Accelerating simulation for the multiple-point statistics algorithm using vector quantization

Chen Zuo, Zhibin Pan,[*] and Hao Liang
School of Electronic and Information Engineering, Xi'an Jiaotong University, Xi'an 710049, China

(Received 20 April 2017; revised manuscript received 11 February 2018; published 2 March 2018)

Multiple-point statistics (MPS) is a prominent algorithm to simulate categorical variables based on a sequential simulation procedure. Assuming training images (TIs) as prior conceptual models, MPS extracts patterns from TIs using a template and records their occurrences in a database. However, complex patterns increase the size of the database and require considerable time to retrieve the desired elements. In order to speed up simulation and improve simulation quality over state-of-the-art MPS methods, we propose an accelerating simulation for MPS using vector quantization (VQ), called VQ-MPS. First, a variable representation is presented to make categorical variables applicable for vector quantization. Second, we adopt a tree-structured VQ to compress the database so that stationary simulations are realized. Finally, a transformed template and classified VQ are used to address nonstationarity. A two-dimensional (2D) stationary channelized reservoir image is used to validate the proposed VQ-MPS. In comparison with several existing MPS programs, our method exhibits significantly better performance in terms of computational time, pattern reproductions, and spatial uncertainty. Further demonstrations consist of a 2D four facies simulation, two 2D nonstationary channel simulations, and a three-dimensional (3D) rock simulation. The results reveal that our proposed method is also capable of solving multifacies, nonstationarity, and 3D simulations based on 2D TIs.

DOI: 10.1103/PhysRevE.97.033302

I. INTRODUCTION

Geological modeling is fundamental to various research topics. Prior to performing a spatial analysis, it is necessary to obtain a set of accurate models that describe the geological phenomenon of interest. Stochastic simulation algorithms provide a feasible way to fulfill such modeling. In general, simulation realizations not only comply with the basic physical law but also honor the available data sampled from reality. In addition, these simulation realizations should have a sufficient variability in order to reflect the uncertainty.

Since its early years, stochastic simulation algorithms have been composed of two basic methods: two-point geostatistics [1,2] and object-based methods [3,4]. The former is based on a sequential procedure that simulates a point at a time. This has a positive effect on honoring local hard data. However, two-point geostatistics uses a variogram or covariance to depict geological structures. These descriptors fail to preserve geometrically complex features. The object-based methods imitate the formation of geological structures. Their realizations agree well with reality. Nevertheless, intensive computations and a limited ability to hard data conditioning are two main obstacles for applications of object-based methods.

Owing to these limitations, the need for the multiple-point statistics (MPS) algorithm arises. As an attractive and versatile algorithm developed about two decades ago, MPS has been widely used in applications including groundwater flow [5–7], geophysics [8,9], petroleum engineering [10–12], mineral deposit modeling [13–15], and so on. The main idea of MPS and the first program, extended normal equation simulation (ENESIM), was introduced by Guardiano and Srivastava [16]. Viewing training images (TIs) as prior conceptual models, this program extracts patterns from TIs and reproduces patterns that are compatible with local data. This allows a computer to model geologically realistic features. In 1993, it was difficult to implement this program because of its high computational burden. Nevertheless, ENESIM provided a new way to explore geostatistical modeling. In particular, it introduced three innovations: (1) the concept of MPS instead of two-point statistics such as covariance or variogram, (2) the importance of TIs to inspire MPS, and (3) the use of nonparametric statistics. In 2002, Strebelle [17] proposed a landmark framework, referred to as single normal equation simulation (SNESIM), to considerably improve central processing unit (CPU) time performance. Using a search tree as a database, patterns from TIs and their occurrences are stored in computer memory. In simulation, informed points compose conditioning data using a template. SNESIM retrieves the database to find compatible patterns. The value of an uninformed point is determined by conditional probabilities computed based on occurrences of compatible patterns. The simulation procedure is repeated point by point until no uninformed point exists in the simulation domain. Because this program simulates points separately, it is called a point-based method.

Since SNESIM, there have been two main ways to develop MPS. The first one explores pattern-based methods instead of point-based methods. The first idea of pattern-based methods was proposed by Arpat and Caers [18] in their program, simulation patterns (SIMPAT). To overcome memory limitations as well as enhance pattern reproduction quality, SIMPAT concisely views stochastic modeling as a randomized puzzle. Patterns are viewed as puzzle pieces that need to be pasted and constrained to local data. In a similar manner, Zhang *et al.* [19]

[*]Corresponding author: zbpan@mail.xjtu.edu.cn

presented a filter-based selection of patterns (FILTERSIM) to accelerate search in SIMPAT at the expense of pattern reproduction. Honarkhah and Caers [20] suggested distance-based pattern clustering (DISPAT) to rapidly find desired patterns without the approximations in SIMPAT. Tahmasebi *et al.* [21] proposed cross-correlation function simulation (CCSIM). The efficiency and pattern reproduction quality are significantly improved via a cross-correlation function. Further research completed by Tan *et al.* [22] indicated that this method outperforms DISPAT in many applications. Tahmasebi *et al.* [23] improved CCSIM by a multiscale representation (MS-CCSIM). Cross-correlation function computations are implemented in the Fourier domain to alleviate computer burden. Yang *et al.* [24] presented a framework called global simulation. In this method, global optimization is applied to eliminate cumulative errors. A modified PatchMatch algorithm is used to accelerate the search procedure.

On the other hand, the second way to develop MPS relies on point-based methods. Strebelle and Claude [25] analyzed the issues of efficiency and memory in SNESIM. They proposed a new multigrid approach, a data template that preferentially includes previous simulated points, and template size optimization. Straubhaar *et al.* [26] presented a list approach and realized improved parallelization (IMPALA). In 2013, Straubhaar *et al.* [27] further modified the structure of databases based on lists and trees. With the aim of avoiding memory limitations in MPS, Mariethoz *et al.* [28] suggested direct sampling (DS). This method directly retrieves compatible patterns from TIs and makes databases unnecessary. Meerschman *et al.* [29] provided a practical guide to operate DS. Abdollahifard and Faez [30] suggested fast direct sampling (FDS), where a fast gradient descent pattern matching strategy is employed to speed up the search. In 2016, Straubhaar *et al.* [31] extended DS to deal with block data by using an additional criterion for the acceptance of candidate points.

As discussed above, the generation and retrieval operations to databases are key procedures in MPS frameworks. However, databases grow with increasing numbers of patterns in the case of geometrically complex, multifacies, multivariate, and high-dimensional simulations. A large amount of redundant data existing in databases consumes considerable CPU time in the simulation procedure. Moreover, existing methods focus on reproducing the majority of patterns in TIs. The difficulty in reproducing lowly proportional patterns is not addressed.

In this paper, we explore a way to implement MPS. To the best of our knowledge, vector quantization (VQ) techniques have not been applied in MPS yet. With the objective to speed up simulation and ensure high quality, we propose an accelerating simulation for MPS using VQ, called VQ-MPS. In our proposed method, a variable representation is presented to make categorical variables applicable for computations with VQ. In addition, tree-structured VQ (TSVQ) is used to compress patterns and store their occurrences so that stationary simulations are realized. Finally, we address nonstationary simulations. A transformed template is adopted to complete spatial transformations. Classified VQ (CVQ) is employed to incorporate secondary variables in databases. An application of a two-dimensional (2D) channelized reservoir simulation was used to validate our method. Qualitative and quantitative comparisons were made to evaluate the results of VQ-MPS and

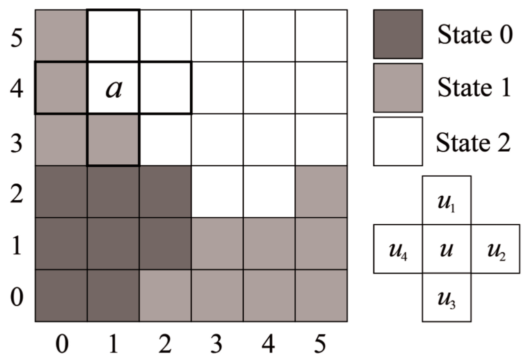

FIG. 1. A TI of size 6×6 and a template of size 4.

several other MPS programs. The proposed method exhibited significantly better performance in terms of computational efficiency, pattern reproduction ability, and spatial uncertainty. Lowly proportional patterns were reproduced in our method. Further applications were composed of a 2D four facies simulation, two 2D nonstationary simulations, and a three-dimensional (3D) simulation from a rock slice. The results indicated that VQ-MPS was also capable of addressing multifacies, solving nonstationarity, and producing a 3D simulation from a 2D TI.

The rest of this paper is organized as follows. Section II provides related work such as previous database structures in MPS and VQ. The proposed algorithm is presented in Sec. III. Section IV shows some applications. Finally, conclusions are drawn in Sec. V.

II. RELATED WORK

A. Previous database structures in the MPS framework

Previous point-based MPS methods, such as SNESIM and IMPALA, consist of two parts: (1) a training procedure and (2) a simulation procedure. These two procedures proceed as follows. In the training procedure, a template of size n, which is denoted by τ_n and contains n points $(u_1, u_2, \ldots, u_n)$, is predefined to find the neighbors surrounding its center u. A training pattern (also referred to as a data event), which is denoted by $d_n(u)$, is extracted by τ_n. In the example shown in Fig. 1, a TI of size 6×6 and a template of size 4 are applied. Therefore, the pattern centered at a is $d_n(a) = (2, 2, 1, 1)$.

To avoid a situation in which points in the template are outside the range of the TI, τ_n is always entirely inside the TI. After scanning all feasible points, the computer obtains a set of patterns and their occurrences. A search tree-structured database, generated by SNESIM in this case, is displayed in Fig. 2. A list-structured database, generated by IMPALA, is shown in Fig. 3.

Because three states are available in the TI, each element in the search tree can be expressed as (o_0, o_1, o_2). Here, o_i denotes an occurrence of a certain pattern with state i at its center. Each arrow line corresponds to a state. For example, the first element in layer 5 was visited along 0, 0, 0, 0 from the root. This indicated that the pattern was (0, 0, 0, 0). The element (1, 0, 0) showed that this pattern occurred in the TI once and its center was state 0. By contrast, the list in Fig. 3 is intuitive. Patterns and occurrences are separately

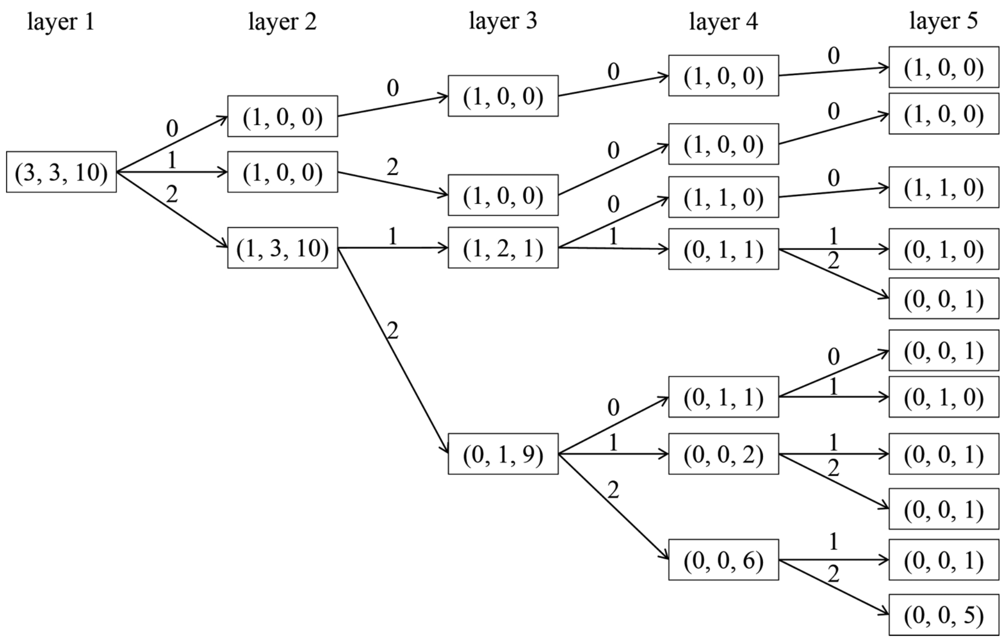

FIG. 2. A search tree-structured database.

stored. To reduce complexity, elements are sorted in ascending order according to the number of state 2 in patterns. Database generation methods are elaborated in Refs. [17,26].

