance to the simulated node and takes the first $k$ nodes. The complexity of the well-known quicksort algorithm is $O(n\log(n))$ (Hoare, 1962). A considerable improvement of this strategy consists of only sorting the first $k$ nodes, thus reducing the complexity to $O(n + k\log(k))$ with partial quicksort (Martínez, 2004). This strategy is optimal for simulations of a small grid with a large neighbourhood size and presents the advantage of being able to treat equally data at any location. The "spiral search" (Deutsch and Journel, 1992) visits each node of the grid by order of proximity to the simulated node, skips the empty nodes, and stops once $k$ simulated nodes have been found. This method is efficient for large grids with a small neighbourhood, particularly when combined with a multi-grid path. Hard data have to be moved to the closest node, at least temporarily. Another trick is to define an exploration distance threshold to limit the search to the nodes within a certain distance to the simulated one. The "superblock search" (Journel and Huijbregts, 1978; Deutsch and Journel, 1992) and "tree-based search" (Hassanpour and Leuangthong, 2006; Manchuk and Deutsch, 2012) provide alternative solutions, which can be more efficient, especially for the simulation of irregular points or when dealing with hard data within a two-part search method (Deutsch and Journel, 1992).

Secondly, the time to compute the kriging weights and variance errors is mainly determined by solving kriging systems, one for each simulated node. This has an overall computational complexity of $O(nk^3)$ (Dimitrakopoulos and Luo, 2004), making the neighbourhood size $k$ the main parameter influencing the computation time. Another common solution to reduce the computational cost in regular grids is to use a covariance lookup table, which pre-computes and stores the covariance of all pairs of nodes within the search ellipsoid.

The total computational savings provided by the constant path are the sum of these two computational costs multiplied by the number of realizations required, such that implementing the constant path approach is only rewarding when several realizations are needed. In addition, the constant path allows for a more flexible choice of parameters because the computational cost associated with these parameters is only paid once. Therefore, using a sub-optimal neighbourhood strategy does not increase the computational cost as much as it would in the traditional approach. This suggest that a good strategy is to increase the neighbourhood size to reduce the simulation error without excessively increasing the computational burden.

The computational benefit of using a constant path is numerically assessed with the speed-up, that is, the ratio of computation time of the traditional SGS $T^{trad}$ over the constant path SGS $T^{cst}$. It can be expressed as a function of the number of realizations $m$ based on the simulation of a single realization,

$$S = \frac{T^{trad}}{T^{cst}} = \frac{mT^{trad}(1)}{T^{cst}(1) + (m-1)T^{cst}_{real}(1)}, \tag{21}$$

where $T^{cst}_{real}(1)$ refers to the time spent in the realizations loop for a single realization (lines 11–16 in Algorithm 3). Fig. 1 shows the speed-up for unconditional simulations performed with a spherical covariance function having a range of 20, using a spiral search and a multi-grid path, and without covariance lookup table. SGS with constant path is simulated with the parallel implementation (Algorithm 3) but running on a single

