



# Fast Gap-Filling of Massive Data by Local-Equilibrium Conditional Simulations on GPU

M. Lach<sup>1</sup> · M. Žukovič<sup>1</sup> 

Received: 2 December 2022 / Accepted: 28 July 2023 / Published online: 1 September 2023  
© The Author(s) 2023

## Abstract

The ever-growing size of modern space–time data sets, such as those collected by remote sensing, requires new techniques for their efficient and automated processing, including gap-filling of missing values. Compute Unified Device Architecture-based parallelization on graphics processing units (GPUs) has become a popular way to dramatically increase the computational efficiency of various approaches. Recently, a computationally efficient and competitive yet simple spatial prediction approach inspired by statistical physics models, called the modified planar rotator method, was proposed. Its GPU implementation allowed additional impressive computational acceleration exceeding two orders of magnitude in comparison with central processing unit calculations. In the current study, a rather general approach to modeling spatial heterogeneity in GPU-implemented spatial prediction methods for two-dimensional gridded data is proposed by introducing spatial variability to model parameters. Predictions of unknown values are obtained from non-equilibrium conditional simulations, assuming “local” equilibrium conditions. It is demonstrated that the proposed method leads to significant improvements in both prediction performance and computational efficiency.

**Keywords** Spatial interpolation · Local-equilibrium simulation · Non-Gaussian model · Heterogeneous data · GPU parallel computing · CUDA

## 1 Introduction

With the emergence and increasing frequency of massive spatiotemporal data sets, such as those collected by remote sensing technologies, scalable numerical techniques are required for their efficient processing. For example, such data often include gaps

---

✉ M. Žukovič  
milan.zukovic@upjs.sk

<sup>1</sup> Department of Theoretical Physics and Astrophysics, Institute of Physics, Faculty of Science, Pavol Jozef Šafárik University in Košice, Park Angelinum 9, 041 54 Košice, Slovakia

that may occur as a result of sensor malfunctions, cloud, vegetation or snow coverage, dense precipitation or other barriers separating the sensed object and the remote sensing device (Lehman et al. 2004; Coleman et al. 2011; Sun et al. 2017; Kadlec and Ames 2017) and may have an unfavorable effect on the statistical assessment of mean values and trends. Therefore, they need to be estimated to generate gapless maps of observed variables and to facilitate prompt and informed decisions (Sickles and Shadwick 2007). Most traditional interpolation methods, however, such as kriging (Wackernagel 2003) are not directly applicable to such massive data due to their computational demands. Consequently, several modifications of kriging-based methods have been developed (Furrer et al. 2006; Cressie and Johannesson 2018; Hartman and Hössjer 2008; Kaufman et al. 2008; Ingram et al. 2008; Marcotte and Allard 2018) in an effort to increase their computational efficiency.

Very recently, a geostatistics-informed machine learning model was proposed by Bai and Tahmasebi (2021) to improve the computational performance of ordinary kriging, and an inverse distance weighted regression method by Emmendorfer and Dimuro (2021) to improve the performance of the standard inverse distance weighted (IDW) method (Shepard 1968). Aiming at the same goal, a statistical physics-inspired approach that employs models based on Boltzmann–Gibbs exponential joint densities has also been proposed (Hristopulos 2003; Hristopulos and Elogne 2007; Hristopulos 2015; Žukovič and Hristopulos 2009a, b, 2018; Hristopulos et al. 2021). Within this concept, spatial correlations are captured using short-range interactions instead of the experimental variogram used in geostatistical methods, which renders the proposed interpolation methods computationally very efficient. In particular, the recently introduced method that employs the modified planar rotator (MPR) model (Žukovič and Hristopulos 2018), due to its computational efficiency (roughly linear-time computation complexity) and ability to operate autonomously without user input, was shown to be appropriate for the automated and efficient processing of massive gridded data, typical in remote sensing.

Nevertheless, spatial simulation performed sequentially is still computationally costly, particularly in the simulation of huge data sets (Mariethoz 2010; Nunes and Almeida 2010; Peredo et al. 2015). With new developments in hardware architecture and their availability in common PCs, in particular multi-core CPUs and general-purpose graphics processing units (GPU), an increasingly popular way of overcoming computational inefficiency is achieved by parallel implementations. Most standard interpolation methods to date, including kriging and IDW, have been parallelized on high-performance and distributed architectures (Kerry and Hawick 1988; Cheng et al. 2010; Guan et al. 2011; Pesquer et al. 2011; Hu and Shu 2015; Misra et al. 2020; Que et al. 2021) and general-purpose computing GPUs (Xia et al. 2011; Tahmasebi et al. 2012; Cheng 2013; de Ravé et al. 2014; Mei 2014; Stojanovic and Stojanovic 2014; Mei et al. 2017; Marcellino et al. 2017; Zhang et al. 2018). It has been shown that by using parallelization, it is possible to achieve computational acceleration up to almost two orders of magnitude compared to traditional single-CPU implementation.

Parallelization of spin model simulations is achievable due to the short-range nature of interactions between the spin variables. Impressively, up to 1000-fold speedup can be achieved (Weigel 2012) by employing a highly parallel architecture of GPUs. A recent GPU implementation of the spin-model-based MPR method to massive data

sets (Žukovič et al. 2020) led to almost 500-fold computational speedup when compared to single-processor calculations. Thus, using an ordinary personal computer, data sets involving several millions of points can be processed in a fraction of a second.

Most geostatistical methods assume spatial homogeneity/stationarity of data, even though a kriging-based interpolation method for non-homogeneous data has recently been proposed (Lajaunie et al. 2020). However, if one targets large spatial data, in which anisotropy and non-stationarity are common, such an assumption is not justified. It is not reasonable to assume that one set of model parameters can capture scale-dependent relationships between covariates and the outcome variable that vary in space. The most common techniques for modeling such data are geographically weighted regression (GWR) and spatially varying coefficient (SVC) methods (Fotheringham et al. 2003; Gelfand et al. 2003; Finley 2011). Some relatively efficient GWR methods have been developed (Harris et al. 2010; Li et al. 2019), but the scalable linear-time implementation required for the application to big data sets has been proposed only very recently (Murakami et al. 2020). Their parallelization via the Message Passing Interface leads to a further increase in computational efficiency (Li et al. 2019). Another recent approach to modeling large data with a non-stationary covariance structure is based on efficient local likelihood estimation in moving windows to infer spatially varying covariance parameters (Pardo-Igúzquiza et al. 2005; Wiens et al. 2020).

