

1 **Submission of revised version of manuscript titled**
2 **“QuickSampling v1.0: a robust and simplified pixel-**
3 **based multiple-point simulation approach”**

4
5 We would like to thank Ute Mueller and Thomas Mejer Hansen for their valuable
6 feedbacks. Their inputs led to significant improvements in the manuscript.

7 We have incorporated the suggestions made by the reviewers in the revised
8 manuscript. We have added cumulants as multiple points statistics, and
9 significantly improve the discussion (and figures) associated with the kernel to
10 provide the readers a better understanding of this new parameter’s benefits.

11
12 Furthermore, due to reviewers comments and external feedbacks we decided to
13 rename the algorithm from Quantile Sampling to QuickSampling in order to
14 prevent ambiguity.

15
16 We hope that with these additions and all modifications discussed below, the
17 revised manuscript is now ready for publication in Geoscientific Model
18 Development.

19
20 Best regards,
21 Mathieu Gravey
22

23 Reply to Reviewer 1 (Prof. Ute Mueller)

24 The paper describes a new algorithm for multiple point simulation of continuous and
25 discrete spatial variables. To start with a short review of the various types of MPS algorithms
26 is provided, which distinguishes patching from pixel based approaches. The algorithm
27 described here falls into the second category. Shortcomings of the method are discussed
28 briefly, including the need for a threshold and sensitivity of the simulation quality to this
29 threshold, but which can also lead to very long simulation times. In this paper the authors
30 exploit a decomposition of the distance measures to apply FFT to speed up computation of
31 mismatch maps with the aim to more quickly identify candidate patterns in the training
32 image, which may be complete or incomplete. The use of the FFT to compute the mismatch
33 map is attractive in that it is fast to compute irrespective of dimension.

34 We thank Prof. Ute Mueller for her feedback and interest in our work.

35 The mismatch map is calculated by computing for each pair (s, t) a dissimilarity measure
36 where t belongs to the training image and s to the conditioning set. It is this dissimilarity
37 measure which is then identified in terms of cross correlation. The authors provide a
38 description of the metrics applied and a rewrite of the metrics in terms of cross correlations,
39 and while the reader gets a general idea as to what is being calculated the derivation is
40 patchy and somewhat sloppy in that summation indices are missing and critical steps are
41 not described satisfactorily, such as the derivation of equation 9, which introduces cross
42 correlations.

43 We improved the overall readability by introducing “sets” in summation. Equation 8 was
44 extended to add the reordering of the summation (Line 236), and the definition of cross-
45 collation was added between equation 8 and 9 (Line 238) to enhance readability of the
46 derivation.

47 Also, is it correct to assume that “ l ” is a grid operator?

48 l was properly defined as a relative displacement (Line 211).

49 Once the mismatch map is computed, the k best matches are identified and a sample is
50 drawn at random from this pool. The possibility of having non-integer values for k is touched
51 upon, and allow unequal weighting of the first ceiling(k) candidates, with the first floor(k)
52 candidates equally likely and the final candidate less likely (probability of being chose): $1-$
53 $\text{floor}(k)/k$. The main advantage appears to lie in being able to choose between 2 instead of
54 just one candidate (case of k between 1 and 2)

55 The interest of partial k , in particular the relation with Direct Sampling (DS) was further
56 developed at Line 304 -308.

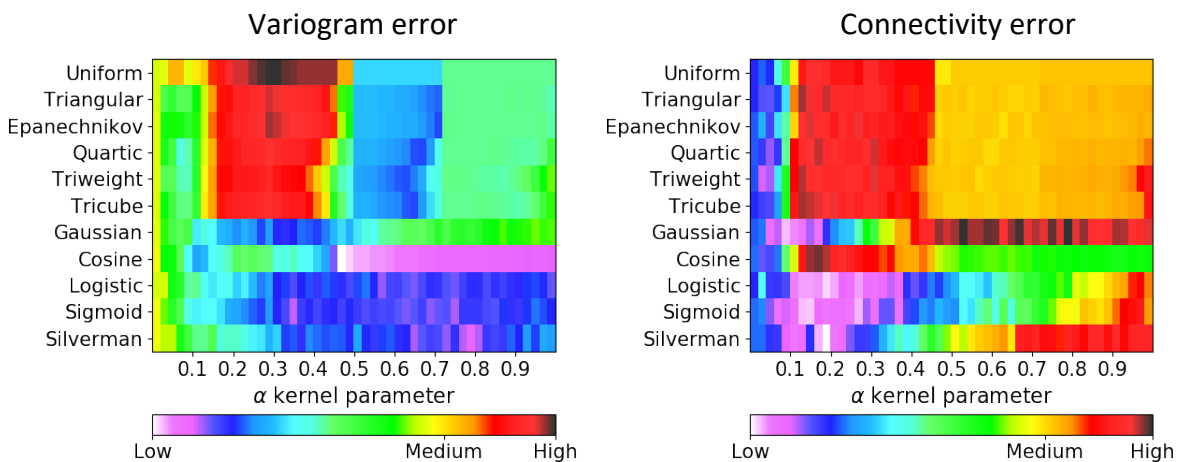
57 Simplifications and computational implementation details for speeding up the computation
58 are discussed reasonably thoroughly and provide other practitioners with useful
59 suggestions on how to potentially improve the efficiency of their own MPS algorithms. The
60 proposed algorithm is benchmarked by means of standard sample data sets and a sensitivity
61 analysis is provided demonstrating that QS performs well subject to the choice of a suitable
62 kernel and that the quality of QS simulations is similar to that of DS simulations. It would
63 have been interesting to see an exploration of kernels other than one of Gaussian type.

64 While we agree that a full exploration of kernel possibilities would be interesting, it will be the
65 subject of a future paper.

66 To be more precise about this point, preliminary experimentations on various kernel radial
67 designs have shown that it is not straightforward to define general guidelines for an optimal
68 kernel. Figure 1 below shows some of these preliminary results, where exhaustive kernel
69 parameter exploration is carried out and identifies areas of higher performance in terms of
70 reproducing variogram and connectivity function. While it is clear that some kernel functions
71 perform better than others, it seems that the results are highly specific to the type of patterns
72 to reproduce and should be further investigated.

73 Furthermore, tests using non-parametric kernels show a potential for future improvements.
74 However, substantial future research is still needed on this topic, which will be the object of a
75 future publication.

76
77



78
79

Figure 1 Errors for different types of radial kernel based on the stone image.

80 In addition, with improved Figure 8 by adding the Euler characteristics, that allows readers to
81 get a better idea of the potential and the interest of this kernels.

82 Because our formulation may have been unclear, in the sense that the example kernel used
83 in this manuscript has an exponential form and not Gaussian. Therefore, we clarified this point
84 at Line 423 by specifying that $\|\cdot\|_2$ represents the Euclidean distance.

85 **Also, the metrics being used to assess the performance would benefit from going beyond**
86 **variograms and connectivity (I acknowledge that the Euler characteristic was also used, but**
87 **what good is it without a definition? Reference to another paper is all fine and well, but a**
88 **definition and an explanation of what it measures would have been nice.)**

89 **It would be really nice to see an evaluation in terms of a multipoint statistics.**

90 We provided some brief description of each metrics (Line 407 – 412). We carried out a
91 validation for our realizations in terms of the reproduction of higher-order statistics, using
92 cumulants as a metric in Figure 6.

93 **Please amend all the formulae to ensure summation indices are clear, eg: Line 149: It is not**
94 **clear over what is summed in equation 1.**

95 Unfortunately, in Equation 1, line 150, it is impossible to know in advance the number of
96 elements or the set for the summation. The description is really generic here and needs to be

97 adapted for each required metric as shown for the L_2 and Hamming metrics. However, we
98 add and define proper ensemble \mathcal{J} for each summation to help the reader.

99 **You clarify this to some extent below in lines 150 to 183, but I find this a little unsatisfying**
100 **Line 174: The description preceding equation 2 talks about vectors, but the formula seems**
101 **to be univariate.**

102 We agree this is unclear as the “vector” in line 173, was referring to the origin of the Hamming-
103 distance. We rephrase this sentence to remove any confusion for the audience as following:
104 “The Hamming distance measures the dissimilarity between two lists by counting the number
105 of elements that have different categories (Hamming, 1950)”.

106 **If you have c categories, is “a” a vector with c entries or simply one of the values from 1 to**
107 **c if you label the categories in that manner?, It looks to me that “a” is simply a category ...**
108 **so looking at the equation, it would seem that it is equal to c, if “a” and “b” are distinct and**
109 **equal to c-1 if they are equal, while the sum on the right is equal to 1 if “a” and “b” are equal**
110 **and 0 else. There are also brackets missing in the middle expression (you should have**
111 **$\sum_{j \in C} (1 - \delta_{aj}) \delta_{bj}$)**

112 The description with categorical cases mentioned by the reviewer is correct (and we don’t
113 need to number from 1 to c, and it is not the case in the implementation either). We thank
114 the reviewer for spotting the error of the equation 3. Indeed, it should be: $1 - \sum_{j \in C} (\delta_{a,j} \cdot \delta_{b,j})$
115 and not $\sum_{j \in C} 1 - (\delta_{a,j} \cdot \delta_{b,j})$. Therefore, now equation 3 is:

116
$$\epsilon_{L^0}(a, b) = 1 - \sum_{j \in C} \delta_{a,j} \cdot \delta_{b,j} \propto \sum_{j \in C} \delta_{a,j} \cdot \delta_{b,j}$$

117 The linear transformation between both sides of the proportional symbol is $y=ax+b$, with $a=-$
118 1 , and $b=1$. The manuscript is updated accordingly (Line 179).

119 **Line 200: $N(t)$ is not just a location but a neighbourhood?**

120 $N(t)$ is indeed a neighborhood. We clarify this point in Line 210-212.

121 **Please clarify Line 230: define the cross-correlation operator. Also, T_i has not been defined.**
122 **You identify “*” with convolution and then apply the convolution theorem. Provide a**
123 **derivation that this is true in an appendix.**

124 * represents the cross-correlation and therefore the “convolution theorem” is applied as
125 follows: $\mathcal{F}(x \star y) = \overline{\mathcal{F}(x)} \circ \mathcal{F}(y)$. This is contrarily to a convolution * where we get $\mathcal{F}(x \star$
126 $y) = \mathcal{F}(x) \circ \mathcal{F}(y)$. Some references and a proper derivation of this property were added in
127 the Annex Line 684. Furthermore, T_i is properly defined in Line 244.

128 **There are also some typos in the figure captions**

129 Captions were checked and corrected in consequence.

130

131 Response to Reviewer 2 (Prof. Thomas Mejer Hansen)

132 The authors present a novel multiple point statistical simulation algorithm that works for
133 both discrete and continuous data, that scales well on parallel computing architectures, and
134 that is available as open-source C++ code (G2S) with interfaces in Matlab, Python and R.

135 At the core of the method is the use of convolution to very efficiently compute to compute
136 a mismatch, between a conditional event (consisting of the 'N' closest hard/simulated data)
137 centered at all locations in the TI (except near the boundaries) (2.3) Then the authors
138 suggest to simulate the current pixel based on a random selection between the 'k' centered
139 pixel values associated with the smallest mismatch (2.4)

140 This leads to an algorithm with only two main 'tuning parameters'. The algorithm is in-itself
141 novel and has obvious potential for used instead of some of the currently widely used MPS
142 methods. The examples in the manuscript nicely describe the potential uses. In addition,
143 the way the algorithm has been implemented should be applauded, as it is available as Open
144 Source code that can be used with ease ranging from a case of "running on a single thread
145 on a laptop in python/Matlab", to "running remote on a large cluster". This makes the code
146 very versatile.

147 Therefore I find the manuscript highly appropriate for publication.

148 Thanks a lot for the positive feedback!

149 I have one major comment, that relate to the name of the algorithm and the way a pixel
150 value is chosen based on 'k' smallest values of E/mismatch. The authors refer to these 'k'
151 smallest values of E as a "quantile" and call the algorithm, for quantile sampling. This I do
152 not understand and find a bit misleading. How can this represent a quantile? I think the term
153 'threshold' would be more fitting than 'quantile'.

154 The question of the algorithm name is something that has been extensively discussed
155 between authors. We believe that the use of the term "threshold" would bring confusion with
156 the Direct Sampling algorithm, which uses a threshold in the error.

157 One name that was originally discussed for our algorithm is "Quick Sampling". According to
158 the justified comment by the reviewer, we propose to use this name as a replacement, as it
159 allows keeping the acronym "QS" with which users are already familiar.

160 The use of the term "quantile" suggests that the selection of the new pixel value is based of
161 a probabilistic measure. Also, say 'k=18', and for a discrete case only 9 pixel values are
162 associated with a mismatch of '0'. Why would one want to use the same probability
163 ($P=9/18$) to select one of these, as opposed to one of the pixel values associated with a non-
164 perfect match ($P=9/10$)? Or more extreme, say that pixel associated with the 18th best
165 mismatch has a mismatch of 10 pixels. Why would one want to assign the same probability
166 ($1/18$) to this, as to the pixels with a mismatch of 0? The use of the 'k'-'threshold' is
167 convenient, but to me it makes the method less clear to describe in terms of the implied
168 statistical assumptions. Some discussion on 'quantile' vs 'threshold' would be good.

169 This hypothesis is similar to the use of a distance threshold in DS and should indeed be
170 discussed. k in QS is statistically similar to DS with a threshold at 0 and a fraction $f=1/k$ (this is
171 one of the reasons for using decimal values for k) under the hypothesis of stationarity.

172 This equivalence was added in section 2.4, to allow readers to get a better understanding
173 about the relation between QS and DS, and the utility of non-integer values of k.

174 The question of the reviewer could be turned around “why to not using the best candidate
175 (k=1)?”. The main answer is to limit verbatim copy, because the random selection between
176 candidates with similar mismatch (algorithm presented in Appendix A.1) significantly limits
177 this problem in cases of a training image with replicated patterns. The problem remains for
178 other images, and especially continuous variable images where often few replicated patterns
179 exist.

180 **Some comments to the text:**

181 **Line 150: Here ‘a’ and ‘b’ are referred to as “univariate pixel values”. It seems ‘a’ and ‘b’ has
182 a different meaning in line 174 (eqn 3)? Here they seem to represent vectors?**

183 “a” and “b” represent each time one possible class. This section was corrected and clarified to
184 remove this ambiguity, also according to the comments by reviewer 1. In addition, in order to
185 help readers to better understand how this metric works, a short example was added in Line
186 176.

187 **Line 185, Eqn 5: Please elaborate a bit on how this allows mixing discrete and continuous
188 variables calculating the mismatch? It seems nontrivial to compute the mismatch between
189 for example a velocity of 2.1 km/s and a “lithology of type A” to a velocity of 2.13 km/s and
190 “lithology of type C”?**

191 The task of combining continuous and categorical variables is indeed challenging, and has
192 been so for all MPS approaches. From the literature and practical use of the software, we
193 know that this problem is general to most MPS methods and that many strategies can be used.
194 One can use a different distance threshold for each variable (as done in the DEESSE
195 implementation), or a linear combination of the normalized errors (as done in the DS
196 implementation). Here we use the second approach, taking advantage of the linearity of the
197 Fourier transform. If the relative importance can be set in the “ f_i ” or “ g_i ” functions in equation
198 1, it is computationally advantageous to use the kernel weights such as to have standard
199 functions for each metric.