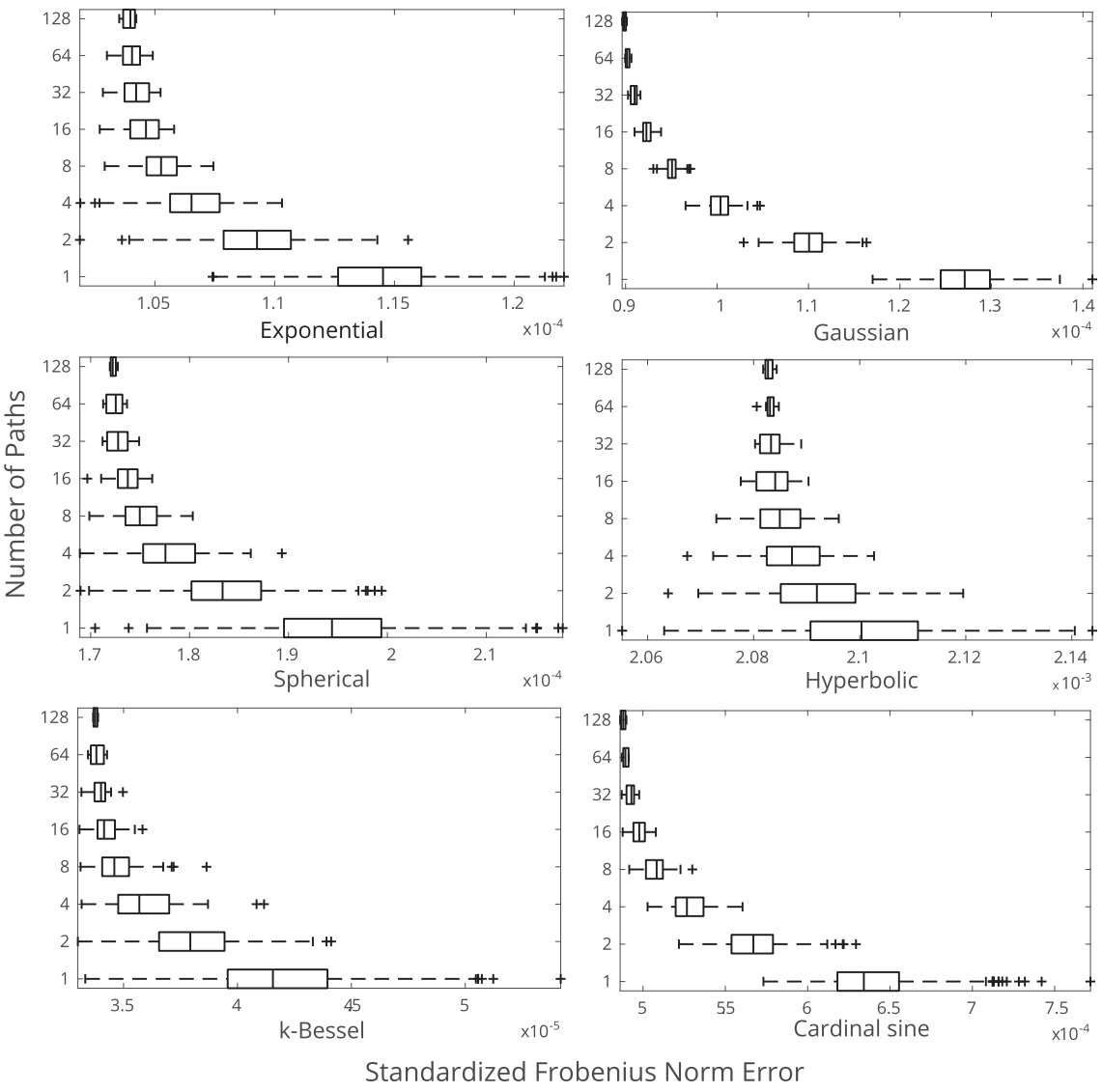


**Fig. 2.** Boxplots of the SFN of simulated RFs $Z_P$ for different numbers of simulation paths $n_P = 1, \ldots, 128$. 512 simulated RFs $Z_{p_i}$ were computed and different numbers of them combined to construct the simulated RFs with a fully randomized path $Z_P$ according to equation (20).

core. The processor used is a AMD Opteron (2300 Mhz). These results show that the speed-up increases with the neighbourhood size and scales well with grid size. In this particular case, around 3′000, 5′000 and 6′000 realizations can be performed with a constant path for the same duration than 100 realizations without constant path for neighbourhood sizes of 20, 52 and 108 nodes, respectively. This case study reveals that, even if only 5 realizations are needed, it is computationally more efficient to generate them with a constant path and a neighbourhood of 108 nodes than without constant path and a neighbourhood of 20 nodes.

## 4. Simulation errors

In the following, six covariance functions (exponential, Gaussian, spherical, hyperbolic, *k*-Bessel, and cardinal sine) are considered because of their variability close to the origin and near the range as well as is their popularity. A correlation range of 20 nodes normalized by their integral value is used. Because of the large cost of computing the exact covariance matrix, a small grid of $65 \times 65$ nodes is used in this section.

### 4.1. Covariance errors

In order to compare large covariance matrices, Emery and Peláez (2011) proposed to aggregate the covariance errors with the standardized Frobenius norm (SFN)

$$\eta^{p_i} = \frac{\left\| \boldsymbol{C}_{Z_{p_i}} - \boldsymbol{C}_Z \right\|}{\boldsymbol{C}_Z}, \tag{22}$$

where $\| \cdot \|$ denotes the $L_{2,2}$ or Frobenius norm. Limitations associated with this metric are discussed in section 5.3.

With this single value, it is possible to compare the error of simulated RFs using different numbers of paths $n_P$ as well as different covariance function types. The SFNs of 512 simulated RFs with a constant path $Z_{p_i}$ were computed. Then, using equation (20), the simulated RFs with a randomized path $Z_P$, with $n_P = 2, \ldots, 128$ were computed. Fig. 2 shows the corresponding results in the form of boxplots. Simulations were performed with a neighbourhood size of 20 nodes and a fully randomized path.