In the present paper, some modifications to the previously introduced GPU-accelerated MPR method are implemented in an effort to enable modeling spatial heterogeneity/non-stationarity, essential for analyzing massive spatial data. The GPU-implemented version can conveniently achieve this by introducing spatial dependence to the MPR model parameter (temperature) by the so-called double-checkerboard decomposition. Then, predictions of unknown values are obtained from non-equilibrium conditional situations, assuming “local” equilibrium conditions corresponding to local temperatures varying in space (MacGillivray et al. 1993).

The rest of the paper is structured as follows: In Sect. 2, an overview of the previously introduced MPR model and its GPU implementations with both spatially uniform and spatially varying parameters is presented; more details are given in Žukovič and Hristopulos (2018) and Žukovič et al. (2020). The statistical and computational performance of the MPR-based models is investigated and compared to the standard approach in Sect. 3. Lastly, in Sect. 4, the findings and present conclusions are summarized.

## 2 MPR Methods

### 2.1 MPR Method with Spatially Uniform Parameter

Let us consider a two-dimensional square grid  $\mathcal{G}$  consisting of  $L \times L$  nodes with partially known values (samples). Let us denote locations of the samples of the spatial process  $Z(\mathbf{s})$  on the grid nodes as  $\mathcal{G}_S = \{\mathbf{s}_n\}_{n=1}^N$ , where  $N < L^2$  and their values as  $\mathbf{Z}_S = (z_1, \dots, z_N)^\top$  (where  $\top$  denotes the matrix transpose). The task is to estimate the missing values  $\hat{\mathbf{Z}}_P = (\hat{z}_1, \dots, \hat{z}_P)^\top$  of the process at the grid nodes  $\mathcal{G}_P = \{\tilde{\mathbf{s}}_p\}_{p=1}^P$ .

Thus, the intersection of the sets  $\mathcal{G}_P$  and  $\mathcal{G}_S$  is empty, and their union represents the full grid  $\mathcal{G}$ .

In the following, the basic idea of the MPR method, recently introduced for efficient and automatic prediction of partially sampled non-Gaussian data on regular grids (Žukovič and Hristopoulos 2018), is briefly outlined. It should be noted that, unlike geostatistical methods, the MPR method makes no restrictive assumptions regarding the probability distribution of the spatial process. Instead, inspired by statistical physics, it assumes that the data represent a spatial random field realization and the correlations are imposed by local (nearest-neighbor) “interactions” between the field variables at the nodes of  $\mathcal{G}$ . In statistical physics, the Boltzmann–Gibbs probability density function of any random field  $X(\mathbf{s})$  governed by an energy functional  $\mathcal{H}[X(\mathbf{s})]$  (also called Hamiltonian) takes the form

$$f_X[X(\mathbf{s})] = \mathcal{Z}^{-1} \exp\{-\mathcal{H}[X(\mathbf{s})]/k_B T\}, \quad (1)$$

where  $\mathcal{Z}$  is the partition function (normalizing constant),  $T$  is the thermodynamic temperature and  $k_B$  is the Boltzmann constant. The Hamiltonian can incorporate various terms resulting from different types of interactions between the field variables. In magnetism, these variables, called spins, represent elementary magnetic moments of atoms localized on a regular lattice (grid). They can be either discrete, for example, binary with the spins only allowed to “point up” or “point down” (Ising model), or continuous, for example, represented by two-dimensional unit vectors that can rotate in a plane (planar rotator model). The Hamiltonian of the latter in its simplest form can be expressed as  $\mathcal{H}[X(\mathbf{s})] = -J \sum_{i,j} S_i \cdot S_j = -J \sum_{i,j} \cos(\phi_i - \phi_j)$ , where  $J > 0$  is the interaction between neighboring spins  $S_i$  and  $S_j$ , with the respective turn angles  $\phi_i$  and  $\phi_j \in [0, 2\pi]$ .

The MPR method employs the modified planar rotator (MPR) spin model in the framework of a Gibbs–Markov random field (GMRF), defined by means of the Boltzmann–Gibbs distribution (Eq. 1). In the first step, the original data are linearly transformed to continuously valued “spin” variables (or spin angle  $\phi$ ) space  $[0, 2\pi]$ , using the transformation

$$\mathbf{Z}_s \mapsto \Phi_s = \frac{2\pi(\mathbf{Z}_s - z_{s,\min})}{(z_{s,\max} - z_{s,\min})}, \quad (2)$$

where  $z_{s,\min}$  and  $z_{s,\max}$  are the minimum and maximum sample values and  $\Phi_s = \{\phi_i\}_{i=1}^N$  and  $\phi_i \in [0, 2\pi]$ , for  $i = 1, \dots, N$ . Then, conditional Monte Carlo (MC) simulations of the MPR Hamiltonian

$$\mathcal{H} = -J \sum_{(i,j)} \cos[q(\phi_i - \phi_j)], \quad (3)$$

where  $J > 0$  is the interaction between neighboring spins and  $q \leq 1/2$  is the modification parameter, are performed at “temperature”  $T$ .

At this point, some comments on the relation between the original planar rotator, used for modeling magnetic materials, and its modified version (MPR), used for spatial



prediction, are in order. In the former, the parameters  $J$ ,  $k_B$  and  $T$  have physical meaning with the values given in physical units. In particular,  $k_B$  is a universal constant,  $J$  is a constant for a given system and the temperature  $T$  is a variable parameter related to a degree of thermal fluctuations (disorder) in the system. Nevertheless, in simulations, these parameters appear as the product  $k_B T/J$ , and thus the constants  $J$  and  $k_B$  can be absorbed in  $T$  (by setting them to 1), and the temperature  $T$  is then measured in dimensionless units. On the other hand, in the MPR model,  $T$  is not real temperature but just a parameter called temperature. Nevertheless, it plays a similar role as in the planar rotator spin model since its increasing (decreasing) value increases (decreases) fluctuations in the simulated data and thus controls the degree of the spatial correlations and smoothness of the resulting realizations. Finally, the role of the modification factor  $q$  in the MPR model is to impose the desirable type of short-range correlations characteristic of geostatistical data.

