

# INVERSE MODELING OF MOVING AVERAGE KERNELS FOR 3D GAUSSIAN SIMULATION

Oscar Peredo <sup>1</sup> & Julián M. Ortiz <sup>1,2</sup> & Oy Leuangthong <sup>3</sup>

<sup>1</sup> *ALGES Lab, Advanced Mining Technology Center, University of Chile*

<sup>2</sup> *Department of Mining Engineering, University of Chile*

<sup>3</sup> *SRK Consulting (Canada) Inc.*

**Abstract.** Moving average simulation can be summarized as a convolution between a spatial kernel of weights and a white noise random field. The kernel can be calculated once the variogram model is known. We propose an inverse approach to moving average simulation, where the kernel of weights is determined based on the experimental variogram map, thus no explicit variogram modelling is required. The omission of structural modeling in the simulation workflow may be particularly attractive if spatial inference is challenging and/or practitioners lack confidence in this task.

A constrained non-linear optimization problem is formulated in order to solve the inverse problem of discrete kernel weight estimation. The objective value of the inverse problem is the squared euclidean distance between experimental variogram values and the convolution of a stationary random field with Dirac covariance and the simulated kernel. Implementation details and examples are presented to demonstrate the performance and potential extensions of this method.

**Keywords.** Moving average, convolution, Gaussian simulation, Variogram, Optimization

## 1 Introduction

Geostatistical simulation is useful to quantify uncertainty of a variable in space and assess its performance to various processes. These transfer functions may be related to flow, transport, thresholding, etc. and they require processing multiple numerical realizations that discretize the geographical space.

The generation of multiple realizations is cumbersome and computationally expensive, since the spatial structure must be inferred and imposed in the numerical models. Many methods exist for generating the simulations once a structural model (a variogram) has been determined. However, a constant challenge is to simplify this process and minimize the time required for generating the realizations.

To avoid the formality of fitting a structural model, we propose an inverse moving average approach to Gaussian simulation. A constrained non-linear optimization problem is posed that solves the inverse problem of discrete kernel weight estimation. The aim is to minimize the squared euclidean distance between experimental variogram values and the convolution of a stationary random field with Dirac covariance and the simulated kernel. A brief review of the

mathematical and computational aspects of moving average kernel estimation for 3D gaussian simulation is presented. This is followed by the proposed methodology, implementation issues, and small synthetic and real examples to demonstrate the viability of this approach.

## 1.1 Forward Moving Averages

The forward method, discussed by [7] and previously applied in one dimension by [4], can be summarized as a convolution between a spatial kernel of weights and a white noise random field. The kernel can be calculated once the variogram model is known.

As explained by [4] and [7], the equation

$$\mathbf{Y}(\mathbf{x}) = \int_U f(\mathbf{x}, \mathbf{x}') Z(\mathbf{x}') d\mathbf{x}' \quad (1)$$

allows us to obtain correlated random fields  $\mathbf{Y}$  with covariance matrix  $\mathbf{C}$  and mean 0, convoluting a kernel function  $f$  with a stationary random field  $\mathbf{Z}$  with a Dirac covariance measure. In order to obtain the function kernel  $f$  we need to know a priori the value of the covariance matrix  $\mathbf{C}$  according to the convolution relation:

$$\mathbf{C}(\mathbf{x}, \mathbf{x}') = \int_U f(\mathbf{x}, \mathbf{x}'') f(\mathbf{x}', \mathbf{x}'') d\mathbf{x}'' \quad (2)$$

Given a known covariance model  $\mathbf{C}(r)$  (e.g. exponential, gaussian, spherical or others), we can perform the inverse Fourier transform of  $\sqrt{\mathcal{F}(\mathbf{C}(r))}$  (see [7] for details):

$$f(r) = \mathcal{F}^{-1} \left( \sqrt{\mathcal{F}(\mathbf{C}(r))} \right)$$

Our approach is to obtain the weights or values of the kernel function  $f$  without knowing a priori the covariance model  $\mathbf{C}$  of the random field  $\mathbf{Y}$ . In order to do this, we will formulate the problem of estimation of the kernel values at discrete locations in terms of a non-linear optimization problem with linear constraints. In this problem the cost function is the squared euclidean distance between two vectors of variographic data, one calculated using experimental values and the other simulated using the unknown weights of the kernel function. Several lag vectors with different lag distances can be incorporated to the cost function. The constraints will be associated to statistical and geometrical properties that must be fulfilled by the values of  $f$  at user-defined neighbour locations.