Prior to the simulation procedure, it is necessary to initialize the simulation domain. In other words, a part of the simulation domain is known in order to prompt the geostatistical modeling program. These known points are viewed as hard data. An unconditional simulation strategy was applied as the initialization in this paper. The computer randomly samples a point from the TI. Next, another point is randomly sampled from the simulation domain. The value of the sampled point from the TI is assigned to the sampled point from the simulation domain. This assignment procedure is repeated until the percentage of hard data exceeds a predefined threshold called the sampling rate.

After the training procedure and the hard data assignment, a simulation procedure starts. The objective of a point simulation is to determine the value of an unknown point based on its surrounding informed data (also referred to as conditioning data). For example, a simulation domain is displayed in Fig. 4. Using the template shown in Fig. 1, the conditioning pattern centered at point b was $d_n(b) = (2,\ 2,\ -1,\ 1)$. Afterwards, the computer retrieved the database, as shown in Figs. 2 and 3. In this case, (2, 2, 0, 1), (2, 2, 1, 1), and (2, 2, 2, 1) were found as compatible patterns. Thus, occurrences (0, 1, 0), (0, 0, 1), and (0, 0, 1) were obtained. The sum of occurrences was 3. The total occurrences for o_0, o_1, and o_2 were 0, 1, and 2, respectively. The probability that point b was state 0 was $0/3 = 0$. In comparison, the probabilities that b was state 1 and state 2 were $1/3 = 0.33$ and $2/3 = 0.67$, respectively. Consequently, the computer selected either state 1 with a probability of 0.33 or state 2 with a probability of 0.67. In this case, state 2 was assigned to point b. The simulation procedure is repeated point by point along either a random or unilateral path [32] until each uninformed point is simulated.

One challenge in MPS is to capture patterns at different scales. Accordingly, a multigrid strategy is presented [17]. An example of the multigrid strategy with $G = 3$ nested grids is illustrated in Fig. 5. The gth $(1 \leqslant g \leqslant G)$ grid is composed of every 2^{g-1}th points of the simulation domain. Thus, $g = 1$

Pattern d_n	Occurrence (o_0, o_1, o_2)
(0, 0, 0, 0)	(1, 0, 0)
(1, 2, 0, 0)	(1, 0, 0)
(2, 1, 0, 0)	(1, 1, 0)
(2, 1, 1, 1)	(0, 1, 0)
(2, 1, 1, 2)	(0, 0, 1)
(2, 2, 0, 0)	(0, 0, 1)
(2, 2, 0, 1)	(0, 1, 0)
(2, 2, 1, 1)	(0, 0, 1)
(2, 2, 1, 2)	(0, 0, 1)
(2, 2, 2, 1)	(0, 0, 1)
(2, 2, 2, 2)	(0, 0, 5)

FIG. 3. A list-structured database.

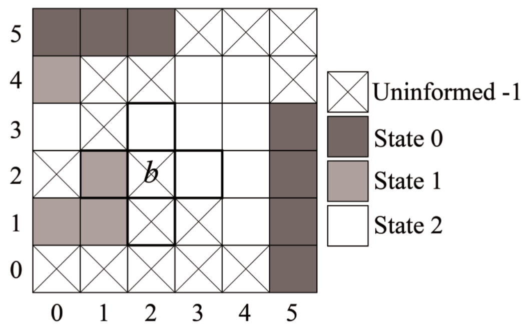

FIG. 4. A simulation domain of size 6 × 6.

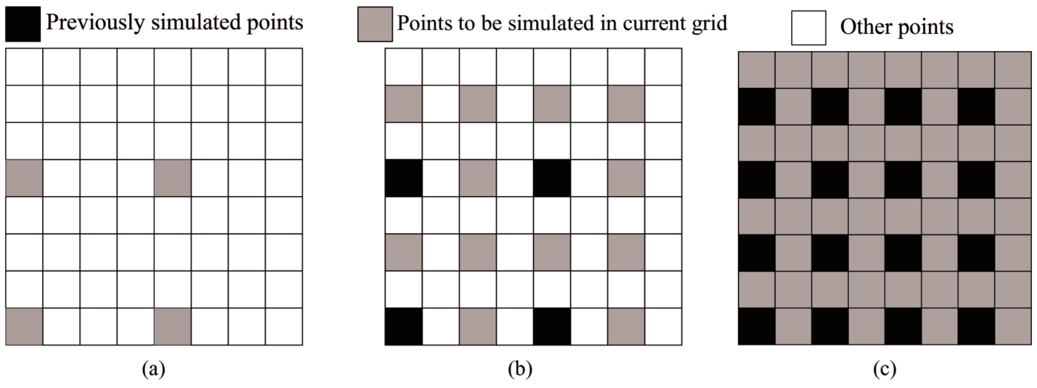
Previously simulated points
Points to be simulated in current grid
Other points
(a)
(b)
(c)

FIG. 5. Multigrid strategy for a simulation domain of size 8 × 8. (a) The coarsest grid. (b) The moderate grid. (c) The finest grid.

and G denote the finest and coarsest grids, respectively. The simulation procedure proceeds in a coarse to fine fashion. The unknown points in the coarsest grid are assigned at the beginning. Based on this design, the long-range structures are simulated prior to reproducing the detailed information.

Although many improvements over SNESIM and IMPALA have been explored, there are still three basic limitations.

(1) The amount of memory used to establish databases exponentially increases with the size n of template τ_n. Supposing that there are K possible states in a TI, the maximum number of possible patterns is K^n. Fortunately, databases only store patterns existing in TIs. Typically, the sizes of the databases used in the following section were in the range of $10^3 - 10^5$.

(2) The template also has a significant effect on computational time. The database grows with the size of the template. A considerable CPU time is required to find the desired elements. This issue is further aggravated in the case of geometrically complex, multifacies, multivariate, and high-dimensional simulations.

(3) The CPU time for calculating a conditional probability is also influenced by the number of uninformed points in the conditioning pattern. Assuming that n' informed points exist in a conditioning pattern d_n, the number of compatible patterns of d_n is $K^{n-n'}$. The more uninformed points, the more compatible patterns there are. Considerable CPU consumption is necessary to solve a wider search scope. However, abundant informed points lead to an absence of compatible patterns in TIs. Previous MPS programs adopted a pruning strategy, where some conditioning data are discarded. The computer has to iteratively check the database until it finds enough compatible patterns.

B. Vector quantization

Vector quantization is a classical quantization technique from signal processing that models the probability density function using prototype vectors. VQ works by partitioning a set of vectors into a number of groups. Each group is represented by its centroid, as designed in the k-means algorithm and other clustering algorithms. In this section, the concept of VQ is theoretically explained. The combination of MPS and VQ is proposed in Sec. III, where we also discuss an example.

In VQ, a codebook, which stores all centroids, is trained in advance. There are various algorithms to train a codebook. Among them, the most frequently used method is the Linde-Buzo-Gray (LBG) algorithm [33]. In this paper, a random selection strategy is applied as an initialization. In other words, S codewords are randomly selected from training vectors. Here, S denotes the size of the codebook and is predefined by users. Then, an iterative procedure generated the final codebook denoted herein as *Book*. The details of LBG are summarized in the following steps.

(1) Generate an initial codebook $Book_0$ of size S. Set the iteration counter $i = 0$ and the maximum iteration counter $I_{\max}$.

(2) For each training vector, find its best-matched codeword with the smallest distance in the current codebook $Book_i$.

(3) Allocate training vectors into S groups. Update the centroid of each group by calculating the mean of vectors in it. Let $i = i + 1$ and a new codebook $Book_i$ is obtained.

(4) If $i = I_{\max}$, output the codebook $Book = Book_i$ and LBG is completed; otherwise, the program returns to step 2.

Codebook *Book* is composed of S M-dimensional codewords $W_s = (w_0^s, w_1^s, \ldots, w_{M-1}^s)$. Here, s denotes the index of codeword W_s in the range $[0, S-1]$. w denotes an elementary variable and M denotes the dimension of the vectors. In step 2 of the algorithm just described, the squared Euclidean distance between an input vector W and codeword W_s is defined as

$$D_E(W, W_s) = \sum_{m=0}^{M-1} \left(w_m - w_m^s\right)^2. \quad (1)$$

Because LBG performs an exhaustive search in step 2, this method is called a full search method. However, a full search is time consuming. Modifying the structure of the codebook is a feasible way to reduce the computational burden. In the following paragraphs, we introduce two typical types of constrained VQ: tree-structured VQ and classified VQ.

TSVQ uses a tree-structured codebook [34,35]. A binary tree is frequently used due to its simplicity. The first step is to calculate the centroid of the collection of training vectors and view this centroid as the root. Next, to find two children nodes of this node, the centroid and a perturbed centroid are chosen as the initial codewords. Using the best-matched search and centroid updating strategy, the local optimal codewords for the two children nodes can be found. Thus, the training vectors are divided into two groups. The procedure recurs on each subtree until the height of the tree exceeds a predefined threshold. To find an appropriate codeword for a given input vector, the tree is traversed from the root down by comparing the input vector with the two children nodes. The computer iteratively follows the node that has the smallest Euclidean distance until it reaches a leaf node. Unlike the full search method, the search result of TSVQ may not be a global optimum because only part of the codebook is checked. However, the result is approximately optimum and the cost of computations is significantly reduced. If the binary tree is reasonably balanced, a single search in a codebook of size S can be completed in $2 \times \log_2 S$ calculations of the Euclidean distance. As a comparison, S calculations of the Euclidean distance are necessary in a full search method.

Ramamurthi and Gersho [36] proposed CVQ to reduce the complexity as well as distortions. The key concept in CVQ is to apply a classification prior to compiling an independent codebook for each class. CVQ consists of two operations. First, training vectors are classified according to their features such as mean and variance. Second, a codebook with a smaller size is designed independently for each class. In searching procedure, a classifier is applied first for an input vector. The best-matched codeword is then found from its corresponding codebook.