For all covariance function types, a decrease of the mean and variance of the SFN is observed for simulated RFs using larger numbers of paths. The reduction of the variance is an expected result of the averaging described in equation (20). But the decrease of the mean error is due to the aggregation of errors in the SFN. This will be further explored in the

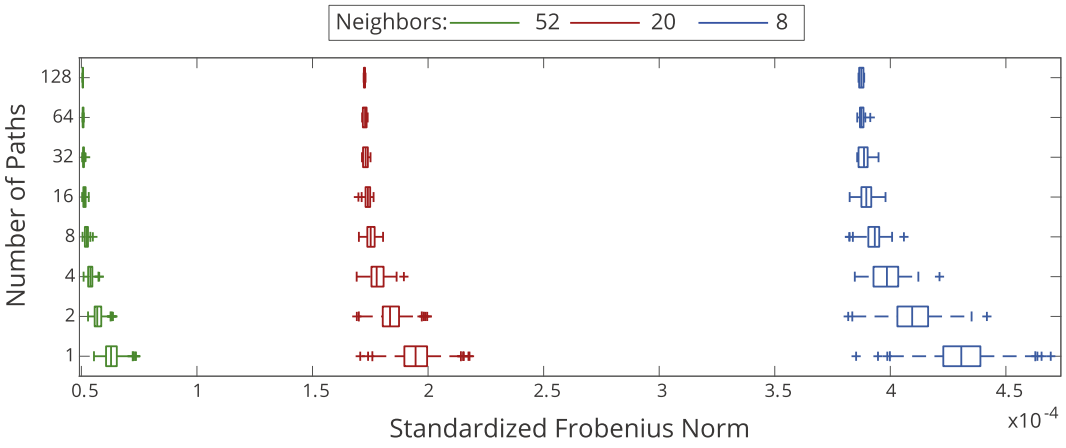


**Fig. 3.** Boxplot of the SFN for several simulated RFs for different numbers of simulation paths and different neighbourhood sizes for a spherical covariance function.

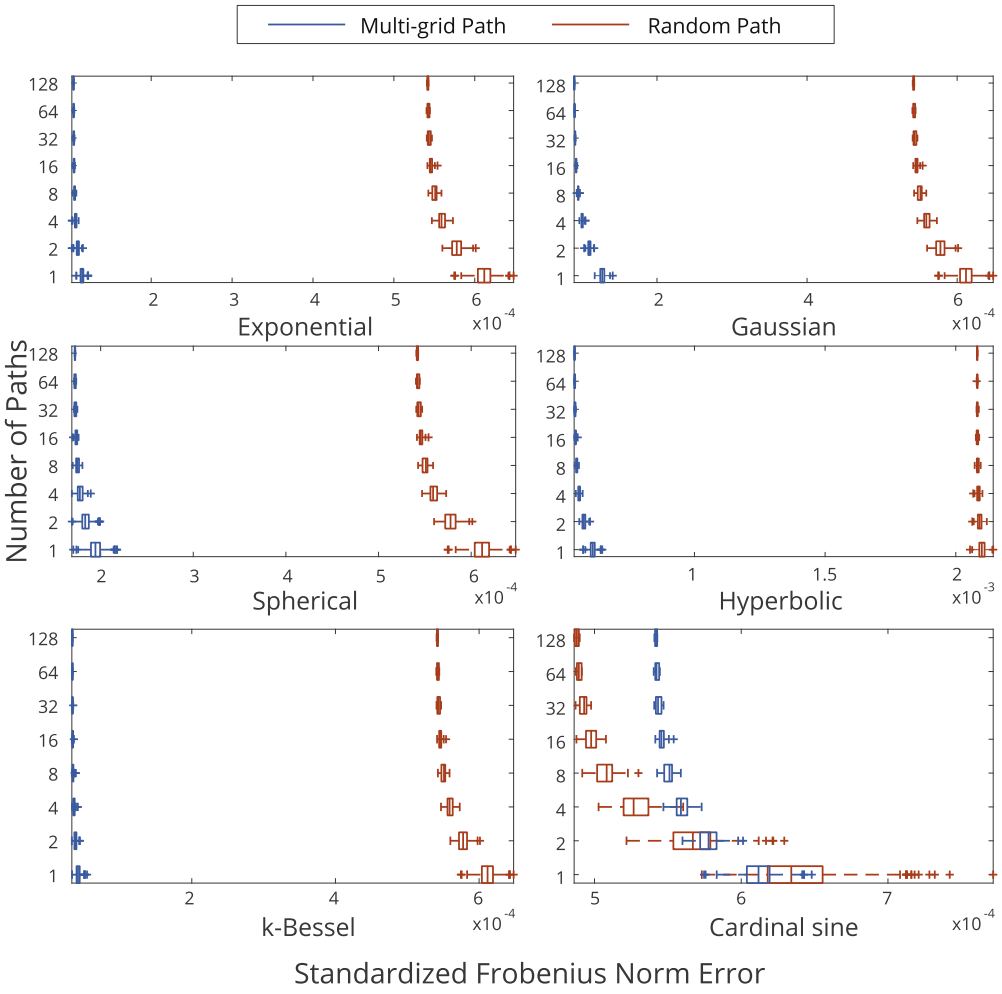


**Fig. 4.** Boxplot of the SFN for several simulated RFs using different numbers of simulation paths for multi-grid or random simulation paths.

discussion part. It can also be noted that the value of the SFN rapidly converges and, in this specific case, with more than approximately 16 different paths no significant error reduction is observed. The magnitude of the error reduction is strongly linked to the shape of covariance function where, for instance, the Gaussian or cardinal sine present larger reductions than hyperbolic or exponential covariance functions.