200 If the task of setting such variable-dependent parameters is complex, one can use the results
201 of recent research to identify the optimal parameterization using stochastic optimization
202 approaches, as in (Baninajar et al.), which can and has been applied on QS. This discussion was
203 added in the revised manuscript Line 515-523.

204

205 Baninajar, E., Sharghi, Y. & Mariethoz, G. MPS-APO: a rapid and automatic parameter
206 optimizer for multiple-point geostatistics. *Stoch Environ Res Risk Assess* 33, 1969–1989
207 (2019). <https://doi.org/10.1007/s00477-019-01742-7>

208

209 **Figure 1: What do the red dots in the middle small figure?**

210 Red/Purple dots represent missing and unusable (border effect) data. The caption of Figure 1
211 was updated accordingly.

212 **Line 287: Please explain clearly what is meant by “verbatim copy”. The term is used several
213 places without a proper definition.**

214 A short explanation and a reference were added in the introduction (Line 64-66) to clarify the
215 meaning of verbatim copy:

216 Verbatim copy (Mariethoz and Caers, 2014) refers to the phenomenon whereby a neighbor of
217 a pixel in the simulation is the neighbor in the training image. This results in large parts of the
218 simulation that are identical to the training image.

219 **Line 338: Please explain “NUMA-aware” or provide a reference.**

220 NUMA stands for “Non-Uniform Memory Access” and refers to memory communication
221 between many CPU sockets (such as bi-Xeon). This connection has a limited bandwidth and
222 therefore minimizing the communication on it can significantly increase the speed of the
223 algorithm on such architectures. This can have a huge impact when running on workstations
224 or clusters computers. The manuscript is updated with a reference.

225 **Line 392: What is meant by “..enables adaption of the parameterization. . .”?**

226 Here we meant that it allows for fine-tuning the parameterization. Line 419 was updated in
227 accordingly.

228 **Figure 5: Please help the reader here: is Qs with a kernel better than QS with no kernel? I
229 am not sure what the figure tells us?**

230 The Figure 5 simply show that the results have similar quality and variability DS, which is
231 considered a reference in the field. The benefits of Kernel is further detailed at Line 450-452
232 and in Figure 8

233 **Line 399, Figure 6: Perhaps you could elaborate a little bit on “Euler characteristic” and
234 whether it is a problem what Figure 6 shows?**

235 It is added to the manuscript, according to the comments by reviewers. This is further
236 discussed by extending the description of the metrics in Line 409-411 and in particular with its
237 interpretation in Line 450-452.

238 **Figure 8: I need some help appreciating how Figure 8 suggests that the use of alpha is useful?**

239 We agree that this figure does not illustrate very well the use of the alpha parameter.
240 However, over many tests, and as confirmed by feedback of early users, this parameter does
241 allow a fine tuning of the simulation and is therefore an interesting tool, especially for
242 conditional simulations with an exhaustively informed covariable. The goal here is to make
243 the reader aware of this possibility. Therefore, we changed figure 8 to better show the
244 sensitivity in particular by adding Euler characteristics curve for each case and improving the
245 discussion.

246 **Figure 9: Please show the ‘dots’ (the actual CPU time measurements) in the figures. Is it fair
247 to say that the main limitation of the using QS is the size of the training image?**

248 The “dots” are added to the figure. We agree that the main limitation of QS is the TI size
249 because its relation to computing time evolves in $O(n \cdot \ln(n))$ due to the FFT computation. Even
250 with the use of solutions such as window convolution (often used in audio processing), in our
251 tests the improvements are only noticeable for huge TIs. While such approaches do bring an
252 improvement and tend to reduce the memory footprint, they also add significant complexity
253 to the algorithm for a minor gain. Over the last decade, convolution techniques have been
254 substantially improved, driven by the needs of Convolutional Neural Networks, but are often
255 applied to small matrices (e.g. 3x3, 5x5, or 7x7). Other solutions are available to increase the
256 speed of the convolution such as GPUs or FPGAs that we are still investigating.

257 Note that a dedicated CUDA implementation of QS is available in our repository, but there is
258 still work in progress and at this stage we prefer not to include a detailed description of it in
259 the paper.

260 A short discussion about this point is added in Line 534-537.

261 **Lines 466-472. It is nice that one can choose to use many conditional point with not extra**
262 **CPU costs. one could though argue that sometimes it is convenient in other MPS methods**
263 **(SNESIM/IMPALA/DS) that the simulation becomes MUCH faster if one uses few**
264 **conditioning data. If you would want to simulate with fewer conditioning data, QS would**
265 **not lead to faster CPU time.. Just to say that the advantage you describe, could in a specific**
266 **context, be seen as the opposite.**

267 We completely agree with this point. However, many researchers in the field tend to reduce
268 the simulation quality in order to gain in time or memory space. The point here was to explain
269 that whatever the parameterization (and quality), the computation time is identical.
270 Therefore, it is better to choose parameters yielding a good quality simulation.

271 This is an important discussion point: QS will be fast with a small image, whereas for
272 (SNESIM/IMPALA) it is only partially true because of the overhead related to the creation of
273 the list/tree. Similarly, QS is insensitive to the complexity of the training image (number of
274 patterns available), whereas SNESIM/IMPALA/DS are highly sensitive to it. QS is therefore
275 more adapted to TIs with complex features and few repetitions. We however agree that
276 SNESIM/IMPALA/DS will simulate significantly faster a simple and repetitive TI than QS, but in
277 such cases computation time is generally not a critical issue anyway.

278 This was already discussed, but we updated it to add the relation with IMPALA and SNESIM.

279 **Some of the figures and tables in Appendix A should be excluded unless they are discussed**
280 **and references in the text.**

281 Missing references are added in the text.

282

283 Response to Short Comment 1 (Executive editor of GMD Astrid Kerkweg)
284

285 **Please add a version number for the QS in the title upon your revised submission to GMD.**

286 *As suggested we add the versioning identifier in the title of the manuscript.*

QuickSampling v1.0: a robust and simplified pixel-based multiple-point simulation approach

Mathieu Gravey¹, Grégoire Mariethoz¹

¹ University of Lausanne, Faculty of Geosciences and Environment, Institute of Earth Surface Dynamics, Switzerland

Correspondence to: Mathieu Gravey (mathieu.gravey@unil.ch)

Highlights

- A new approach is proposed for pixel-based multiple-point geostatistics simulation.
- The method is flexible and straightforward to parametrize.
- It natively handles continuous and multivariate simulations.
- High computational performance with predictable simulation times.
- A free and open-source implementation is provided.

Abstract

Multiple-point geostatistics enable the realistic simulation of complex spatial structures by inferring statistics from a training image. These methods are typically computationally expensive and require complex algorithmic parametrizations. The approach that is presented in this paper is easier to use than existing algorithms, as it requires few independent algorithmic parameters. It is natively designed for handling continuous variables, and quickly implemented by capitalizing on standard libraries. The algorithm can handle incomplete training images of any dimensionality, with categorical or/and continuous variables, and stationarity is not explicitly required. It is possible to perform unconditional or conditional simulations, even with exhaustively informed covariates. The method provides new degrees of freedom by allowing kernel weighting for pattern matching. Computationally, it is adapted to modern architectures and runs in constant time. The approach is benchmarked against a state-of-the-art method. An efficient open-source implementation of the algorithm is released and can be found here (<https://github.com/GAIA-UNIL/G2S>), to promote reuse and further evolution.

Keywords

Multiple-point statistics, stochastic simulation, continuous variable, training image, cross-correlation, Fourier transform.

1. Introduction

Geostatistics is used widely to generate stochastic random fields for modeling and characterizing spatial phenomena such as Earth surface features and geological structures. Commonly used methods, such as the sequential Gaussian simulation (Gómez-Hernández and Journel, 1993) and turning bands algorithms (Matheron, 1973), are based on kriging (e.g., Graeler et al., 2016; Li and Heap, 2014; Tadić et al., 2017; 2015). This family of approaches implies spatial relations using exclusively pairs of points and expresses these relations using

37 covariance functions. In the last two decades, multiple point statistics (MPS) emerged as a
38 method for representing more complex structures using high-order nonparametric statistics
39 (Guardiano and Srivastava, 1993). To do so, MPS algorithms rely on training images, which
40 are images with similar characteristics to the modeled area. Over the last decade, MPS has been
41 used for stochastic simulation of random fields in a variety of domains such as geological
42 modeling (e.g., Barfod et al., 2018; Strebelle et al., 2002), remote sensing data processing (e.g.,
43 Gravey et al., 2019; Yin et al., 2017), stochastic weather generation (e.g., Oriani et al., 2017;
44 Wojcik et al., 2009), geomorphological classification (e.g., Vannamettee et al., 2014) and
45 climate model downscaling (a domain that has typically been the realm of kriging-based
46 methods (e.g., Bancheri et al., 2018; Jha et al., 2015; Latombe et al., 2018)).

47 In the world of MPS simulations, one can distinguish two types of approaches. The first
48 category is the patch-based methods, where complete patches of the training image are imported
49 into the simulation. This category includes methods such as SIMPAT (Arpat and Caers, 2007)
50 and DISPAT (Honarkhah and Caers, 2010), which are based on building databases of patterns,
51 and image quilting (Mahmud et al., 2014), which uses an overlap area to identify patch
52 candidates, which are subsequently assembled using an optimal cut. CCSIM (Tahmasebi et al.,
53 2012) uses cross-correlation to rapidly identify optimal candidates. More recently, Li (2016)
54 proposed a solution that uses graph-cuts to find an optimal cut between patches, which has the
55 advantage of operating easily and efficiently independently of the dimensionality of the
56 problem. Tahmasebi (2017) propose as a solution that is based on “warping” in which the new
57 patch is distorted to match the previously simulated areas. For a multivariate simulation with
58 an informed variable, Hoffmann (2017) presented an approach for selecting a good candidate
59 based on the mismatch of the primary variable, and on the mismatch rank of the candidate
60 patches for auxiliary variables. Although patch-based approaches are recognized to be fast, they
61 are typically difficult to use in the presence of dense conditioning data. Furthermore, patch-
62 based approaches often suffer from a lack of variability due to the pasting of large areas of the
63 training image, which is a phenomenon that is called verbatim copy. Verbatim copy (Mariethoz
64 and Caers, 2014) refers to the phenomenon whereby the neighbor of a pixel in the simulation
65 is the neighbor in the training image. This results in large parts of the simulation that are
66 identical to the training image.

67 The second category of MPS simulation algorithms consists of pixel-based algorithms, which
68 import a single pixel at the time instead of full patches. These methods are typically slower than
69 patch-based methods. However, they do not require a procedure for the fusion of patches, such
70 as an optimal cut, and they allow more flexibility in handling conditioning data. Furthermore,
71 in contrast to patch-based methods, pixel-based approaches rarely produce artifacts when
72 dealing with complex structures. The first pixel-based MPS simulation algorithm was
73 ENESIM, which was proposed by Guardiano and Srivastava, 1993, where for a given
74 categorical neighborhood – usually small – all possible matches in the training image are
75 searched. The conditional distribution of the pixel to be simulated is estimated based on all
76 matches, from which a value is sampled. This approach could originally handle only a few
77 neighbors and a relatively small training image; otherwise, the computational cost would
78 become prohibitive and the number of samples insufficient for estimating the conditional
79 distribution. Inspired by research in computer graphics, where similar techniques are developed

80 for texture synthesis (Mariethoz and Lefebvre, 2014), an important advance was the
81 development of SNESIM (Strebelle, 2002), which proposes storing in advance all possible
82 conditional distributions in a tree structure and using a multigrid simulation path to handle large
83 structures. With IMPALA, Straubhaar (2011) proposed reducing the memory cost by storing
84 information in lists rather than in trees. Another approach is direct sampling (DS) (Mariethoz
85 et al., 2010), where the estimation and the sampling of the conditional probability distribution
86 are bypassed by sampling directly in the training image, which incurs a very low memory cost.
87 DS enabled the first use of pixel-based simulations with continuous variables. DS can use any
88 distance formulation between two patterns; hence, it is well suited for handling various types
89 of variables and multivariate simulations.

90 In addition to its advantages, DS has several shortcomings: DS requires a threshold – which is
91 specified by the user – that enables the algorithm to differentiate good candidate pixels in the
92 training image from bad ones based on a predefined distance function. This threshold can be
93 highly sensitive and difficult to determine and often dramatically affects the computation time.
94 This results in unpredictable computation times, as demonstrated by Meerschman (2013). DS
95 is based on the strategy of randomly searching the training image until a good candidate is
96 identified (Shannon, 1948). This strategy is an advantage of DS; however, it can also be seen
97 as a weakness in the context of modern computer architectures. Indeed, random memory access
98 and high conditionality can cause 1) suboptimal use of the instruction pipeline, 2) poor memory
99 prefetch, 3) substantial reduction of the useful memory bandwidth and 4) impossibility of using
100 vectorization (Paul Shen, 2018). While the first two problems can be addressed with modern
101 compilers and pseudorandom sequences, the last two are inherent to the current memory and
102 CPU construction.

103 This paper presents a new and flexible pixel-based simulation approach, namely,
104 [QuickSampling](#) (QS), which makes efficient use of modern hardware. Our method takes
105 advantage of the possibility of decomposing the standard distance metrics that are used in MPS
106 (L^0, L^2) as sums of cross-correlations. As a result, we can use fast Fourier transforms (FFTs) to
107 quickly compute mismatch maps. To rapidly select candidate patterns in the mismatch maps,
108 we use an optimized partial sorting algorithm. A free, open-source and flexible implementation
109 of QS is available, which is interfaced with most common programming languages (C/C++,
110 MATLAB, R, and Python 3).

111 The remainder of this paper is structured as follows: Section 2 presents the proposed algorithm
112 with an introduction to the general method of sequential simulation, the mismatch measurement
113 using FFTs and the sampling approach of using partial sorting followed by methodological and
114 implementation optimizations. Section 3 evaluates the approach in terms of quantitative and
115 qualitative metrics via simulations and conducts benchmark tests against DS, which is the only
116 other available approach that can handle continuous pixel-based simulations. Section 4
117 discusses the strengths and weaknesses of QS and provides guidelines. Finally, guidelines and
118 the conclusions of this work are presented in Section 5.

119 2. Methodology and Implementation

120 2.1. Pixel-based sequential simulation

121 We recall the main structure of pixel-based MPS simulation algorithms (Mariethoz and Caers,
122 2014, p.156), which is summarized and adapted for QS in Pseudocode 1. The key difference
123 between existing approaches is in lines 3 and 4 of Pseudocode 1, when candidate patterns are
124 selected. This task is the most time-consuming in many MPS algorithms and we focus only on
125 computing it in a way that reduces its cost and minimizes the parameterization.

