https://doi.org/10.5194/gmd-2022-229
Preprint. Discussion started: 17 October 2022

Geoscientific Model Development
Discussions

AutoQS v1: Automatic parameterization of QuickSampling based on training images analysis

Mathieu Gravey[1,2], Grégoire Mariethoz[1]

[1]University of Lausanne, Faculty of Geosciences and Environment, Institute of Earth Surface Dynamics, Switzerland
[2]Department of Physical Geography, Faculty of Geosciences, Utrecht University, Utrecht, Netherlands

Correspondence to: Mathieu Gravey (research@mgravey.com)

Highlights

- Adaptative calibration as a function of the simulation progression
- Calibration depends only on the training image
- Robust parameterization based on a rapid prior analysis of the training image

Abstract. Multiple-point geostatistics are widely used to simulate complex spatial structures based on a training image. The use of these methods relies on the possibility of finding optimal training images and parametrization of the simulation algorithms. While methods for selecting training images are available, parametrization can be cumbersome. Here, we propose finding an optimal set of parameters using only the training image as input. The difference between this and previous work that used parametrization optimization is that it does not require the definition of an objective function. It is based on the analysis of the errors that occur when filling artificially constructed patterns that have been borrowed from the training image. The main advantage of our approach is to remove the risk of overfitting an objective function, which may result in underestimating the variance or in a verbatim copy of the training image. Since it is not based on optimization, our approach finds a set of acceptable parameters in a predictable manner by using the knowledge and understanding of how the algorithms work. The technique is explored in the context of the recently developed QuickSampling algorithm, but it can be easily adapted to other pixel-based multiple-point statistics algorithms using pattern matching, such as Direct Sampling or Single Normal Equation Simulation (SNESIM).

1 Introduction

Geostatistics is extensively used in natural sciences to map spatial variables such as surface properties (e.g., soils, geomorphology, meteorology) and subsurface geological features. Its main applications involve the estimation and simulation of natural phenomena. In this paper, we focus on simulation approaches.

Traditional two-point geostatistical simulations preserve the histogram and variogram inferred from point data (Matheron, 1973). However, inherent limitations make the reproduction of complex structures difficult. Multiple-point statistics (MPS), by accounting for more complex relations, enables the reproduction of such complex structures (Guardiano and Srivastava, 1993). However, MPS has its own limitations (Mariethoz and Caers, 2014). To perform satisfactorily, MPS algorithms require analog images (called training images) and appropriate parametrization. Training images can often be provided by expert knowledge. Indeed, the training image is related to the property that is being simulated, and therefore it is common to all MPS algorithms. In addition, several methods have been proposed to automatically select an appropriate training image among a set of candidates (Pérez et al., 2014; Abdollahifard et al., 2019) . However, the parametrization of an MPS algorithm depends not only on the chosen training image but also on the specifics of the algorithm. This makes the task of finding good parametrization cumbersome, and therefore, users often have to resort to a trial-and-error approaches (Meerschman et al., 2013).

Over the last few years, several studies have addressed the challenge of finding good MPS parameters. These can be categorized into two different philosophies. The first approach is focused on the "simulation grid", which assumes that a parametrization is related to the simulation grid, the training image and the MPS algorithm. Dagasan et al. (2018) proposed using the known hard data from the simulation grid as a reference to compute statistical metrics and then trying to improve the parametrization through a simulated annealing optimization process until the metrics matched as closely as possible. The second approach is focused on the "training image" and assumes that the parametrization is only related to the training image and MPS algorithm. Along these lines, Baninajar et al. (2019) proposed the MPS Automatic Parameter Optimizer (MPS-APO) method based on the cross-validation of the training image (TI) to quantify simulation quality and CPU cost. In this approach, artificially generated gaps in the high gradient areas of the training image are created, and MPS algorithms are used to simulate the gaps. The performance of a particular parameterization is quantified by assessing the correspondence between the filled and original training data. By design, this approach is extremely interesting for gap-filling problems. The authors state that it can be used for the parametrization of unconditional simulations; however, the use of limited gaps cannot guarantee the reproduction of long-range dependencies. Furthermore, due to the design of the framework for generating gaps, each MPS algorithm needs to be able to handle gap-filling problems for the error to be estimated properly.

If both approaches show good results, then they are both related to optimization methods, and therefore, the user has no control over the duration of the optimization process. Furthermore, an objective function is needed. Finding this objective function is a challenge in itself because it can change depending on the training image used. Using optimization approaches, many metrics can be accounted for in the objective function, such as histogram, variogram, pattern histogram, connectivity function, Euler characteristic, etc., (Boisvert et al., 2010; Renard and Allard, 2013; Tan et al., 2013) or a weighted combination of these. Similarly, one has to define the meta-parameters linked to the optimization algorithm itself, such as the cooling rate in simulated annealing or maximum number of iterations. As a result, MPS parameter optimization approaches tend to be complex and difficult to use.

In this contribution, we propose skipping the complexity of an optimization algorithm and instead simplifying the optimization procedure to a key element: the simulation of a single pixel. The underlying principle of our approach is that a sequence of well-simulated pixels converges to a good simulation overall. Therefore, the goal is to find the optimal parameters to simplify the simulation of a single pixel using the training image as the only reference. Baninajar et al. (2019) showed that computing the prediction error (i.e., the error between the simulation and the reference) is an appropriate metric to find optimal parameters. Following this approach, we propose exhaustively exploring the parameter space by performing pixel predictions over patterns extracted from the training image,

https://doi.org/10.5194/gmd-2022-229
Preprint. Discussion started: 17 October 2022

and compute the associated prediction error. This results in a prediction error map for each combination of parameters.

The remainder of this paper is structured as follows: Section 2 presents the proposed method. Section 3 evaluates the approach in terms of quantitative and qualitative metrics. Section 4 discusses the strengths and weaknesses of the proposed approach and presents the conclusions of this work.

2 Challenges related to inappropriate parameters

The hypothesis behind multiple point simulation is that the neighborhood of a given pixel *x* (the pattern generated by known or previously simulated pixels) is informative enough to constrain the probability density function of the value Z(*x*). Therefore, it requires a training image with enough repetition of the pattern (large enough) to allow the computation of such a conditional probability distribution. The Extended Normal Equation Simulation (ENESIM) (Guardiano and Srivastava, 1993) algorithm computes this distribution for the simulation of each pixel, therefore ensuring such a property. To provide a similar guarantee, the SNESIM (Strebelle, 2002) algorithm and the Improved Multiple-Point Parallel Algorithm using a List Approach (IMAPLA) (Straubhaar et al., 2011), include a parameter to define a minimum number of replicates. Direct Sampling (DS) (Mariethoz et al., 2010) adopts a different strategy by allowing for the interrupted exploration of the training image. It includes a distance threshold parameter that defines what is an acceptable match for a neighborhood; however, too small a threshold typically results in a verbatim copy of the training image.

To reduce this issue, a maximal fraction of the explored training image is introduced, which is also called the exploration ratio. Since QuickSampling (QS) (Gravey and Mariethoz, 2020) also suffers from the verbatim copy issue with the use of the number of candidates being k=1, the authors allow and recommend the use of k>1, as k is similar to the number of replicates in SNESIM or IMPALA. A value k=1.5 in QS can be visualized as SNESIM with a minimum number of replicates of 1 for 50% of the time and 2 for the remaining 50% of the time.

The phenomenon of verbatim copy is the complete pasting of a section from the training image to the simulation (an unintentionally similar process as that in patch-based approaches (Rezaee et al., 2013)). This means that the relative position of the simulated pixels is the same as that in the training image. This occurs when the neighborhood constraints on the simulated pixels are too strong and only the exact same patterns as those in the training image are acceptable. To detect this issue, a common strategy is to create a position map (similar to the index map), which represents the provenance of simulated values by mapping their original coordinates in the training image, as shown in **Figure 1**.