### 4.2. *Sensitivity to neighbourhood and path type*

To provide a perspective on these error reductions obtained by

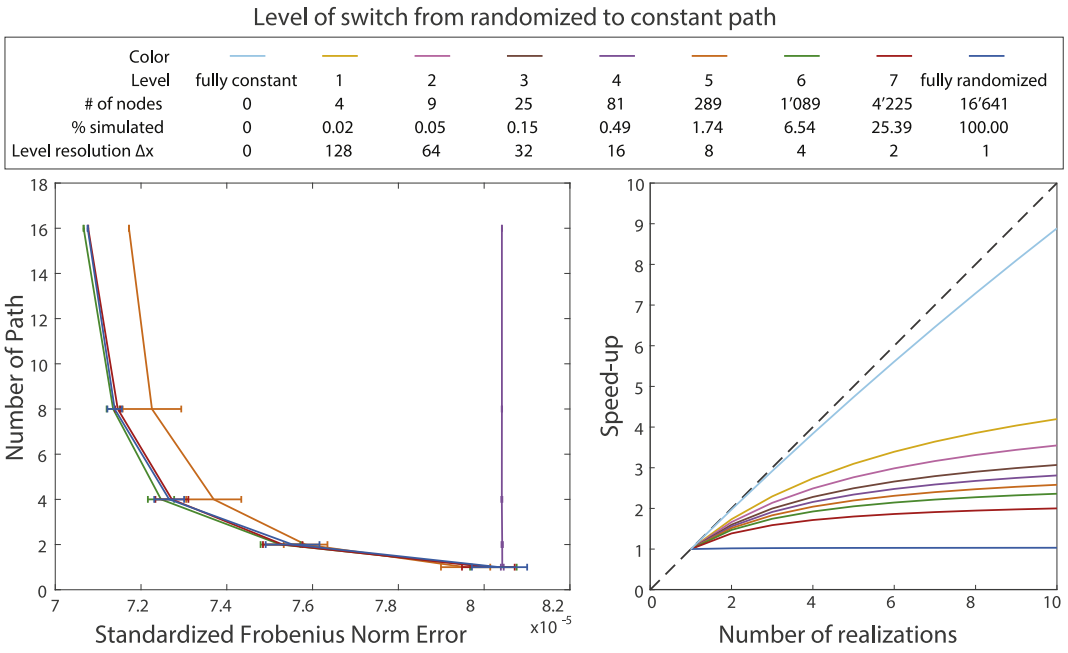


**Fig. 5.** (Left) Mean and standard deviation of the SFN using up to 16 different randomized multi-grid paths for simulations where the constant path approach is switched on at different grid level. Note that the curves for levels 1–4 and for the fully constant path are superimposed. (Right) Corresponding speed-up of simulations.

varying the path, we compare them to those obtained for an increase in neighbourhood size and the use of different types of paths.

Firstly, the influence of neighbourhood size is analyzed. Fig. 3 shows the SFN for simulated RFs for a variable number of paths and several neighbourhood sizes. The number of neighbours used in the kriging has a stronger influence on the error than varying the path. Only the simulated RF for a spherical covariance function is shown because the results for all other covariance function types considered in this study are qualitatively similar. These results reinforce the suggestion made in section 3.4 to take advantage of the computational savings associated with the use of a constant path to increase the neighbourhood size. Indeed, a much larger reduction of covariance errors is achieved by increasing the neighbourhood size than by using different randomized paths.

Secondly, two different path types are compared: a random path which visits each cell of the grid equiprobably and a multi-grid path using a series of nested grids to guide the order of the simulation. More specifically, the first grid simulates the four corners of the final grid, then each subsequent grid doubles the previous grid resolution, re-using the previously simulated nodes and simulating the empty nodes randomly. Note that both random and multi-grid paths can be randomized. Fig. 4 compares the SFN of simulated RFs with random and multi-grid paths for different numbers of randomized paths.

As already shown in Nussbaumer et al. (2017), Fig. 4 illustrates that a multi-grid path generally improves the reproduction of the covariance matrix. However, it also shows that using multiple randomized multi-grid paths only reduces slightly both the magnitude of the SFN and its variability. Note that the cardinal sine covariance function presents an interesting exception where increasing the number of randomized paths results in greater SFN reduction with a random path than with a multi-grid path. This can be explained by the fact that when few neighbours are available the reproduction of the covariance function can follow a non-linear behaviour due to the cyclical nature of this covariance function. Fig. 10 in Nussbaumer et al. (2017) shows that the type of path has a limited effect on the cardinal sine. With this particular covariance function, the neighbourhood size is the most important parameter, as illustrated by Fig. 9 in Nussbaumer et al. (2017).