126

127 Pseudocode 1: QS Algorithm

128

129 Inputs:

130 T the training images

131 S the simulation grid, including the conditioning data

132 P the simulation path

133 The choice of pattern metric

134

135 1. **For** each unsimulated pixel x following the path P :

136 2. Find the neighborhood $N(x)$ in S that contains all previously simulated or conditioning
137 nodes in a specified radius

138 3. Compute the mismatch map between T and $N(x)$: [Section 2.3](#)

139 4. Select a good candidate using quantile sorting over the mismatch map: [Section 2.4](#)

140 5. Assign the value of the selected candidate to x in S

141 6. **End**

142

143 2.2. Decomposition of common mismatch metrics as sums of products

144 Distance-based MPS approaches are based on pattern matching (Mariethoz and Lefebvre,
145 2014). Here, we rely on the observation that many common matching metrics can be expressed
146 as weighted sums of the pixelwise mismatch ε . This section explores the pixelwise errors for a
147 single variable and for multiple variables. For a single variable, the mismatch metric ε between
148 two pixels is the distance between two scalars or two classes. In the case of many variables, it
149 is a distance between two vectors that are composed by scalars, by classes, or by a combination
150 of the two. Here, we focus on distance metrics that can be expressed in the following form:

151

Equation 1

152

$$\varepsilon(a, b) \propto \sum_{j \in \mathcal{J}} f_j(a) \cdot g_j(b)$$

153

154

155

156

157

158

159

160

161

162

163

164

165

166

170

171

where a and b represent the values of two univariate pixels and f_j and g_j are functions that depend on the chosen metric. \mathcal{J} is defined by the user depending on the metric used. Here, we use the proportion symbol because we are interested in relative metrics rather than absolute metrics, namely, the objective is to rank the candidate patterns. We show below that many of the common metrics or distances that are used in MPS can be expressed as Equation 1.

For the simulation of continuous variables, the most commonly used mismatch metric is the L^2 -norm, which can be expressed as follows:

Equation 2

$$\varepsilon_{L^2}(a, b) = (a - b)^2 = a^2 - 2ab + b^2$$

Using Equation 1, this L^2 -norm can be decomposed into the following series of functions f_j and g_j :

$$f_0: x \rightarrow x^2 \qquad \qquad \qquad g_0: x \rightarrow 1$$

$$f_1: x \rightarrow -2x \qquad \qquad \qquad g_1: x \rightarrow x$$

$$f_2: x \rightarrow 1 \qquad \qquad \qquad g_2: x \rightarrow x^2$$

172

173 A similar decomposition is possible for the L^0 -norm (also called Hamming distance), which is
174 commonly used for the simulation of categorical variables. The Hamming distance measures
175 the dissimilarity between two lists by counting the number of elements that have different
176 categories (Hamming, 1950). Example the dissimilarity between a,b,b,c,b,a and a,c,b,a,c,a is
177 0,1,0,1,1,0 and the associated Hamming distance is 3.

178

Equation 3

179

$$\varepsilon_{L^0}(a, b) = (a - b)^0 = 1 - \sum_{j \in \mathcal{C}} (\delta_{a,j} \cdot \delta_{b,j}) \propto \sum_{j \in \mathcal{C}} \delta_{a,j} \cdot \delta_{b,j}$$

180

where $\delta_{x,y}$ is the Kronecker delta between x and y , which is 1 if x equals y and 0 otherwise,
181 and \mathcal{C} is the set of all possible categories of a specified variable. Here $\mathcal{J} = \mathcal{C}$.

182

Using Equation 1, this L^0 distance can be decomposed (Arpat and Caers, 2007) into the
183 following series of functions f_j and g_j :

184

$$f_j: x \rightarrow -\delta_{xj}$$

185

$$g_j: x \rightarrow \delta_{xj}$$

186

with a new pair of f_j and g_j for each class j of \mathcal{C} .

187

For multivariate pixels, such as a combination of categorical and continuous values, the
188 mismatch ε can be expressed as a sum of univariate pixelwise mismatches.

189

Equation 4

190

$$\varepsilon(\mathbf{a}, \mathbf{b}) \propto \sum_{i \in \mathcal{J}} \sum_{j \in \mathcal{J}_i} f_j(a_i) \cdot g_j(b_i)$$

191

where \mathbf{a} and \mathbf{b} are the compared vectors and a_i and b_i are the individual components of \mathbf{a} and
192 \mathbf{b} . \mathcal{J}_i represents the set related to the metric used for the i^{st} variable, and \mathcal{J} represents the set of
193 variables.

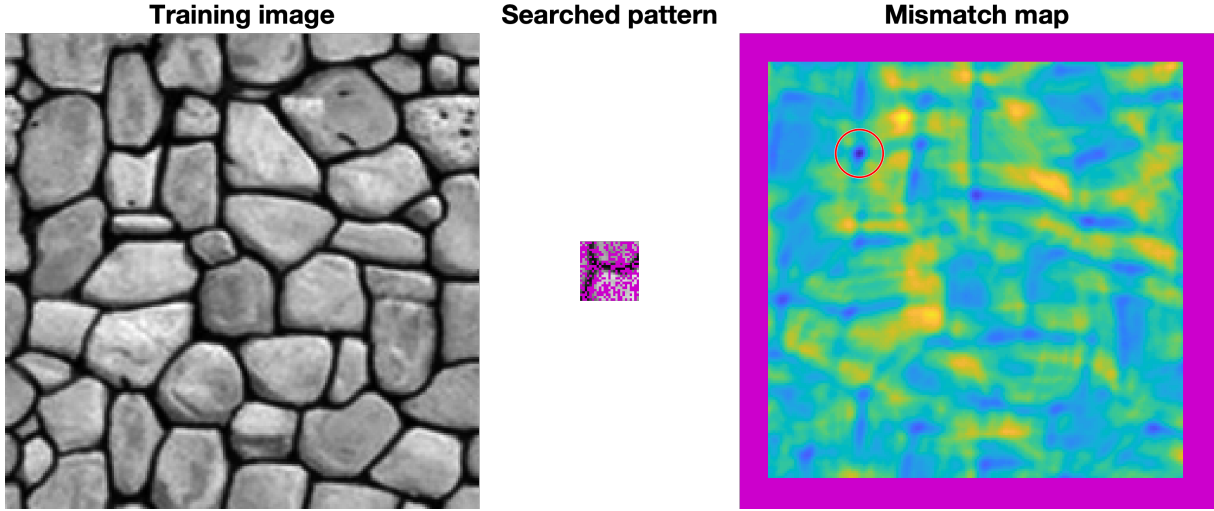
194

195

2.3. Computation of a mismatch map for an entire pattern

196

The approach that is proposed in this work is based on computing a mismatch map in the TI for
197 each simulated pixel. The mismatch map is a grid that represents the pattern-wise mismatch for
198 each location of the training image and enables the fast identification of a good candidate, as
199 shown by the red circle in Figure 1.



200
 201 *Figure 1 Example of a mismatch map for an incomplete pattern. Blue represents good matches,*
 202 *yellow bad matches and purple missing and unusable (border effect) data.* The red circle
 203 *highlights the minimum of the mismatch map, which corresponds to the location of the best*
 204 *candidate.*

205
 206 If we consider the neighborhood $N(s)$ around the simulated position s , then we can express a
 207 weighted dissimilarity between $N(s)$ and a location in the TI $N(t)$:

208 *Equation 5*

209
$$E(N(t), N(s)) = \sum_{l \in N(t,s)} \omega_l \varepsilon(N_l(t), N_l(s))$$

210 where $N(t, s) = \{l \mid N_l(t) \text{ and } N_l(s) \text{ exist}\}$

211 and $N_l(p)$ is the neighbors of p (p can represent either s or t) with a relative displacement l
 212 from p , therefore $N(p) = \{l \mid N_l(p)\}$. l is the lag vector that defines the relative position of
 213 each value within N , and ω_l is a weight for each pixelwise error according to the lag vector l .
 214 By extension, ω is the matrix of all weights, which we call the weighting kernel or, simply, the
 215 kernel. E represents the mismatch between patterns that are centered on s and $t \in T$, where T
 216 is the training image.

217 Some lags may not correspond to a value, for example, due to edge effects in the considered
 218 images or because the patterns are incomplete. Missing patterns are inevitable during the course
 219 of a simulation using a sequential path. Furthermore, in many instances, there can be missing
 220 areas in the training image. This is addressed by creating an indicator variable to be used as a
 221 mask, which equals 1 at informed pixels and 0 everywhere else:

222 *Equation 6*

223
$$\mathbb{1}_l(p) = \begin{cases} 1 & \text{if } N_l(p) \text{ is informed} \\ 0 & \text{otherwise} \end{cases}$$

224 Let us first consider the case in which for a specified position, either all or no variables are
 225 informed. Expressing the presence of data as a mask enables the gaps to be ignored because the
 226 corresponding errors are multiplied by zero.

227 Then, [Equation 5](#) can be expressed as follows:

Equation 7

$$229 \quad E(N(t), N(s)) = \sum_{\mathbf{l}} \omega_{\mathbf{l}} \cdot \mathbb{1}_{\mathbf{l}}(t) \cdot \mathbb{1}_{\mathbf{l}}(s) \cdot \varepsilon(N_{\mathbf{l}}(t), N_{\mathbf{l}}(s))$$

230 . Combining [Equation 4](#) and [Equation 7](#), we get:

Equation 8

$$\begin{aligned}
 233 \quad E(N(t), N(s)) &\propto \sum_{\mathbf{l}} \omega_{\mathbf{l}} \cdot \mathbb{1}_{\mathbf{l}}(t) \cdot \mathbb{1}_{\mathbf{l}}(s) \sum_{i \in \mathcal{J}} \sum_{j \in \mathcal{J}_i} f_j(N_{\mathbf{l}}(t)_i) \cdot g_j(N_{\mathbf{l}}(s)_i) \\
 234 &= \sum_{\mathbf{l}} \sum_{i \in \mathcal{J}} \sum_{j \in \mathcal{J}_i} \omega_{\mathbf{l}} \cdot \mathbb{1}_{\mathbf{l}}(t) \cdot \mathbb{1}_{\mathbf{l}}(s) \cdot f_j(N_{\mathbf{l}}(t)_i) \cdot g_j(N_{\mathbf{l}}(s)_i) \\
 235 &= \sum_{i \in \mathcal{J}} \sum_{j \in \mathcal{J}_i} \sum_{\mathbf{l}} \omega_{\mathbf{l}} \cdot \left(\mathbb{1}_{\mathbf{l}}(t) \cdot f_j(N_{\mathbf{l}}(t)_i) \right) \cdot \left(\mathbb{1}_{\mathbf{l}}(s) \cdot g_j(N_{\mathbf{l}}(s)_i) \right) \\
 236 &= \sum_{i \in \mathcal{J}} \sum_{j \in \mathcal{J}_i} \sum_{\mathbf{l}} \left(\mathbb{1}_{\mathbf{l}}(t) \cdot f_j(N_{\mathbf{l}}(t)_i) \right) \cdot \left(\omega_{\mathbf{l}} \cdot \mathbb{1}_{\mathbf{l}}(s) \cdot g_j(N_{\mathbf{l}}(s)_i) \right)
 \end{aligned}$$

232 .

237 After rewriting, [Equation 8](#) can be expressed as a sum of cross-correlations that encapsulate
 238 spatial dependencies, [using the cross-correlation definition \$f \star g = \sum_i f_i \cdot g_i\$, as follows:](#)

Equation 9

$$240 \quad E(N(t), N(s)) \propto \sum_{i \in \mathcal{J}} \sum_{j \in \mathcal{J}_i} \left(\mathbb{1}(t) \circ f_j(N(t)_i) \right) \star \left(\omega \circ \mathbb{1}(s) \circ g_j(N(s)_i) \right)$$

241 where ω and $\mathbb{1}(\cdot)$ represent the matrices that are formed by $\omega_{\mathbf{l}}$ and $\mathbb{1}_{\mathbf{l}}(\cdot)$ for all possible vectors
 242 \mathbf{l} , \star denotes the cross-correlation operator, and \circ is the element-wise product (or Hadamard-
 243 product).

244 [Finally, with \$T = \{T_i, i \in \mathcal{J}\}\$, \$T_i\$ represents the training image for the \$i\$ -th variable, and](#) by
 245 applying cross-correlations for all positions $t \in T$, we obtain a mismatch map, which is
 246 expressed as:

Equation 10

$$248 \quad E(T, N(s)) \propto \sum_{i \in \mathcal{J}} \sum_{j \in \mathcal{J}_i} \left(\mathbb{1}(T) \circ f_j(T_i) \right) \star \left(\omega \circ \mathbb{1}(s) \circ g_j(N(s)_i) \right)$$

249 . The term $\mathbb{1}(T)$ allows the consideration of the possibility of missing data in the training image
 250 T .

251 Let us consider the general case in which only some variables are informed and the weighting
 252 can vary for each variable. [Equation 10](#) can be extended for this case by defining separate masks
 253 and weights ω_i for each variable:

254 *Equation 11*

$$255 \quad E(T, N(s)) \propto \sum_{i \in \mathcal{J}} \sum_{j \in \mathcal{J}_i} (\mathbb{1}(T_i) \circ f_j(T_i)) \star (\omega_i \circ \mathbb{1}(s_i) \circ g_j(N(s)_i))$$

256 . [Equation 11](#) can be expressed using the convolution theorem [applied to cross-correlation](#):

257 *Equation 12*

$$258 \quad E(T, N(s)) \propto \sum_{i \in \mathcal{J}} \sum_{j \in \mathcal{J}_i} \mathcal{F}^{-1} \left\{ \overline{\mathcal{F}\{\mathbb{1}(T_i) \circ f_j(T_i)\}} \circ \mathcal{F}\{\omega_i \circ \mathbb{1}(s_i) \circ g_j(N(s)_i)\} \right\}$$

259 , where \mathcal{F} represents the Fourier transform, \mathcal{F}^{-1} the inverse transform, and \bar{x} the conjugate of
 260 x .

261 By linearity of the Fourier transform, the summation can be performed in Fourier space, thereby
 262 reducing the number of transformations:

263 *Equation 13*

$$264 \quad E(T, N(s)) \propto \mathcal{F}^{-1} \left\{ \sum_{i \in \mathcal{J}} \sum_{j \in \mathcal{J}_i} \overline{\mathcal{F}\{\mathbb{1}(T_i) \circ f_j(T_i)\}} \circ \mathcal{F}\{\omega_i \circ \mathbb{1}(s_i) \circ g_j(N(s)_i)\} \right\}$$

265 . [Equation 13](#) is appropriate for modern computers, which are well-suited for computing FFTs
 266 (Cooley et al., 1965; Gauss, 1799). Currently, FFTs are well implemented in highly optimized
 267 libraries (Rodríguez, 2002). [Equation 13](#) is the expression that is used in our QS implementation
 268 because it reduces the number of Fourier transforms, which are the most computationally
 269 expensive operations of the algorithm. One issue with the use of FFTs is that the image T is
 270 typically assumed to be periodic. However, in most practical applications, it is not periodic.
 271 This can be simply addressed by cropping the edges of $E(T, N(s))$ or by adding a padding
 272 around T .

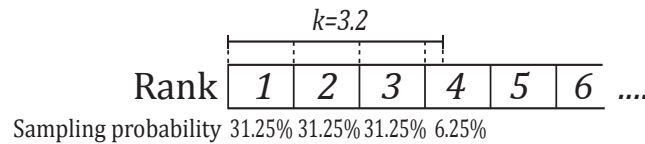
273 The computation of the mismatch map ([Equation 13](#)) is deterministic; as a result, it incurs a
 274 constant computational cost that is independent of the pixel values. Additionally, [Equation 13](#)
 275 is expressed without any constraints on the dimensionality. Therefore, it is possible to use the
 276 n -dimensional FFTs that are provided in the above libraries to perform n -dimensional
 277 simulations without changing the implementation.