Verbatim copy can appear in many forms; **Figure 1** shows the most common ones. The pure verbatim (the most common type of verbatim copy) is a simple copy of the image, with all pixels in the same order inside of the patches. Block verbatim typically appears when there are many replicates of a very specific type of pattern in the training image and few replicates of all other patterns. Therefore, the MPS algorithm is forced to switch at the position of this common pattern. Structural verbatim is an example of pure verbatim over the white pixels. Structural verbatim tends to appear when large-scale structures are unique in the training image, which often allows a visually satisfying image to be quickly obtained. Often, users are ready to sacrifice verbatim on large-scale structures, but this can easily introduce bias, which is one of the hardest types of verbatim to detect. This can typically occur when the maximum neighborhood radius is too large. We then perfectly duplicate the large and initial structure. Finally, no verbatim, which is the expected result of simulations, is when the position pixel does not have any particular relations (their position is unpredictable).

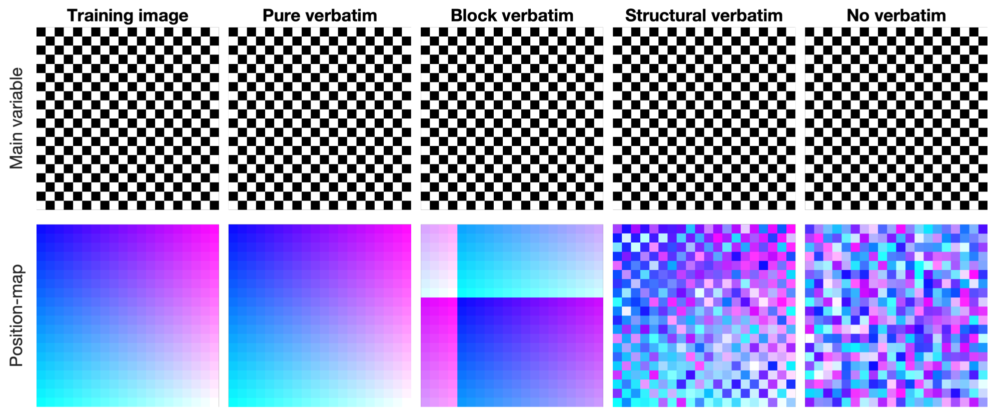


Figure 1 Visualization of verbatim copies using a position map. This is an extreme case that highlights that verbatim is not defined by the value simulated but by the source of the value.

3 Method

The objective of the presented approach is to find an optimal set of parameters using only the training image and knowledge of the mechanics of the simulation algorithm as information. The simulation algorithm is not used in this context; in fact, simulations are not required to obtain a proper calibration. The main target application of the presented approach is the pattern matching simulation algorithm QuickSampling (QS), where the values, at a pixel scale, are directly sampled from the training image. The method is suitable for the simulation of continuous and/or categorical variables. Binary variables are a particular case of continuous and categorical variables.

Simulation algorithms, such as QS, can be summarized by Algorithm 1. The key operation occurs at Line 3, which is when the algorithm searches for an optimal match based on the neighboring conditioning data.

Algorithm 1 The QS algorithm

```
Inputs:
$T$: training images
$S$: simulation grid, including the conditioning data
$P$: simulation path
$\theta$: parametrization (including $n$: number of neighbors)

1.  For each unsimulated pixel $x$ following the path $P$:
2.      Find the neighborhood $N(x)$ in $S$ composed of the $n(\theta)$ closest neighbors
3.      Find a candidate in $T$ those matches $N(x)$ using$\theta$
4.      Assign the value $A$ of the selected candidate to $x$ in $S$
5.  End
```

Here, we propose applying a divide and conquer approach by dividing any pixel-based sequential simulation into its atomic operation: the simulation of a single pixel. We assume that if all pixels are perfectly simulated, then the

https://doi.org/10.5194/gmd-2022-229
Preprint. Discussion started: 17 October 2022

resulting simulation should also be good. A perfectly simulated pixel is a pixel that respects the conditional probability distribution.

Considering the simulation of a pixel, many values can potentially be valid.

$$|\{A|P(A|N(x)) > 0\}| \geq 1 \tag{1}$$

where $|.|$ represents the cardinality of a set. $P\big(A\big|N(x)\big)$ denotes the probability of A (a given value) knowing $N(x)$, the neighborhood. Each possibility will still respect the probability distribution.

The proposed approach consists of finding a set of parameters that results in accurate samples for each pattern. An extreme and undesired situation occurs from the simulation of a value that came from the sampling of perfect matches (the neighborhood is available in the training image), which results in a simulation identical to the training image and therefore constitutes verbatim copy.

The search for the optimal parametrization is carried out by exhaustive exploration (Algorithm 2). The prediction error is computed, which is the difference between the original value of the pattern and the value of the selected training image pattern (Figure 2).

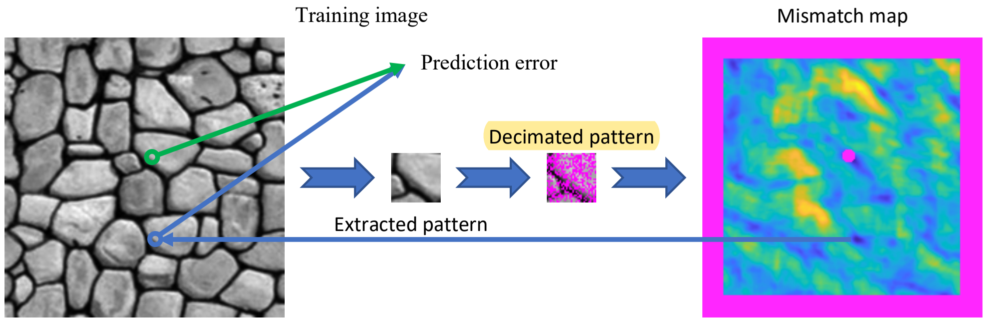


Figure 2 All steps for a single pattern, summarizing Algorithm 2, Lines 2-4.

The proposed algorithm explores a discretized parameter space θ (Algorithm 2, Line 1). While this discretization is natural for some parameters, such as n, which is an integer, it can require an explicit discretization for other parameters, such as the kernel in QS (or potentially the th in DS). Furthermore, a key component of our method is the exploration of the parameter space for several representative stages D of the simulation (Algorithm 2, Line 1). In the case of a random path, the progression of a simulation is directly related to the density of the neighborhoods, therefore D represents the density of a neighborhood. For each combination D and θ, multiple measures over a set of random locations $\mathcal{V}$ ($500 < |\mathcal{V}| < 10000$) occur at Lines 1-5 in Algorithm 2 and their mathematical expression is shown in Equation 2,

Algorithm 2

```
Inputs:
List of $D$, and list of $\theta$
$T$ the training image
1. **For** each possible combination of $D$ and $\theta$ **do** for all $v \in \mathcal{V}$ ($\mathcal{V}$ is randomly generated):
2.     Sample a neighborhood $N(v)$ from $T$ respecting $D$
3.     Using $\theta$, find a candidate in $T$ that matches $N(v)$, excluding for $v$ itself
4.     Compute the error $\varepsilon$ between the selected candidate and$Z(v)$
5. **End**
6. Analyze the errors $\varepsilon$ to determine the best $\theta$ for each $D$.
```

$$\varepsilon(\theta, D, T) = \sqrt{\frac{1}{|\mathcal{V}|}\sum_{v\in\mathcal{V}}\left(Z(v) - Z\left(\underset{T\setminus\{v\}}{\mathrm{Cand}}\big(\theta, N(v, D)\big)\right)\right)^2} \qquad (2)$$

where $\mathrm{Cand}(\theta, N)$ returns a single candidate position for a given neighborhood N and follows the parametrization θ. $N(v, D)$ denotes a decimated neighborhood around v that respects the condition D. $\mathcal{V}$ represents a random set of positions in the training image, and $Z(v)$ refers to the actual value at position $v \in \mathcal{V}$ in the training image. Finally, for each stage considered, the set of parameters with the minimum associated error ε is considered optimal (in Algorithm 2, Line 6).

$$\varepsilon(\theta_{optimal}, D, T) = \min_{\theta} \varepsilon(\theta, D, T) \qquad (3)$$

To avoid over-constrained situations from generating a verbatim copy of the training image, the position v and its direct neighbors (in a small radius, usually around 5 pixels, but can we increase depending of the small scale structure of the training image) are removed from the set of potential candidates. Furthermore, in the case of equality between several optimal options, we propose the simple rules of taking the cheapest parameter set in terms of computational cost (e.g., the smallest n).