A compelling asset of the multi-grid path is that it allows for switching from either randomized to constant path or vice versa at each grid level. This feature is attractive because a randomized path at the initial coarse levels maximizes the variability of the poorly constrained nodes. Conversely, a constant path at the subsequent finer levels allows for important computational gains while not compromising the variability, because these nodes are anyway well constrained by the coarser grid. Fig. 5 shows the SFNs and corresponding speed-ups for simulations starting with a randomized path and switching to constant path at differents multi-grid levels. The simulations were performed on a $129 \times 129$ grid with a spherical covariance function and 20 neighbours.

Switching to a constant path during levels 1 to 4 results in similar SFN values than with a fully constant path for any number of paths. This means that changing the order of simulated nodes belonging to these levels does not result in any measurable improvement in the reproduction of the final covariance matrix. In this case study, switching path at level 5 results in a relatively limited error reduction when several randomized paths are used. Switching path at the following levels has little or no impact on the reproduction of the covariance matrix, which can be explained by the grid spacing of these levels being smaller than the covariance function range. Indeed, during the simulation of those levels, the simulation of nodes is well constrained by the previous multi-grids. In all cases, the magnitude of the errors is small, even when using fully constant path, as shown in Fig. 3. As a general rule, multi-grid levels with a grid spacing much lower than the covariance range can be simulated with a constant path without affecting the SFN values. The corresponding speed-up indicates that using a constant path on the last levels of the multi-grid provides the largest computational improvement due to the large number of nodes.

### 4.3. Conditional simulation

For conditional simulations, the constant path strategy can be implemented in the same way as for the unconditional case by computing once and storing the kriging weights associated with the hard data for all simulated nodes and then re-using them for each additional realization.

Conditional simulations can be considered as a special case of unconditional simulations, for which the first nodes of the path are the hard data with a constant outcome and whose sampling outcome is constant. Emery and Peláez (2011) show that the assessment of a conditional simulation is based on the reproduction of the covariance matrix of the simulated nodes and the reproduction of the expected field. They

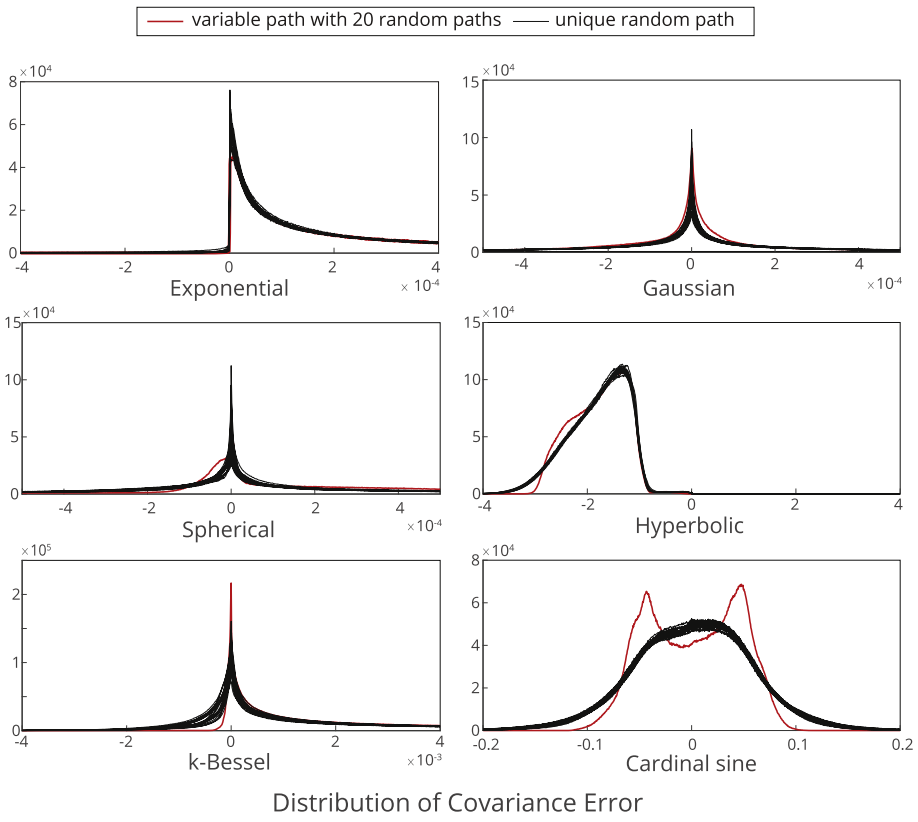


**Fig. 6.** Distributions of all values $\varepsilon_{\alpha,\beta}$ (equation (19)) of 20 simulated RFs with a constant path (black lines) and of a simulated RF with a randomized path (red lines). (For interpretation of the references to colour in this figure legend, the reader is referred to the Web version of this article.)

demonstrate that the covariance matrix of the simulated nodes is similar to that of the unconditional case (Equation (11) in Emery and Peláez (2011)). The exact reproduction of the expected value, which corresponds to matching the kriging predictor, is only achieved when all hard data are used in every kriging system (Appendix 1 in Emery and Peláez (2011)). In this case, using a constant or randomized path has the same benefits and consequences as it does for unconditional simulations where a constant path is enforced on the first nodes equivalent to the hard data.