278 **2.4. Selection of candidates based on a quantile**

279 The second contribution of this work is the k -sampling strategy for selecting a simulated value
 280 among candidates. The main idea is to use the previously calculated mismatch map to select a
 281 set of potential candidates that are defined by the k smallest (i.e. a quantile) values of E . Once
 282 this set has been selected, we randomly draw a sample from this pool of candidates. This differs

283 from strategies that rely on a fixed threshold, which can be cumbersome to determine. This
 284 strategy is highly similar to the ϵ -replicate strategy that is used in image quilting (Mahmud et
 285 al., 2014) in that we reuse and extend to satisfy the specific requirements of QS. It has the main
 286 advantage of rescaling the acceptance criterion according to the difficulty; i.e. the algorithm is
 287 more tolerant of rare patterns while requiring very close matches for common patterns.

288 In detail, the candidate selection procedure is as follows: All possible candidates are ranked
 289 according to their mismatch and one candidate is randomly sampled among the k best. This
 290 number k can be seen as a quantile over the training dataset. However, parameter k has the
 291 advantage of being an easy representation for users, who can associate $k = 1$ with the best
 292 candidate, $k = 2$ with the two best candidates, etc. For fine-tuning parameter k , the sampling
 293 strategy can be extended to non-integer values of k by sampling the candidates with
 294 probabilities that are not uniform. For example, if the user sets $k = 1.5$, the best candidate has
 295 a probability of $2/3$ of being sampled and the second best a probability of $1/3$. For $k = 3.2$,
 296 (Figure 2) each of the 3 best candidates are sampled with an equal probability of 0.3125 and
 297 the 4th best with a probability of 0.0625. This feature is especially useful for tuning k between
 298 1 and 2 and for avoiding a value of $k = 1$, which can result in the phenomenon of verbatim
 299 copy.



300

301 *Figure 2 Illustration of the k -sampling strategy*

302 An alternative sampling strategy for reducing the simulation time is presented in Appendix A.3.
 303 However, this strategy can result in a reduction in the simulation quality.

304 The value of non-integer k -values is not only in the fine tuning of parameters. It also allows
 305 direct comparisons between QS and DS. Indeed, under the hypothesis of a stationary training
 306 image, using DS with a given max fraction of scanned training image (f) and a threshold (t) of
 307 0 is statistically similar to using QS with $k=1/f$. In both situations, the best candidate is
 308 sampled in a fraction f of the training image.

309

310 **2.5. Simplifications in the case of a fully informed training image**

311 In many applications, spatially exhaustive TIs are available. In such cases, the equations above
 312 can be simplified by dropping constant terms from Equation 1, thereby resulting in a simplified
 313 form for Equation 13. Here, we take advantage of the ranking to know that a constant term will
 314 not affect the result.

315 As in Tahmasebi (2012), in the L^2 -norm, we drop the squared value of the searched pattern,
 316 namely, b^2 , from Equation 2. Hence, we can express Equation 4 as follows:

317

Equation 14

318

$$\varepsilon(\mathbf{a}, \mathbf{b}) = \sum_{i \in \mathcal{J}} a_i^2 - 2 \sum_{i \in \mathcal{J}} a_i \cdot b_i$$

319

320

321

322

323

324

325

Equation 15

326

$$\varepsilon(\mathbf{a}, \mathbf{b}) = - \sum_{i \in \mathcal{J}} \sum_{j \in \mathcal{C}} \delta_{a_i, j} \cdot \delta_{b_i, j}$$

327

328

2.6. Efficient Implementation

329

330

331

An efficient implementation of QS was achieved by 1) performing precomputations, 2) implementing an optimal partial sorting algorithm for selecting candidates and 3) optimal coding and compilation. These are described below.

332

333

334

335

According to [Equation 13](#), $\overline{\mathcal{F}\{\mathbb{1}(T_i) \circ f_j(T_i)\}}$ is independent of the searched pattern $N(s)$. Therefore, it is possible to precompute it at the initialization stage for all i and j . This improvement typically reduces the computation time for an MPS simulation by a factor of at least 2.

336

337

338

339

340

341

342

343

344

345

346

347

348

349

350

351

352

In the QS algorithm, a substantial part of the computation cost is incurred in identifying the k best candidates in the mismatch map. In the case of [non-integer](#) k , the upper limit $\lceil k \rceil$ is used. Identifying the best candidates requires sorting the values of the mismatch map and retaining the candidates in the top k ranks. For this, an efficient sorting algorithm is needed. The operation of finding the k best candidates can be implemented with a partial sort, in which only the elements of interest are sorted, while the other elements remain unordered. This results in two sets: \mathfrak{S}_s with the k smallest elements and \mathfrak{S}_l with the largest elements. The partial sort guarantees that $x \leq y \mid (x, y) \in \mathfrak{S}_s \times \mathfrak{S}_l$. More information about our implementation of this algorithm is available in Appendix A.1. Here, we use a modified vectorized online heap-based partial sort (Appendix A.1). With a complexity of $O(n \cdot \ln(k))$, it is especially suitable for small values of k . Using the cache effect, the current implementation yields results that are close to the search of the best value (the smallest value of the array). The main limitation of standard partial sort implementations is that in the case of equal values, either the first or the last element is sampled. Here, we develop an implementation that can uniformly sample a position among similar values with a single scan of the array. This is important because systematically selecting the same position for the same pattern will reduce the conditional probability density function to a unique sample, thereby biasing the simulation.

353 Due to the intensive memory access by repeatedly scanning large training images, interpreted
354 programming languages, such as MATLAB and Python, are inefficient for a QS
355 implementation and, in particular, for a parallelized implementation. We provide a NUMA-
356 aware ([Blagodurov et al., 2010](#)) and flexible C/C++/OpenMP implementation of QS that is
357 highly optimized. Following the denomination of Mariethoz (2010), we use a path-level
358 parallelization with a waiting strategy, which offers a good trade-off between performance and
359 memory requirements. In addition, two node-level parallelization strategies are available: if
360 many training images are used, a first parallelization is performed over the exploration of the
361 training images; then, each FFT of the algorithm is parallelized using natively parallel FFT
362 libraries.

363 The FFTw library (Frigo and Johnson, 2018) provides a flexible and performant architecture-
364 independent framework for computing n -dimensional Fourier transformations. However, an
365 additional speed gain of approximately 20% was measured by using the Intel MKL library (Intel
366 Corporation, 2019) on compatible architectures. We also have a GPU implementation that uses
367 cIFFT for compatibility. Many Fourier transforms are sparse and, therefore, can easily be
368 accelerated in n -dimensional cases with “partial FFT” since Fourier transforms of only zeros
369 result in zeros.

370 **3. Results**

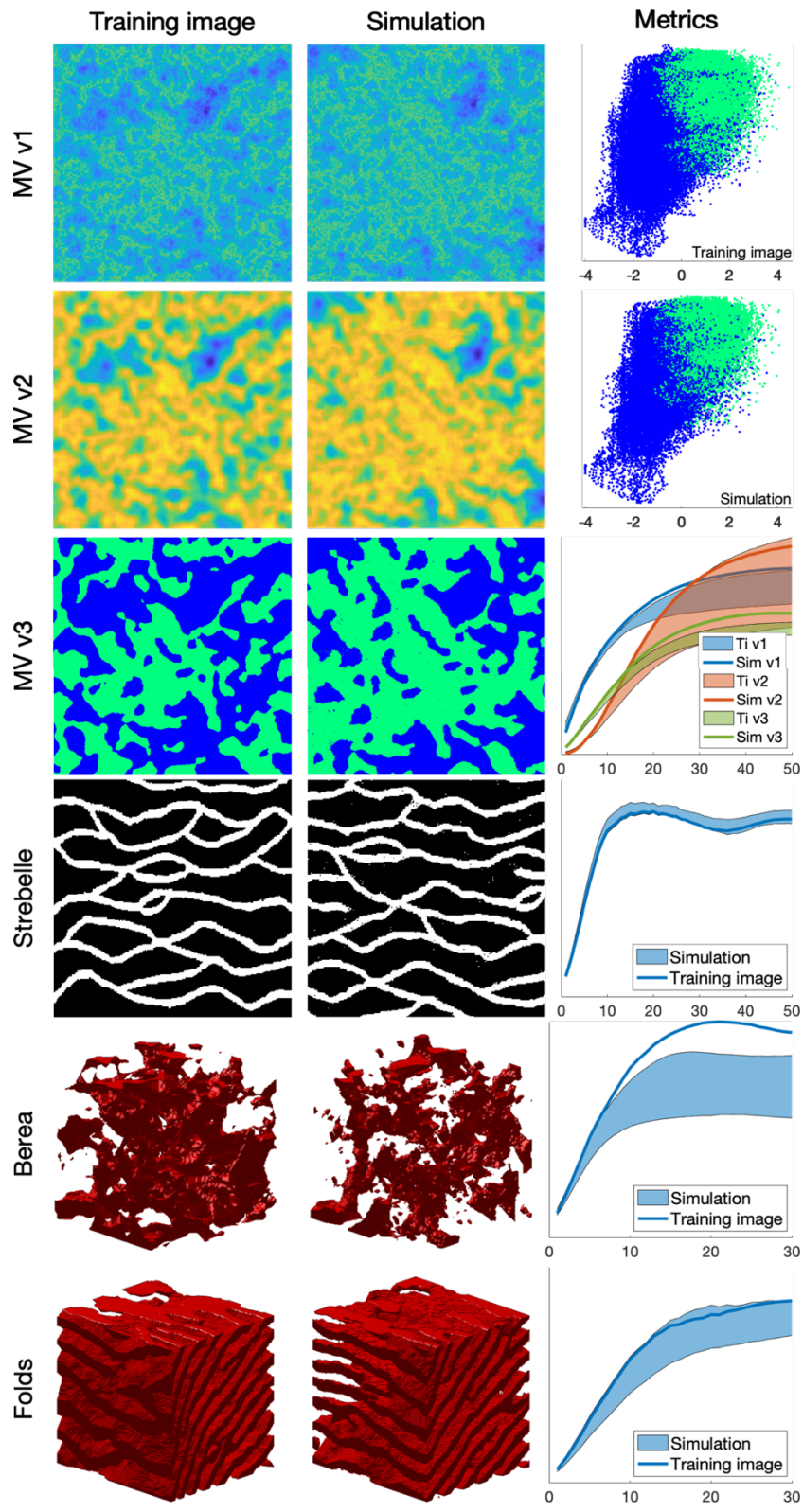
371 **3.1. Simulation examples**

372 This section presents illustrative examples for continuous and categorical case studies in 2D
373 and in 3D. Additional tests are reported in Appendix 0. The parameters that are used for the
374 simulations of [Figure 3](#) are reported in [Table 1](#).

375 The results show that simulation results are consistent with what is typically observed with
376 state-of-the-art MPS algorithms. While simulations can accurately reproduce TI properties for
377 relatively standard examples with repetitive structures (e.g., MV, Strebelle, and Folds), training
378 images with long-range features (typically larger than the size of the TI) are more difficult to
379 reproduce, such as in the Berea example. For multivariate simulations, the reproduction of the
380 joint distribution is satisfactory, as observed in the [scatterplots \(Figure 3\)](#). [More examples are](#)
381 [available in Annex A4, in particular the Figure A2 for 2D examples and the Figure A3 for 3D](#)
382 [examples.](#)

383

384



385

386

387

388

389

390

Figure 3 Examples of unconditional continuous and categorical simulations in 2D and 3D and their variograms. The first column shows the training images that were used, the second column one realization, and the third column quantitative quality metrics. MVs v1, v2 and v3 represent a multivariate training image (and the corresponding simulation) using 3 variables. The first two metrics are scatter plots of MV v1 vs. MV v2 of the training image and the simulation,

391 *respectively. The third metric represents the reproduction of the variogram for each of MVs v1,*
392 *v2 and v3.*

	MVs v1, v2, v3	Strebelle	Berea	Folds
Source	(Mariethoz and Caers, 2014)	(Strebelle, 2002)	Doi:10.6084/m9.figshare.1153794	(Mariethoz and Caers, 2014)
Size of the training image (px)	490 × 490	250 × 250	100 × 100 × 100	180 × 150 × 120
Size of the simulation (px)	490 × 490	250 × 250	100 × 100 × 100	180 × 150 × 120
Computation time (s)	1456	54	1665	76270
k	1.2			
N	80		125	

394
395

Table 1 Parameters that were used for the simulations in [Figure 3](#). Times are specified for simulations without parallelization.

396

3.2. Comparison with direct sampling simulations

397
398
399
400
401
402
403
404

QS simulations are benchmarked against DS using the “Stone” training image ([Figure 4](#)). The settings that are used for DS are based on optimal parameters that were obtained via the approach of Baninajar et al. (2019), which uses stochastic optimization to find optimal parameters. In DS, we use a fraction of scanned TI of $f = 1$ to explore the entire training image via the same approach as in QS and we use the L^2 -norm as in QS. To avoid the occurrence of verbatim copy, we include 0.1% conditioning data, which are randomly sampled from a rotated version of the training image. The number of neighbors N is set to 20 for both DS and QS and the acceptance threshold of DS is set to 0.001.

405
406
407
408
409
410
411
412
413
414
415
416
417
418

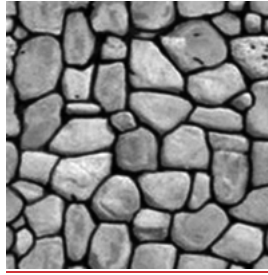
The comparison is based on qualitative ([Figure 5](#)) and quantitative ([Figure 6](#)) metrics, which include directional and omnidirectional variograms, along with the connectivity function, the Euler characteristic (Renard and Allard, 2013) and cumulants (Dimitrakopoulos, 2010). The connectivity represents the probability for 2 random pixels to be in the same connected component. This metric is suited to detect broken structures. The Euler characteristic represents the number of objects subtracted by the number of holes of the objects, and is particularly adapted to detect noise in the simulations such as salt and pepper. Cumulants are high order statistics and therefore allow considering the relative positions between elements. The results demonstrate that the simulations are of a quality that is comparable to DS. With extreme settings (highest pattern reproduction regardless of the computation time), both algorithms perform similarly, which is reasonable since both are based on sequential simulation and both directly import data from the training image. The extra noise present in the simulation is shown in the Euler characteristic. Furthermore, it demonstrates that the use of a kernel can reduce this noise to get better simulations.