3.1 An efficient implementation

In practice, the implementation of Algorithm 2 divides θ into two subsets of parameters: θ_h and θ_s. θ_h contains all the parameters that affect the computation of the match of a single pattern. This is dependent on the algorithm; in the case of QS, these are the number of neighbors n, and the kernel (in DS, it would be th, the threshold, and n). θ_s includes the parameters related to the sampling process of the training image. In the case of QS, these are the number of matches to retain k, the number of candidates (and in the case of DS, f, which is the fraction of the training image that is scanned). Interestingly, we can precompute and store of all matches for a given parameterization θ_h. Then, the saved matches of θ_h can be used to quickly measure all possibilities for the parameters in $\theta = \theta_h + \theta_s$. This two-step approach allows to significantly reduce redundant computations. It is possible to further accelerate this algorithm by aborting the estimation of ε if the error remains high after having tested only a small number (at least 500) of samples from $\mathcal{V}$. To carry out this last operation efficiently, the algorithm increases $\mathcal{V}$ for the parameter combinations of interest. At each increase step, it checks if more computations are needed. The following rules proved a good trade-off:

$$\varepsilon(\theta, D, T) - \frac{1}{2}\sigma(\theta, D, T) > \varepsilon(\theta_{min}, D, T) + \frac{1}{2}\sigma(\theta_{min}, D, T)$$
$$\varepsilon(\theta_{min}, D, T) = \min_{\theta} \varepsilon(\theta, D, T)$$
$$\sigma(\theta, D, T) = \sqrt{\frac{1}{|\mathcal{V}|}\sum_{v \in \mathcal{V}}\left(\left(Z(v) - Z\left(\underset{T\setminus\{v\}}{\text{Cand}}(\theta, N(v, D))\right)\right) - \varepsilon(\theta, D, T)\right)^2} \quad (4)$$

With $\varepsilon(.)$ the error, and $\sigma(.)$ the standard deviation. Therefore, a given parametrization is only further explored if the error is a range of a σ.

4 Result

All experimental tests in this section are performed using the training image shown in Figure 2. The stages D are distributed following a logarithmic scale. Experimentation shows that the nodes simulated in the initial stages of the path are critical for the overall simulation

4.1 Automatic calibration for QS

In the case of QS, the method finds optimal values for k the number of candidates, n the number of neighbors and ω the kernel.

4.1.1 Automatic calibration for QS with a uniform kernel

In this first test, we use the configuration $\theta_h = \{n\}$ and $\theta_s = \{k\}$. The results are shown in Figure 3. It shows the optimal number of candidates k and number of neighbors as a function of the simulation progress (equivalent to the neighborhood density D). The ignorance threshold is defined by computing the average error between elements of the marginal distribution. It represents the error value beyond which no information is extracted from the neighborhood, stage where the simulated value can then be drawn from the marginal distribution without introducing bias.

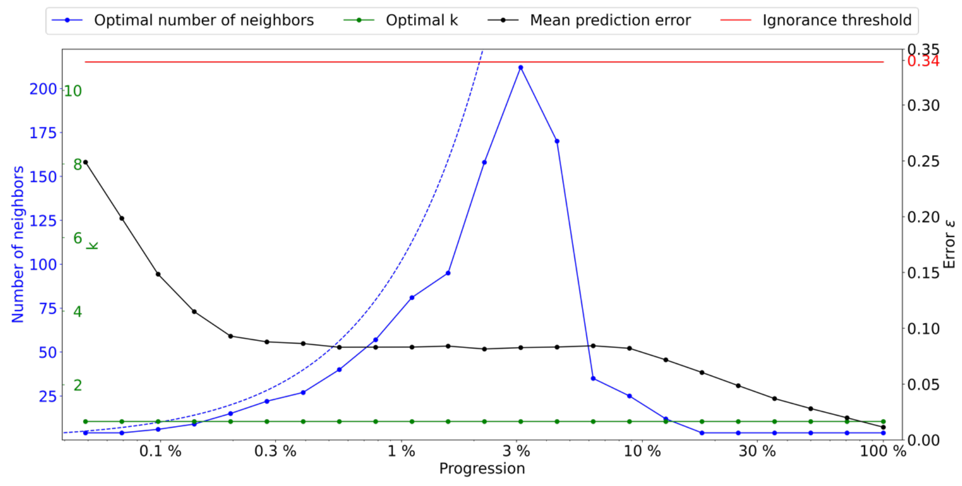


Figure 3 Optimal parameters for QS (k in green and number of neighbors in blue) as a function of the progression, with the associated prediction error (in black). The red line represents the ignorance threshold. The dashed blue line indicates the average density for the neighborhood considered.

The optimal k remains small throughout the simulation because the training image is not sufficiently repetitive (large). At early stages of the simulation, it seems important to use many neighbors. The number of neighbors increases until approximately 3% of the simulation, followed by a subsequent drastic reduction. This tends to indicate that once the large structures are informed, only the few direct neighbors are important. We also note that even if the parametrization is logical, it is generally difficult to predict. This indicates that the use of a single parametrization for the entire MPS simulation is generally suboptimal. Figure 3 also shows that the first few simulated pixels are hardly predictable (close to the ignorance threshold).

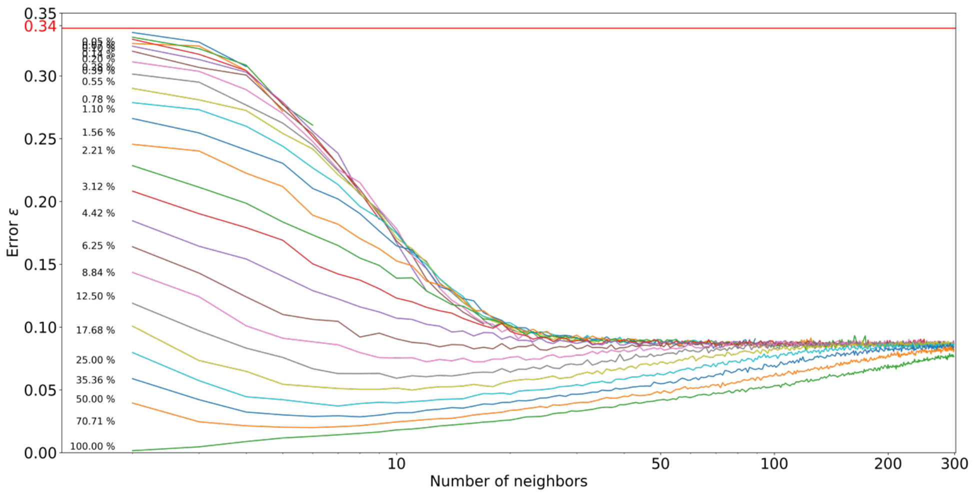


Figure 4 Error as a function of the number of neighbors, with k=1, where each curve represents the associated density of the neighborhood D (which is equivalent to the fraction of the simulation path).

Figure 4 shows the evolution of ε as a function of the number of neighbors in the simulation stage. This indicates that two regimes exist in the simulation. In the first percentages of the simulation, an optimal prediction can be obtained even with a small number of neighbors. However, as the neighborhoods become denser, the importance of spatial continuity takes over. This two-step process is expected, as random large-scale features are generated first; then, in a second step, the MPS algorithm fills the image with a consistent fine-scale structures. Furthermore, it shows that using a large number of neighbors at the end of the simulation generates suboptimal results, which could explain the small-scale noise that is sometimes visible in some MPS simulations.