## 2 Inverse Moving Averages

According to [4], we can estimate the semi-variogram  $\gamma(h)$  from available data  $\{y(x_i)\}_{i=1}^{N_d}$  located in a regular 3D lattice using the formula:

$$\gamma(h) = \frac{1}{2N_h} \sum_{i=1}^{N_h} [y(x_i) - y(x_i + h)]^2 \quad (3)$$

with  $N_h$  the number of pairs  $\{y(x_i), y(x_i + h)\}$  of data separated by the lag vector  $h$ . Equivalently we can formulate the semi-variogram using the following definition:

$$\gamma(h) = \frac{1}{2N_h} \sum_{\{(p,q):x_q=x_p+h\}} [y(x_p) - y(x_q)]^2 \quad (4)$$

If we assume that the random field  $\mathbf{Y}$  was generated using the formula (1), for some kernel  $f$  and some stationary random field  $\mathbf{Z}$  with Dirac covariance measure, we have:

$$\begin{aligned} \gamma(h) &= \frac{1}{2N_h} \sum_{\{(p,q):x_q=x_p+h\}} [y(x_p) - y(x_q)]^2 \\ &\approx \frac{1}{2N_h} \sum_{\{(p,q):x_q=x_p+h\}} \left[ \sum_{i=-D}^D \sum_{j=-D}^D \sum_{k=-D}^D w_{i,j,k} (z_{p(i,j,k)} - z_{q(i,j,k)}) \right]^2 \end{aligned} \quad (5)$$

with  $D$  the *radius* of the kernel's neighbourhood (a cube of side  $2D + 1$ ) and the vectors

$$z_{p(i,j,k)} =: z(x_p + ib\hat{i} + jb\hat{j} + kb\hat{k}) \quad (6)$$

$$w_{i,j,k} =: f(ib\hat{i} + jb\hat{j} + kb\hat{k}) \quad (7)$$

with  $b$  the step length of the underlying lattice,  $i, j, k \in \{-D, \dots, D\}$ ,  $(\hat{i}, \hat{j}, \hat{k})$  the canonical vectors in each dimension and  $p(i, j, k), q(i, j, k) \in \{1, \dots, N_h\}$  indexes of the corresponding pair. If we traverse the indexes  $(i, j, k) \in \{(-D, -D, -D), \dots, (D, D, D)\}$  using a one-dimensional index  $r \in \{1, \dots, \underbrace{(2D + 1)^3}_{N_w}\}$ , the expression (5) is equivalent to:

$$\begin{aligned} \gamma(h) &\approx \frac{1}{2N_h} \sum_{\{(p,q):x_q=x_p+h\}} \left[ \sum_{r=1}^{N_w} w_r \underbrace{(z_{p(r)} - z_{q(r)})}_{\delta(p,q)_r} \right]^2 \\ &= \frac{1}{2N_h} \sum_{\{(p,q):x_q=x_p+h\}} (\boldsymbol{\delta}(p, q)^T \mathbf{w})^2 \\ &= \mathbf{w}^T \boldsymbol{\Delta}(h)^T \boldsymbol{\Delta}(h) \mathbf{w} \end{aligned} \quad (8)$$

with  $\boldsymbol{\Delta}(h)^T = \frac{1}{\sqrt{2N_h}} \left[ \boldsymbol{\delta}(p_1, q_1) \mid \dots \mid \boldsymbol{\delta}(p_{N_h}, q_{N_h}) \right] \in \mathbb{R}^{N_w \times N_h}$  (dense real matrix). The equation (8) says that the semi-variogram values for each lag distance  $h_i$  can be modeled as a quadratic function of the kernel weights  $\mathbf{w}$ , using the random field  $\mathbf{Z}$  to build the matrix  $\boldsymbol{\Delta}(h)$ . We will try to find optimal weights  $\mathbf{w}$  such that this quadratic function matches experimental semi-variogram values obtained without previous knowledge of the structural model of  $\mathbf{Y}$ .