419 With QS, kernel weighting allows fine tuning of the parametrization to improve the results, as
420 shown in Figure 5. In this paper, we use an exponential kernel:

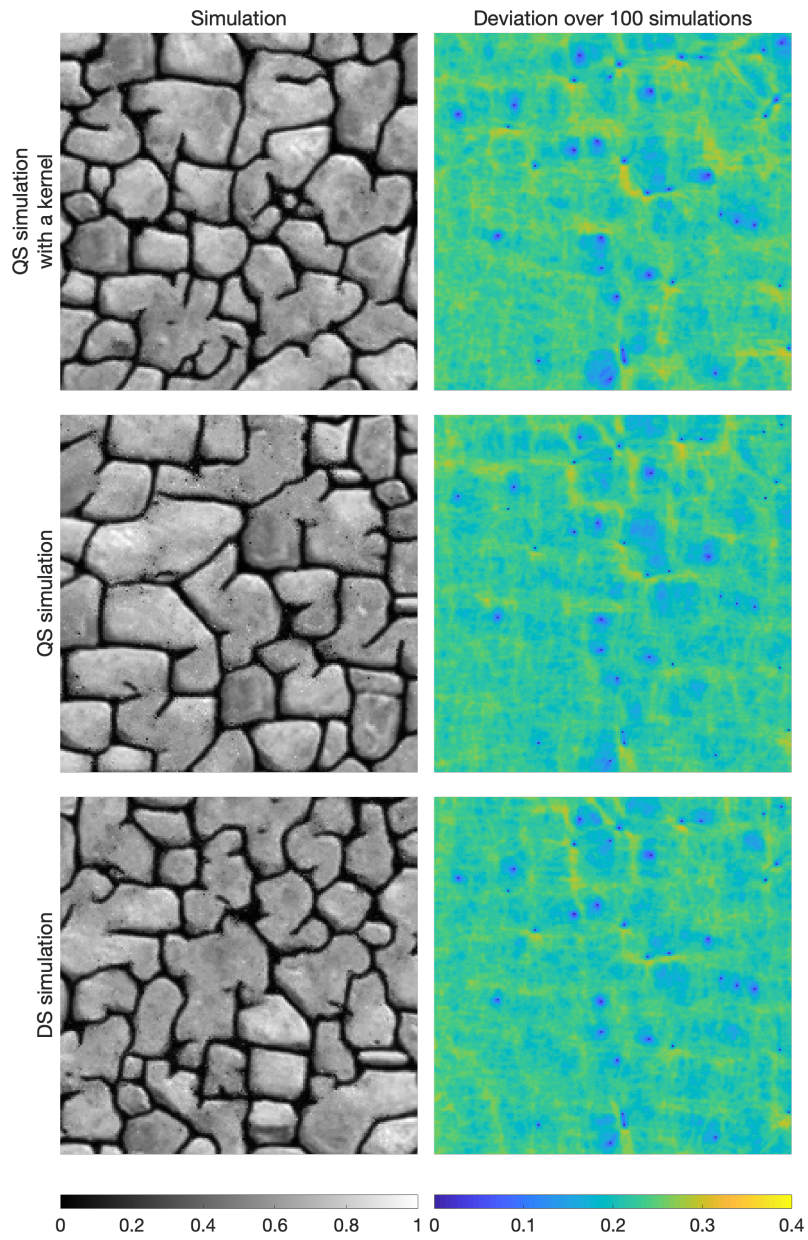
421 *Equation 16*

422
$$\omega_l = e^{-\alpha \|l\|_2}$$

423 where α is a kernel parameter and $\| \cdot \|_2$ the Euclidean distance. The validation metrics of Figure
424 6 show that both QS and DS tend to slightly underestimate the variance and the connectivity.
425 Figure 6 shows that an optimal kernel improves the results for all metrics, with all training
426 image metrics in the 5-95% realization interval, except for the Euler characteristic.



427
428 *Figure 4 Training image that was used for benchmarking and sensitivity analysis.*



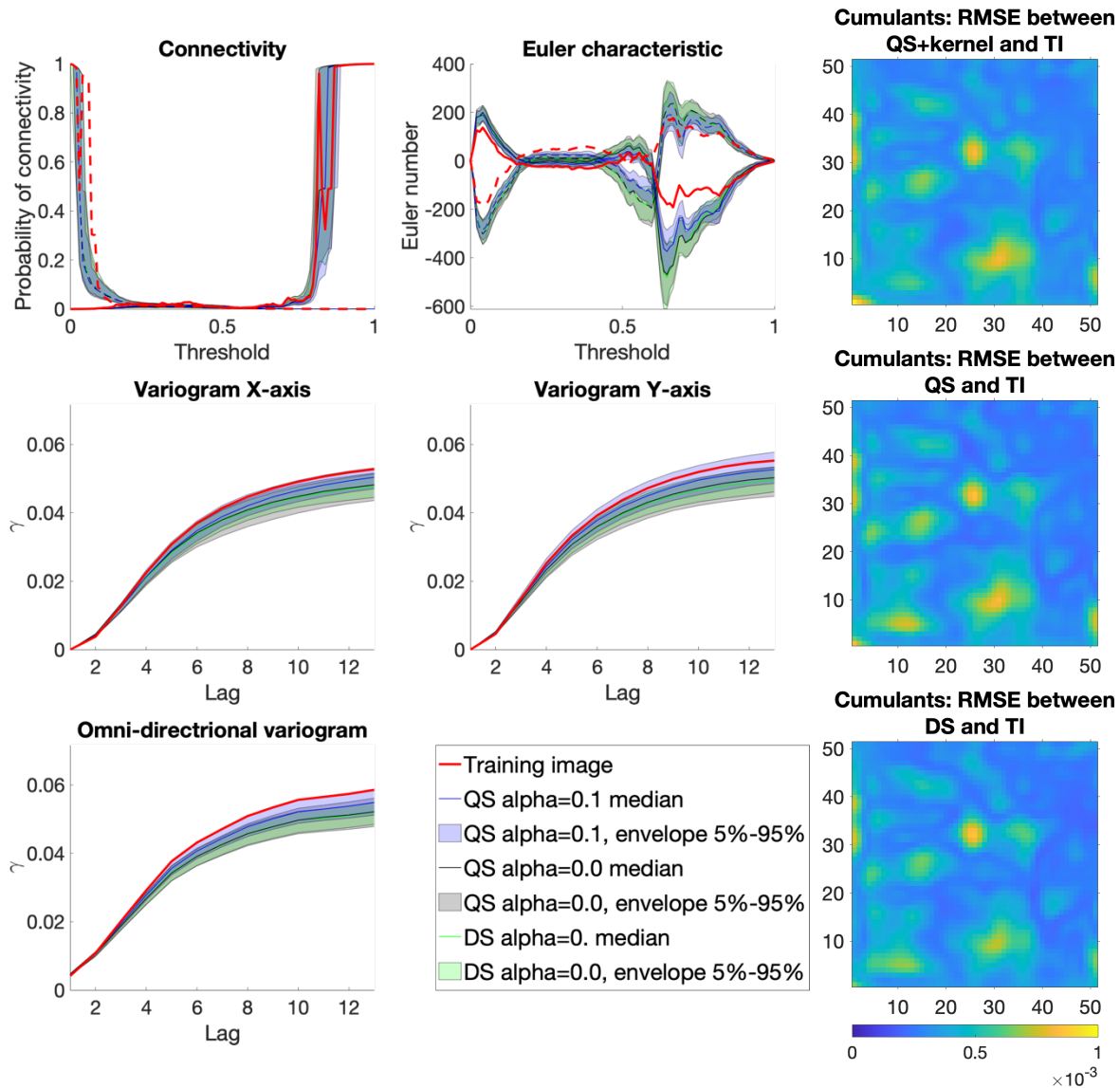
429

430

431

432

Figure 5 Examples of conditional simulations and their standard deviation over 100 realizations that are used in the benchmark between QS and DS.



433
434
435

Figure 6 Benchmark between QS (with and without kernel) and DS over 6 metrics Using each time 100 unconditional simulation.

436

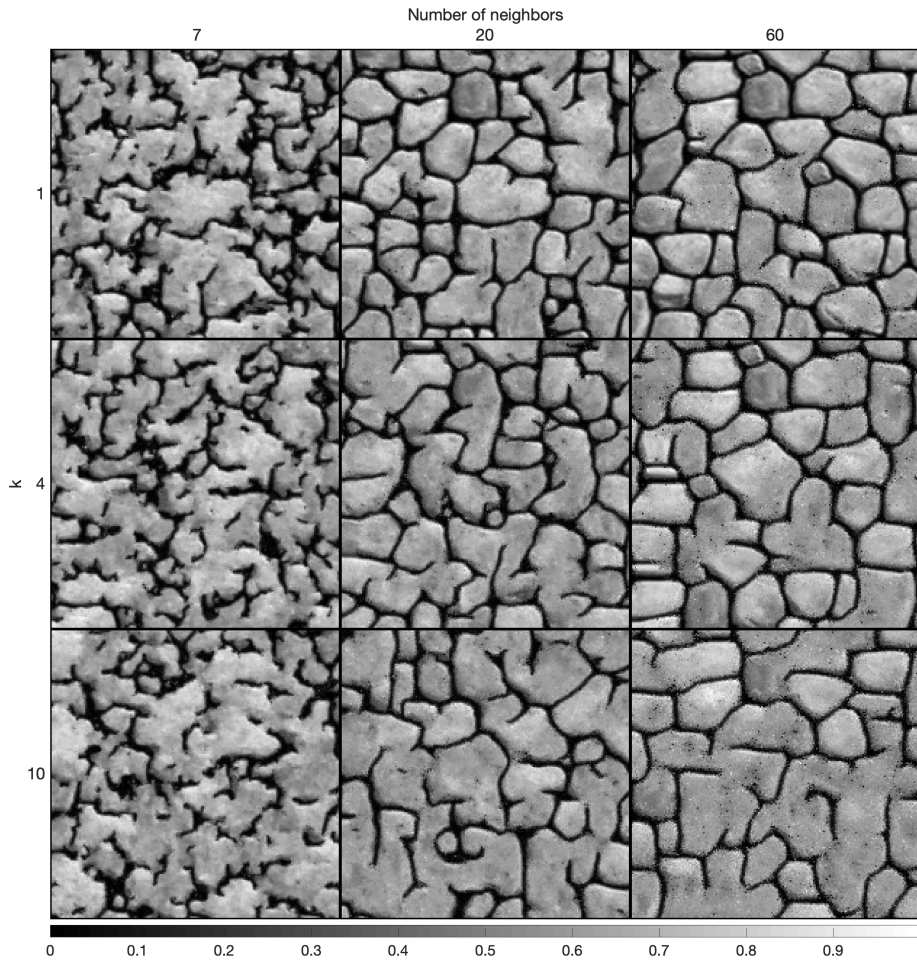
3.3. Parameter sensitivity analysis

437
438
439
440

In this section, we perform a sensitivity analysis on the parameters of QS using the training image in [Figure 4](#). Only essential results are reported in this section ([Figure 7](#) and [Figure 8](#)); more exhaustive test results are available in Appendix 0 ([Figure A 4](#) and [Figure A 5](#)). The two main parameters of QS are the number of neighbors N and the number of used candidates k .

441
442
443

[Figure 7](#) (and Appendix 0 [Figure A 4](#)) shows that large N values and small k values improve the simulation performance; however, tend to induce verbatim copy in the simulation. Small values of N result in noise with good reproduction of the histogram.

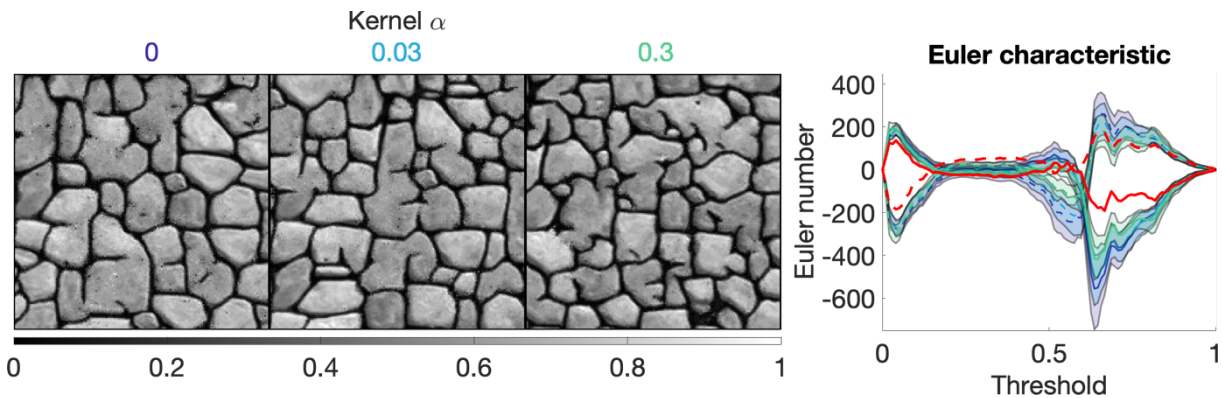


444
445
446

Figure 7 Sensitivity analysis on one simulation for the two main parameters of QS using a uniform kernel.

447
448
449
450
451
452

ω can be a very powerful tool, typically using the assumption that the closest pixels are more informative than remote pixels. The sensitivity analysis of the kernel value α are explored in Figure 8 and Figure A 5. They show that α provides a unique tool for improving the simulation quality. In particular, using a kernel can reduce the noise in simulations, which is clearly visible by comparing the Euler characteristic curves. However, reducing too much the importance of distant pixels results in ignoring them altogether, therefore damaging long-range structures.



453
454
455
456

Figure 8 Sensitivity analysis on the kernel parameter α , with fixed parameters $k=1.5$ and $N=40$. The values of the kernels are shown in colors that correspond to the Euler characteristic lines (red is the training image).

457

3.4. Computational efficiency and scalability

458 In this section, we investigate the scalability of QS with respect to the size of the simulation
 459 grid, the size of the training image grid, the number of variables, incomplete training images,
 460 and hardware. According to the test results, the code will continue to scale with new-generation
 461 hardware.

462 As explained in Section 2.3 and 2.4, the amounts of time that are consumed by the two main
 463 operations of QS (finding candidates and sorting them) are independent of the pixel values.
 464 Therefore, the training image that is used is not relevant (here, we use simulations that were
 465 performed with the TI of [Figure 4](#) and its classified version for categorical cases). Furthermore,
 466 the computation time is independent of the parametrization (k and N). However, the
 467 performance is affected by the type of mismatch function that is used; here, we consider both
 468 continuous ([Equation 2](#) and [Equation 14](#)) and categorical cases ([Equation 3](#) and [Equation 15](#)).

469 We also test our implementation on different types of hardware, as summarized in [Table 2](#). We
 470 expect Machine (2) to be faster than Machine (1) for medium-sized problems due to the high
 471 memory bandwidth requirement of QS. Machine (3) should also be faster than Machine (1)
 472 because it takes advantage of a longer vector computation (512-bit VS. 256-bit instruction set).