III. PROPOSED METHODOLOGY

A. Categorical variables representation

As mentioned above, a key procedure in VQ is updating centroids. However, it is the categorical variable that is simulated in this paper, in contrast with the continuous variable typically used in VQ. It is necessary to develop a way to make categorical variables applicable for VQ. Before presenting our method, several features of categorical variables are listed. An important feature is that categorical variables are nominal attributions. In general, each state reflects a level or label, which is a logically independent concept. Therefore, categorical variables can only be manipulated by distinction and set-related operations. Furthermore, the criterion that measures the distance between two vectors of categorical variables is the Hamming distance. Let $u_k = 0, 1, \ldots, K-1$ denote all possible states. The Hamming distance between two states u_i and u_j is defined as

$$D_H(u_i,u_j) = \begin{cases} 0 & \text{if} \quad u_i = u_j \\ 1 & \text{if} \quad u_i \neq u_j \end{cases}. \tag{2}$$

Let $d_n(a) = (u_0^a, \quad u_1^a, \ldots, u_{n-1}^a)$ and $d_n(b) = (u_0^b, u_1^b, \ldots, u_{n-1}^b)$ denote two patterns obtained in MPS. The Hamming distance between $d_n(a)$ and $d_n(b)$ is defined as

$$D_H[d_n(a),d_n(b)] = \sum_{i=0}^{n-1} D_H\left(u_i^a,u_i^b\right). \tag{3}$$

In addition, u_i^a may be unknown in a simulation procedure. Uninformed items are ignored in Eq. (3). For example, the distance between (0, 0, 0, 0) and (0, −1, 0, 1) is 1, where −1 denotes an unknown state.

Classifying patterns is not new in MPS literature. In the previous methods proposed by Zhang *et al.* [19] and Honarkhah and Caers [20], centroids are created by the pointwise averaging of all training patterns in a class. This strategy is reasonable to solve the continuous variable. However, it is not suitable to adopt pointwise averaging to address categorical variables. The reason is that the pointwise averaging strategy violates the intrinsic property of categorical variables and generates a biased centroid. We explain this by an intuitive example in water resources research. Let state 0, state 1, and state 2 denote mudstone, channel, and levee, respectively. It is clear that these three states are mutually independent concepts. Assume that two training patterns, (0, 0, 0, 0) and (2, 2, 2, 2), and one conditioning pattern (1, 1, 1, 1) are found. Using pointwise averaging, the centroid of these two training patterns is (1, 1, 1, 1). On the one hand, this centroid is in complete agreement with the conditioning pattern. On the other hand, there is no channel state in those two training patterns. The dissimilarity between each training pattern and the conditioning pattern is high. This inconsistency indicates that pointwise averaging is not an accurate method to generate centroids.

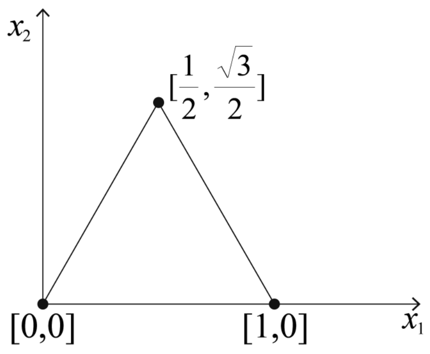

FIG. 6. A 2D plane to represent three states.

To make categorical variables suitable for VQ and keep their intrinsic properties, we propose a representation method. Suppose that a categorical variable with K possible states is used. Our method maps this variable into a $K-1$ dimensional space, where each state is expressed as a fixed point. The distance between any two points is always equal to 1. Therefore, our representation method agrees with Eq. (2). For instance, an example for $K = 3$ is illustrated in Fig. 1. Using our representation method, this categorical variable is mapped into a 2D plane. Three points that constitute an equilateral triangle are assigned to three states, which is shown in Fig. 6. State 0 is assigned to the origin of [0, 0]. State 1 is assigned to the point of [1, 0]. State 2 is assigned to the point of $[1/2, \sqrt{3}/2] \approx [0.50, 0.87]$. In the following, all coordinates are expressed as two decimal fractions. Indeed, coordinates of three points can be arbitrarily assigned if only these points constitute an equilateral triangle with one side length equal to 1. Using this representation method, the patterns shown in Fig. 3 can be transformed in codewords shown in Fig. 7. In order to be coherent with the notation in VQ, the transformed patterns are called codewords. Although this database looks complex, it is easily implemented by a computer. In a similar manner, a categorical variable for $K = 4$ possible states can be mapped into a three-dimensional space with the four assigned points constituting a triangular pyramid. With the increasing number of possible states, the concepts of hyperplanes and hyperpyramids play a central role.

Codeword W	Occurrence (o_0, o_1, o_2)
([0.00, 0.00], [0.00, 0.00], [0.00, 0.00], [0.00, 0.00])	(1, 0, 0)
([1.00, 0.00], [0.50, 0.87], [0.00, 0.00], [0.00, 0.00])	(1, 0, 0)
([0.50, 0.87], [1.00, 0.00], [0.00, 0.00], [0.00, 0.00])	(1, 1, 0)
([0.50, 0.87], [1.00, 0.00], [1.00, 0.00], [1.00, 0.00])	(0, 1, 0)
([0.50, 0.87], [1.00, 0.00], [1.00, 0.00], [0.50, 0.87])	(0, 0, 1)
([0.50, 0.87], [0.50, 0.87], [0.00, 0.00], [0.00, 0.00])	(0, 0, 1)
([0.50, 0.87], [0.50, 0.87], [0.00, 0.00], [1.00, 0.00])	(0, 1, 0)
([0.50, 0.87], [0.50, 0.87], [1.00, 0.00], [1.00, 0.00])	(0, 0, 1)
([0.50, 0.87], [0.50, 0.87], [1.00, 0.00], [0.50, 0.87])	(0, 0, 1)
([0.50, 0.87], [0.50, 0.87], [0.50, 0.87], [1.00, 0.00])	(0, 0, 1)
([0.50, 0.87], [0.50, 0.87], [0.50, 0.87], [0.50, 0.87])	(0, 0, 5)

FIG. 7. A list represented by the proposed representation method.

In summary, our method generates $(K-1)$-dimensional coordinates $[x_1(u), x_2(u), \ldots, x_{K-1}(u)]$ to represent state u. This coordinate vector is applied as an element w in a codeword W. The addition operation and centroid between u_i^a and u_i^b can be expressed, respectively, as

$$v_{\text{add}}(u_i^a, u_i^b) = [x_1(u_i^a) + x_1(u_i^b), x_2(u_i^a) + x_2(u_i^b), \ldots, x_{K-1}(u_i^a) + x_{K-1}(u_i^b)], \quad (4)$$

$$v_{\text{cent}}(u_i^a, u_i^b) = \{\tfrac{1}{2}[x_1(u_i^a) + x_1(u_i^b)], \tfrac{1}{2}[x_2(u_i^a) + x_2(u_i^b)], \ldots, \tfrac{1}{2}[x_{K-1}(u_i^a) + x_{K-1}(u_i^b)]\}. \quad (5)$$

Therefore, the centroid of patterns $d_n(a)$ and $d_n(b)$ can be expressed as

$$\begin{aligned} W &= (w_0, w_1, \ldots, w_{n-1}) \\ &= [v_{\text{cent}}(u_0^a, u_0^b), v_{\text{cent}}(u_1^a, u_1^b), \ldots, v_{\text{cent}}(u_{n-1}^a, u_{n-1}^b)]. \end{aligned} \quad (6)$$

Moreover, the Euclidean distance can be applied as a measure of distance between two codewords. The distances between u_i^a and u_i^b, $d_n(a)$, and $d_n(b)$ can be expressed, respectively, as

$$D_E(u_i^a, u_i^b) = D_E(w_i^a, w_i^b) = \sum_{j=1}^{K-1} [x_j(u_i^a) - x_j(u_i^b)]^2, \quad (7)$$

$$D_E[d_n(a), d_n(b)] = D_E[W(a), W(b)] = \sum_{i=0}^{n-1} [D_E(u_i^a, u_i^b)]^2. \quad (8)$$

Equations (7) and (8) are used not only in the training procedure but also in the simulation procedure. Like original MPS, the uninformed items are ignored.

The preceding example was used to test the proposed representation. Because three states appear, two training patterns, (0, 0, 0, 0) and (2, 2, 2, 2), were, respectively, described as ([0, 0], [0, 0], [0, 0], [0, 0]) and ([0.50, 0.87], [0.50, 0.87], [0.50, 0.87], [0.50, 0.87]). Thus, the centroid is ([0.25, 0.43], [0.25, 0.43], [0.25, 0.43], [0.25, 0.43]). The conditioning pattern (1, 1, 1, 1) was depicted as ([1, 0], [1, 0], [1, 0], [1, 0]). Therefore, the distance between the conditioning pattern and the centroid was $(\sqrt{3}/2) \times 4 \approx 3.46$. In comparison, the distance between the conditioning pattern and each training pattern is 4. These distances imply that the training patterns are incompatible with the conditioning pattern. The training patterns are properly characterized by our centroid.

Codeword W	Corresponding pattern d_n
([1.00, 0.00], [0.50, 0.87], [0.00, 0.00], [0.00, 0.00])	(1, 2, 0, 0)
([0.50, 0.87], [1.00, 0.00], [1.00, 0.00], [1.00, 0.00])	(2, 1, 1, 1)
([0.50, 0.87], [1.00, 0.00], [1.00, 0.00], [0.50, 0.87])	(2, 1, 1, 2)
([0.50, 0.87], [0.50, 0.87], [0.00, 0.00], [0.00, 0.00])	(2, 2, 0, 0)

FIG. 8. The codebook $Book_0$.

B. Principles of VQ-MPS

After representing each pattern occurring in TIs, VQ can be applied to compress databases. The list shown in Fig. 7 is employed as an example. Regardless of the occurrences, each codeword is viewed as an input vector. We set the size of the codebook $S = 4$ and the maximum iteration counter $I_{\max} = 10$ in this case. With a random selection strategy, a codebook $Book_0$ is shown in Fig. 8. The pattern corresponding to each codeword is also displayed. LBG is applied to train the codebook. A codebook $Book_{10}$ is obtained, as shown in Fig. 9. The left column displays the codewords, while the right column notes the subordinated patterns to each group. After codebook generation, the computer allocates the occurrences for each codeword. Consequently, the final codebook $Book$ is shown in Fig. 10. After the training procedure, only the $Book$ will be used. The TIs and previous databases can be removed from computer memory.

The program described above uses a full search VQ which degrades CPU time performance. TSVQ, which was introduced in detail in Sec. II B, provides an effective way to reduce the computational burden. In this case, the computer used a binary tree, the height of which was assigned to 3. The root node in layer 1 is the centroid of all patterns occurring in the TI. In layer 2, LBG divides a set of patterns into two groups. This procedure is repeated until the height of the tree reaches a certain threshold. After codebook generation, the occurrences are stored in leaf-level nodes. The codebook $Book_{\text{tree}}$ in this case is shown in Fig. 11. It is worth noting that this tree-structured codebook is significantly different from the search tree. Our method produces a centroid as an element in the tree-structured codebook, as opposed to each element in the search tree being a set of occurrences. Moreover, TSVQ generates two children using a clustering technique. In comparison, the search tree extends children by means of visiting the next point in a pattern. There is no grouping or classification operation in the search tree.