4.1.2 Automatic calibration of QS considering different kernels

Here, we use the following configuration $\theta_h = \{n, \omega\}$ and $\theta_s = \{k\}$. We consider kernels as having a radial exponential shape, $w_i = e^{-\alpha . d_i}$.

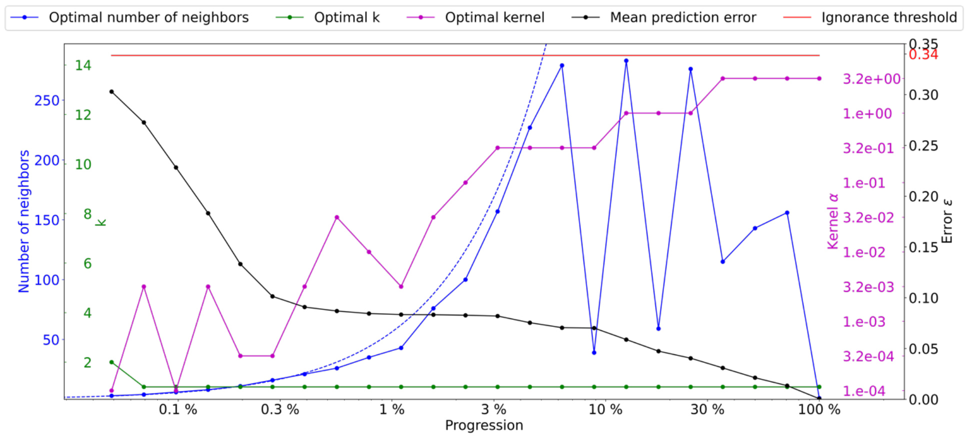


Figure 5 Optimal parameters for QS (k in green, number of neighbors in blue, and best kernel in magenta) as a function of the progression, with the associated prediction error (in black). The dashed blue line indicates the average density for the neighborhood considered.

Figure 5 shows the evolution of the QS parameters, where interferences between the number of neighbors and skewed kernels (high α) are visible. This interaction can be explained by the fact that the last neighbors will receive negligible weights with large α values, and thus n becomes insensitive. In that case, the differences between possible configurations are negligible, with random noise in the metric. As expressed in the methodology section, in cases of a similar error, the cheapest solution is considered. In the case of QS, having a large number of neighbors can marginally increase the computational time; therefore, we introduce a small tolerance that results in favoring small n values. This tolerance is introduced as a small extra cost for each extra neighbor, for example adding 5e-5 for each extra neighbor. When the gain during simulations was limited, up to a 10% computational gain was observed using Equation 4.

https://doi.org/10.5194/gmd-2022-229
Preprint. Discussion started: 17 October 2022

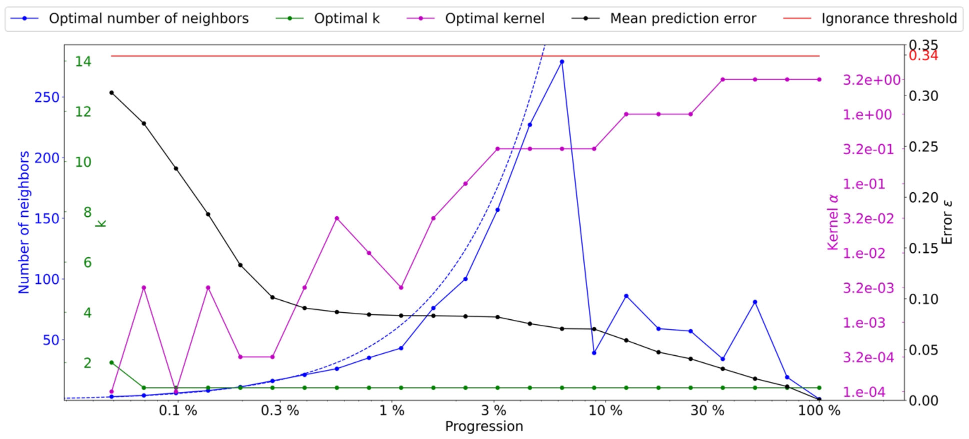


Figure 6 Optimal parameters for QS (k in green and number of neighbors in blue, and the best kernel in magenta) as a function of the progression, with the associated prediction error (in black). The dashed blue line is the average density for the neighborhood considered.

Figure 6 shows similar quality (ε curves) as Figure 5. However, the number of neighbors required during the simulation drastically decreases as advanced simulation stages, and the wild oscillations are avoided.

4.2 Sequential simulation using automatic calibration

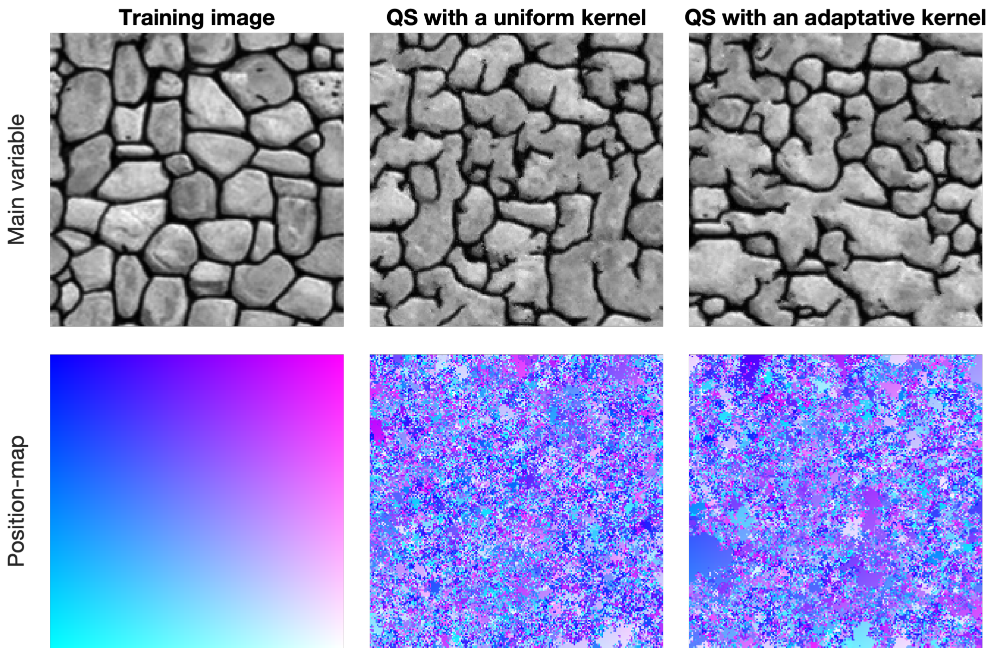


Figure 7 Simulation using QS with parameters generated by the automatic calibration.