However, if too many hard data are present, not all can be retained in every kriging system. In this case, the expected field does not match the kriging predictor, and, in turn, Equation (17) does not hold. The consequences on the effects of using a constant path are difficult to assess in general. However, if hard data are included more generously with a two part search strategy, it can be expected that the effect of the missing hard data is limited. Consequently, from a qualitative point view, the overall results of this study remain valid.

Another effect of including hard data is that the presence of numerous and scattered hard data constrains the simulation, such that, randomizing the path results in a smaller reduction of errors. An equivalent effect can be observed in Fig. 5, where simulations switching from a randomized to constant path for the last levels of the multi-grid produce similar errors reductions.

## 5. Discussion

### 5.1. Unlucky path

An unlucky path is a particular path which leads to especially bad covariance matrix reproduction. These paths can be discussed by comparing the tails of the errors distribution in Figs. 2–4. With the reduction of variance demonstrated in equation (20), maximal SFN values in Fig. 2 also decrease with a largest number of simulation paths used. Based on Figs. 3 and 4, increasing the neighbourhood size or using a multi-grid approach reduces even further the occurrence of unlucky paths.

### 5.2. Empirical covariance

In this section, the assessment of error based on aggregating multiple realizations errors with empirical inter-realization statistics is discussed.

Firstly, even if the covariance error of several realizations is small, it does not mean that each realization has a smaller error. Indeed, each realization could contain large errors, but averaging these errors may lead to small inter-realization errors. This is an important consideration for our comparison because each realization of a simulation RF with a randomized path is produced with a single path and, therefore, with the same potential errors as a realization of a simulation RF with a constant path. This means that although a simulation RF with a randomized path can improve the overall inter-realizations statistics, each individual realization still contains a similar amount of errors.

Secondly, when covariance errors are empirically computed, the number of realizations is limited and the ergodic fluctuations should be taken into account. In theory, for equation (18) to be correct, an infinite number of realizations of each constant path simulation is required. As the path changes for each realization, discrepancies between the simulation covariance and the model covariance are expected (Matheron, 1989; Emery, 2004).

### 5.3. Aggregation of the covariance matrix errors

While the SFN provides a reliable and simple measure of error (equation (22)), there are also some limitations that need to be highlighted. Essentially, the SFN can be viewed as a normalized root mean square error (RMSE) measure

$$\eta^{p_i} = \frac{\sum_{\alpha}^{N}\sum_{\beta}^{N}\left(\varepsilon_{\alpha,\beta}^{p_i}\right)^2}{\sum_{\alpha}^{N}\sum_{\beta}^{N}\left(C_Z(\boldsymbol{u}_\alpha - \boldsymbol{u}_\beta)\right)^2}, \tag{23}$$

which has the well-known effect of enhancing the influence of larger errors. Thus, a few pairs of nodes with large covariance errors are predominant over many small errors. While this is a commonly accepted procedure, a thorough investigation of the effect of varying the path requires analysis of the exact distribution of the covariance errors. Fig. 6 displays the distribution of all values of $\varepsilon_{\alpha,\beta}$ (equation (19)) of 20 simulated RFs with a constant path and the simulated RF with a randomized path. The latter is constructed by combining the previous 20 simulated RFs according to equation (20).

Each covariance function type produces different distributions, but in each of them, the 20 simulated RFs with a constant path generate similar distributions. In comparison, the simulated RF with a randomized path generally has a smaller number of covariance pairs with large and small errors. However, the number of intermediate errors is increased. This can be explained by the averaging of the covariance error for each pair of nodes in equation (20). Indeed, averaging tends to reduce extreme values and to increase the medium-range values. This means that, although extreme errors are avoided, the number of pairs of nodes with no errors is also reduced. Therefore, if instead of looking at the root mean square error, the number of correct pairs of nodes was computed, a simulated RF with a constant path would produce a more favourable result.

## 6. Conclusions

This study analyzed in detail the benefits and limitations of using a constant simulation path in SGS. Compared to the traditional implementation of SGS, the use of a constant path provides computational gains of several orders of magnitude. While randomizing the path among realizations slightly reduces the covariance errors, this reduction is only significant for a limited number of randomized paths. Most importantly, our analyses demonstrated that, for the majority of our simulations, these error reductions were relatively small in comparison to the effects of increasing the neighbourhood size or, to a lesser extent, using a multi-grid path. The optimal approach with regard to computational efficiency and statistical fidelity is to switch from a randomized path to a constant path in the course of a multi-grid path. The timing of this switch should correspond to a grid spacing smaller than the covariance range. However, as this solution can be rather tedious to implement and given the relatively limited gains it provides in terms of error reduction, our recommendation is to use a fully constant path and to employ the associated computational gains to increase the neighbourhood size. The constant path approach can readily be extended to other sequential simulation methods as long as some computationally expensive operations are the same for all realizations. In order to be common, such operations have to be independent of the data values, such as, for instance, the computation of the kriging weights or the neighbourhood search. This includes methods such as Sequential Cosimuation, Sequential Indicator Simulation (Isaaks, 1984) or Direct Sequential Simulation (Billings et al., 2002).