## 2.1 A numerical solution

Using equation (8) we can define a least-squares type of cost function, measuring the squared euclidean distance between a vector of experimental semi-variogram values at different lag distances with different lag vectors  $h \in \{d_1, \dots, d_n\}$  and the simulated semi-variogram depending on the weight values  $\mathbf{w}$ :

$$\text{minimize}_{\mathbf{w} \in \mathbb{R}^{N_w}} \sum_{i=1}^n (\|\Delta(d_i)\mathbf{w}\|^2 - \gamma^{target}(d_i))^2 \quad (9)$$

Several constraints can be added to the optimization problem in order to reduce the size of the search space and get more accurate results. In this work we have considered the following constraints:

1. *Upper and lower bounds*:  $w_i \in [0, B]$  for some constant  $B > 0$ .
2. *Radial symmetry and isotropy*: we are assuming that the kernel function  $f$  has radial dependency and its values are isotropic. With this assumption we can reduce the number of variables  $N_w = (2D+1)^3$  by imposing  $w_p = w_q$  if their corresponding coordinates are equidistant to the origin. The reduction is approximately  $N_w \approx \lambda(2D+1)^3$  with  $\lambda = \frac{1}{8}$  and  $\lambda = \frac{1}{32}$  in 2D and 3D cases respectively.

The non-linear optimization problem that must be solved is as follows:

$$\begin{aligned} &\text{minimize}_{\mathbf{w} \in \mathbb{R}^{N_w}} \sum_{i=1}^n (\|\Delta(d_i)\mathbf{w}\|^2 - \gamma^{target}(d_i))^2 \\ &\text{subject to} \quad \begin{aligned} w_i &\leq B, \quad \forall i \\ w_i &\geq 0, \quad \forall i \\ w_i &= w_j, \quad i, j \text{ are radially symmetric} \end{aligned} \end{aligned} \quad (10)$$

## 3 Implementation

Simulated annealing ([5]) will be used to solve problem (10). The calculation of the current simulated semi-variogram values  $\{\|\Delta(d_i)\mathbf{w}\|^2\}_{i=1}^n$  is done using the routines `gam` or `gamv` (depending on the spatial regularity of the experimental data) from the GSLIB package ([2]) with the same parameter file used to obtain the target experimental semi-variogram  $\{\gamma^{target}(d_i)\}_{i=1}^n$ . Optionally, the user can standardize the experimental or simulated data using GSLIB's `nscore` routine before executing the semi-variogram calculation.

Algorithm 1 summarizes the methodology. Some important aspects of this proposed algorithm include:

**Input:**

- Initial kernel weights  $\mathbf{w}^0 := f^0(r)$  and neighbourhood radius  $D$  as in equation (7)
- Sample data base values  $\mathbf{V}$  defined in a domain  $\Omega$  (irregular)
- GSLIB's gam/gamv parameter file `params.dat`
- Simulated annealing parameters: initial temperature  $T_0$ , cooling scheme  $\lambda$ , number of iterations  $N_{iters}$ , convergence tolerance  $\epsilon_{tol}$