After codebook training, the simulation procedure proceeds along a random path. As shown in Fig. 4, an unconditional

Codeword W	Subordinated pattern d_n
([0.50, 0.00], [0.25, 0.43], [0.00, 0.00], [0.00, 0.00])	(0, 0, 0, 0), (1, 2, 0, 0)
([0.50, 0.87], [0.67, 0.58], [0.83, 0.28], [1.00, 0.00])	(2, 1, 1, 1), (2, 2, 1, 1), (2, 2, 2, 1)
([0.50, 0.87], [0.67, 0.58], [0.83, 0.28], [0.50, 0.87])	(2, 1, 1, 2), (2, 2, 1, 2), (2, 2, 2, 2)
([0.50, 0.87], [0.67, 0.58], [0.00, 0.00], [0.33, 0.00])	(2, 1, 0, 0), (2, 2, 0, 0), (2, 2, 0, 1)

FIG. 9. The codebook $Book_{10}$.

Codeword W	Occurrence (o_0, o_1, o_2)
([0.50, 0.00], [0.25, 0.43], [0.00, 0.00], [0.00, 0.00])	(2, 0, 0)
([0.50, 0.87], [0.67, 0.58], [0.83, 0.28], [1.00, 0.00])	(0, 1, 2)
([0.50, 0.87], [0.67, 0.58], [0.83, 0.28], [0.50, 0.87])	(0, 0, 7)
([0.50, 0.87], [0.67, 0.58], [0.00, 0.00], [0.33, 0.00])	(1, 2, 1)

FIG. 10. The final codebook *Book*.

simulation strategy is applied. For each successive point b in the path, the following steps apply.

(1) Find the neighbors and generate a pattern. A template, which was already used in the training procedure, is employed again to find informed points. In Fig. 4, $d_n(b) = (2,\ 2,\ -1,\ 1)$ was obtained. If no informed point is found, the computer randomly samples a point from the TI and assigns the value of this sampled point to point b.

(2) Transform the pattern. Using the representation method defined in Sec. III A, $d_n(b)$ was transformed into a codeword $W(b) = ([0.50,\ 0.87],\ [0.50,\ 0.87],\ [-1.00,\ -1.00],\ [1.00,\ 0.00])$.

(3) Find the best-matched codeword in the codebook. In the full search VQ scheme, each codeword in codebook *Book* shown in Fig. 10 was successively searched. The codeword ([0.50, 0.87], [0.67, 0.58], [0.83, 0.28], [1.00, 0.00]) was found. In TSVQ, the program traversed the tree from the root. The algorithm followed the child node with the smallest Euclidean distance. Therefore, the codeword ([0.50, 0.87], [0.67, 0.58], [0.83, 0.28], [1.00, 0.00]) was found again.

(4) Calculate the conditional probabilities. Conditional probabilities are computed using the occurrences in the best-matched codeword. In this case, the occurrences of three states were 0, 1, and 2. Therefore, the conditional probabilities of three states were $0/3 = 0.00$, $1/3 = 0.33$, and $2/3 = 0.67$, respectively. In other words, the computer selected either state 1 with a probability of 0.33 or state 2 with a probability of 0.67. In this case, state 2 was assigned to point b.

The point simulation mentioned above iterates over the remaining uninformed points in the simulation domain. Previous simulated points are viewed as hard data and become conditioning data for subsequent simulations. Because our method focuses on improving the database structure, all of the usual extensions of MPS (such as a multigrid strategy, a unilateral simulation path, and post- and syn-processing) can be applied straightforwardly.

C. Features and advantages of VQ-MPS

In summary, VQ technique is applied to refine databases in SNESIM or IMPALA. Although the classification or clustering methods have already been introduced in previous MPS programs (FILTERSIM and DISPAT), our proposed method has the following features.

(1) VQ-MPS is a point-based program. In other words, our method simulates a point at a time. This scheme has a positive effect on honoring hard data. Similar to SNESIM and IMPALA, the value of an unknown point is determined by the occurrences of compatible patterns.

(2) In training procedure, the similarity between two patterns is directly calculated in the spatial domain. The dimensionality of patterns is not reduced. There is no feature extraction technique in our method. With the objective to reduce the size of patterns, FILTERSIM and DISPAT describe a pattern as several features. Thus, the distance measure is carried out in the feature domain. FILTERSIM creates six scores to represent a two-dimensional pattern. In DISPAT, a pattern is mapped into a Cartesian space using multidimensional scaling. However, dimension reduction is a lossy operation. A feature vector

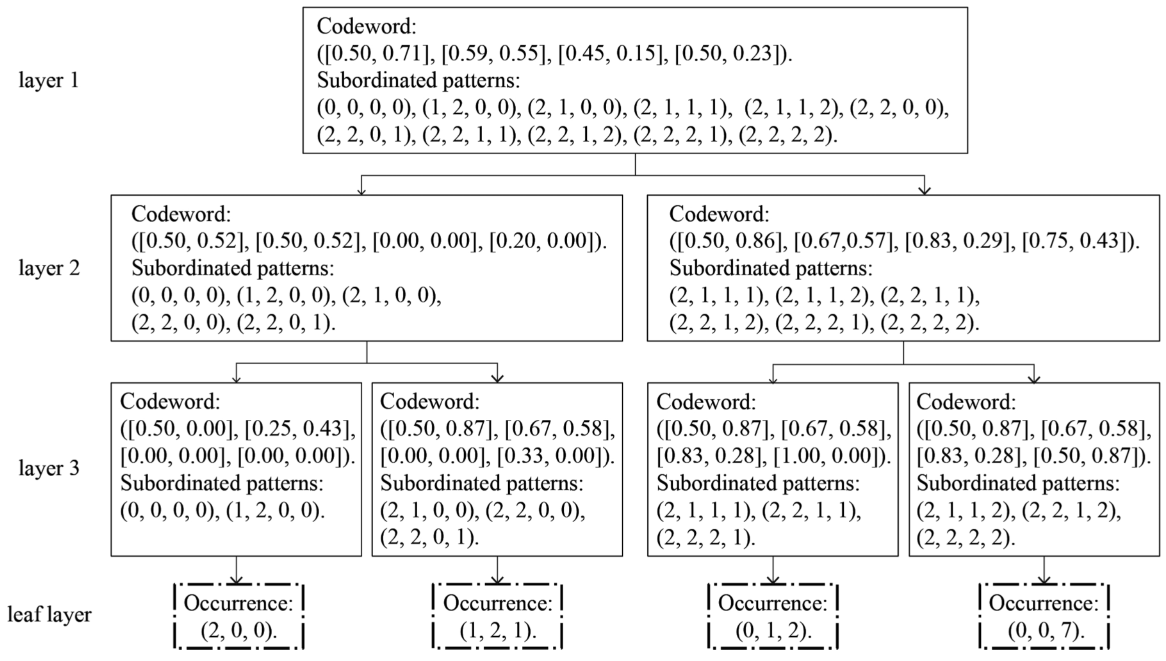
layer 1
Codeword: ([0.50, 0.71], [0.59, 0.55], [0.45, 0.15], [0.50, 0.23]). Subordinated patterns: (0, 0, 0, 0), (1, 2, 0, 0), (2, 1, 0, 0), (2, 1, 1, 1), (2, 1, 1, 2), (2, 2, 0, 0), (2, 2, 0, 1), (2, 2, 1, 1), (2, 2, 1, 2), (2, 2, 2, 1), (2, 2, 2, 2).
layer 2
Codeword: ([0.50, 0.52], [0.50, 0.52], [0.00, 0.00], [0.20, 0.00]). Subordinated patterns: (0, 0, 0, 0), (1, 2, 0, 0), (2, 1, 0, 0), (2, 2, 0, 0), (2, 2, 0, 1).
Codeword: ([0.50, 0.86], [0.67,0.57], [0.83, 0.29], [0.75, 0.43]). Subordinated patterns: (2, 1, 1, 1), (2, 1, 1, 2), (2, 2, 1, 1), (2, 2, 1, 2), (2, 2, 2, 1), (2, 2, 2, 2).
layer 3
Codeword: ([0.50, 0.00], [0.25, 0.43], [0.00, 0.00], [0.00, 0.00]). Subordinated patterns: (0, 0, 0, 0), (1, 2, 0, 0).
Codeword: ([0.50, 0.87], [0.67, 0.58], [0.00, 0.00], [0.33, 0.00]). Subordinated patterns: (2, 1, 0, 0), (2, 2, 0, 0), (2, 2, 0, 1).
Codeword: ([0.50, 0.87], [0.67, 0.58], [0.83, 0.28], [1.00, 0.00]). Subordinated patterns: (2, 1, 1, 1), (2, 2, 1, 1), (2, 2, 2, 1).
Codeword: ([0.50, 0.87], [0.67, 0.58], [0.83, 0.28], [0.50, 0.87]). Subordinated patterns: (2, 1, 1, 2), (2, 2, 1, 2), (2, 2, 2, 2).
leaf layer
Occurrence: (2, 0, 0).
Occurrence: (1, 2, 1).
Occurrence: (0, 1, 2).
Occurrence: (0, 0, 7).

FIG. 11. The final tree-structured codebook $Book_{\text{tree}}$.

only preserves a part of the geometry information about the original pattern. Dissimilar patterns may be grouped into the same cluster. This is certainly harmful for pattern clustering and further simulation procedure.

(3) Training patterns are clustered. VQ compresses a huge number of patterns into a codebook and uses centroids to represent all training patterns. A database of small size helps the program to improve efficiency in the simulation procedure. In comparison, SNESIM and IMPALA do not use a pattern classification analysis. Each training pattern is directly stored in computer memory. FILTERSIM classifies patterns by means of dividing each score into five equal frequency bins. However, this scheme is a scalar quantization and ignores the correlation between scores. For this reason, FILTERSIM exhibits limited classification accuracy.

(4) A new pattern representation and centroid generation method are presented in this paper. Our scheme attempts to conserve the intrinsic properties of categorical variables. The price of this advantage is an increase in the pattern dimensionality. By contrast, FILTERSIM and DISPAT use pointwise averaging. A biased centroid cannot properly represent a group of patterns.

(5) Centroids are organized by a tree. The computer only visits a part of a tree-structured codebook to find the best-matched element. Therefore, our method significantly saves CPU time in a retrieve operation. However, FILTERSIM and DISPAT directly store centroids in computer memory. The program has to test each centroid in order to find the desired one.

(6) In simulation procedure, VQ-MPS only finds the best-matched codeword. This means the codebook is checked only once. However, SNESIM and IMPALA adopt a pruning strategy. If compatible patterns do not exist in the database, the last point in the conditioning pattern is dropped. The program has to test the database again until it finds enough compatible patterns.

(7) The appearance of lowly proportional patterns enriches the diversity of patterns in VQ-MPS. Suppose that the radius of the template is r and the TI is of size $height \times width$. Accordingly, the training procedure extracts $(height-2r) \times (width-2r)$ patterns from the TI and stores N patterns in the database. Here, N denotes the size of database. In general, N is much lower than $(height-2r) \times (width-2r)$ because certain patterns repeatedly appear in the TI. The patterns that frequently appear in a TI are referred to as highly proportional patterns, as opposed to lowly proportional patterns, which rarely occur.