In the MPR method,  $T$  is estimated by matching the specific energy of the whole grid (including sample and prediction points) with that calculated only from samples. The former is calculated as  $e(T, L) = \langle H \rangle / N_{GP}$ , where  $\langle H \rangle$  is the expectation of the MPR energy over all probable spin states on  $G$ , and  $N_{GP} = 2L(L - 1)$  is the number of nearest-neighbor pairs on the  $L \times L$  grid with open boundary conditions. The value of  $\langle H \rangle$  is determined by running unconditional MC simulations within some temperature range. The latter takes the form

$$e_s = -\frac{1}{N_{SP}} \sum_{i=1}^N \sum_{j \in nn(i)} \cos[q(\phi_i - \phi_j)], \quad (4)$$

where  $j \in nn(i)$  is the sum over the sample (non-missing) nearest neighbors of the site  $s_i$  (i.e.,  $s_j \in \mathcal{G}_S$ ), and  $N_{SP}$  is the total number of the nearest-neighbor sample pairs. Then the value of the parameter  $T$  is taken as the temperature at which the energies  $e$  and  $e_s$  become equal. In thermodynamic equilibrium, the spin realizations follow the GMRF distribution (Eq. 1), controlled by the parameter  $T$ . After reaching thermodynamic equilibrium, the spin values at the prediction locations are back-transformed to the original values, using Eq. (2). Finally, spatial prediction of missing data is based on taking the mean of the respective conditional distribution at the target site given the incomplete measurements. Its high computational efficiency makes the MPR method applicable to massive data, such as remotely sensed images. Additional details about the MPR algorithm can be found in the paper by Žukovič and Hristopulos (2018).

## 2.2 SV-MPR Method with Spatially Varying Parameter

While the MPR model has proved to be competitive for non-Gaussian data, its reliance on the single parameter—the reduced temperature  $T$ —for the whole data set naturally restricts its applicability. In particular, since  $T$  is related to spatial variability, a single parameter value cannot adequately capture the spatial variability of the data showing some heterogeneity, a feature commonly present in massive data. For example, if the studied data set includes domains of almost constant values as well as domains with large spatial fluctuations, no single value of  $T$  can be optimal for both of these regimes.

The sample-specific energy given by Eq. (4) in a subsystem with nearly constant values will be very low, which will result in  $T$  taking values close to 0. Such values would not be representative of a subsystem with high variability and vice versa. The sample-specific energy calculated from all data (including domains with low and high variability) and subsequently estimated temperature thus may not be characteristic for either of the involved domains. As a result, the spatial variability of the predictions in the domains with low (high) variability will be over- (under)estimated. Generally, such a problem can be addressed by introducing a spatial dependence into the model parameters, as is done in the GWR approach (Fotheringham et al. 2003; Gelfand et al. 2003). The current implementation in CUDA offers a very straightforward way to implement this in the MPR method.

### 2.2.1 GPU Implementation in CUDA

NVIDIA's Compute Unified Device Architecture (CUDA) is a general-purpose parallel computing platform and programming model that can be used to accelerate complex computations by harnessing the massively parallel architecture of modern GPUs. CUDA's software environment extends the C programming language by introducing additional syntax, keywords and application programming interface (API) functions to enable the use of the computational resources of the GPU. Its compiler, `nvcc`, splits the code into two parts—host code executed on the CPU (host) and the device code executed on the GPU (device). In a typical CUDA program workflow, the host accesses the input computational data stored in memory (i.e., random access memory, RAM) and copies it to the memory allocated on the device, which then performs computations using device functions called kernels. These kernels utilize the parallel features of GPUs to efficiently solve the problem, and afterwards, the device copies the results back to the host memory. The performance of a CUDA application depends on how efficiently the kernels can utilize the GPU's resources—its computational cores and memory, of which there are multiple types with different properties.

A CUDA kernel is a C function that is executed on many GPU threads in parallel. Threads are virtual processors that can be mapped to hardware resources to perform computations for individual (or multiple) data points or spins. These threads all perform the same instructions written in a single kernel function and are organized into blocks consisting of up to 1,024 threads arranged in one, two or three dimensions (here, square blocks are used). Thread blocks are in turn organized into a grid of up to three dimensions. Block and grid dimensions must be specified at the launch of each kernel. The blocks of threads are scheduled to run on the GPU cores organized into streaming multiprocessors (SMs). A single SM can execute one or more thread blocks. Individual threads are executed in groups called warps. Each warp consists of 32 threads and is scheduled to run on an SM automatically by the warp scheduler. The warp scheduler ensures that as many warps as possible run concurrently by stalling warps waiting for data or memory transactions and scheduling others in their place. If SMs are occupied with a sufficiently large number of warps, this scheduler mechanism can effectively hide the high latency of memory operations, which is crucial for a high-performing CUDA application.

The three main types of GPU memory are global memory, shared memory and registers. Global memory is physically a part of the graphics card but is not a part of the GPU chip itself. The information stored in it can be accessed by any thread summoned by a kernel. It is both the largest and the slowest type of GPU memory. Its size is typically in the tens of gigabytes and it takes about 500 clock cycles to read or write data, compared to one clock cycle for an arithmetic operation. Shared memory is located on the GPU chip and provides low-latency communication between threads that belong to the same block. It is much faster than global memory and much smaller, by default 48 KB per thread block. It is most often used as a cache for intermediate results and computations. Registers facilitate the local memory available to each thread to store its local variables. It is the smallest and fastest memory type available. On newer GPUs (2012 Kepler architecture and newer), threads that belong to the same warp can access each other's memory. It is important to utilize the faster memory types as much as possible and try to maximize the arithmetic intensity—the number of arithmetic operations per memory read/write.