- 1:  $\{\gamma^{target}(d_i)\}_{i=1}^n \leftarrow$  Execute gam/gamv over  $(\mathbf{V}, \Omega)$  using `params.dat`. Optional: apply `nscore` over  $(\mathbf{V}, \Omega)$  before executing gam/gamv.
- 2:  $\Omega_h \leftarrow$  Generate a regular lattice  $\Omega_h$ , with similar dimensions of  $\Omega$
- 3:  $\mathbf{Z} \leftarrow$  Generate random field with Dirac covariance defined in  $\Omega_h$  (random 1s and 0s)
- 4:  $\mathbf{Y}^0 \leftarrow$  Convolute  $\mathbf{w}^0 := f^0(r)$  and  $\mathbf{Z}$  in domain  $\Omega_h$  using formula (1)
- 5:  $\{\|\Delta(d_i)\mathbf{w}^0\|^2\}_{i=1}^n \leftarrow$  Execute gam/gamv over  $(\mathbf{Y}^0, \Omega_h)$  using `params.dat` Optional: apply `nscore` over  $(\mathbf{Y}^0, \Omega_h)$  before executing gam/gamv.
- 6:  $cost^0 \leftarrow$  Calculate cost function of formula (9)
- 7:  $k \leftarrow 0, T \leftarrow T_0$
- 8: **while**  $\frac{cost^k}{cost^0} < \epsilon_{tol}$  **or**  $k > N_{iters}$  **do**
- 9:  $w_r^k \leftarrow$  Select random kernel weight from  $\mathbf{w}^k$
- 10:  $\tilde{\mathbf{w}}_r^k \leftarrow$  Modify  $w_r^k$  into  $\tilde{w}_r^k$
- 11: Force feasibility of  $\tilde{\mathbf{w}}^k$  using constraints of problem (10)
- 12:  $\tilde{\mathbf{Y}}^k \leftarrow$  Convolute  $\tilde{\mathbf{w}}^k := \tilde{f}^k(r)$  and  $\mathbf{Z}$  in domain  $\Omega_h$  using formula (1)
- 13:  $\{\|\Delta(d_i)\tilde{\mathbf{w}}^k\|^2\}_{i=1}^n \leftarrow$  Execute gam/gamv over  $(\tilde{\mathbf{Y}}^k, \Omega_h)$  using `params.dat` Optional: apply `nscore` over  $(\tilde{\mathbf{Y}}^k, \Omega_h)$  before executing gam/gamv.
- 14:  $cost^k \leftarrow$  Calculate cost function of formula (9) using  $\{\|\Delta(d_i)\tilde{\mathbf{w}}^k\|^2\}_{i=1}^n$
- 15:  $\delta \leftarrow \begin{cases} 1 & , \text{ if } \widetilde{cost^k} < cost^k \\ e^{\frac{cost^k - \widetilde{cost^k}}{T \cdot cost^k}} & , \text{ otherwise} \end{cases}$
- 16:  $(\mathbf{w}^{k+1}, cost^{k+1}) \leftarrow \begin{cases} (\tilde{\mathbf{w}}^k, \widetilde{cost^k}) & , \text{ if } \text{rand}(0,1) < \delta \\ (\mathbf{w}^k, cost^k) & , \text{ otherwise} \end{cases}$
- 17:  $k \leftarrow k + 1$
- 18: Apply cooling scheme  $\lambda$  to the temperature  $T$
- 19: **end while**

**Output:** Optimal kernel weights  $\mathbf{w}^*$  and best convoluted image  $\mathbf{Y}^*$  (just used for visualization purposes)

**Algorithm 1:** Simulated annealing to solve problem (10) using GSLIB procedures `gam/gamv` to calculate semi-variogram values.

- The regular lattice  $\Omega_h$  can be defined in any way with the only requirement that it contains enough data to reproduce the semi-variogram using all lags defined in the parameter file `params.dat`.
- Feasibility enforcement consists in verifying if  $\tilde{w}_r^k \in [0, B]$  and modify all radial-symmetric weights in the same way as  $\tilde{w}_r^k$ .
- The probability of acceptance  $\delta$  is given by the Boltzmann distribution, as described in several sources ([1, 8]). The terminology of simulated annealing resembles the one used in the physical process of annealing. The idea is to decrease the temperature  $T$  as the process algorithm runs, emulating the cooling that occurs during the physical process that lets the molecules reorganize to lower energy states.
- The last step is the update of the temperature  $T$  applying a cooling scheme  $\lambda$ . This scheme can be implemented in many ways ([6]). The easiest way is to update  $T \leftarrow \lambda T$  once every  $m$  iterations. Theoretical properties that must fulfill the cooling scheme can be studied in [3].

The target semi-variogram values of step 1, denoted  $\{\gamma^{target}(d_i)\}_{i=1}^n$ , must be calculated by the user before the program's execution, using GSLIB's `gamv` if the data is not structured, or `gam` if it is structured in a regular lattice. The current implementation supports 2D and

3D images, using the same code without special treatment depending on image dimensions. However, 3D simulations require several minutes/hours to converge making it impractical to obtain fast simulations in tight time schedules (so called *dimensionality curse*). Future versions of the application will use last-generation technologies to accelerate the elapsed time of 3D simulations.