## Acknowledgments

This work has been supported by a grant from the Swiss National Science Foundation.

## References

Banerjee, S., Gelfand, A.E., Finley, A.O., Sang, H., sep 2008. Gaussian predictive process models for large spatial data sets. J. Roy. Stat. Soc. B 70 (4), 825–848. http://doi.wiley.com/10.1111/j.1467-9868.2008.00663.x.

Barry, R.P., Kelley Pace, R., 1997. Kriging with large data sets using sparse matrix techniques. Commun. Stat. Simulat. Comput. 26 (2), 619–629.

Billings, S.D., Beatson, R.K., Newsam, G.N., nov 2002. Interpolation of geophysical data using continuous global surfaces. Geophysics 67 (6), 1810–1822. http://library.seg.org/doi/10.1190/1.1527082. http://library.seg.org/doi/10.1190/1.1527081.

Boisvert, J.B., Deutsch, C.V., 2011. Programs for kriging and sequential Gaussian simulation with locally varying anisotropy using non-Euclidean distances. Comput. Geosci. 37 (4), 495–510. https://doi.org/10.1016/j.cageo.2010.03.021.

Boucher, A., 2007. Random Function : a New Formalism for Applied Geostatistics. Tech. rep. Stanford University.

Cáceres, A., Emery, X., Godoy, M., 2010. Speeding up Conditional Simulation: Using Sequential Gaussian Simulation with Residual Substitution. Tech. rep.

Chilès, J.-P., Delfiner, P., mar 1999. Geostatistics. Vol. 497 of Wiley Series in Probability and Statistics. John Wiley & Sons, Inc., Hoboken, NJ, USA. http://doi.wiley.com/10.1002/9781118136188. http://doi.wiley.com/10.1002/9780470316993.

Cressie, N., Johannesson, G., jan 2008. Fixed rank kriging for very large spatial data sets. J. Roy. Stat. Soc. B 70 (1), 209–226. http://doi.wiley.com/10.1111/j.1467-9868.2007.00633.x.

Deutsch, C.V., Journel, A.G., 1992. GSLIB: Geostatistical Software Library and User's Guide. Oxford University Press, New York.

Dimitrakopoulos, R., Luo, X., jul 2004. Generalized sequential gaussian simulation on group size and screen-effect approximations for large field simulations. Math. Geol. 36 (5), 567–591. http://link.springer.com/10.1023/B: MATG.0000037737.11615.df.

Emery, X., dec 2004. Testing the correctness of the sequential algorithm for simulating Gaussian random fields. Stoch. Environ. Res. Risk Assess. 18 (6), 401–413. http://link.springer.com/10.1007/s00477-004-0211-7.

Emery, X., Peláez, M., sep 2011. Assessing the accuracy of sequential Gaussian simulation and cosimulation. Comput. Geosci. 15 (4), 673–689. http://link.springer.com/10.1007/s10596-011-9235-5.

Furrer, R., Genton, M.G., Nychka, D., sep 2006. Covariance tapering for interpolation of large spatial datasets. J. Comput. Graph Stat. 15 (3), 502–523. http://www.tandfonline.com/doi/abs/10.1198/106186006X132178.

Gómez-Hernández, J.J., Journel, A.G., 1993. Geostatistics Tróia '92. Vol. 5 of Quantitative Geology and Geostatistics. Springer Netherlands, Dordrecht. http://link.springer.com/10.1007/978-94-011-1739-5.

Goovaerts, P., 1997. Geostatistics for Natural Resources Evaluation. Oxford University Press.

Gribov, A., Krivoruchko, K., 2004. Geostatistical mapping with continuous moving neighborhood. Math. Geol. 36 (2), 267–281. http://link.springer.com/10.1023/B:MATG.0000020473.63408.17.

Hartman, L., Hössjer, O., jan 2008. Fast kriging of large data sets with Gaussian Markov random fields. Comput. Stat. Data Anal. 52 (5), 2331–2349. http://linkinghub.elsevier.com/retrieve/pii/S0167947307003568.

Hassanpour, R.M., Leuangthong, O., 2006. On the Use of a Quadtree Search for Estimation and Simulation, 1–11.

Hoare, C.A.R., jan 1962. Quicksort. Comput. J. 5 (1), 10–16. https://academic.oup.com/comjnl/article-lookup/doi/10.1093/comjnl/5.1.10.

Isaaks, E.H., 1984. Indicator simulation: application to the simulation of a high grade uranium mineralization. In: Verly, G.W. (Ed.), Geostatistics for Natural Resources Characterization. Part 2. D. Reidel Publishing, pp. 1057–1069.