VQ-MPS uses N patterns that are stored in the database to train a codebook. Thus, the lowly proportional patterns have the same influence as the highly proportional patterns in the cluster operation. After codebook generation, occurrences are assigned to each codeword. Therefore, some codewords are created by lowly proportional patterns. For example, the first codeword in Fig. 9 contains two lowly proportional patterns. In simulation, VQ-MPS determines the value of an unknown point by the occurrences of the best-match codewords. If a lowly proportional codeword is selected, the program completes a point simulation in accordance with the occurrences of certain lowly proportional patterns. This design allows VQ-MPS to reproduce the lowly proportional patterns and enrich the diversity of patterns in simulation realizations. For one thing, the appearance of lowly proportional patterns makes the realization more similar to the TI. For another thing, the increasing diversity enlarges the difference between simulation realizations.

However, the value of an unknown point is principally determined by the highly proportional patterns in the original MPS. In SNESIM and IMPALA, a point simulation is conducted on the basis of the occurrences of compatible patterns. If a highly proportional pattern is selected as a compatible pattern, its high occurrences principally determine the result of a point simulation. Therefore, the program always reproduces highly proportional patterns. Some lowly proportional patterns do not appear in the simulation realizations. FILTERSIM and DISPAT both use $(height-2r) \times (width-2r)$ patterns in the classification task, while VQ-MPS applies N patterns. The centroids produced by FILTERSIM and DISPAT are biased to highly proportional patterns. As a result, lowly proportional patterns do not have enough influence on the simulation procedure. The absence of lowly proportional patterns is harmful for pattern reproduction as well as spatial uncertainty.

D. Addressing nonstationarity

Nonstationarity is not unusual in geological processing. A competitive algorithm should have competence in addressing nonstationarity. For existing MPS programs, many methods have been reported to tackle this issue. A frequently used strategy is to provide auxiliary information to describe nonstationarity. The categorical variables in patterns are called the first variables (also called the primary variables), and the auxiliary characteristics of patterns are referred to as the secondary variables (also referred to as the auxiliary variables). In this paper, two types of nonstationarity are addressed: (1) spatial transformations of patterns and (2) restricted locations of reproductive patterns.

For the first type of nonstationarity, traditional MPS transforms the patterns appearing in the TI. The first step is to scan a stationary TI and construct various databases using a transformed template. As the second step, the nonstationary simulation domain is exhaustively partitioned into several stationary areas. The index of each area reflects auxiliary characteristics of spatial transformations. The simulation procedure proceeds using a nontransformed template. Two widely applied transformations are rotation and affinity.

However, we do not adopt this strategy. The reason is that the transformations are defined not only by indices but also by several continuous variables. The former reveals that transformations are identical within each area, whereas the latter implies that transformations change continuously at each point. Therefore, our method is inspired by a strategy in DS [28]. As the first step, the training procedure scans a stationary TI using a nontransformed template. TSVQ technique is applied to create databases. Second, the simulation procedure visits an unknown point b and finds its neighborhoods. The form of template is distorted according to parameters of spatial transformation. Let $[x, y]$ denote the coordinates of any point in a transformed template centered at point b in a nonstationary simulation domain and let $[x', y']$ denote the coordinates of any point in a template used in stationary TIs. The spatial

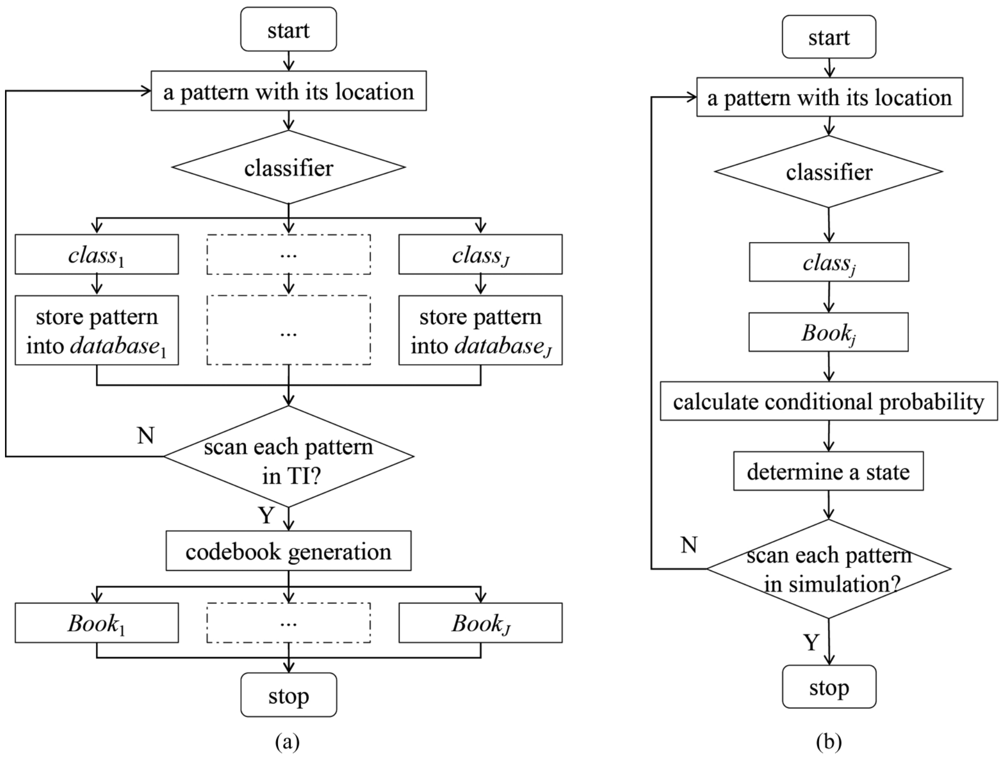
start
a pattern with its location
classifier
$class_1$
...
$class_J$
store pattern into $database_1$
store pattern into $database_J$
N
scan each pattern in TI?
Y
codebook generation
$Book_1$
$Book_J$
stop
(a)
start
a pattern with its location
classifier
$class_j$
$Book_j$
calculate conditional probability
determine a state
N
scan each pattern in simulation?
Y
stop
(b)

FIG. 12. CVQ in the proposed method. (a) The training procedure. (b) The simulation procedure.

transformation is processed as

$$\begin{aligned}[x' \quad y' \quad 1] &= [x \quad y \quad 1] \times \mathbf{T} \\ &= [x \quad y \quad 1] \times \begin{bmatrix} t_{11} & t_{12} & 0 \\ t_{21} & t_{22} & 0 \\ t_{31} & t_{32} & 1 \end{bmatrix}. \end{aligned} \quad (9)$$

Scaling, rotation, translation, and sheering can be easily realized, depending on the values chosen for the elements of the matrix **T**. Note that the matrix **T** transforms patterns in the nonstationary simulation domain into patterns in stationary TIs. By adopting this transformation to each point, a nonstationary pattern is transformed into a stationary pattern. Similar to the procedure in the stationary case, the program finds the best-matched codeword for a given pattern and determines the value of an unknown point. The program successively proceeds until each uninformed point is determined in the simulation domain. Consequently, the geometric spatial transformation of the pattern is dealt with.

For the second type of nonstationarity, the TIs are divided into several areas. The index of each area is used as a secondary variable to record locations where patterns appear. Prior to simulation, each point in the simulation domain is exhaustively assigned a secondary variable. The significance of this exhaustive assignment is to restrict locations of reproduced patterns. For example, patterns occurring in the ith area in a TI must be reproduced in the ith area in the simulation domain. For this type of nonstationarity, a useful approach is to create an independent database for each area. CVQ technique described in Sec. II B provides a feasible framework. Figure 12 shows a simplified flowchart of our method. A key step of CVQ is to design a classifier that categorizes secondary variables into J classes. The value of J is problem dependent. In the training procedure, each pattern occurring in TIs is classified into a certain class. J databases are then created. Afterwards, an independent codebook for each class is generated by LBG or TSVQ. In the simulation, the program obtains a value of the secondary variable from each unknown point and allocates this point to class j. $Book_j$ is chosen and checked to find the best-matched codeword. Finally, the state of each point is determined based on the occurrences. Point simulation progresses iteratively along a random path, as described above. Note that CVQ can easily be expanded to applications with multiple secondary variables by modifying the classifiers.

IV. APPLICATIONS OF THE METHODOLOGY

A. A 2D application with a binary stationary training image

As the first application, a 2D stationary channelized TI of size 101×101, shown in Fig. 13, was provided to validate the proposed method. This image has been widely used in many

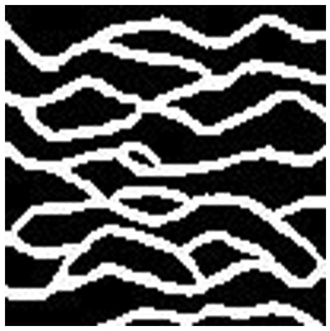

FIG. 13. A 2D stationary channelized TI.

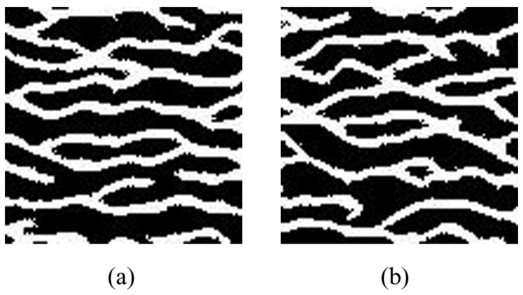
FIG. 14. The first two simulated realizations generated by full search VQ with codebooks of size 128.

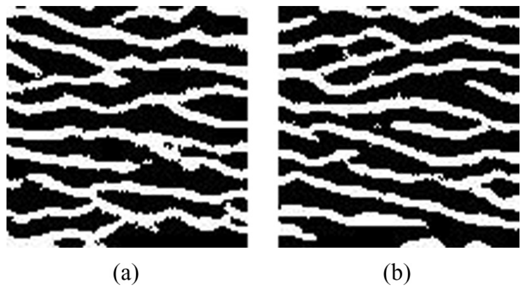
FIG. 16. The first two simulated realizations generated by TSVQ with codebooks of size 256.

MPS programs [22–25,28]. Our method was implemented in Java and executed on a 3.2-GHz Windows computer. The multigrid scheme with $G = 2$ was adopted for this application. Therefore, two databases were required. A template of size 7×7 and simulation domain of size 101×101 were applied. The sampling rate was set to 0.5% so that each simulated domain initially contained $101 \times 101 \times 0.5\% = 51$ hard data. A median filter was used as a postprocessing step in order to eliminate isolated points. Three schemes in the proposed method were considered: a full search VQ with codebooks of size 128, a TSVQ with codebooks of size 128, and a TSVQ with codebooks of size 256. Fifty unconditional simulations were generated for each scheme. The first two for each scheme are displayed in Figs. 14–16. For the first scheme, the computer required about 5 s to create two databases, and 427 s to simulate 50 realizations. For the second scheme, the computer cost 10 s to create databases, and 41 s to simulate realizations. For the third scheme, the training procedure and the simulation procedure took 12 and 45 s, respectively. In this application, an ideal simulated realization has similar connectivity and smoothness as the TI. By comparison with the TI, channels in simulated realizations were connected from left to right sides of the images. The width of the channels matched the TI. Qualitatively, this visual interpretation indicated that all three schemes effectively completed the simulation.