## 4 Examples

Two sets of examples are presented: (1) three synthetic datasets, and (2) one based on real data. The synthetic examples are designed to show extreme scenarios where the initial kernel weights are far from the target kernel weights.

In all cases, the initial and final/best images are presented. The convergence of the algorithm is also provided via the minimization of the cost function over the 6000 iterations. Comparisons between the initial, final and targeted semi-variogram and the kernel functions are also shown.

### 4.1 Synthetic data examples

In the first two examples the semi-variogram of the convoluted field  $\mathbf{Y}$  is calculated over 2D images with dimension  $256 \times 256 \times 1$ , where each node is located in a regular lattice of step length 1.0. The semi-variogram is calculated with `gam` using the direction  $h = (1.0, 0.0, 0.0)$  with 20 lags of separation 1.0. The random field  $\mathbf{Z}$  has dimension  $(256 + D) \times (256 + D) \times D$ , keeping a buffer zone of nodes in the boundaries. The kernel radius is  $D = 9$ , reaching a neighbourhood cube of  $19 \times 19 \times 19$  centered in the node being simulated. The third synthetic example is similar constructed; however, the global dimensions are reduced by half to  $128 \times 128 \times 1$ .

The first example (Fig. 1) illustrates convergence of the algorithm when it is initialized with a very smooth semi-variogram while the targeted semi-variogram is less continuous. Specifically, we begin with Gaussian values for each kernel weight:

$$f(r) = \sqrt{\frac{4}{a^2\pi}} e^{\frac{-2r^2}{a^2}}, \quad a = 9.0 \quad (11)$$

with the target specified as 2D spherical kernel weights:

$$f(r) = \begin{cases} \frac{2}{a\sqrt{\pi}} & \text{for } r \leq \frac{a}{2} \\ 0 & \text{for } r > \frac{a}{2} \end{cases}, \quad a = 9.0 \quad (12)$$

In contrast, the second example (Fig. 2) begins with short range continuity (see equation (12)), while the targeted semi-variogram is much more continuous. An exponential kernel weight becomes our target (3D expression according to [7], displacing the origin to  $r_0 = -1$ ):

$$f(r) = \frac{1}{(r+1)\sqrt{2\pi a}} e^{\frac{-(r+1)}{a}}, \quad a = 9.0 \quad (13)$$

In both instances, the final image matches the target image remarkably well (after re-scaling). This is further confirmed when the final and target semi-variograms and kernel weights functions are compared. A possible way to reduce the scale difference could be to introduce conditioning data in the simulations.