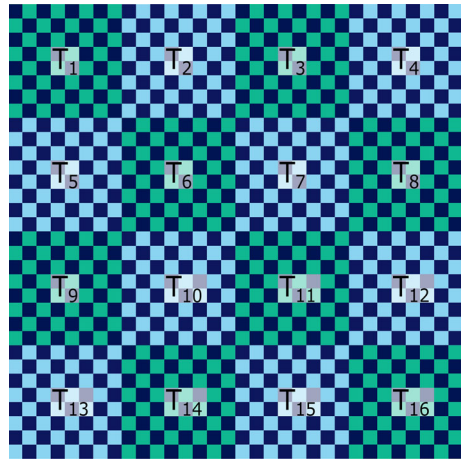
### 2.2.2 Block-Specific Parameter Inference and Simulation

The energy of the system is calculated using the parallel reduction algorithm (Harris 2007). Each thread is mapped to a single spin and computes the energy of a bond with two of its neighbors using the Hamiltonian (3) in such a way that all the bonds present in the system are counted. Then, the results from each thread within a given block are reduced (summed) to a single value representing the energy contribution of that block. Since at this point the energy is known for each block, the same energy matching procedure as described in Sect. 2.1 can be used to assign an individual value of  $T$  to each block. Thus, one obtains multiple values of the simulation parameter  $T$  for different regions of the system. In the extreme case, when a large percentage of the block's data are missing and there are no bonds between samples (no nearest neighbors) for energy calculation and temperature estimation, such a block will be assigned the temperature corresponding to the median value of the available block temperatures.

Since during simulation threads representing individual spins are also grouped into blocks, instead of a single global value of  $T$ , each thread can use the value that corresponds to the block it belongs to. One must ensure that the same configuration of thread blocks used for the energy computation is also used for the Metropolis update; otherwise, spins from one part of a system may end up using a temperature based on a different part of the system with different behavior. This approach is synergized with an optimization technique called double-checkerboard (DC) decomposition (Weigel 2012), which is an extension of the single-checkerboard (SC) decomposition described in our previous paper (Žukovič et al. 2020).

Assuming the data are localized on a two-dimensional square grid (generalization to any regular grid is straightforward), they can be split into two sets sitting on two interpenetrating sub-grids, A and B. The nearest neighbors of any node on sub-grid A belong to sub-grid B, and vice versa. Therefore, the updating algorithm can be applied to all the spins on the same sub-grid in parallel. In Fig. 1, the two sub-grids are depicted using light and dark small squares. Each small square represents a grid node with its

**Fig. 1** Schematic representation of a double-checkerboard decomposition of the spin grid with block-specific temperatures  $T_i$ ,  $i = 1, \dots, 16$



associated spin variable. Each thread performs calculations for a single spin. Sub-grid A comprises the dark nodes, while sub-grid B includes the light ones. The grid is further decomposed into larger dark and light tiles corresponding to individual thread blocks responsible for the numerical operations on the spins included within the tile. Each tile can be simulated using a different temperature computed only from the spins within the tile. The computation kernels call one thread for each sub-grid spin.

The DC decomposition further splits the grid into larger tiles, corresponding to thread blocks. At each MC step, at first, only the even (odd)-numbered tiles are updated in parallel, and then only the odd (even) ones. Each tile can be loaded into the block's shared memory, which is orders of magnitude faster than the GPU's global memory, and one can perform multiple Metropolis updates in quick succession only within the even-numbered blocks, before doing the same for the odd-numbered blocks. As mentioned above, since the temperature is computed individually for each block, one can assign a different value to each tile of the decomposed grid, as depicted in Fig. 1 using light and dark large squares. In the present case, this is the main reason for using the DC decomposition, as the update procedure is not applied multiple times for individual tiles. Nevertheless, because the tiles are first loaded into shared memory, looking up the neighboring spins uses shared instead of global memory, and one still gains a small performance benefit, as will be discussed below.

This version of the algorithm with the DC decomposition will be referred to as the SV-MPR method with block-specific temperatures (BST). It is worth mentioning that the greatest challenge in its implementation is related to the indexing of spins. For each spin, it is necessary to compute the indices of their nearest neighbors and implement boundary conditions for those at the edges. Due to performance considerations, one should aim to coalesce global memory access whenever possible. This means storing the two-dimensional spin grid as a one-dimensional array, which greatly complicates index arithmetic. Furthermore, implementing the double-checkerboard decomposition brings additional complexity and radically changes the Metropolis algorithm implementation. Spins within the block make use of shared memory, and the spins at the boundary of each tile have to be treated specially. This requires careful testing of the

indexing on smaller grids and its graphical visualization, before scaling up to larger grids. It is perhaps the most important challenge, as the correct mapping of threads onto spins is required in almost every implemented CUDA kernel.

### 2.2.3 Site-Specific Parameter Inference and Simulation

The second approach aims to eliminate the undesirable block boundary effect that occurs in the above-described SV-MPR method with BST. It results from the fact that the temperature varies on the grid discontinuously as a step function. Therefore, neighboring blocks, assigned with significantly different temperatures, will be characterized by significantly different spatial variations of the simulated values, which can generate unnatural edges between blocks on the prediction map. Such an effect can be partially reduced by decreasing the size of the blocks. However, this approach has some limitations, since very small blocks may either completely lack samples or their reduced numbers within blocks may lead to imprecise block-specific temperature estimation. An alternative approach allows us to reduce the block size so that each of them contains only one spin and thus to obtain a smooth variation of temperatures on the grid. Assigning a value of  $T$  to each spin individually can be viewed as a limiting case of decreasing the block size  $l_b \rightarrow 1$ . However, instead of actually decreasing the block size, a simple smoothing algorithm is applied to the block temperatures. This can be performed by starting from the BST state and recursively replacing the temperature at each site with an average value of the surrounding area with some radius  $r_s$ . To reach the desired level of smoothness, it can be applied  $n_s$  times in succession. In the present tests, the radius is arbitrarily fixed to  $r_s = 8$ , and the value of  $n_s$  is changed to achieve various degrees of smoothing. This version of the algorithm will be referred to as the SV-MPR method with site-specific temperatures (SST).