In order to discuss computational complexity as well as CPU time performance, IMPALA was adopted for this application. With the intention of objectively evaluating performance of our method, IMPALA was implemented by our own version. The reason is that programming language and computer configurations have an important effect on CPU time. Because the parallelization strategy has not been applied, we use sequential IMPALA (S-IMPALA) to denote IMPALA of our version. Using the same template, multigrid, and other configurations described in VQ-MPS, 50 realizations were simulated by S-IMPALA. The first two were shown in Fig. 17. It costs 1 s to create two databases and 512 s to generate 50 simulated realizations. This implies that VQ technique significantly accelerates simulation in the MPS framework. In the following, we theoretically analyze the computational efficiency of VQ-MPS and S-IMPALA.

On the one hand, the efficiency of Hamming distance in Eq. (3) and Euclidean distance in Eq. (8) is considered. Because a template of size 7×7 was used, each pattern was a 48-dimensional vector. As an instance, the computer calculated the two distances between two arbitrary 48-dimensional vectors 1×10^8 times. The CPU time cost for both distance measures was 6 s. This indicates that the computational efficiencies of the two distances are similar. However, computations of Hamming distance can be further reduced. The list in Fig. 3 is used as an example. Suppose that a pattern $d_n(c) = (0, -1, -1, 0)$ is obtained, which already has two informed states. Therefore, the possible number of state 2 in compatible patterns is in the range [0, 2]. The search scope is from (0, 0, 0, 0) to (2, 2, 1, 1). In a search, patterns like (1, 2, 0, 0) can be rapidly eliminated because its first state is not equal to the first state in $d_n(c)$. It is unnecessary to test all elements. During a VQ search, each informed state has to be checked. In this 2D channels simulation, Hamming distance involved vectors of dimension 20 on average, while calculations of Euclidean distance involved vectors of dimension 48. On the other hand, for a single point simulation, 128 calculations of Euclidean distance were required for the full search VQ in this application. However, two TSVQs only performed

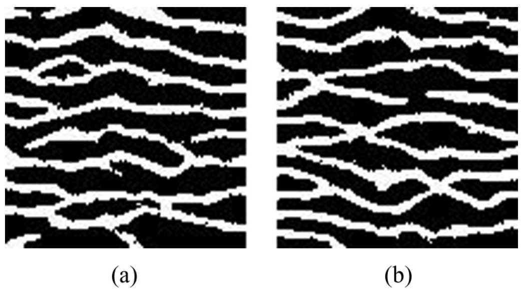
FIG. 15. The first two simulated realizations generated by TSVQ with codebooks of size 128.

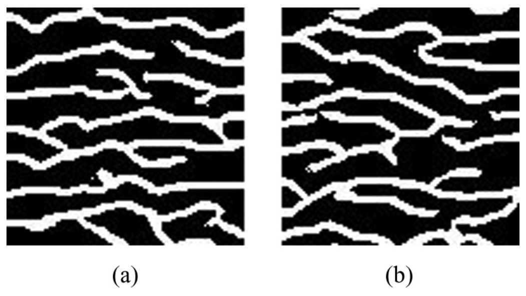
FIG. 17. The first two simulated realizations generated by S-IMPALA.

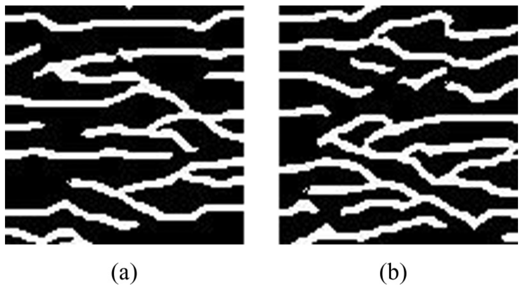
FIG. 18. The first two simulated realizations generated by SNESIM.

$2 \times \log_2 128 = 14$ and $2 \times \log_2 256 = 16$ calculations of Euclidean distance, respectively. In comparison, the two lists in IMPALA were of size 4291 and 7008. On average, Hamming distance was calculated 365.8 times for a single point simulation. Taking into account all previous factors, VQ-MPS with three schemes requires fewer computations by a factor of 1.2, 10.89, and 9.45 than S-IMPALA, respectively. This was verified by experimental data.

An analysis of distance (ANODI) [22] is an important method to quantitatively compare TI-based algorithms in geostatistics. For unconditional simulations, simulated realizations should contain two types of variability: spatial uncertainty and pattern reproduction. The former summarizes the difference between simulated realizations, while the latter represents average similarity between a TI and each simulated realization. The best algorithm should extend spatial uncertainty, as well as maximize pattern reproduction. In this paper, a cluster-based histogram of patterns in ANODI was adopted. Three ratios of $r_{k,m}^{\text{between}}$, $r_{k,m}^{\text{within}}$, and $r_{k,m}^{\text{overall}}$ were defined to relatively rank algorithm k and algorithm m. A small value for $r_{k,m}^{\text{between}}$ indicates that algorithm k has less spatial uncertainty than m. A large value for $r_{k,m}^{\text{within}}$ reveals algorithm k has worse pattern reproducibility than m. The value $r_{k,m}^{\text{overall}}$ summarizes the former two aspects and $r_{k,m}^{\text{overall}} < 1$ tells us that algorithm m outperforms k. The detailed procedure of ANODI is presented in Ref. [22].

For testing the performance of our proposed method, four more MPS programs were considered as benchmarks: SNESIM [17], FDS [30], CCSIM [21], and MS-CCSIM [23]. The template, multigrid strategy, and other configurations in SNESIM were identical to those of VQ-MPS. DS [28] is an important point-based MPS method. FDS accelerates DS using a gradient descent strategy. According to a practical guide in Ref. [29], the neighbors $n = 30$ and threshold $t = 0.2$ were chosen. The above programs were implemented in our version. In addition, CCSIM and MS-CCSIM were used because they are relatively new and competitive methods in pattern-based MPS. Both the two programs are developed by their authors via MATLAB and can be open accessed. We executed these programs using default parameters in Ref. [23]. The first two results obtained by the above four methods are displayed in Figs. 18–21. By applying each scheme in VQ-MPS as algorithm m, the results of comparisons made by ANODI are shown in Tables I–III.

The data in Table II were analyzed first. The values $r_{k,m}^{\text{between}}$ were all greater than 1. This implies that existing MPS programs have larger spatial uncertainty than the second scheme in VQ-MPS. Except for an item in the last row, $r_{k,m}^{\text{within}}$ were all greater than 1, which indicates that the second scheme has a better pattern reproduction quality. The first four $r_{k,m}^{\text{overall}}$ were approximately equal to 1. This shows that TSVQ with codebooks of size 128 yields comparable simulation results to S-IMPALA, SNESIM, FDS, and CCSIM. However, the last $r_{k,m}^{\text{overall}}$ was greater than 1. This indicates that MS-CCSIM is a better algorithm. Then, the data in Tables I and III were analyzed. All $r_{k,m}^{\text{between}}$ were smaller than 1, which indicates that the two schemes of VQ-MPS expand spatial uncertainty. The values $r_{k,m}^{\text{within}}$ were greater than 1, except for the last item in Table I, which reveals that our method improves pattern reproduction ability. As a result, $r_{k,m}^{\text{overall}}$ were all greater than 1. This quantitative analysis validated the improvement made by our proposed VQ-MPS in terms of spatial uncertainty and pattern reproduction. The reason for this phenomenon is that our method is capable of reproducing lowly proportional patterns, as explained in detail in Sec. III C. On the one hand, reproductions of lowly proportional patterns allow the simulated realizations to be more similar to the TI. On the other

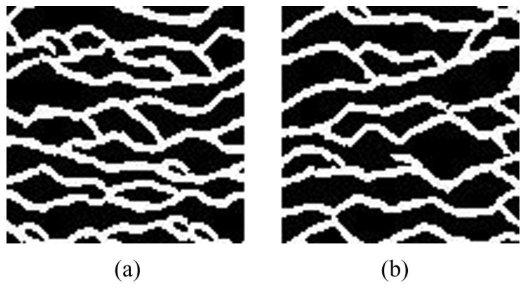
FIG. 20. The first two simulated realizations generated by CCSIM.

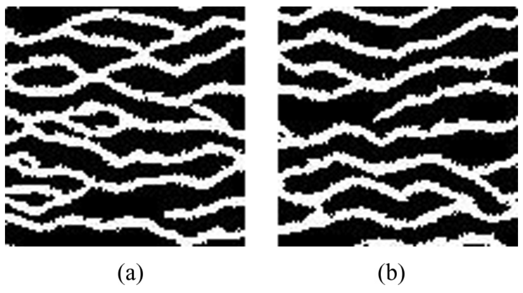
FIG. 19. The first two simulated realizations generated by FDS.

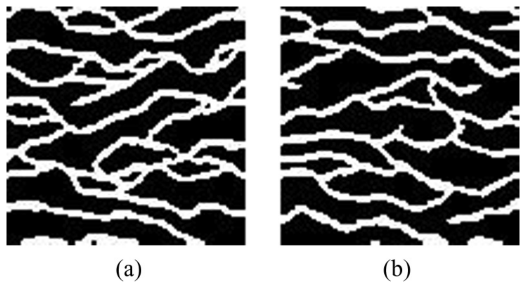
FIG. 21. The first two simulated realizations generated by MS-CCSIM.

TABLE I. Comparison between VQ-MPS with full search VQ with codebooks of size 128 and existing MPS programs.

Algorithm k	$r_{k,m}^{\text{between}}$	$r_{k,m}^{\text{within}}$	$r_{k,m}^{\text{overall}}$
S-IMPALA	0.8480: 1.0000	1.2760: 1.0000	0.6646: 1.0000
SNESIM	0.8440: 1.0000	1.2298: 1.0000	0.6863: 1.0000
FDS	0.8418: 1.0000	1.0401: 1.0000	0.8094: 1.0000
CCSIM	0.8936: 1.0000	1.1019: 1.0000	0.8110: 1.0000
MS-CCSIM	0.9271: 1.0000	0.9808: 1.0000	0.9452: 1.0000

hand, the diversity of patterns results in a greater difference between simulated realizations. As a result, our method yields the best simulation quality. Moreover, the full search VQ has a better performance than TSVQ under the same conditions. The reason is that TSVQ improves computational efficiency at the cost of codebook performance. However, a codebook of a larger size is easy to implement to mitigate this degradation. TSVQ with a codebook of size 256 yielded a competitive result.

B. A 2D application with a four facies stationary training image

In this section, a multifacies simulation was carried out to test VQ-MPS. As Fig. 22(a) shows, a training image of size 101×101 with four facies was applied. This image has already been used in Ref. [20] in order to compare FILTERSIM and DISPAT. In this example, the simulation domain was assumed to be of the same size as the TI. An unconditional simulation strategy was applied and the sample rate was 0.5%. The program adopted a template of size 9×9 and a multigrid strategy with $G = 3$. The height of the tree-structured codebook was 12. Therefore, TSVQ was performed with a codebook of size 2048. Using our proposed categorical variable representation method, four states were mapped into a three-dimensional space. The coordinates of these four fixed points were [0, 0, 0], [1, 0, 0], [0.5, 0.87, 0], and [0.5, 0.29, 0.82], respectively. An extending dimensionality of patterns and a large size of the template bring an increase in the computational load. Consequently, the computer spent 92 s refining three databases. Next, 140 s was consumed to achieve 50 realizations. The first two simulation realizations are shown in Figs. 22(b) and 22(c). It is obvious that VQ-MPS properly preserves the geometrical features of the TI. In the simulation realizations, facies 1 occupied the largest percentage of the simulation domain. Facies 2 exhibited a good connectivity and was surrounded by facies 3. Facies 4 was modestly reproduced near facies 2. Considering the efficiency and effectiveness, we believe that the proposed VQ-MPS provides a feasible way to conduct multifacies simulations.