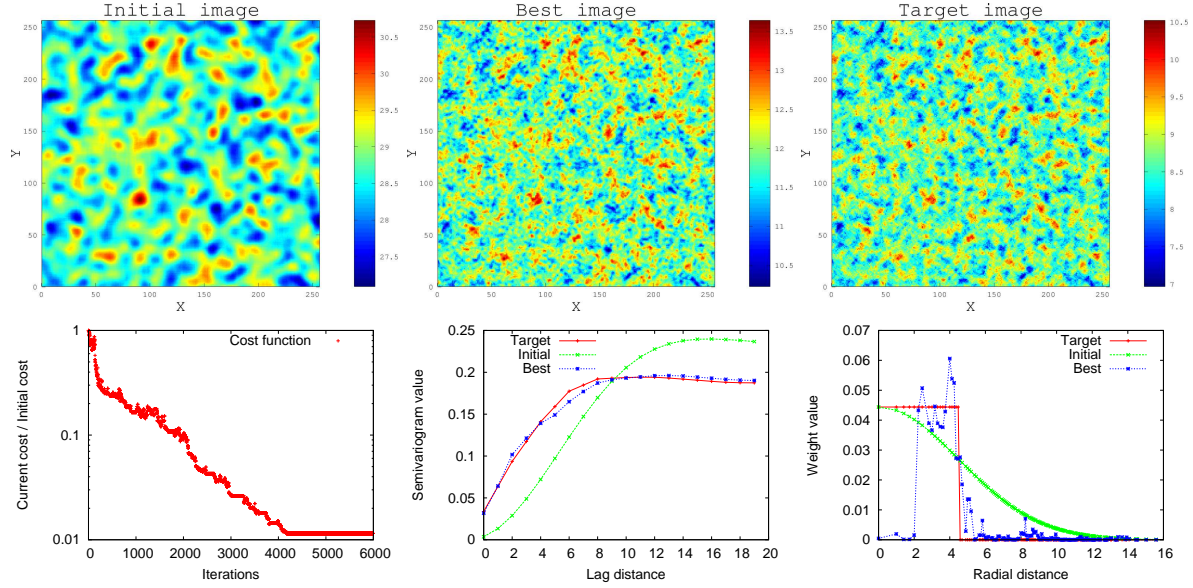


Figure 1: Example 1, synthetic: initial (top-left), best (top-center) and target (top-right) images; cost function (bottom-left), semi-variograms (bottom-center) and kernel weights (bottom-right).

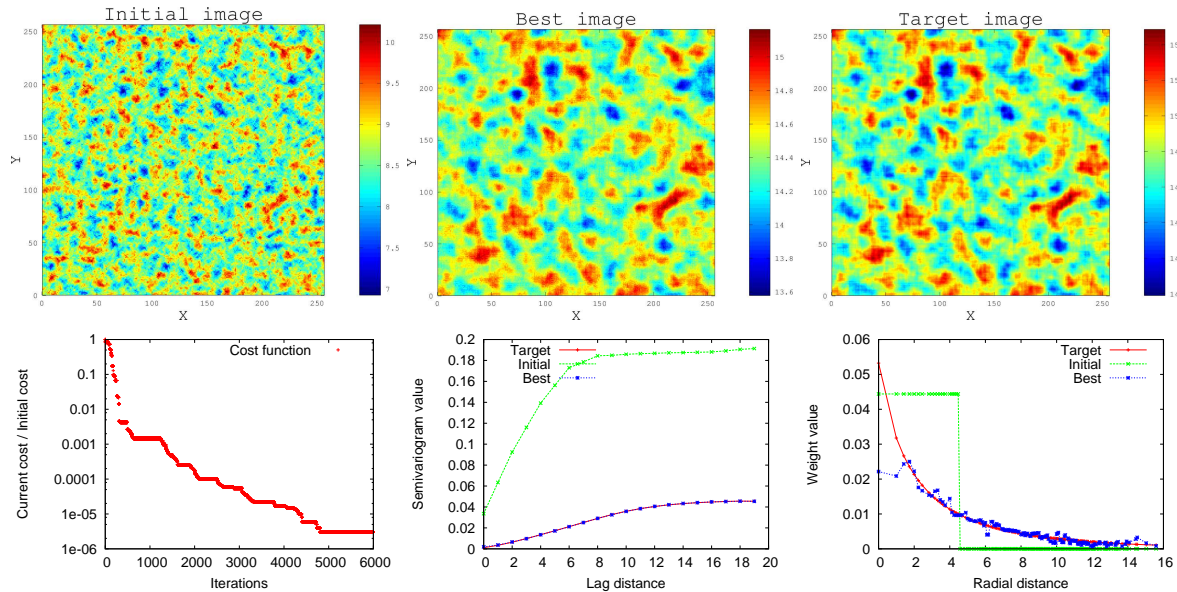


Figure 2: Example 2, synthetic: initial (top-left), best (top-center) and target (top-right) images; cost function (bottom-left), semi-variograms (bottom-center) and kernel weights (bottom-right).

A third example (Fig. 3) shows the effect of computing the normal scores before calculating the target and simulated semi-variograms. The final kernel weights do not match exactly with the target weights, yet semi-variograms match almost perfectly (final cost function value is less than 0.5%). The reason of this mismatch lies in the convex (quadratic) relation between the kernel weights and the semi-variogram. Standardization of the data using `nscore` breaks this convex relation (cubic polynomial of square root of a logarithmic function applied to the convoluted data, according to `gauinv` routine from GSLIB), generating several local optimal kernel weights that can produce images with semi-variogram curves similar to the target semi-variogram.