The implementations of the MPR algorithm and its BST and SST variants on the GPU are illustrated in Algorithms 1 and 2, as well as the flowchart shown in Fig. 2. The differences between the respective approaches can be summarized as follows. The main difference lies in the definition of the parameters  $a$ ,  $T$  and  $e_s$ . While in the MPR method they are constants, meaning that they are the same for all the prediction points, in the BST implementation they are vectors of length equal to the number of the blocks, that is, they are shared by all the prediction points in a given block, and in the SST implementation, they are vectors of length equal to the number of prediction points, that is, they have generally different values at each prediction point.

## 3 Results

### 3.1 Data

To assess the effect of introducing spatial dependence into the model parameter, both the prediction performance and the computational efficiency of the present SV-MPR model are compared with the original MPR model. The comparison is performed on several large real-world data sets showing a heterogeneous nature of their spatial variability (see Fig. 3) and non-Gaussian (skewed, multimodal, etc.) distributions (see

**Algorithm 1** Parallel restricted Metropolis updating algorithm for the original MPR method and its BST and SST implementations.  $\hat{\Phi}^{\text{old}}$  is the initial spin state, and  $\hat{\Phi}^{\text{new}}$  is the new spin state.  $\hat{\Phi}_{-p}^{\text{old}}$  is the initial spin state excluding the point labeled by  $p$ .  $\mathbf{u}$  denotes a set of random numbers from uniform probability distribution in  $[0, 1]$ . The restriction parameter  $\mathbf{a}$  and temperature  $\hat{\mathbf{T}}$  in the MPR method are constants, that is, the same for all the prediction points; in the BST implementation, vectors of length equal to the number of blocks, that is, shared by the prediction points in a given block; and in the SST implementation, vectors of length equal to the number of prediction points, that is, generally different at each prediction point.

---

```

procedure UPDATE( $\hat{\Phi}^{\text{new}}, \hat{\Phi}^{\text{old}}, \mathbf{u}, \mathbf{a}, \hat{\mathbf{T}}$ )
  for all  $p = 1, \dots, P$  in parallel do ▷ Apply the update procedure to prediction sites in parallel
    1:  $\hat{\Phi}'_p \leftarrow \hat{\Phi}_p^{\text{old}}$  ▷ Generate  $\hat{\Phi}'_p$ 
    if  $\mathbf{a} \neq 1$  then
      1.1:  $\hat{\Phi}'_p \leftarrow \mathcal{R}\{\hat{\Phi}_p^{\text{old}}\}$  ▷ Over-relaxation step applied only in the non-equilibrium regime
    else
      1.2:  $\hat{\Phi}'_p \leftarrow \hat{\Phi}_p^{\text{old}}$  ▷ In the equilibrium regime keep the old state
    end if
    2:  $\hat{\Phi}''_p \leftarrow \hat{\Phi}'_p + 2\pi(u_p - 0.5)/a_p \pmod{2\pi}$  ▷ Propose updated spin values
    3:  $\Delta\mathcal{H} = \mathcal{H}(\hat{\Phi}''_p, \hat{\Phi}_{-p}^{\text{old}}) - \mathcal{H}(\hat{\Phi}'_p, \hat{\Phi}_{-p}^{\text{old}})$  ▷ Evaluate change of energy
    4:  $A P = \min\{1, \exp(-\Delta\mathcal{H}/\hat{\mathbf{T}}_p)\}$  ▷ Evaluate probability of acceptance
    5:  $\hat{\Phi}_{-p}^{\text{new}} \leftarrow \hat{\Phi}_{-p}^{\text{old}}$  ▷ Execute update according to Metropolis rule
    if  $A P > r \leftarrow U(0, 1)$  then
      5.1:  $\hat{\Phi}_p^{\text{new}} \leftarrow \hat{\Phi}''_p$  ▷ Update the old state
    else
      5.2:  $\hat{\Phi}_p^{\text{new}} \leftarrow \hat{\Phi}'_p$  ▷ Keep the old state
    end if
  end for ▷ End of parallel prediction process
  6: return  $\hat{\Phi}^{\text{new}}$  ▷ Return the “updated” new state
end procedure

```

---

Fig. 4). The first one represents the synthetic pollutant concentration data derived from a digital elevation model of the Walker Lake area in Nevada (Isaaks and Srivastava 1989). A map showing a two-dimensional projection of the pollution field is presented in Fig. 3a along with the histogram on a semi-log scale in Fig. 4a. The units used for the  $Z$  values are arbitrarily set to parts per million (ppm). The map shows the presence of both larger dark regions with values close to zero and almost no variability, and brighter regions with very large values and relatively high variability. (Note the adjusted color map, showing all the values  $Z \gtrsim 4,000$  in yellow, to better visualize the extreme data in the tail.) The histogram shows that the distribution is highly positively skewed, indicating the predominance of very low concentrations, with just a small portion of extremely large values.

Further, much larger data sets with linear grid sizes  $L = 2,048$  and  $L = 8,192$ , collected using airborne light detection and ranging (LiDAR) technology, are considered. They represent the canopy height (in meters) of the forests around Kaibab Plateau, Arizona (US Forest Service 2019) (Figs. 3b, 4b), and the digital surface model of the Wasatch Front, Utah (State of Utah 2015) (Figs. 3c, 4c). These data sets include values over extensive spatial domains with nontrivial distributions, as shown in the