Figure 7 shows qualitative results using the evolutive parametrization resulting of the proposed autocalibration. QS with kernel refers to the use of different values of alpha for the kernel. In this case, the results are similar to state-of-the-art simulations using a manual calibration. Tests using QS with a uniform kernel fail to reproduce

https://doi.org/10.5194/gmd-2022-229
Preprint. Discussion started: 17 October 2022

some structures; in particular, the size of the objects is incorrect. Each verbatim map shows few homogenous areas; therefore, realizations are produced with a low rate of verbatim copy.
From a quantitative point of view, Figure 8 illustrates different metrics across a set of 100 simulations. The automatic calibration method proposed here allows for obtaining better quality simulations than in the original QS article.
Figure 9 shows that variogram and connectivity metrics are well reproduced, although they have not been directly constrained in the calibration process. Indeed, the parameter optimization only considers the simulation of single pixels and never computes global metrics over an entire grid.

https://doi.org/10.5194/gmd-2022-229
Preprint. Discussion started: 17 October 2022

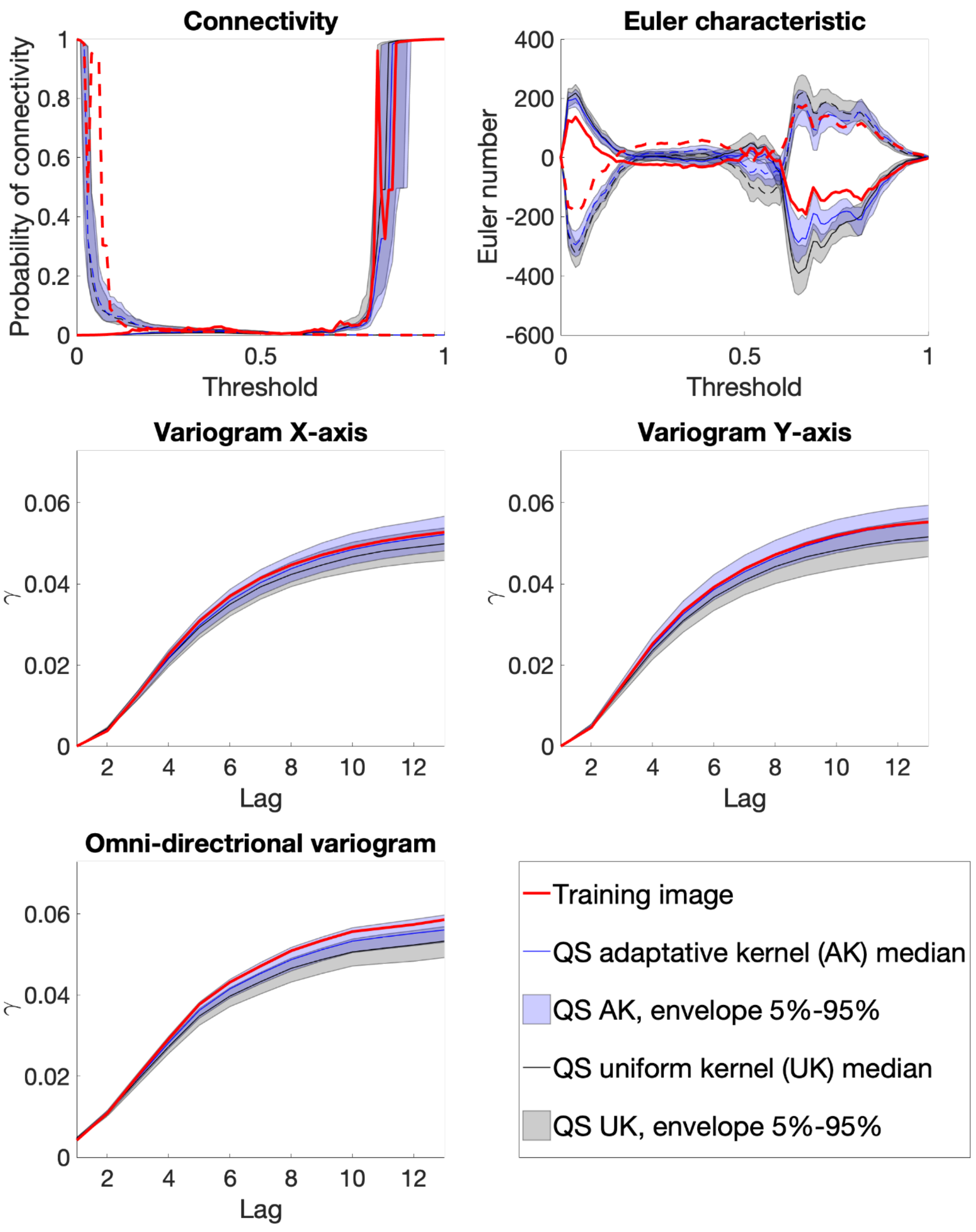


Figure 8 Benchmark between QS with an adaptive kernel (Figure 6) and a uniform (without) kernel (Figure 3) over 100 simulations for 5 different metrics.

5 Discussion

The proposed method allows for the automatic calibration of QS and potentially similar pixel-based MPS approaches, reaching a similar quality as that of manual parameterization from both quantitative and qualitative

https://doi.org/10.5194/gmd-2022-229
Preprint. Discussion started: 17 October 2022

points of view. The metrics confirm the good reproduction of training patterns because the method finds a calibration that avoids verbatim copy.

The major advantage of our approach is the absence of a complex objective function, which often itself requires calibration. The method runs in a predictable maximum time, which depends of the number of patterns tested, that relate on the expected quality of the calibration σ. The calibration can even be refined based on previous results, without running all the processes again, by adding steps, kernels, or by increasing $|\mathcal{V}|$.

Our approach cannot be used to determine an optimal simulation path because it focuses on the simulation of a single pixel. Furthermore, the method does not take into consideration the computational cost required for a simulation.

The computation time of the optimal parameters depends on the expected quality. For example, sometimes a kernel provides only a small improvement but requires many computations. A full exploration of a 250x250 image takes approximately 30 min. However, a result using a degraded parameter space can provide close results in less than 10 min. This degraded space can be constructed, for example, by subsampling the number of neighbors following a squared function or using some external/expert knowledge.

The method was explored for multivariable images, resulting in a larger parametrization space than with a single variable. The method provides good results independent of the task. Unfortunately, when performing this approach, each new parameter increases the computation time. This can lead to impractical scenarios, especially in the case of 4 or more variables.

In the context of testing the generality of the proposed approach, calibration was computed on multiple training images (found in the Appendix B, C). Unexpectedly, the calibration pattern with two regimes (n large, then n smaller) seems to be universal, at least for single variable simulations. While the position of the abrupt transition between each regime seems to vary greatly (between 0.5% and 20% of the path), the overall shape remains the same. Therefore, the approach proposed by Baninajar et al. (2019), in which long ranges are not considered, can be extended by using large n values in the early stages of the simulation.

6 Conclusion

The proposed approach allows for the automatic calibration of pixel-based MPS algorithms. Furthermore, it demonstrates that for optimal results, the parametrization cannot remain constant during the simulation and instead needs to evolve with the simulation progression. A visually appealing result of complex features without verbatim copy is difficult to simulate, especially when using a uniform kernel.

The proposed method allows for the calibration of a parametric kernel. However, in future work one can envision the optimization of a nonparametric kernel where the weight of each individual neighbor w_i is considered a variable to optimize using ε as an objective function (e.g., using machine learning frameworks).

The study of the evolution of parameters shows a smooth behavior of the average error. Therefore, the use of a multivariate fitting approaches to estimate the error surface with fewer evaluations could be an interesting solution to speed up the parametrization by capitalizing on neighbors (in parameter space). The use of machine learning to take advantage of transfer learning between training images also has a high potential. These two solutions will allow for the interpolation between parameters such as α of the kernel.

Code availability

The source code of the autoQS algorithm is available as part of the G2S package at: https://github.com/GAIA-UNIL/G2S (last access: 15 Sept 2022) under the GPLv3 license. Platform: Linux/macOS/Windows 10+. Language: C/C++. Interfacing functions in MATLAB, Python3, and R.

Author contributions.

MG proposed the idea, implemented and optimized the autoQS approach and wrote the article. GM provided supervision, methodological insights and contributed to the writing of thearticle.

Competing interests

The authors declare that they have no conflict of interest.

Acknowledgements

This research was funded by the Swiss National Science Foundation.

6.1 Financial support

This research has been supported by the Swiss National Science Foundation (grant no. 200021_162882).