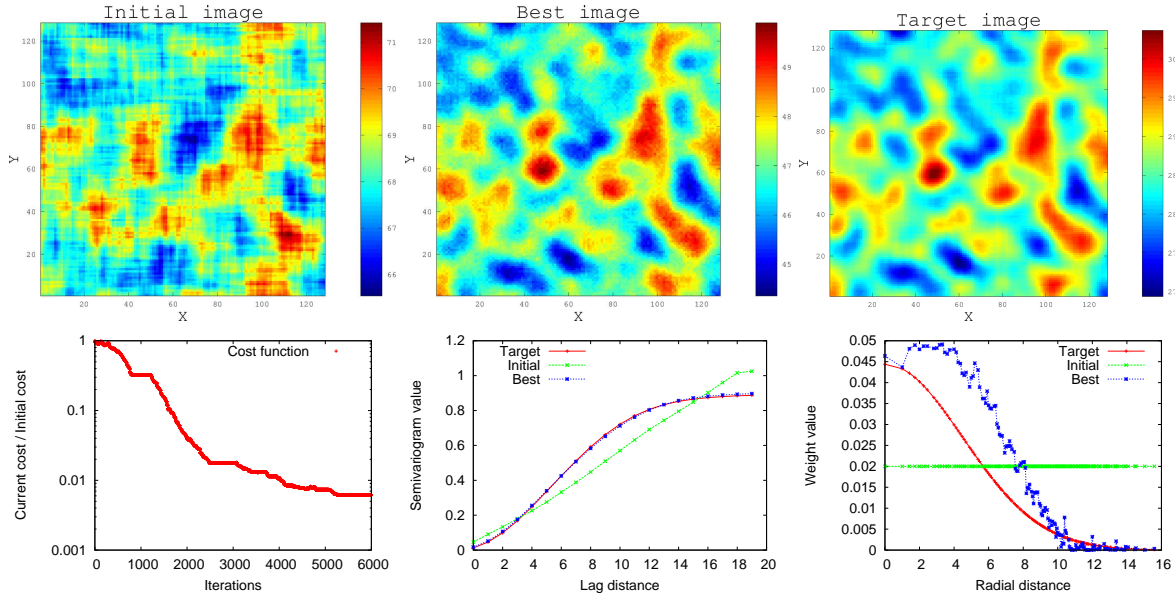


Figure 3: Example 3, Synthetic data: initial (top-left), best (top-center) and target (top-right) images; cost function (bottom-left), semivariograms (bottom-center) and kernel weights (bottom-right).

## 4.2 Real data example

A 3D dataset of copper grades (2376 composites) is used as real data example. The parameters used by `gamv` to calculate the target semivariograms can be viewed in table 1. The example uses a convoluted field  $Y$  calculated over a 2D image of size  $50 \times 50 \times 1$  using step length of  $h = 3.0$  and kernel radius is  $D = 14$  (see Fig. 4).

Unlike the synthetic examples, no reference or target image is available. However, the final image is noticeably closer in resemblance to a 'typical' simulated realization, without the banding that is apparent in the initial image. Clearly, the small size of the image contributes to this 'pixelation' effect that is not apparent in the synthetic examples.



	Example (omni-horizontal direction)
number of lags	7
lag separation distance	20.0
lag tolerance	10.0
number of directions	1
azm,atol,bandh,dip,dtol,bandv	0.0,90.0,20.0,0.0,20.0,5.0
standardize sills? (0=no, 1=yes)	0
number of variograms	1
tail var., head var., variogram type	1,1,1