TABLE II. Comparison between VQ-MPS with TSVQ with codebooks of size 128 and existing MPS programs.

Algorithm k	$r_{k,m}^{\text{between}}$	$r_{k,m}^{\text{within}}$	$r_{k,m}^{\text{overall}}$
S-IMPALA	1.0367: 1.0000	1.2570: 1.0000	0.8247: 1.0000
SNESIM	1.0320: 1.0000	1.2115: 1.0000	0.8518: 1.0000
FDS	1.0289: 1.0000	1.0251: 1.0000	1.0037: 1.0000
CCSIM	1.0928: 1.0000	1.0860: 1.0000	1.0062: 1.0000
MS-CCSIM	1.1336: 1.0000	0.9666: 1.0000	1.1728: 1.0000

TABLE III. Comparison between VQ-MPS with TSVQ with codebooks of size 256 and existing MPS programs.

Algorithm k	$r_{k,m}^{\text{between}}$	$r_{k,m}^{\text{within}}$	$r_{k,m}^{\text{overall}}$
S-IMPALA	0.8165: 1.0000	1.3950: 1.0000	0.5853: 1.0000
SNESIM	0.8126: 1.0000	1.3443: 1.0000	0.6045: 1.0000
FDS	0.8107: 1.0000	1.1372: 1.0000	0.7129: 1.0000
CCSIM	0.8602: 1.0000	1.2023: 1.0000	0.7155: 1.0000
MS-CCSIM	0.8925: 1.0000	1.0742: 1.0000	0.8309: 1.0000

C. A 2D application with spatial transformations of patterns

Affinity and rotation are two typical spatial transformations. Figure 23 shows an illustration of using these transformations to realize a nonstationary simulation. The intensity of each point in Fig. 23(a) represents a rotation angle the value of which is in the range [−180, 180°]. The intensity of each point in Fig. 23(b) represents an affinity ratio that ranges from 1 at the center to 0.4 in the corners. Both Figs. 23(a) and 23(b) are of size 1024×1024. Figure 23(c) shows a stationary channelized TI of size 256×256.

Let θ denote a degree of angle and α denote an affinity ratio at any point in the simulation domain. Equation (9) can be specified as

$$\begin{aligned}[x' \quad y' \quad 1] &= [x \quad y \quad 1] \times \mathbf{T} \\ &= [x \quad y \quad 1] \times \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ &\quad \times \begin{bmatrix} 1/\alpha & 0 & 0 \\ 0 & 1/\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}. \qquad (10)\end{aligned}$$

Our method simulated a nonstationary realization of size 1024×1024, as shown in Fig. 23(d). In this realization, the channels originate from the center and flow in all directions. Meanwhile, the channels gradually become thin at points away from center. This shows our method realizes the spatial transformations of patterns in the simulation domain.

D. A 2D application with one secondary variable

In this application, a secondary variable was incorporated into a 2D simulation. Figure 24(a) shows a channelized reservoir image of size 256×256. We suppose that the origin

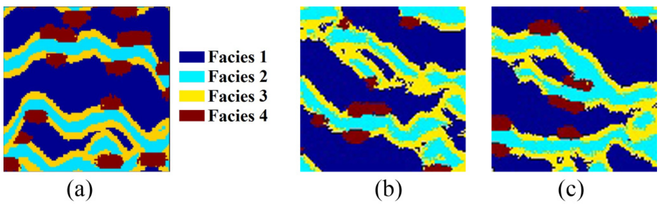

FIG. 22. A 2D application with four facies simulation. (a) The TI. (b), (c) Two realizations.

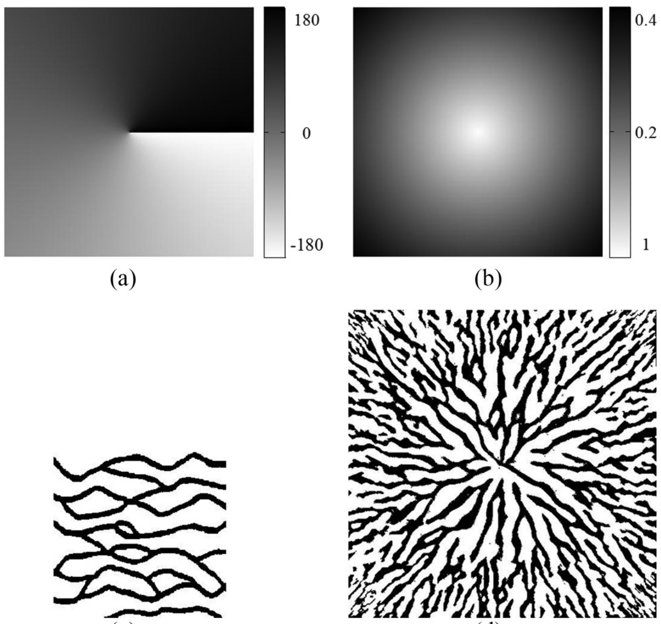

FIG. 23. A 2D application with spatial transformation of patterns. (a) Rotation. (b) Affinity. (c) The TI. (d) A simulated realization.

is at the upper left corner, with the positive x axis extending to the right and the positive y axis extending downward. The directions of channels change smoothly depending on their x coordinates. Based on this interpretation, x coordinates were applied as a secondary variable to describe nonstationarity in the TI. However, this secondary variable was normalized into the range [0, 1] prior to simulation, which is shown in Fig. 24(b).

In the simulation, a realization of size 256×256 with the following conditions was targeted: horizontal channels

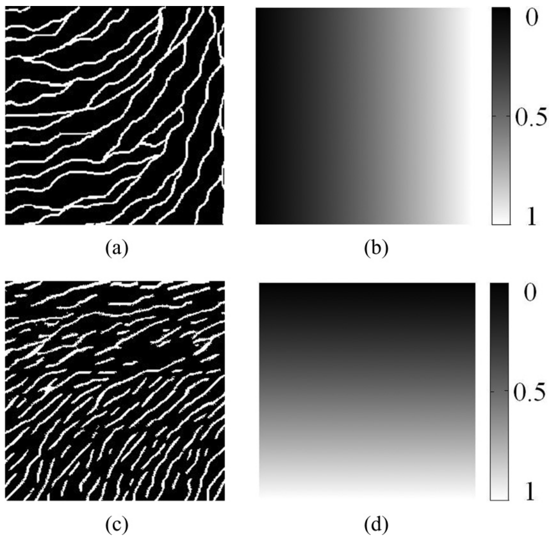

FIG. 24. A 2D application with one secondary variable. (a) The first variable for a nonstationary TI. (b) Secondary variable describing the nonstationarity of the first variable in the TI. (c) A simulated realization for the first variable. (d) Secondary variable describing the nonstationarity of the first variable in the simulation domain.

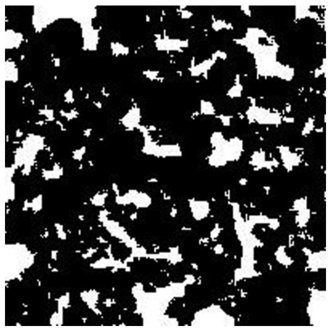

FIG. 25. A rock slice.

that are reproduced at the top, vertical channels that are reproduced on the left, and smooth transition occurring in between. According to these conditions, y coordinates in the simulation domain were used as a secondary variable. Therefore, an image of size 256×256 was created and displayed in Fig. 24(d). The intensity of any point in this image was used to define a value of the secondary variable. Setting the number of categories $J = 10$ and book size $S = 64$, CVQ was applied. In other words, ten codebooks of size 64 were generated. Our method yielded the realization shown in Fig. 24(c). In this realization, horizontal channels occurred at the top, vertical channels occurred on the left, and smooth transitions occurred in between. According to visual interpretation, this realization satisfies the targeted conditions.

E. A 3D simulation from a rock slice

In petroleum engineering, it is not uncommon to simulate a high-dimensional rock model based on a low-dimensional slice. Comunian *et al.* [37] reported several practical methods to complete 3D MPS simulations using 2D TIs. We incorporate VQ-MPS into a framework called probability aggregation to perform 3D simulations. Because of the absence of a 3D TI, the 3D conditional probability is obtained by aggregating probabilities calculated from 2D planes. The linear pooling formula was adopted in this paper to aggregate several probabilities. The framework of probability aggregation is elaborated in Ref. [37].

In this section, a porous medium of sandstone samples was selected for simulations. Suppose that white areas represent pores and black areas represent another material. A binary 2D rock slice of size 200×200, shown in Fig. 25, was used as the TI. This image was created using a microcomputed tomography (CT) technique with a resolution of $10\,\mu$m. The dimensions of the simulated domain were set to $200 \times 200 \times 200$. A template of size 7×7 and a TSVQ with a codebook of size 128 were applied. The sampling rate was set to 0.5%. The porosity of initial hard data was limited to 20.2% because this was the porosity of the TI. In addition, a multigrid scheme $G = 2$ was used. A simulated realization is displayed in Figs. 26(a)–26(c). These figures are compared with the 3D image of original CT sample in Figs. 26(d)–26(f). In exterior view and cross view, green areas represent pores and black areas represent another material. In the perspective image, pore structures are expressed by gray areas and another material is transparent. According to human visual interpretation, pores

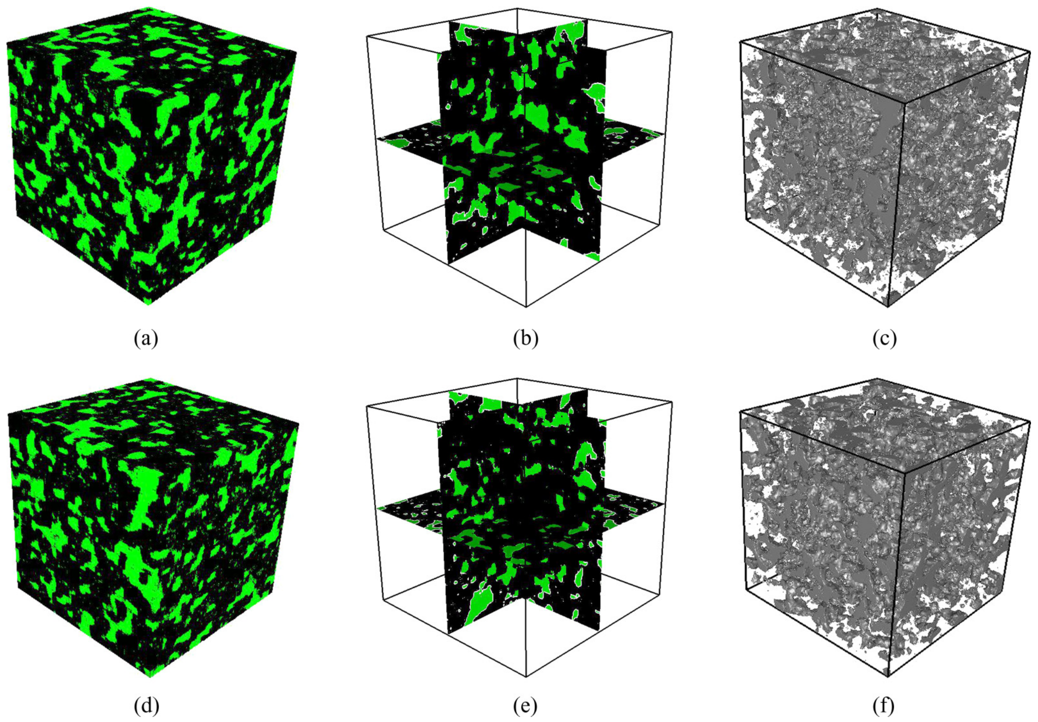

FIG. 26. A simulated realization and a CT sample. (a) 3D exterior view of the simulated realization. (b) A cross section of the simulated realization. (c) The perspective image of the simulated realization. (d) 3D exterior view of the CT sample. (e) A cross section of the CT sample. (f) The perspective image of the CT sample.

in the simulated realization have similar contours and spatial distributions to the CT sample.