7 References

Abdollahifard, M. J., Baharvand, M., and Mariéthoz, G.: Efficient training image selection for multiple-point geostatistics via analysis of contours, Computers &amp; Geosciences, 128, 41–50, https://doi.org/10.1016/j.cageo.2019.04.004, 2019.
Baninajar, E., Sharghi, Y., and Mariethoz, G.: MPS-APO: a rapid and automatic parameter optimizer for multiple-point geostatistics, Stoch Environ Res Risk Assess, 33, 1969–1989, https://doi.org/10.1007/s00477-019-01742-7, 2019.
Boisvert, J. B., Pyrcz, M. J., and Deutsch, C. V.: Multiple Point Metrics to Assess Categorical Variable Models, Nat Resour Res, 19, 165–175, https://doi.org/10.1007/s11053-010-9120-2, 2010.
Dagasan, Y., Renard, P., Straubhaar, J., Erten, O., and Topal, E.: Automatic Parameter Tuning of Multiple-Point Statistical Simulations for Lateritic Bauxite Deposits, Minerals, 8, 220, https://doi.org/10.3390/min8050220, 2018.
Gravey, M. and Mariethoz, G.: QuickSampling v1.0: a robust and simplified pixel-based multiple-point simulation approach, Geosci. Model Dev., 13, 2611–2630, https://doi.org/10.5194/gmd-13-2611-2020, 2020.
Guardiano, F. B. and Srivastava, R. M.: Multivariate Geostatistics: Beyond Bivariate Moments, in: Quantitative Geology and Geostatistics, Springer Netherlands, 133–144, https://doi.org/10.1007/978-94-011-1739-5_12, 1993.
Mariethoz, G., Caers, J., 2014. Multiple-point geostatistics: stochastic modeling with training images. Wiley.
Mariethoz, G., Renard, P., and Straubhaar, J.: The Direct Sampling method to perform multiple-point geostatistical simulations, Water Resour. Res., 46, https://doi.org/10.1029/2008wr007621, 2010.

Matheron, G.: The intrinsic random functions and their applications, Advances in Applied Probability, 5, 439–468, https://doi.org/10.2307/1425829, 1973.

Meerschman, E., Pirot, G., Mariethoz, G., Straubhaar, J., Van Meirvenne, M., and Renard, P.: A practical guide to performing multiple-point statistical simulations with the Direct Sampling algorithm, Computers &amp; Geosciences, 52, 307–324, https://doi.org/10.1016/j.cageo.2012.09.019, 2013.

Pérez, C., Mariethoz, G., and Ortiz, J. M.: Verifying the high-order consistency of training images with data for multiple-point geostatistics, Computers &amp; Geosciences, 70, 190–205, https://doi.org/10.1016/j.cageo.2014.06.001, 2014.

Renard, P. and Allard, D.: Connectivity metrics for subsurface flow and transport, Advances in Water Resources, 51, 168–196, https://doi.org/10.1016/j.advwatres.2011.12.001, 2013.

Rezaee, H., Mariethoz, G., Koneshloo, M., and Asghari, O.: Multiple-point geostatistical simulation using the bunch-pasting direct sampling method, Computers &amp; Geosciences, 54, 293–308, https://doi.org/10.1016/j.cageo.2013.01.020, 2013.

Straubhaar, J., Renard, P., Mariethoz, G., Froidevaux, R., and Besson, O.: An Improved Parallel Multiple-point Algorithm Using a List Approach, Math Geosci, 43, 305–328, https://doi.org/10.1007/s11004-011-9328-7, 2011.

Strebelle, S.: Mathematical Geology, 34, 1–21, https://doi.org/10.1023/a:1014009426274, 2002.

Tan, X., Tahmasebi, P., and Caers, J.: Comparing Training-Image Based Algorithms Using an Analysis of Distance, Math Geosci, 46, 149–169, https://doi.org/10.1007/s11004-013-9482-1, 2013.

https://doi.org/10.5194/gmd-2022-229
Preprint. Discussion started: 17 October 2022

Appendix

This supplementary material contains a similar calibration for other training images.

A. Stone

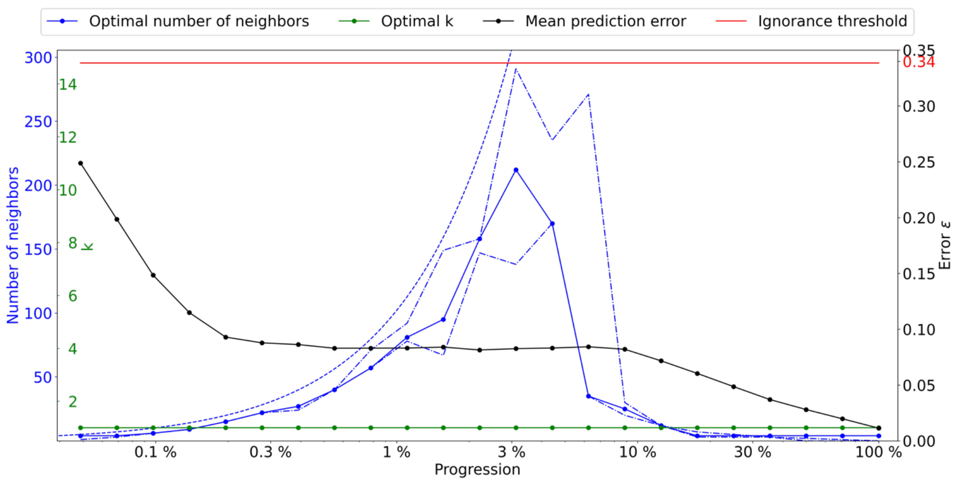


Figure A.1 Optimal parameters for QS (k in green and number of neighbors in blue) as a function of the progression, with the associated prediction error (in red). The red line represents the ignorance threshold. The dashed blue line is the average density for the neighborhood considered. The dot-dashed line represents the variability in 1% of the error.

B. Strebelle

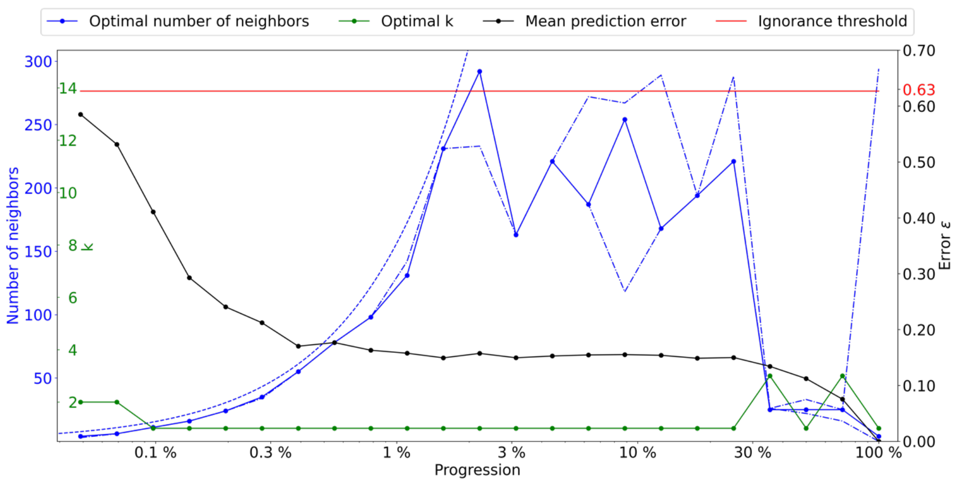


Figure B.1 Optimal parameters for QS (k in green and number of neighbors in blue) as a function of the progression, with the associated prediction error (in red). The red line represents the ignorance threshold. The dashed blue line is the average density for the neighborhood considered. The dot-dashed line represents the variability in 1% of the error.

https://doi.org/10.5194/gmd-2022-229
Preprint. Discussion started: 17 October 2022

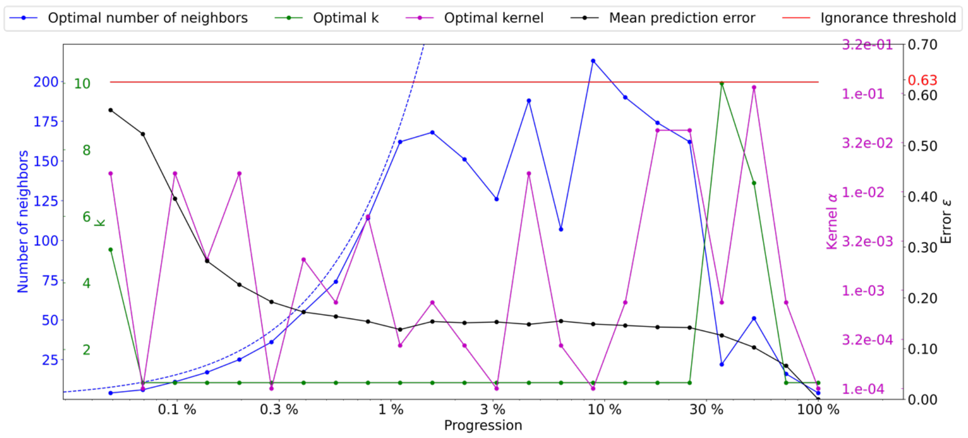


Figure B.2 Optimal parameters for QS (k in green and number of neighbors in blue, and the best kernel in magenta) as a function of the progression, with the associated prediction error (in red). The dashed blue line is the average density for the neighborhood considered.