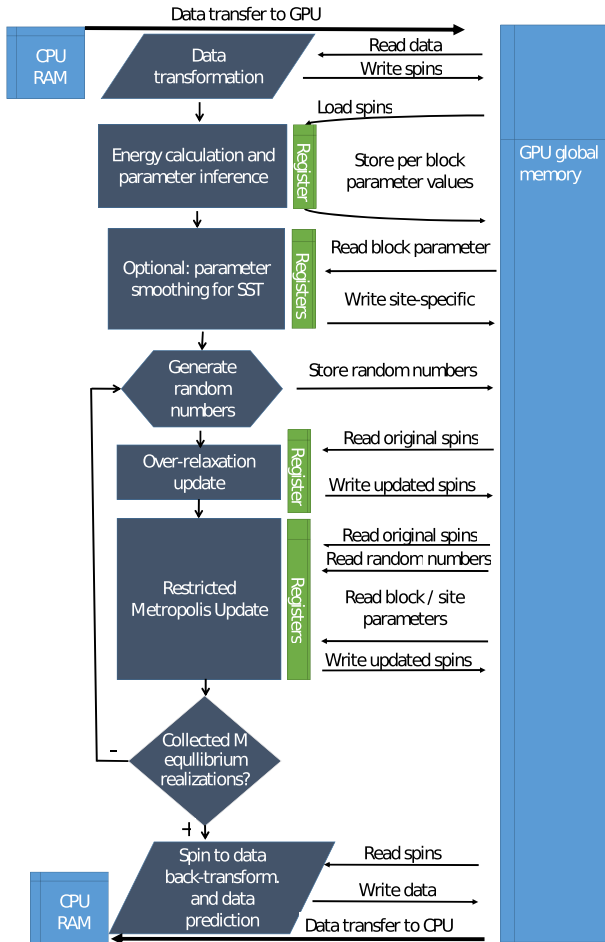


Fig. 2 Flowchart of the main computational steps and memory transactions performed on the GPU

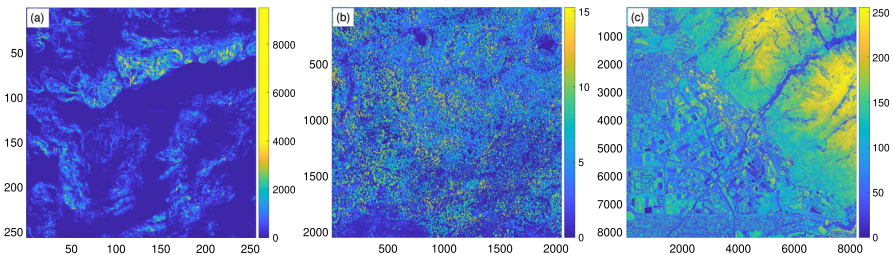


Fig. 3 Spatial distributions of a Walker Lake, b Kaibab Plateau and c Wasatch Front data sets



**Algorithm 2** Simulation of MPR model and its BST and SST implementations. The algorithm includes the parallel restricted Metropolis UPDATE presented in Algorithm 1.  $\Phi_s$  represents the vector of known spin angles at the sampling points.  $\hat{\Phi}$  is the vector of estimated spin angles at the prediction points.  $e_s$  and  $\hat{T}$  represent respectively the vectors of the sample energies and the estimated temperatures with lengths equal to either 1 (MPR), or to the number of blocks (BST) or the number of prediction points (SST).  $G(\cdot)$  represents the transformation from the field of the original data to the spin angle field, and  $G^{-1}(\cdot)$  represents its inverse.  $\hat{Z}(j)$ ,  $j = 1, \dots, M$  is the  $j$ th realization of the original data field.

1: Set simulation parameters with fixed values

1.1: Set the parameters:  $M$  - number of configurations in equilibrium for statistics assessment,  $n_f$  - frequency of equilibrium conditions verification,  $n_{\text{fit}}$  - number of points involved in fitting of energy evolution function,  $A_{\text{targ}}$  - desired acceptance ratio for the Metropolis update,  $k_a$  - specifies variation rate of the vector of perturbation control factors  $\mathbf{a}$ ,  $i_{\text{max}}$  - maximum number of MC steps (optional). The effect of these parameters on prediction performance was found to be marginal, and one can safely use the default values set by Žukovič and Hristopulos (2018).

2: Original data to spin angle transformation

2.1:  $\Phi_s \leftarrow G(\mathbf{Z}_s)$  applying (2) ▷ Set sample spin angles

3: Initialize simulation parameters with varying values

3.1:  $i \leftarrow 0$  ▷ Reset counter of simulated states

3.2:  $\hat{\Phi}(0) \leftarrow \bar{\Phi}_s$  ▷ Initialize missing spin values as per-block averages of sample points

3.3:  $k(0) \leftarrow -1$  ▷ Initialize the energy evolution function slope

3.4:  $\mathbf{a}(0) \leftarrow \mathbf{1}$  ▷ Set control parameters for spin angle perturbation

4: Parameter inference

4.1: Evaluate  $e_s$  using (4) ▷ Calculate sample energy

4.2:  $\hat{T} \leftarrow e^{-1}(e_s|L)$  ▷ Estimate temperature according to  $e(\hat{T}, L) = e_s$

4.3:  $\hat{T} \leftarrow \mathcal{S}\{\hat{T}\}$  ▷ Only for SST: apply the smoothing algorithm to block-specific temperatures to obtain site-specific values

5: Spin relaxation during non-equilibrium period

**while** [ $k(i) < 0$ ]  $\wedge$  [ $i \leq i_{\text{max}}$ ] **do** ▷ Update spins with restricted Metropolis algorithm

5.1:  $\mathbf{u}(i) \in \mathbf{U}(0, 1)$  ▷ Generate a set of uniform random numbers from [0, 1]

5.2: UPDATE( $\hat{\Phi}(i+1)$ ,  $\hat{\Phi}(i)$ ,  $\mathbf{u}(i)$ ,  $\mathbf{a}(i)$ ,  $\hat{T}$ )

**if**  $A_p < A_{\text{targ}}$  **then** ▷ Check the acceptance ratio at each point in parallel

5.3:  $a_p(i+1) \leftarrow 1 + (i+1)/k_a$  ▷ Update perturbation control parameters in parallel

**end if**

5.4: Calculate  $e(i+1) \leftarrow \mathcal{H}/N_{GP}$  ▷ Evaluate current value of specific energy

**if** [ $i \geq n_{\text{fit}}$ ]  $\wedge$  [ $i \equiv 0 \pmod{n_f}$ ] **then** ▷ Check whether slope of  $e$  should be updated

5.5:  $k(i+1) \leftarrow \text{SG}$  ▷ Perform updating of the slope by applying SG method with last  $n_{\text{fit}}$  values

**end if**

5.6:  $i \leftarrow i+1$  ▷ Update counter of MC steps

**end while**

6: Simulation during equilibrium period

6.1:  $\hat{\Phi}^{\text{eq}}(0) \leftarrow \hat{\Phi}(i)$  ▷ Set the initial equilibrium state