Isaaks, E.H., Srivastava, R.M., 1989. An Introduction to Applied Geostatistics. Oxford University Press, New York.

Journel, A.G., Huijbregts, C.J., 1978. Mining Geostatistics. Academic press, London.

Kammann, E.E., Wand, M.P., 2003. Geoadditive models. Appl. Sci. 52 (1), 1–22.

Manchuk, J.G., Deutsch, C.V., 2012. Implementation aspects of sequential Gaussian simulation on irregular points. Comput. Geosci. 16 (3), 625–637.

Mariethoz, G., jul 2010. A general parallelization strategy for random path based geostatistical simulation methods. Comput. Geosci. 36 (7), 953–958. http://linkinghub.elsevier.com/retrieve/pii/S0098300410000774.

Martínez, C., 2004. Partial quicksort. In: 6th ACM-SIAM Workshop on Algorithm Engineering and Experiments and 1st ACM-SIAM Workshop on Analytic Algorithmics and Combinatorics. p. 5.

Matheron, G., 1965. Les Variables Régionalisées et Leur Estimation. Masson et Cie, Paris.

Matheron, G., 1989. Estimating and Choosing. Springer Berlin Heidelberg, Berlin, Heidelberg. http://link.springer.com/10.1007/978-3-642-48817-7.

Memarsadeghi, N., Mount, D.M., 2007. Efficient implementation of an optimal interpolator for large spatial data sets. In: Computational ScienceICCS 2007. Springer, pp. 503–510. http://link.springer.com/chapter/10.1007/978-3-540-72586-2_74. http://link.springer.com/10.1007/978-3-540-72586-2_74.

Memarsadeghi, N., Raykar, V.C., Duraiswami, R., Mount, D.M., 2008. Efficient kriging via fast matrix-vector products. In: IEEE Aerospace Conference Proceedings.

Meyer, T.H., 2004. The discontinuous nature of kriging interpolation for digital terrain modeling. Cartogr. Geogr. Inf. Sci. 31 (4), 209–216. http://www.tandfonline.com/doi/abs/10.1559/1523040042742385.

Nunes, R., Almeida, J.a, aug 2010. Parallelization of sequential Gaussian, indicator and direct simulation algorithms. Comput. Geosci. 36 (8), 1042–1052. http://linkinghub.elsevier.com/retrieve/pii/S0098300410001627.

Nussbaumer, R., Mariethoz, G., Gloagen, E., Holliger, K., Aug 2017. Which path to choose in sequential gaussian simulation. Math. Geosci. Issn: 1874-8953 http://link.springer.com/10.1007/s11004-017-9699-5.

Omre, H., Sølna, K., Tjelmeland, H., 1993. Simulation of random functions on large lattices. In: Soares, A. (Ed.), Geostatistics Tròia '92. Kluwer Academic Publishers, Dordrecht, pp. 179–199. http://www.springerlink.com/index/10.1007/978-94-011-1739-5_16.

Pebesma, E.J., Wesseling, C.G., jan 1998. Gstat: a program for geostatistical modelling, prediction and simulation. Comput. Geosci. 24 (1), 17–31. http://linkinghub.elsevier.com/retrieve/pii/S0098300497000824.

Rasera, L.G., Machado, P.L., Costa, J.F.C.L., 2015. A conflict-free, path-level parallelization approach for sequential simulation algorithms. Comput. Geosci. 80, 49–61. https://doi.org/10.1016/j.cageo.2015.03.016.

Rivoirard, J., Romary, T., 2011. Continuity for kriging with moving neighborhood. Math. Geosci. 43 (4), 469–481. http://link.springer.com/10.1007/s11004-011-9330-0.

Rue, H., Tjelmeland, H., mar 2002. Fitting gaussian Markov random fields to gaussian fields. Scand. J. Stat. 29 (1), 31–49. http://doi.wiley.com/10.1111/1467-9469.00058.

Sakata, S., Ashida, F., Zako, M., 2004. An efficient algorithm for kriging approximation and optimization with large-scale sampling data. Comput. Meth. Appl. Mech. Eng. 193 (3–5), 385–404.

Srinivasan, B.V., Duraiswami, R., Murtugudde, R., 2008. Efficient kriging for real-time spatio-temporal interpolation Linear kriging. In: 20th Conference on Probablility and Statistics in Atmospheric Sciences, 228–235.

Trefethen, L.N., Bau III, D., 1997. Numerical Linear Algebra, vol. 50. SIAM.

Vargas, S.H., Caetano, H., Filipe, M., 2007. Parallelization of sequential simulation procedures. In: EAGE Conference on Petroleum Geostatistics. http://www.earthdoc.org/publication/publicationdetails/?publication=7859.

Verly, G.W., 1993. Sequential gaussian cosimulation: a simulation method integrating several types of information. In: Soares, A. (Ed.), Geostatistics Tròia '92. Kluwer Academic Publishers, pp. 543–554. http://www.springerlink.com/index/10.1007/978-94-011-1739-5_42.