Table 1: gamv parameters for real data example

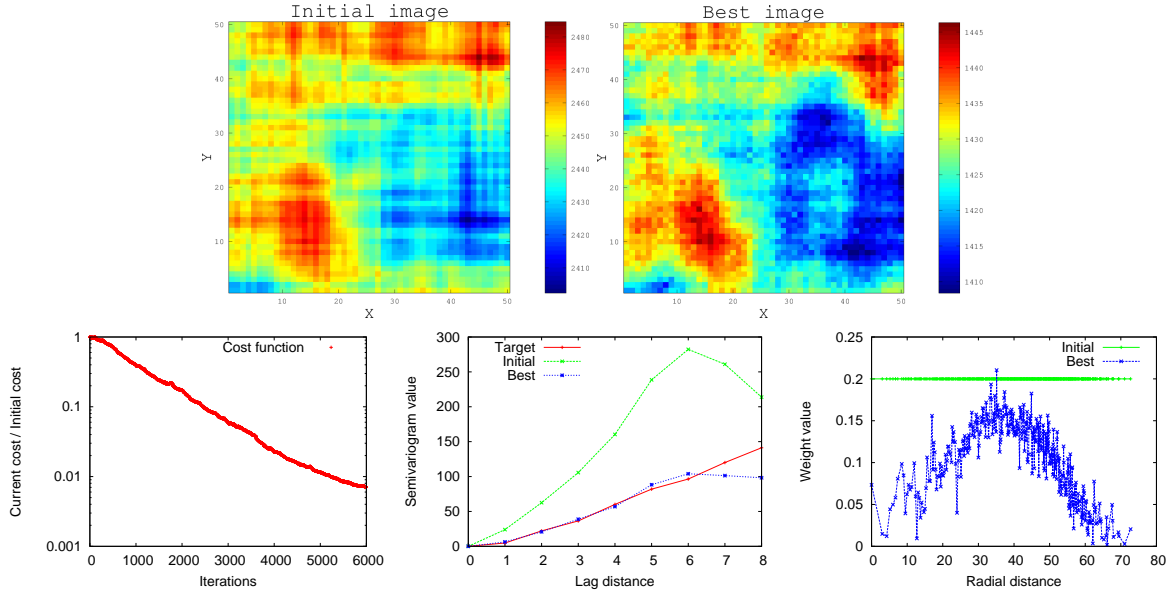


Figure 4: Real data example: initial (top-left), best (top-right) images; cost function (bottom-left), semivariograms (bottom-center) and kernel weights (bottom-right).

## 5 Conclusions

We have shown a novel approach to Gaussian simulation that does not require explicit variogram modelling by the practitioner. The method is based on an inverse moving average simulation, where kernel estimations from experimental semivariogram values is solved via a constrained optimization problem. Simulated annealing is used to solve the optimization problem.

Early application of this methodology to synthetic and real data examples show reasonable convergence and kernel weight values. These results are encouraging; however, much remains to be done in the development of this method. Future tasks include:

1. Consideration of anisotropy in the kernel weights, which may be modeled by relaxing the third constraint of problem (10), allowing each weight  $w_i$  to fit different values without radial symmetric dependency. This relaxation imposes larger execution times and harder convergence conditions, because the number of variables in the optimization problem, namely  $N_w$ , increases 8x or 32x times, depending on the dimension of the images.

2. Conditioning of simulated results should be straightforward and could be incorporated into the convoluted image without significant effort. The main effect of conditioning in the proposed algorithm could be a small increment of the total elapsed time, which can be mitigated improving the current implementation.
3. Increase computational efficiency to make this approach more practical. System calls are launched inside of ANSI C code in order to execute GSLIB routines. This configuration is intended to work as a preliminary version, because several performance bottlenecks arise after profiling our current implementation. Efficient integration of C and Fortran codes and acceleration using multi-core programming models or graphical processing units is left as a future work.

## References

- [1] Deutsch, C. V. (1992). *Annealing Techniques Applied to Reservoir Modeling and the Integration of Geological and Engineering (Well Test) Data*. PhD thesis, Stanford University, Stanford, CA.
- [2] Deutsch, C. V. and Journel, A. G. (1998). *GSLIB: Geostatistical Software Library and User's Guide*. Oxford University Press.
- [3] Hajek, B. (1988). Cooling schedules for optimal annealing. *Mathematics of Operations Research*, 13:311–329.
- [4] Journel, A. G. and Huijbregts, C. J. (1978). *Mining Geostatistics*. Academic Press, London.
- [5] Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220(4598):671–680.
- [6] Laarhoven, P. J. M. and Aarts, E. H. L., editors (1987). *Simulated annealing: theory and applications*. Kluwer Academic Publishers, Norwell, MA, USA.
- [7] Oliver, D. S. (1995). Moving averages for gaussian simulation in two and three dimensions. *Mathematical Geology*, 27(8):939–960.
- [8] Peredo, O. and Ortiz, J. M. (2011). Parallel implementation of simulated annealing to reproduce multiple-point statistics. *Computers & Geosciences*, 37(8):1110 – 1121.