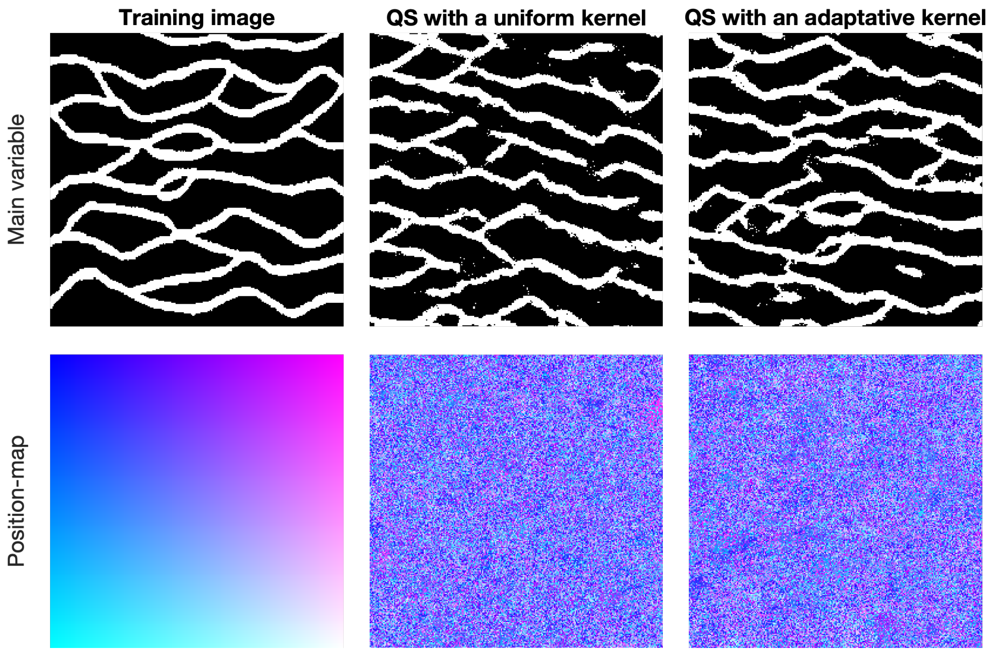


Figure B.3 Simulation using QS using parameters generated by the automatic calibration.

https://doi.org/10.5194/gmd-2022-229
Preprint. Discussion started: 17 October 2022

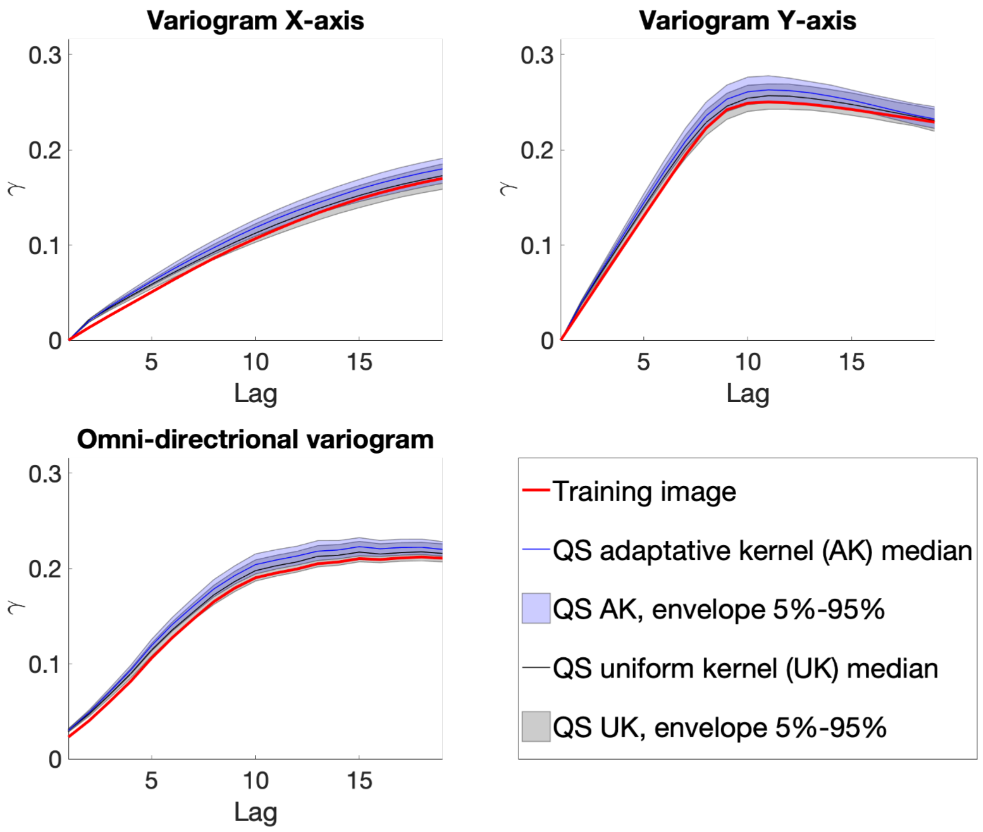


Figure B.4 Benchmark between QS with adaptative kernel (Figure B.2) and uniform (without) kernel (Figure B.1) over 100 simulations for 5 different metrics.

C. Delta Lena

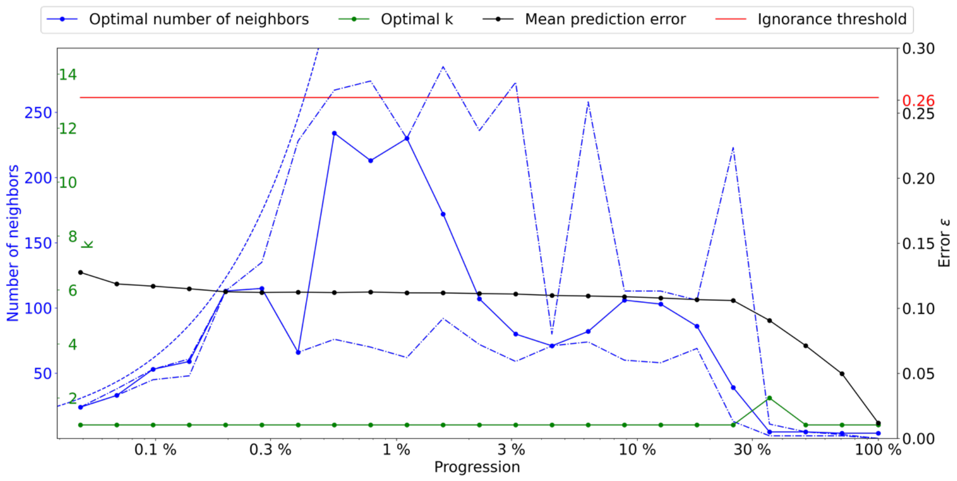


Figure C.1 Optimal parameters for QS (k in green and number of neighbors in blue) as a function of the progression, with the associated prediction error (in red). The red line represents the ignorance threshold. The dashed blue line is the average density for the neighborhood considered. The dot-dashed line represents the variability in 1% of the error.