**for**  $j = 0, \dots, M-1$  **do**

6.2:  $\mathbf{u}(j) \in \mathbf{U}(0, 1)$  ▷ Generate a set of uniform random numbers from [0, 1]

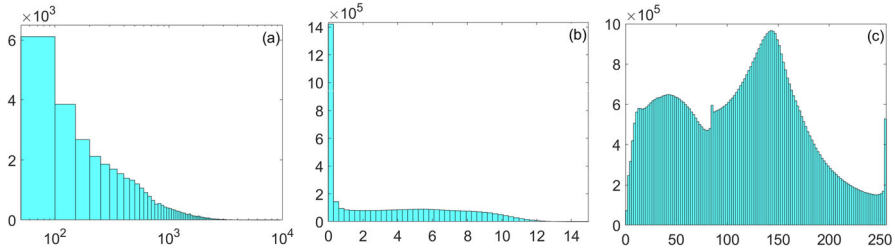
6.3: UPDATE( $\hat{\Phi}^{\text{eq}}(j+1)$ ,  $\hat{\Phi}^{\text{eq}}(j)$ ,  $\mathbf{u}(j)$ ,  $\mathbf{1}$ ,  $\hat{T}$ ) ▷ Produce equilibrium realizations

6.4:  $\hat{Z}(j+1) \leftarrow G^{-1}[\hat{\Phi}^{\text{eq}}(j+1)]$  ▷ Back-transform spin states to original field representation

**end for**

7: **return** Statistics of  $M$  realizations  $\hat{Z}(j)$ ,  $j = 1, \dots, M$

histograms in Fig. 4. The statistical properties of all the data sets are summarized in Table 1.



**Fig. 4** Histograms of **a** Walker Lake, **b** Kaibab Plateau and **c** Wasatch Front data sets

**Table 1** Summary statistics of the data sets used

| Data set       | Number of points              | Range   | $\bar{z}$ | $z_{0.50}$ | $\sigma_z$ | Skewness | Kurtosis |
|----------------|-------------------------------|---------|-----------|------------|------------|----------|----------|
| Walker Lake    | 65,535 ( $L = 256$ )          | 0–9,500 | 289       | 64         | 516        | 3.6      | 23.3     |
| Kaibab Plateau | 4, 194, 304 ( $L = 2, 048$ )  | 0–15    | 3.6       | 2.7        | 3.5        | 0.58     | 2        |
| Wasatch Front  | 67, 108, 864 ( $L = 8, 192$ ) | 0–255   | 112       | 117        | 62         | 0.13     | 2.2      |

### 3.2 Prediction Validation

To evaluate the performance of the MPR-based prediction algorithms, missing values are simulated by setting aside a portion of the complete data to be used as a validation set. Typically,  $M = 100$  different configurations are generated by randomly removing  $p = 30\% - 90\%$  (or  $p = 0.3 - 0.9$ ) of the data points. The MPR predictions are based on the conditional mean as evaluated from the conditional MC simulation. The reconstructions are compared with the true values, first by visually inspecting the reconstructed data and then statistically, using two validation measures: the average absolute error (AAE) defined as

$$AAE = \frac{1}{P} \sum_{\mathbf{r}_p \in G_p} |\epsilon(\mathbf{r}_p)|, \tag{5}$$

and the root average squared error (RASE)

$$RASE = \sqrt{\frac{1}{P} \sum_{\mathbf{r}_p \in G_p} \epsilon^2(\mathbf{r}_p)}, \tag{6}$$

where  $\epsilon(\mathbf{r}_p) = Z(\mathbf{r}_p) - \hat{Z}(\mathbf{r}_p)$  is the difference between the true value  $Z(\mathbf{r}_p)$  and the predicted value  $\hat{Z}(\mathbf{r}_p)$  at the site  $\mathbf{r}_p$ , and  $P = pL^2$  is the number of prediction sites. Both of these quantities are then averaged over the  $M = 100$  different sample configurations to calculate the mean AAE (MAAE) and mean RASE (MRASE). CUDA-based calculations are executed on a PC with NVIDIA GeForce RTX 2080 SUPER GPU, using CUDA version 10. The CPU host system is equipped with an

8-core 3 GHz Intel(R) Core(TM) i7-9700F CPU with 32 GB RAM, running Ubuntu 20.04.3 long-term support (LTS) Linux.

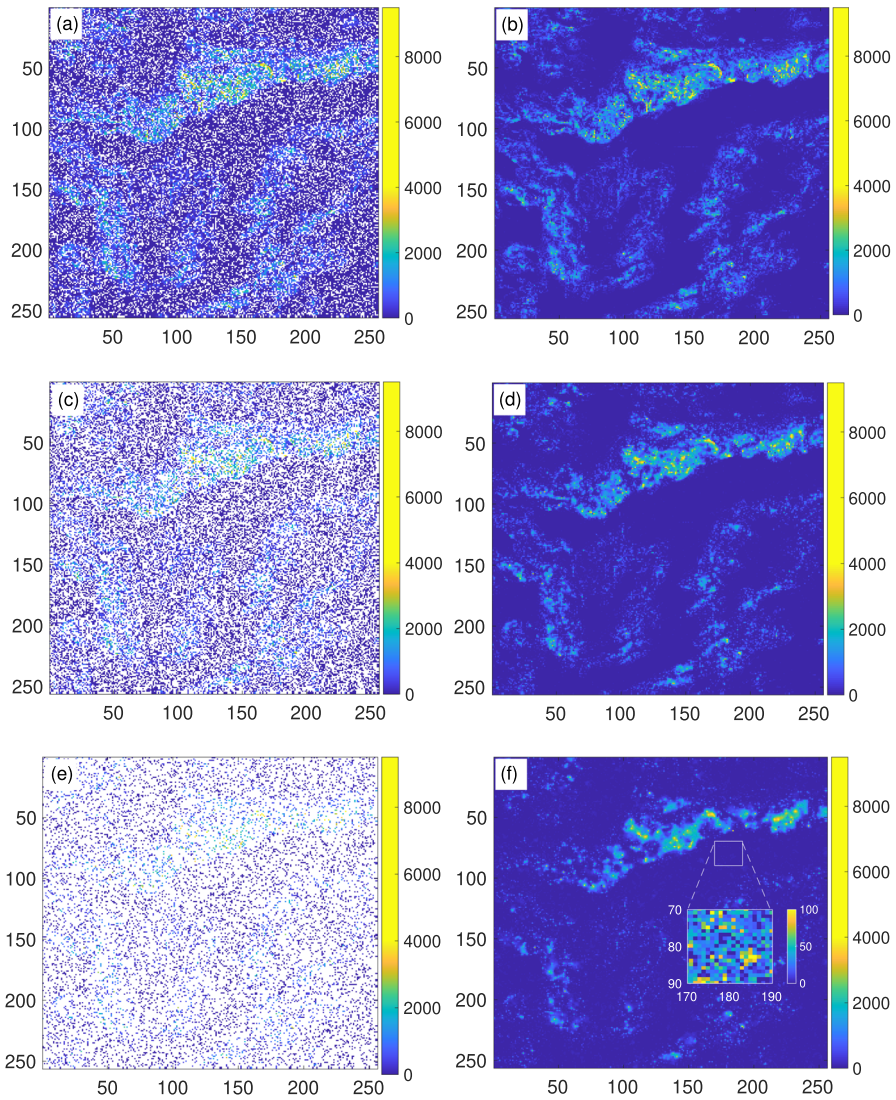
### 3.3 Visual Inspection of Reconstructions