Besides qualitative comparison, two quantitative comparisons are also conducted. The autocorrelation function (ACF) and linear path function (LPF) are two frequently used measures of variability and continuity in 3D rock models. The ACF is defined as

$$R(r) = \frac{\langle [I(u) - \phi][I(r+u) - \phi] \rangle}{\phi - \phi^2}, \tag{11}$$

where u is any point in 3D models. $I(u)$ is an indicator function such that $I(u) = 1$ if u lies within a pore and $I(u) = 0$ otherwise. The angular bracket denotes a mean operation. ϕ denotes the porosity of the image and is defined as $\phi = \langle I(u) \rangle$. ACFs of the TI and CT sample and simulated realization are displayed in Fig. 27. It is clear that the ACF of simulated realization agrees with the CT sample very well.

The LPF, which describes local connectivity of pores, is defined as

$$L(r) = \frac{\text{Prob}[I(u) = 1, I(u+1) = 1, \ldots, I(u+r) = 1]}{\phi}, \tag{12}$$

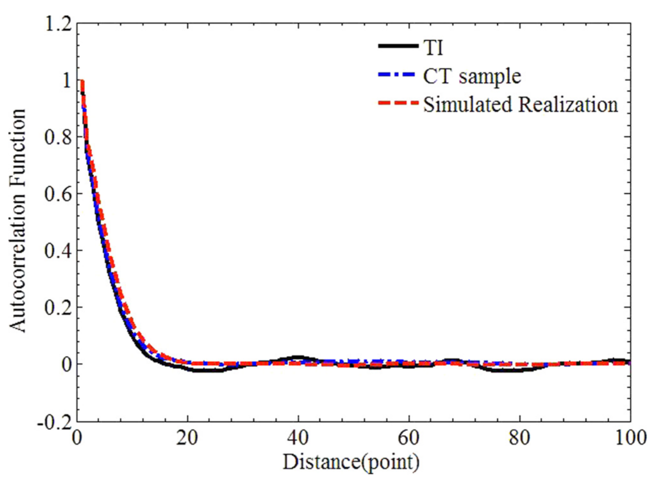

FIG. 27. Comparison of ACFs for TI, CT sample, and simulated realization.

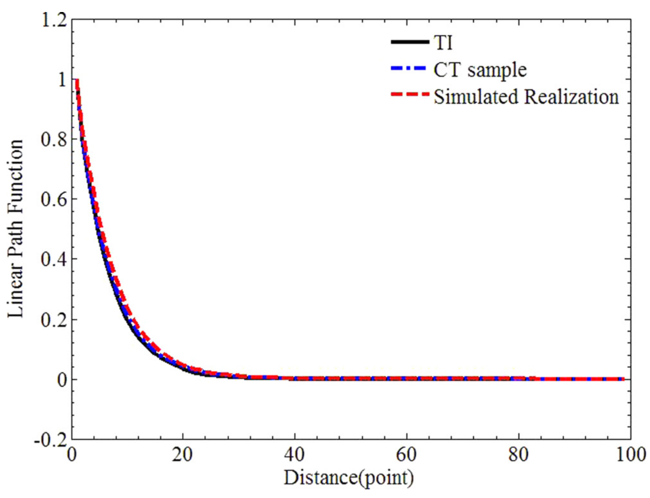

FIG. 28. Comparison of LPFs for TI, CT sample, and simulated realization.

where u is the outset of a line segment l_r of length r. The LPF provides the probability that a line segment l_r is entirely in pore areas. LPFs of the TI and CT sample and simulated realization are displayed in Fig. 28. It is obvious that the simulated realization has matched connectivity with the CT sample. According to previous analyses, the proposed VQ-MPS is capable of reproducing spatial patterns and microstructure of a porous medium from a single 2D image.

V. CONCLUSION

We proposed a VQ-MPS method with the goal of accelerating simulation and improving simulation quality in the MPS framework by using VQ technique to compress databases. First, a variable representation for categorical variables is presented. Each state of categorical variables is mapped into a fixed point in a high-dimensional space. Addition operations and Euclidean distance measurement can be performed with this representation. Second, TSVQ is applied to improve database structure in MPS in order to speed up simulation and ensure high-level simulation quality. Stationary simulations are dealt with. Furthermore, two types of nonstationarity are considered. A transformed template is adopted to address spatial transformations of patterns. CVQ is adopted to incorporate secondary variables into simulations.

We tested our method via a 2D channelized reservoir image. Compared with several other state-of-the-art MPS programs, our method exhibited significantly better performance in terms of computational efficiency, pattern reproduction ability, and spatial uncertainty. Further applications contained a 2D four facies simulation, two nonstationary simulations, and a 3D simulation from a rock slice. The properties of the simulated realizations agreed well with the targeted properties and the real sample. These applications demonstrate that VQ-MPS is a powerful and versatile tool to address various types of geological modelings. For instance, a variety of benchmark experiments are achieved in order to objectively test the proposed VQ-MPS. Furthermore, we do not restrict our attention to a specific research field. However, the core of our paper aims to explore a new way to develop MPS programs. An enhanced technique has a positive influence on expanding applications of MPS. Considering its efficiency and accuracy, we believe that the proposed method of VQ-MPS may potentially have a wide range of practical applications. Since the categorical variable representation increases the pattern dimensionality, reducing the memory usage will be further studied. Moreover, the database compression operation brings computations. Improving the efficiency of the training procedure will also require further research.

ACKNOWLEDGMENTS

This work was supported in part by the Major Projects of the National Natural Science Foundation of China under Grants No. 41390450 and No. 41390454. The authors would like to acknowledge constructive comments from the anonymous referee that prompted us to improve the paper. We are grateful to Dr. Mingliang Gao for helpful discussions, and we also thank Xue Han for providing parts of the materials in the 3D rock simulation.

[1] C. V. Deutsch and A. G. Journel, *GSLIB: Geostatistical Software Library and User's Guide* (Oxford University, New York, 1992).

[2] P. Goovaerts, *Geostatistics for Natural Resources Evaluation* (Oxford University, Oxford, 1997).

[3] C. Deutsch and T. Tran, Comput. Geosci. **28**, 525 (2002).

[4] M. Pyrcz, J. Boisvert, and C. Deutsch, Comput. Geosci. **35**, 1671 (2009).

[5] N. E. M. Dickson, J. C. Comte, P. Renard, J. A. Straubhaar, J. M. McKinley, and U. Ofterdinger, Hydrogeol. J. **23**, 883 (2015).

[6] M. L. Coz, J. Bodin, and P. Renard, J. Hydrol. **545**, 109 (2017).

[7] G. Pirot, J. Straubhaar, and P. Renard, Geomorphology **214**, 148 (2014).

[8] K. S. Cordua, T. M. Hansen, and K. Mosegaard, Math. Geosci. **47**, 317 (2015).

[9] Y. Melnikova, A. Zunino, K. Lange, K. S. Cordua, and K. Mosegaard, Math. Geosci. **47**, 397 (2015).

[10] M. L. Gao, X. H. He, Q. Z. Teng, C. Zuo, and D. D. Chen, Phys. Rev. E **91**, 013308 (2015).

[11] M. L. Gao, Q. Z. Teng, X. H. He, C. Zuo, and Z. J. Li, Phys. Rev. E **93**, 012140 (2016).

[12] T. Zhang, Y. Du, T. Huang, and X. Li, Stoch. Environ. Res. Risk. Assess. **29**, 727 (2015).

[13] H. Rezaee, O. Asghari, M. Koneshloo, and J. M. Ortiz, Stoch. Environ. Res. Risk. Assess. **28**, 1913 (2014).

[14] P. Renard and G. Mariethoz, Math. Geosci. **46**, 129 (2014).

[15] G. Mariethoz and P. Renard, Math. Geosci. **46**, 517 (2014).

[16] F. B. Guardiano and R. M. Srivastava, Geostatistics Tróia '92 **5**, 133 (1993).

[17] S. Strebelle, Math. Geol. **34**, 1 (2002).

[18] G. B. Arpat and J. Caers, Math. Geol. **39**, 177 (2007).

[19] T. Zhang, P. Switzer, and A. Journel, Math. Geol. **38**, 63 (2006).

[20] M. Honarkhah and J. Caers, Math. Geosci. **42**, 487 (2010).

[21] P. Tahmasebi, A. Hezarkhani, and M. Sahimi, Comput. Geosci. **16**, 779 (2012).

[22] X. Tan, P. Tahmasebi, and J. Caers, Math. Geosci. **46**, 149 (2014).

[23] P. Tahmasebi, M. Sahimi, and J. Caers, Comput. Geosci. **67**, 75 (2014).

[24] L. Yang, W. Hou, C. Cui, and J. Cui, Comput. Geosci. **89**, 57 (2016).

[25] S. Strebelle and C. Claude, Math. Geosci. **46**, 171 (2013).

[26] J. Straubhaar, P. Renard, G. Mariethoz, R. Froidevaux, and O. Besson, Math. Geosci. **43**, 305 (2011).

[27] J. Straubhaar, A. Walgenwitz, and P. Renard, Math. Geosci. **45**, 131 (2013).

[28] G. Mariethoz, P. Renard, and J. Straubhaar, Water Resour. Res. **46**, 11536 (2011).

[29] E. Meerschman, G. Piro, G. Mariethoz, J. Straubhaar, M. V. Meirvenne, and P. Renard, Comput. Geosci. **52**, 307 (2013).

[30] M. J. Abdollahifard and K. Faez, Arab. J. Geosci. **7**, 1927 (2014).

[31] J. Straubhaar, P. Renard, and G. Mariethoz, Spat. Stat. **16**, 53 (2016).

[32] M. J. Abdollahifard, Comput. Geosci. **86**, 64 (2016).
[33] Y. Linde, A. Buzo, and R. M. Gray, IEEE T. Commun. **28**, 84 (1980).
[34] A. Buzo, A. H. Gray, R. M. Gray, and J. D. Markel, IEEE Trans. Signal. Proces. **28**, 562 (1980).
[35] J. Makhoul, S. Roucos, and H. Gish, P. IEEE **73**, 1551 (1985).
[36] B. Ramamurthi and A. Gersho, IEEE T. Commun. **34**, 1105 (1986).
[37] A. Comunian, P. Renard, and J. Straubhaar, Comput. Geosci. **40**, 49 (2012).