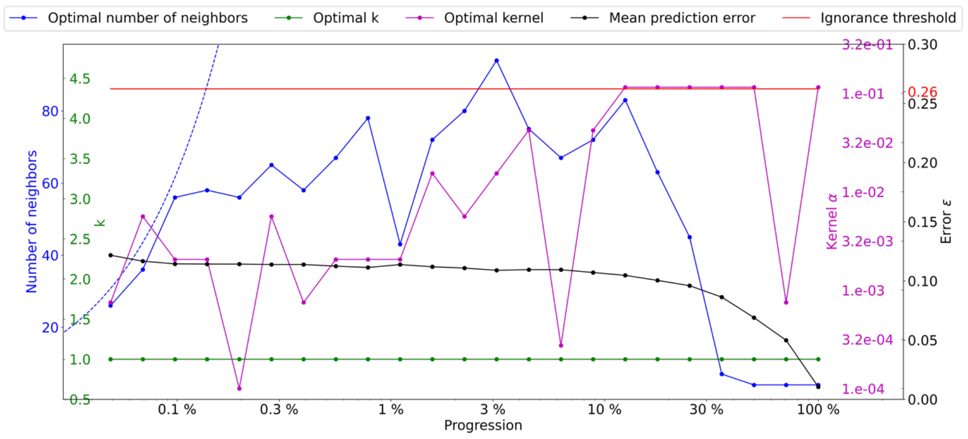
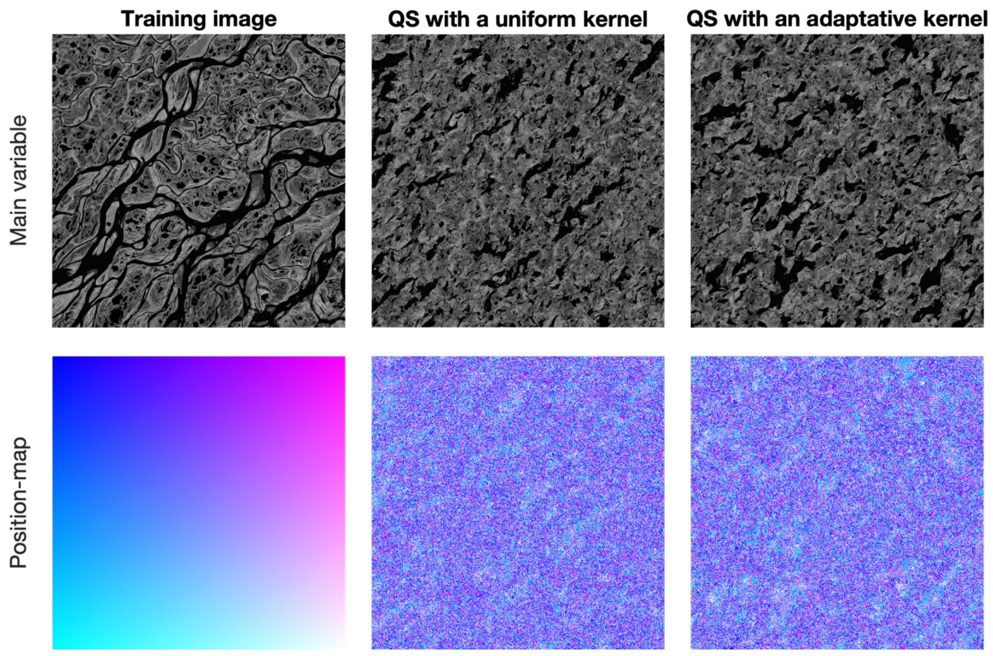


Figure C.2 Optimal parameters for QS (k in green and number of neighbors in blue, and the best kernel in magenta) as a function of the progression, with the associated prediction error (in red). The dashed blue line is the average density for the neighborhood considered.

409 

410 **Figure C.3 Simulation using QS using parameters generated by the automatic calibration.**

https://doi.org/10.5194/gmd-2022-229
Preprint. Discussion started: 17 October 2022

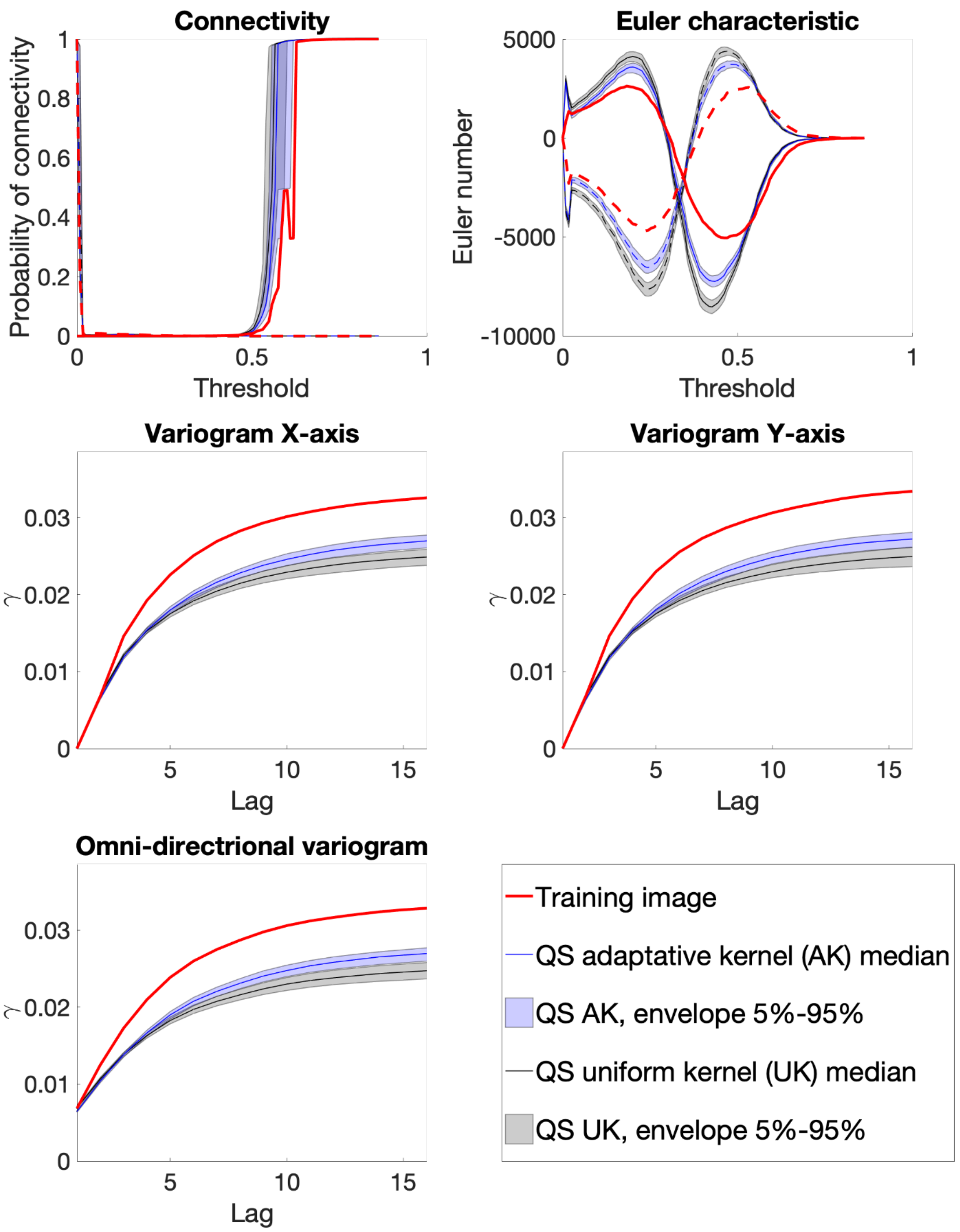


Figure C.4 Benchmark between QS with adaptative kernel (Figure C.2) and uniform (without) kernel (Figure C.1) over 100 simulations for 5 different metrics.