Name of the machine	Machine (1)	Machine (2)	Machine (3)
CPU	-2x Intel(R) Xeon(R) CPU E5-2680 v2 @ 2.80 GHz	-Xeon Phi, Intel(R) Xeon Phi (TM) CPU 7210 @ 1.30 GHz	-2x Intel(R) Xeon(R) Gold 6128 Processor @ 3.40 GHz
Memory type	- DDR3	- MCDRAM / DDR4	- DDR4
OS, compiler and compilation flags	Linux, Intel C/C++ compiler 2018 with -xhost		

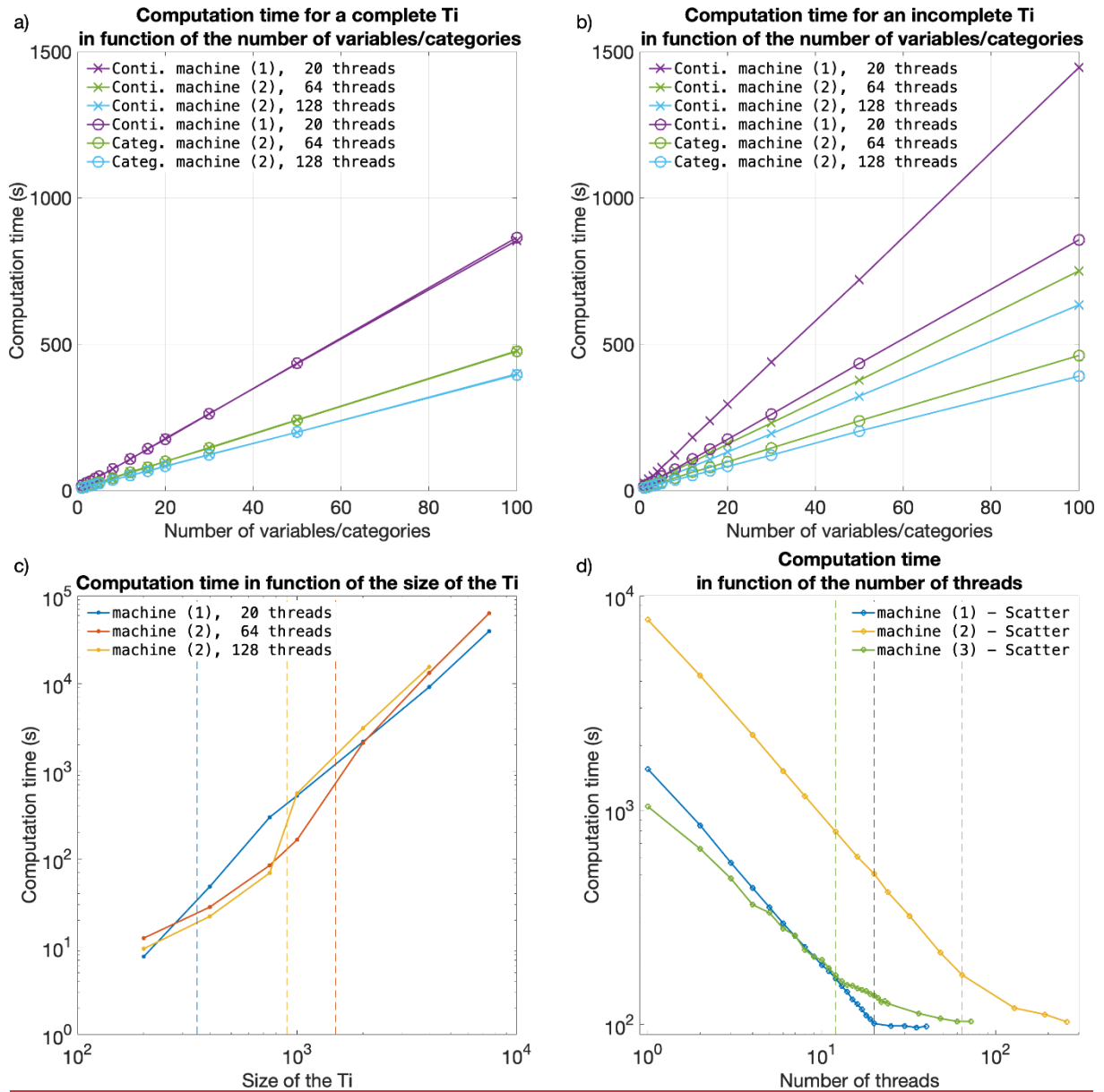
473

Table 2 Hardware that was used in the experiments

474 [Figure 9](#) plots the execution times on the 3 tested machines for continuous and categorical cases
 475 and with training images of various sizes. Since QS has a predictable execution time, the
 476 influence of the parameters on the computation time is predictable: linear with respect to the
 477 number of variables ([Figure 9a](#), [Figure 9b](#)), linear with respect to the size of the simulation grid
 478 and following a power function of the size of the training image ([Figure 9c](#)). Therefore, via a
 479 few tests on a set of simulations, one can predict the computation time for any other setting.

480 [Figure 9d](#) shows the scalability of the algorithm when using the path-level parallelization. The
 481 algorithm scales well until all physical cores are being used. Machine (3) has a different scaling
 482 factor (slope). This suboptimal scaling is attributed to the limited memory bandwidth. Our
 483 implementation of QS scales well with an increasing number of threads ([Figure 9d](#)), with an
 484 efficiency above 80% using all possible threads. The path-level parallelization strategy that was
 485 used involves a bottleneck for large number of threads due to the need to wait for neighborhood
 486 conflicts to be resolved (Mariethoz 2010). This effect typically appears for large values of N or

487 intense parallelization (>50 threads) on small grids. It is assumed that small grids do not require
 488 intense parallelization; hence, this problem is irrelevant in most applications.



489
 490 *Figure 9 Efficiency of QS with respect to all key parameters. a) and b) are the evolution of the*
 491 *computation time for complete and incomplete training images, respectively, with continuous*
 492 *and categorical variables. c) shows the evolution of the computation time as the size of the*
 493 *training image is varied; the dashed lines indicate that the training image no longer fits in the*
 494 *CPU cache. d) shows the evolution of the computation time as the number of threads is*
 495 *increased. The dashed lines indicate that all physical cores are used.*

496

497 4. Discussion

498 The parameterization of the algorithm (and therefore simulation quality) has almost no impact
 499 on the computational cost, which is an advantage. Indeed, many MPS algorithms impose trade-

500 offs between the computation time and the parameters that control the simulation quality,
501 thereby imposing difficult choices for users. QS is comparatively simpler to set up in this
502 regard. In practice, a satisfactory parameterization strategy is often to start with a small k value
503 (say 1.2) and a large N value (> 50) and then gradually change these values to increase the
504 variability if necessary (Figure 6 and Figure A 4).

505 QS is adapted for simulating continuous variables using the L^2 -norm. However, a limitation is
506 that the L^1 -norm does not have a decomposition that satisfies Equation 1 and, therefore, cannot
507 be used with QS. Another limitation is that for categorical variables, each class requires a
508 separate FFT, which incurs an additional computational cost. This renders QS less
509 computationally efficient for categorical variables (if there are more than 2 categories) than for
510 continuous variables. For accelerated simulation of categorical variables, a possible alternative
511 to reduce the number of required operations is presented in Appendix A.2. The strategy is to
512 use encoded variables, which are decoded in the mismatch map. While this alternative yields
513 significant computational gains, it does not allow the use of a kernel weighting and is prone to
514 numerical precision issues.

515 Combining multiple continuous and categorical variables can be challenging for MPS
516 approaches. Several strategies have been developed to overcome this limitation, using either a
517 different distance threshold for each variable, or a linear combination of the errors. Here we use
518 the second approach, taking advantage of the linearity of the Fourier transform. The relative
519 importance can be set in f_i and g_i functions in Equation 1. However, it is computationally
520 advantageous to use the kernel weights in order to have standard functions for each metric.
521 Setting such variable-dependent parameters is complex. Therefore in order to find optimal
522 parameters, stochastic optimization approaches (such as Baninajar et al., 2019) are applied to
523 QS. The computational efficiency of QS is generally advantageous compared to other pixel-
524 based algorithms: for example, in our tests it performed faster than DS. QS requires more
525 memory than DS, especially for applications with categorical variables with many classes and
526 with a path-level parallelization. However, the memory requirement is much lower compared
527 to MPS algorithms that are based on a pattern database, such as SNESIM.

528 There may be cases where QS slower than DS, in particular when using a large training image
529 that is highly repetitive. In such cases, using DS can be advantageous as it must scan only a
530 very small part of the training image. For scenarios of this type, it is possible to adapt QS such
531 that only a small subset of the training image is used; this approach is described in Appendix
532 A3. In the cases of highly repetitive training images, this observation remains true also for
533 SNESIM and IMPALA.

534 Furthermore, QS is designed to efficiently handle large and complex training images (up to 10
535 million pixels), with high variability of patterns and few repetitions. Larger training images
536 may be computationally burdensome, which could be alleviated by using a GPUs
537 implementation allowing gains up to two orders of magnitude.

538 QS can be extended to handle the rotation and scaling of patterns by applying a constant rotation
539 or affinity transformation to the searched patterns (Strebelle, 2002). However, the use rotation-
540 invariant distances and affinity-invariant distances (as in Mariethoz and Kelly, 2011), while
541 possible in theory, would substantially increase the computation time. Mean-invariant distances

542 can be implemented by simply adapting the distance formulation in QS. All these advanced
543 features are outside the scope of this paper.

544 **5. Conclusions**

545 QS is an alternative approach for performing n -dimensional pixel-based simulations, which
546 uses an L^2 -distance for continuous cases and an L^0 -distance for categorical data. The framework
547 is highly flexible and allows other metrics to be used. The simple parameterization of QS
548 renders it easy to use for nonexpert users. Compared to other pixel-based approaches, QS has
549 the advantage of generating realizations in constant and predictable time for a specified training
550 image size. Using the quantile as a quality criterion naturally reduces the small-scale noise
551 compared to DS. In terms of parallelization, the QS code scales well and can adapt to new
552 architectures due to the use of external highly optimized libraries.

553 The QS framework provides a complete and explicit mismatch map, which can be used to
554 formulate problem-specific rules for sampling or even solutions that take the complete
555 conditional probability density function into account, for example, such as a narrowness
556 criterion for the conditional pdf of the simulated value (Gravey et al., 2019; Rasera et al., 2019),
557 or to use the mismatch map to infer the optimal parameters of the algorithm.

558 **6. Code availability**

559 The source code and documentation of the QS simulation algorithm are available as part of the
560 G2S package at: <https://github.com/GAIA-UNIL/G2S> under GPLv3 license. Or permanently
561 at <https://doi.org/10.5281/zenodo.3546338>

562 Platform: Linux / macOS / Windows 10 Language: C/C++

563 Interfacing functions in MATLAB, Python3, R

564 A package is available with our unbiased partial sort at:
565 <https://github.com/mgravey/randomKmin-max>

566 **7. Author contribution**

567 MG proposed the idea, implemented and optimized the QS approach and wrote the manuscript.
568 GM provided supervision, methodological insights and contributed to the writing of the
569 manuscript.

570 8. Appendices

571 A.1. Partial sorting with random sampling

572 Standard partial sorting algorithms resolve tie ranks deterministically, which does not accord
573 with the objective of stochastic simulation with QS, where variability is sought. Here, we
574 propose an online heap-based partial sort. It is realized with a single scan of the array of data
575 using a heap to store previously found values. This approach is especially suitable when we are
576 interested in a small fraction of the entire array.

577 Random positions of the k best values are ensured by swapping similar values. If $k = 1$, the
578 saved value is switched with a smaller value each time it is encountered. If an equal value is
579 scanned, a counter c is increased for this specific value and a probability of $1/c$ of switching to
580 the new position is applied. If $k > 1$, the same strategy is extended by carrying over the counter
581 c .

582 This partial sort outperforms random exploration of the mismatch map. However, it is difficult
583 to implement efficiently on GPUs. A solution is still possible for shared-memory GPUs by
584 performing the partial sort on the CPU. This is currently available in the proposed
585 implementation.

586 k : the number of values of interest

587 D : the input data array

588 S : the array with the k smallest values (sorted)

589 Sp : the array with the positions that are associated with the values of S

590

591 1. **for** each value v of D

592 2. **if** v is smaller than the smallest value of S

593 3. search in S for the position p at which to insert v and insert it

594 4. **if** $p = k$ // last position of the array

595 5. reinitialize the counter c to 0

596 6. insert v at the last position

597 7. **else**

598 8. increment c by one

599 9. swap the last position with another of the same value

600 10. insert the value at the expected position p

601 11. **end**

602 12. **else if** v is equal to the smallest value of S

603 13. increment c by one

604 14. change the position of v to one of the n positions of equal value with a probability of
605 $n/(n + c)$

606 15. **end**

607 16. **end**

608 **A.2. Encoded categorical variables**

609 To handle categorical variables, a standard approach is to consider each category as an
 610 independent variable. This requires as many FFTs as classes. This solution renders it expensive
 611 to use QS in cases with multiple categories.

612 An alternative approach is to encode the categories and to decode the mismatch from the cross-
 613 correlation. It has the advantage of only requiring only a single cross-correlation for each
 614 simulated pattern.

615 Here, we propose encoding the categories as powers of the number of neighbors, such that their
 616 product is equal to one if the class matches. In all other cases, the value is smaller than one or
 617 larger than the number of neighbors.

618
$$\varepsilon_{L^0}(a, b) = \psi((a - b)^0 \propto -(N + 1)^{-p(a)} \cdot (N + 1)^{-p(b)})$$

619 where N is the largest number of neighbors that can be considered and $p(c)$ is an arbitrary
 620 function that maps index classes of \mathcal{C} , $c \in \mathcal{C}$.

621 In this scenario, in [Equation 1](#) this encoded distance L_e^0 can be decomposed into the following
 622 series of functions f_j and g_j :

623 $f_0: x \rightarrow -(N + 1)^{p(x)}$

624 $g_0: x \rightarrow (N + 1)^{-p(x)}$

625 and the decoding function is

626
$$\psi(x) = \lfloor x \rfloor \bmod N$$

627 Table A 1 describes this process for 3 classes, namely, a, b , and c , and a maximum of 9
 628 neighbors. Then, the error can be easily decoded by removing decimals and dozens.

Products	$g_0(a) = 1$	$g_0(b) = 0.1$	$g_0(c) = 0.01$
$f_0(a) = 1$	1	0.1	0.01
$f_0(b) = 10$	10	1	0.1
$f_0(c) = 100$	100	10	1

629 *Table A 1 Example of encoding for 3 classes and 9 neighbors and their associated products*

630 Consider the following combination:

631 $f_0(a, b, a, c, c, b, a, a, b)$

632 $\times g_0(c, b, b, a, a, b, c, a, a)$

633 $-(0.01, 1, 0.1, 100, 100, 1, 0.01, 1, 10) = -213.12$

634 The decoding $\lfloor -213.12 \rfloor \bmod 10 = -213 \bmod 10 = -3$ yields 3 matches (in green).

635 This encoding strategy provides the possibility of drastically reducing the number of FFT
636 computations. However, the decoding phase is not always implementable if a nonuniform
637 matrix ω is used. Finally, the test results show that the method suffers quickly from numerical
638 precision issues, especially with many classes.

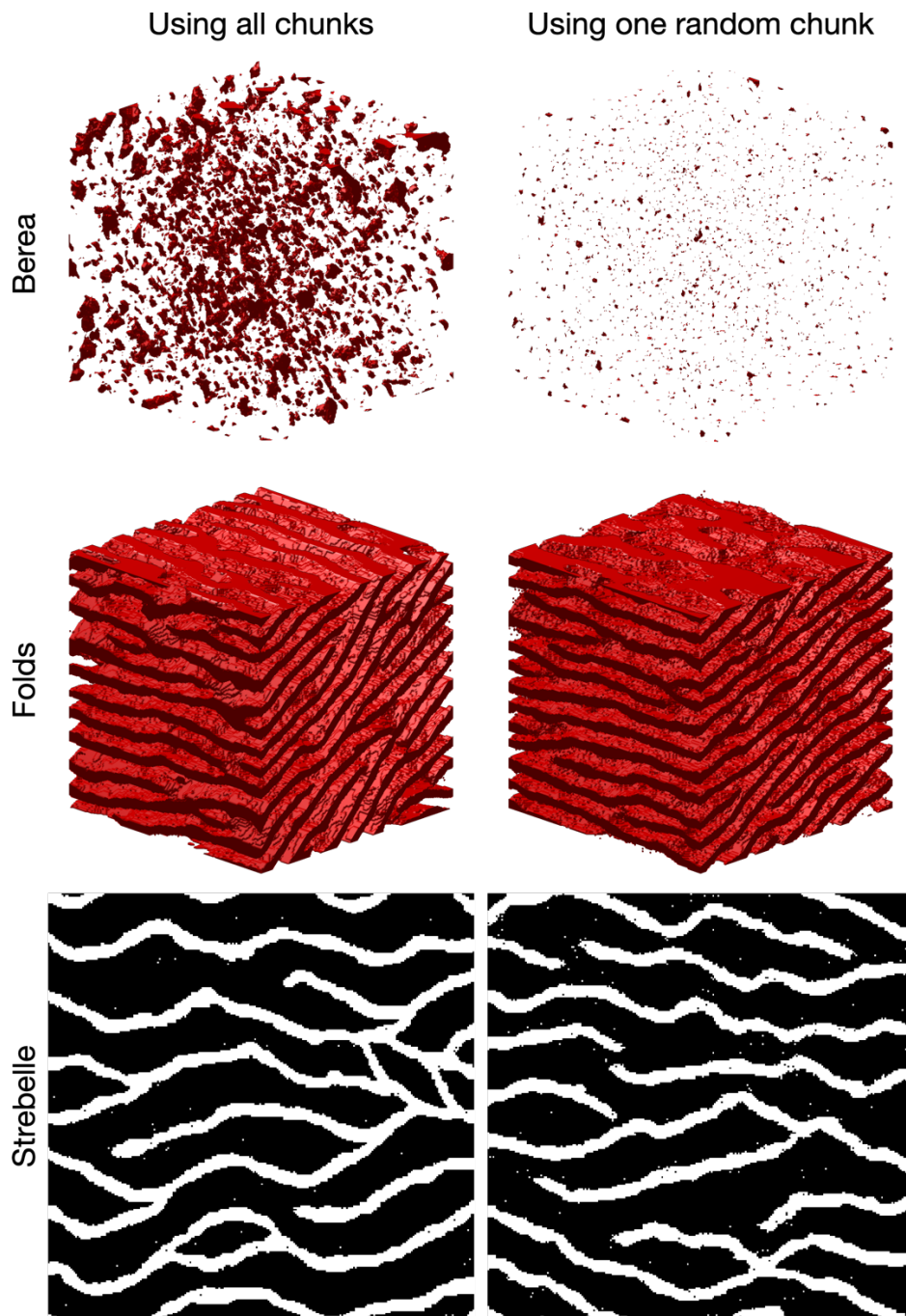
639 **A.3. Sampling strategy using training image splitting**

640 The principle of considering a fixed number of candidates can be extended by instead of taking
641 the k^{th} best candidate, sampling the best candidate in only a portion $\frac{1}{k}$, of the TI. For instance,
642 as an alternative to considering $k = 4$, this strategy searches for the best candidate in one fourth
643 of the image. This is more computationally efficient. However, if all the considered candidates
644 are contiguous (by splitting the TI in k chunks), this approximation is only valid if the TI is
645 completely stationary and all k equal subdivisions of the TI are statistically identical. In
646 practice, real-world continuous variables are often nonstationary. However, in categorical
647 cases, especially in binary ones, the number of pattern replicates is higher and this sampling
648 strategy could be interesting.

649 The results of applying this strategy are presented in Table A 2 and Figure A 1. The
650 experimental results demonstrate that the partial exploration approach that is provided by
651 splitting substantially accelerates the processing time. However, Figure A 1 shows that the
652 approach has clear limitations when dealing with training images with complex and

653 nonrepetitive patterns. The absence of local verbatim copy can explain the poor-quality
654 simulation results.

655



656

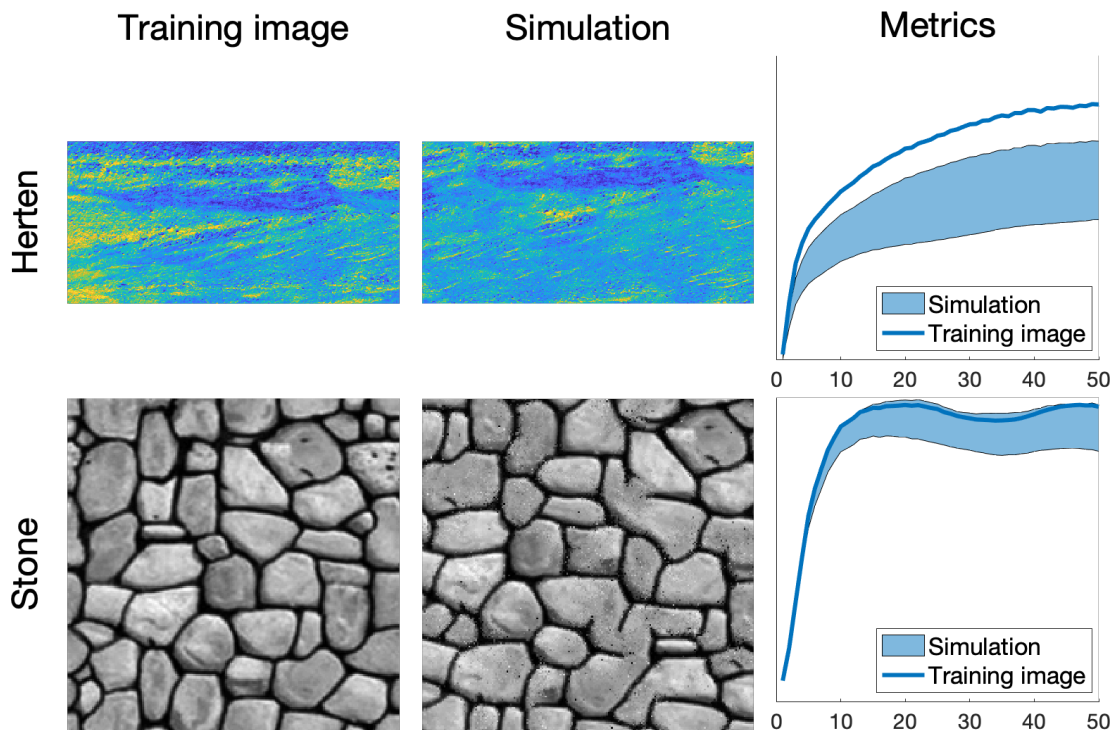
657 *Figure A 1 Comparison of QS using the entire training image and using training image*
658 *splitting. In these examples, the training image is split into two images over each dimension.*
659 *The original training images are presented in Figure 2.*

660

Training image	Using all chunks	Using one random chunk	Speedup
Berea	11 052 s	1 452 s	7.61x
Folds	35 211 s	4 063 s	8.66x
Strebelle	7.95 s	3.16 s	2.51x

661 *Table A 2 Computation times and speedups for the full and partial exploration approaches.*
662 *Times are specified for simulations with path level parallelization.*

663 **A.4. Additional results**

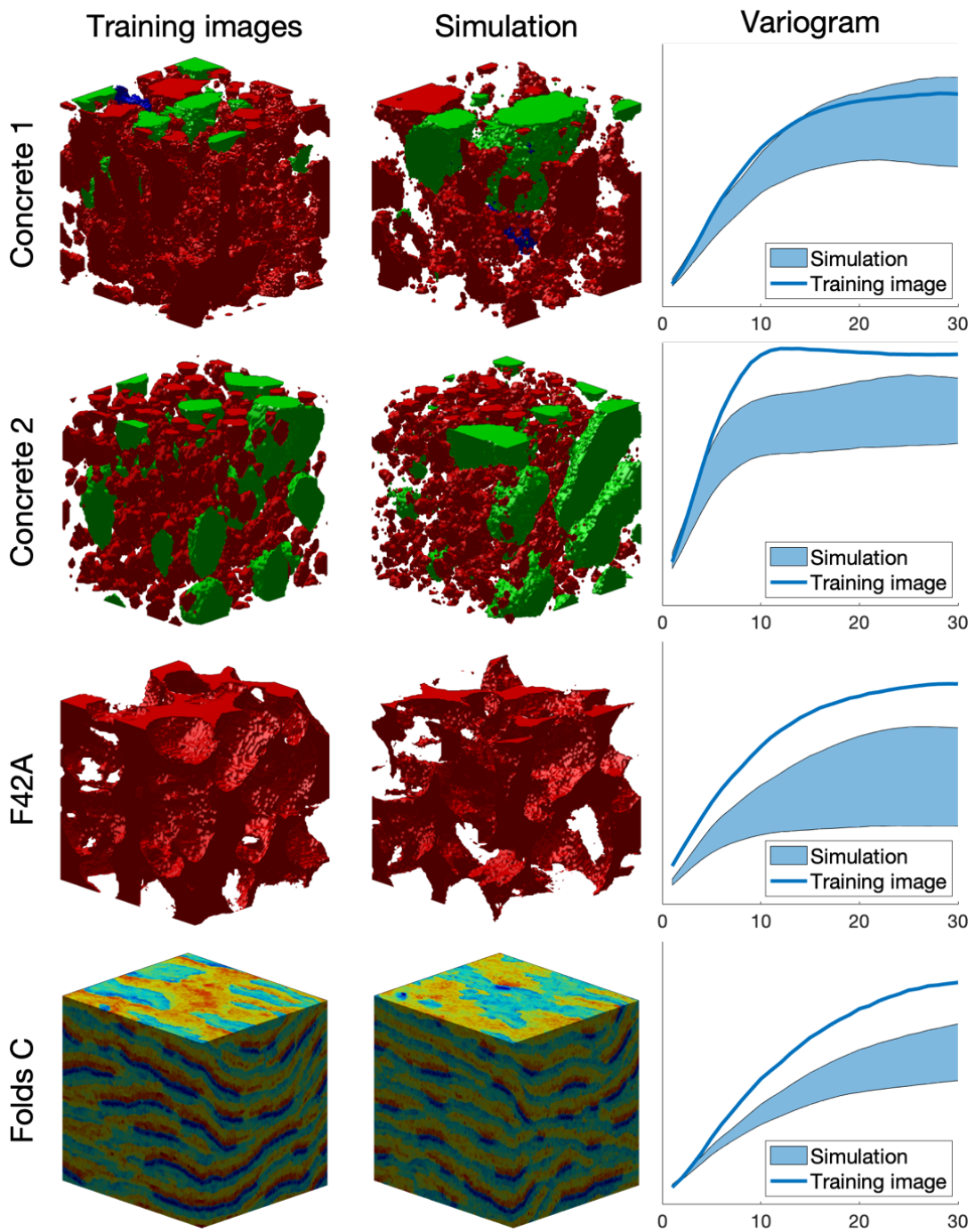


664
665 *Figure A 2 Examples of 2D simulations: the first 3 rows represent 3 variables of a single*
666 *simulation. [Parameters available in Table A 3](#)*

667

668

669



670

671

Figure A 3 Examples of 3D simulation results. *Parameters available in Table A 4*

	Herten	Stone
Source	(Mariethoz and Caers, 2014)	(Mariethoz and Caers, 2014)
Size of the training image (px)	716 × 350	200 × 200
Size of the Simulation (px)	716 × 350	200 × 200
Computation time (s)	1133	21
k	1.2	
N	80	

672
673

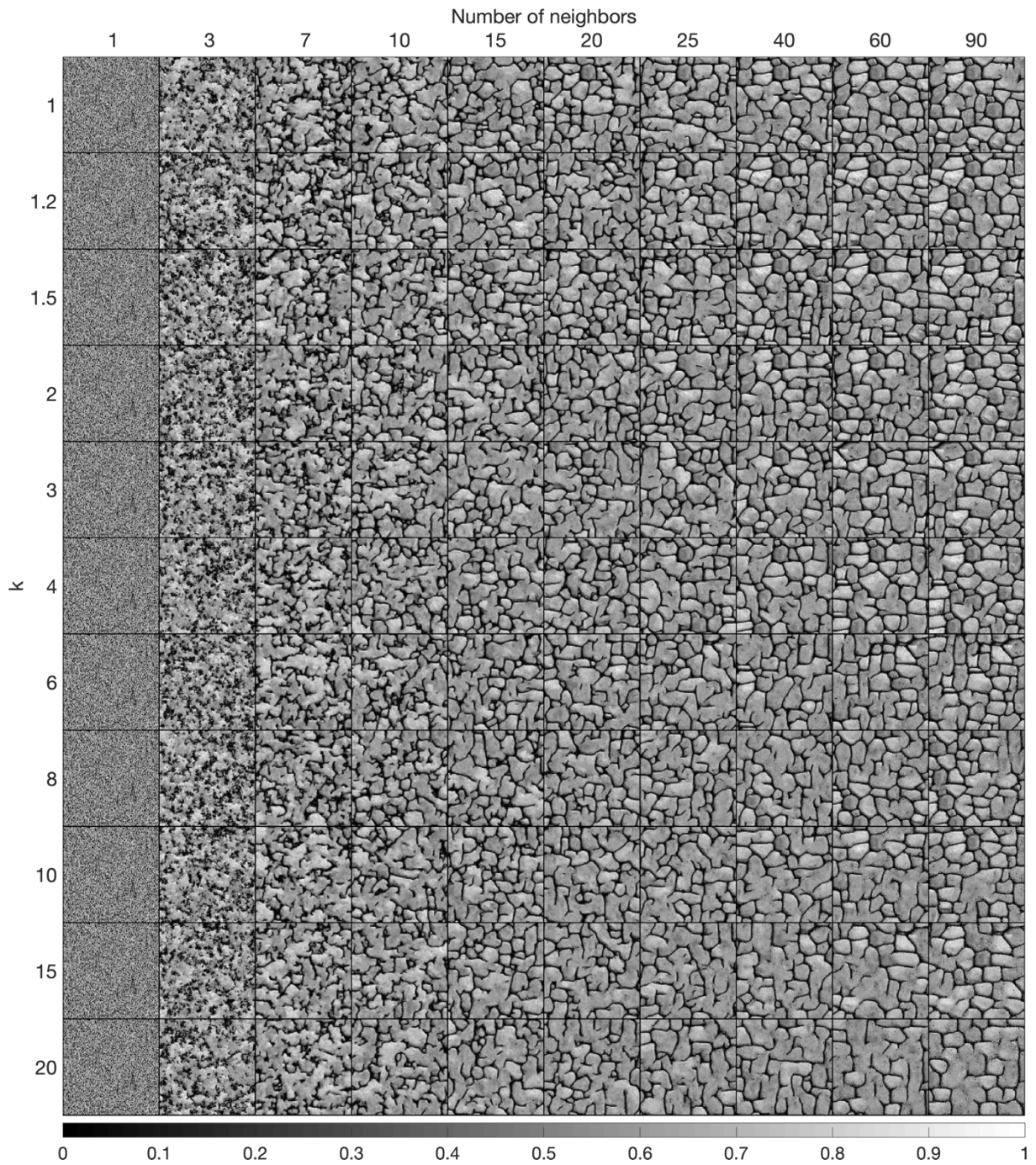
Table A 3 Simulation parameters for Figure A 2. Times are specified for simulations without parallelization.

674

	Concrete 1	Concrete 2	F42A	Folds continues
Source	(Meerschman et al., 2013)	(Meerschman et al., 2013)	Doi:10.6084/m9.fig share.1189259	(Mariethoz and Caers, 2014)
Size of the training image (px)	150 × 150 × 150	100 × 90 × 80	100 × 100 × 100	180 × 150 × 120
Size of the simulation (px)	100 × 100 × 100	100 × 100 × 100	100 × 100 × 100	180 × 150 × 120
Computation time (s)	11436	1416	1638	7637
k	1.2			
N	50		125	

675
676

Table A 4 Simulation parameters for Figure A 3. Times are specified for simulations without parallelization.

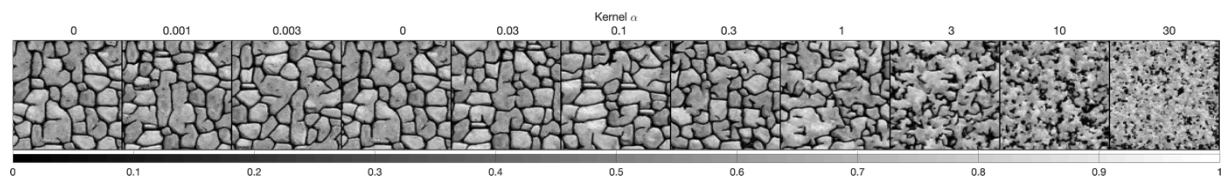


677

678

679

Figure A 4 Complete sensitivity analysis, with one simulation for the two main parameters of QS_{α}



680

681

682

Figure A 5 Complete sensitivity analysis, with one simulation for each kernel with $k=1.5$ and $N=40$

683 A.5. Mathematical derivation

684 The convolution theorem (Stockham, 1966; Krant, 1999; Li et al., 2019) can be easily extended
685 to cross-correlation (Bracewell, 2000). The following derivation shows the validity of the theorem
686 for any function f and g.

$$\begin{aligned} 687 \mathcal{F}\{f \star g\} &= \int (f \star g)(t) e^{it \cdot \xi} dt = \int \int \overline{f(s)} g(s+t) ds e^{it \cdot \xi} dt \\ 688 &= \int \int \overline{f(s)} e^{i(-s) \cdot \xi} ds \cdot g(s+t) ds e^{i(t+s) \cdot \xi} dt \\ 689 &= \int \int \overline{f(s)} e^{i(s) \cdot \xi} ds \cdot g(s+t) ds e^{i(t+s) \cdot \xi} dt = \overline{\mathcal{F}\{f\}} \cdot \mathcal{F}\{g\} \end{aligned}$$

690 The discretization of this property can be obtained using two piecewise continuous functions
691 associated to each discrete representation.

692 9. Acknowledgments

693 This research was funded by the Swiss National Science Foundation, grant number
694 200021_162882. Thanks to Intel for allowing us to conduct numerical experiments on their
695 latest hardware using the AI DevCloud. Thanks to Luiz Gustavo Rasera for his comments,
696 which greatly improved the manuscript; to [Dr. Ehsanollah](#) Baninajar for running his
697 optimization method, which improved the reliability of the benchmarks; and to all the early
698 users of QS for their useful feedback and their patience in waiting for this manuscript. [A](#)
699 [particular thanks to Prof. Ute Mueller and Prof. Thomas Mejer Hansen that accepted to review](#)
700 [the paper and provided constructive comments which significantly improved the quality of the](#)
701 [paper.](#)

702 10. References

- 703 Arpat, G. B. and Caers, J.: Conditional Simulation with Patterns, *Mathematical Geology*,
704 39(2), 177–203, doi:10.1007/s11004-006-9075-3, 2007.
- 705 Bancheri, M., Serafin, F., Bottazzi, M., Abera, W., Formetta, G. and Rigon, R.: The design,
706 deployment, and testing of kriging models in GEOframe with SIK-0.9.8, *Geosci. Model Dev.*,
707 11(6), 2189–2207, doi:10.5194/gmd-11-2189-2018, 2018.
- 708 Baninajar, E., Sharghi, Y. & Mariethoz, G.: MPS-APO: a rapid and automatic parameter
709 optimizer for multiple-point geostatistics, *Stoch Environ Res Risk Assess*, 33: 1969–1989,
710 doi:10.1007/s00477-019-01742-7, 2019
- 711 Barfod, A. A. S., Vilhelmsen, T. N., Jørgensen, F., Christiansen, A. V., Høyer, A.-S.,
712 Straubhaar, J. and Møller, I.: Contributions to uncertainty related to hydrostratigraphic
713 modeling using multiple-point statistics, *Hydrol. Earth Syst. Sci.*, 22(10), 5485–5508,
714 doi:10.5194/hess-22-5485-2018, 2018.

- 715 [Blagodurov, S., Fedorova, A., Zhuravlev, S., & Kamali, A.: A case for NUMA-aware](#)
716 [contention management on multicore systems. In 2010 19th International Conference on](#)
717 [Parallel Architectures and Compilation Techniques \(PACT\) \(pp. 557-558\), IEEE, 2010.](#)
- 718 [Bracewell, R. N.: The fourier transform and its applications. Boston: McGraw-hill, 2000](#)
- 719 Cooley, J. W., computation, J. T. M. O.1965: An algorithm for the machine calculation of
720 complex Fourier series, JSTOR, 19(90), 297, doi:10.2307/2003354, 1965.
- 721 [Dimitrakopoulos, R., Mustapha, H. & Gloaguen, E.: High-order Statistics of Spatial Random](#)
722 [Fields: Exploring Spatial Cumulants for Modeling Complex Non-Gaussian and Non-linear](#)
723 [Phenomena. Math Geosci 42, 65, doi: 10.1007/s11004-009-9258-9, 2010](#)
- 724 Dong, H. and Blunt, M. J.: Pore-network extraction from micro-computerized-tomography
725 images, Phys. Rev. E, 80(3), 84–11, doi:10.1103/PhysRevE.80.036307, 2009.
- 726 Frigo, M. and Johnson, S. G.: FFTW, [online] Available from: <http://www.fftw.org/fftw3.pdf>,
727 2018.
- 728 Gauss, C. F.: Demonstratio nova theorematis omnem functionem algebraicam. 1799.
- 729 Gómez-Hernández, J. J. and Journel, A. G.: Joint Sequential Simulation of MultiGaussian
730 Fields, in Geostatistics Tróia '92, vol. 5, pp. 85–94, Springer, Dordrecht, Dordrecht. 1993.
- 731 Graeler, B., Pebesma, E. and Heuvelink, G.: Spatio-Temporal Interpolation using gstat, R
732 Journal, 8(1), 204–218, 2016.
- 733 Gravey, M., Rasera, L. G. and Mariethoz, G.: Analogue-based colorization of remote sensing
734 images using textural information, ISPRS Journal of Photogrammetry and Remote Sensing,
735 147, 242–254, doi:10.1016/j.isprsjprs.2018.11.003, 2019.
- 736 Guardiano, F. B. and Srivastava, R. M.: Multivariate Geostatistics: Beyond Bivariate
737 Moments, in Geostatistics Tróia '92, vol. 5, pp. 133–144, Springer, Dordrecht, Dordrecht.
738 1993.
- 739 Hamming, R. W.: Error detecting and error correcting codes, edited by The Bell system
740 technical, The Bell system technical, 29(2), 147–160, doi:10.1002/j.1538-
741 7305.1950.tb00463.x, 1950.
- 742 Hoffmann, J., Scheidt, C., Barfod, A., Caers, J.2017: Stochastic simulation by image quilting
743 of process-based geological models, Elsevier, doi:10.1016/j.cageo.2017.05.012, 2017.
- 744 Honarkhah, M. and Caers, J.: Stochastic Simulation of Patterns Using Distance-Based Pattern
745 Modeling, Math Geosci, 42(5), 487–517, doi:10.1007/s11004-010-9276-7, 2010.
- 746 Intel Corporation: Intel® Math Kernel Library Reference Manual - C, 1–2606, 2019.
- 747 Jha, S. K., Mariethoz, G., Evans, J., McCabe, M. F. and Sharma, A.: A space and time scale-
748 dependent nonlinear geostatistical approach for downscaling daily precipitation and
749 temperature, Water Resources Research, 51(8), 6244–6261, doi:10.1002/2014WR016729,
750 2015.

- 751 John Paul Shen, M. H. L.: Modern Processor Design: Fundamentals of Superscalar
752 Processors, 1–658, 2018.
- 753 [Krantz, S. G.: A panorama of harmonic analysis. Washington, D.C.: Mathematical](#)
754 [Association of America, 1999](#)
- 755 Latombe, G., Burke, A., Vrac, M., Levavasseur, G., Dumas, C., Kageyama, M. and Ramstein,
756 G.: Comparison of spatial downscaling methods of general circulation model results to study
757 climate variability during the Last Glacial Maximum, *Geosci. Model Dev.*, 11(7), 2563–2579,
758 doi:10.5194/gmd-11-2563-2018, 2018.
- 759 [Li, B., & Babu, G. J.: A graduate course on statistical inference. New York: Springer, 2019](#)
- 760 Li, J. and Heap, A. D.: Spatial interpolation methods applied in the environmental sciences: A
761 review, *Environ Model Softw*, 53(C), 173–189, doi:10.1016/j.envsoft.2013.12.008, 2014.
- 762 Li, X., Mariethoz, G., Lu, D. and Linde, N.: Patch-based iterative conditional geostatistical
763 simulation using graph cuts, *Water Resources Research*, 52(8), 6297–6320,
764 doi:10.1002/2015WR018378, 2016.
- 765 Mahmud, K., Mariethoz, G., Caers, J., Tahmasebi, P. and Baker, A.: Simulation of Earth
766 textures by conditional image quilting, *Water Resources Research*, 50(4), 3088–3107,
767 doi:10.1002/2013WR015069, 2014.
- 768 Mariethoz, G.: A general parallelization strategy for random path based geostatistical
769 simulation methods, *Computers and Geosciences*, 36(7), 953–958,
770 doi:10.1016/j.cageo.2009.11.001, 2010.
- 771 Mariethoz, G. and Caers, J.: Multiple-point geostatistics: stochastic modeling with training
772 images, Wiley. 2014.
- 773 Mariethoz, G. and Kelly, B. F. J.: Modeling complex geological structures with elementary
774 training images and transform-invariant distances, *Water Resources Research*, 47(7), 959–14,
775 doi:10.1029/2011WR010412, 2011.
- 776 Mariethoz, G. and Lefebvre, S.: Bridges between multiple-point geostatistics and texture
777 synthesis_ Review and guidelines for future research, *Computers and Geosciences*, 66(C),
778 66–80, doi:10.1016/j.cageo.2014.01.001, 2014.
- 779 Mariethoz, G., Renard, P. and Straubhaar, J.: The Direct Sampling method to perform
780 multiple-point geostatistical simulations, *Water Resources Research*, 46(11),
781 doi:10.1029/2008WR007621, 2010.
- 782 Matheron, G.: The intrinsic random functions and their applications, *Advances in Applied*
783 *Probability*, 5(3), 439–468, doi:10.2307/1425829, 1973.
- 784 Meerschman, E., Pirot, G., Mariethoz, G., Straubhaar, J., Van Meirvenne, M. and Renard, P.:
785 A practical guide to performing multiple-point statistical simulations with the Direct
786 Sampling algorithm, *Computers and Geosciences*, 52(C), 307–324,
787 doi:10.1016/j.cageo.2012.09.019, 2013.

- 788 Oriani, F., Ohana-Levi, N., Marra, F., Straubhaar, J., Mariethoz, G., Renard, P., Karnieli, A.
789 and Morin, E.: Simulating Small-Scale Rainfall Fields Conditioned by Weather State and
790 Elevation: A Data-Driven Approach Based on Rainfall Radar Images, *Water Resources*
791 *Research*, 15(4), 265, doi:10.1002/2017WR020876, 2017.
- 792 Rasera L.G., Gravey M., Lane S. N., Mariethoz G. Downscaling images with trends using
793 multiple-point statistics simulation: An application to digital elevation models, *Mathematical*
794 *Geosciences*, 1–43, doi:10.1007/s11004-019-09818-4, 2019
- 795 Renard, P. and Allard, D.: Connectivity metrics for subsurface flow and transport, *Advances*
796 *in Water Resources*, 51(C), 168–196, doi:10.1016/j.advwatres.2011.12.001, 2013.
- 797 Rodríguez, P.: A radix-2 FFT algorithm for modern single instruction multiple data (SIMD)
798 architectures, doi:10.1109/ICASSP.2002.5745335, 2002.
- 799 Shannon: A mathematical theory of communication, Wiley Online Library, 1948.
- 800 Straubhaar, J., Renard, P., Mariethoz, G., Froidevaux, R. and Besson, O.: An Improved
801 Parallel Multiple-point Algorithm Using a List Approach, *Math Geosci*, 43(3), 305–328,
802 doi:10.1007/s11004-011-9328-7, 2011.
- 803 Strebelle, S.: Conditional simulation of complex geological structures using multiple-point
804 statistics, *Mathematical Geology*, 34(1), 1–21, doi:10.1023/A:1014009426274, 2002.
- 805 Strebelle, S., Payrazyan, K. and Caers, J.: Modeling of a Deepwater Turbidite Reservoir
806 Conditional to Seismic Data Using Multiple-Point Geostatistics, *Society of Petroleum*
807 *Engineers*. 2002.
- 808 [Stockham, T. G., Jr.: High-speed convolution and correlation. Proceedings of the April 26-28,](#)
809 [1966, Spring Joint Computer Conference on XX - AFIPS '66 \(Spring\). Presented at the the](#)
810 [April 26-28, 1966, Spring joint computer conference, doi:10.1145/1464182.1464209, 1966](#)
- 811 Tadić, J. M., Qiu, X., Miller, S. and Michalak, A. M.: Spatio-temporal approach to moving
812 window block kriging of satellite data v1.0, *Geosci. Model Dev.*, 10(2), 709–720,
813 doi:10.5194/gmd-10-709-2017, 2017.
- 814 Tadić, J. M., Qiu, X., Yadav, V. and Michalak, A. M.: Mapping of satellite Earth observations
815 using moving window block kriging, *Geosci. Model Dev.*, 8(10), 3311–3319,
816 doi:10.5194/gmd-8-3311-2015, 2015.
- 817 Tahmasebi, P.: Structural Adjustment for Accurate Conditioning in Large-Scale Subsurface
818 Systems, *Advances in Water Resources*, 1–52, doi:10.1016/j.advwatres.2017.01.009, 2017.
- 819 Tahmasebi, P., Sahimi, M., Mariethoz, G. and Hezarkhani, A.: Accelerating geostatistical
820 simulations using graphics processing units (GPU), *Computers and Geosciences*, 46(C), 51–
821 59, doi:10.1016/j.cageo.2012.03.028, 2012.
- 822 Vannamettee, E., Babel, L. V., Hendriks, M. R., Schuur J.: Semi-automated mapping of
823 landforms using multiple point geostatistics, Elsevier, doi:10.1016/j.geomorph.2014.05.032,
824 2014.

- 825 Wojcik, R., McLaughlin, D., on, A. K. I. T.2009: Conditioning stochastic rainfall replicates
826 on remote sensing data, doi:10.1109/TGRS.2009.2016413, 2009.
- 827 Yin, G., Mariethoz, G. and McCabe, M.: Gap-Filling of Landsat 7 Imagery Using the Direct
828 Sampling Method, Remote Sensing, 9(1), 12, doi:10.3390/rs9010012, 2017.