#### 3.3.1 Standard MPR Method

Let us first demonstrate the performance of the original MPR algorithm with a single (spatially uniform) parameter  $T$  and its weak point, particularly when applied to the data with highly heterogeneous spatial variability, such as the Walker Lake data set. The gappy data along with their reconstructions using the MPR algorithm are depicted in Fig. 5 for three values of the thinning ratio  $p = 30\%$ ,  $60\%$  and  $85\%$ . The MPR algorithm performs fairly well for small values of  $p$ , thanks to the abundance of the conditioning sample data. However, with increasing sparsity of samples, the prediction performance quickly deteriorates. In particular, the predicted values in the regions corresponding to very low or zero concentrations (dark regions) noticeably overestimate the true values, which is reflected in the appearance of speckles with lighter colors. On the other hand, the extremely large values are greatly underestimated, although it is less conspicuous in the reconstructed map due to the scarcity of such data. This phenomenon becomes much more pronounced at larger concentrations of missing data (lack of conditioning sample data), such as for  $p = 85\%$  presented in Fig. 5f. This problem is not conspicuous on the large scale but it is evident if one zooms in on a small area, as shown in the inset. In this particular area, the true values are  $Z(\mathbf{r}_p) = 0$ , nevertheless, the predictions  $\hat{Z}(\mathbf{r}_p) \in [0, 100]$ . Because the MPR model includes only one parameter, this averaging effect is not surprising. The temperature  $\hat{T}$  is estimated based on all the samples, involving regions with different degrees of spatial variability, and thus the resulting mean value of  $\hat{T}$  cannot be representative in all these areas and consequently cannot reflect the local conditions.

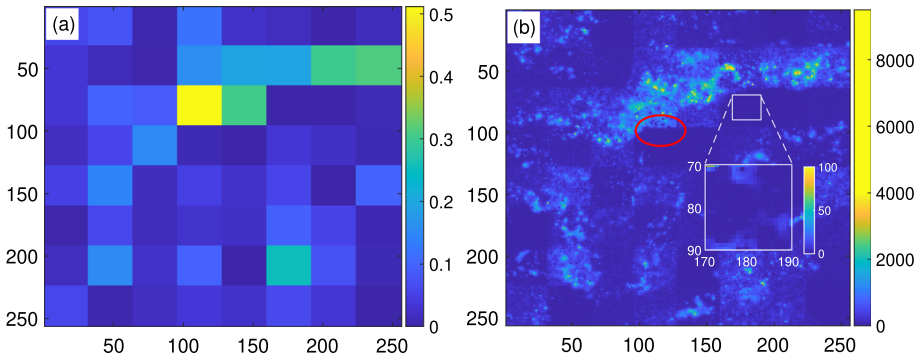
#### 3.3.2 SV-MPR Method-BST Implementation

To best demonstrate the beneficial effects of implementing spatial variability of the parameter in the SV-MPR versions of the algorithm introduced herein, the high sample sparsity case of  $p = 0.85$  (85%) is chosen. The MPR reconstruction is depicted in Fig. 5f, and the corresponding (spatially uniform) parameter value, estimated from the specific energy matching principle based on all samples, is  $\hat{T} = 0.0672$ . By using the double-checkerboard decomposition in the SV-MPR implementation, the spatial distribution of the estimated block-specific temperatures (BST) with square blocks of linear size  $l_b = 32$  is shown in Fig. 6a. One can observe great variability in BST, the values of which correlate with the sample variation in the respective blocks. In particular, the blocks with almost constant sample values close to zero are assigned very low values of  $\hat{T} \approx 0$ , while those with spatially highly variable samples are assigned much higher values of up to  $\hat{T} \approx 0.5$ . It is worth noting that the mean value of  $\hat{T} = 0.0693$ , calculated over all blocks, coincides rather well with the MPR estimate  $\hat{T}$ . Consequently, one can expect that the spatially variable parameter can better model



**Fig. 5** Original Walker Lake data with **a**  $p = 30\%$ , **c**  $p = 60\%$  and **e**  $p = 85\%$  of the data (white color) randomly removed and the corresponding MPR-based reconstructions obtained at the (spatially uniform) temperatures **b**  $\hat{T} = 0.0378$ , **d**  $\hat{T} = 0.0378$  and **f**  $\hat{T} = 0.0672$ , respectively

the local data variability than the spatially uniform one. Indeed, performing simulations using the SV-MPR (BST) implementation with the block-specific temperatures yields a reconstruction which suffers much less from the averaging effect than the MPR method, as demonstrated in Fig. 6b. Visually, the reconstruction is much closer to the original data than that of the original MPR algorithm, especially in the (dark) regions with low spatial variability. For comparison, the inset shows the SV-MPR



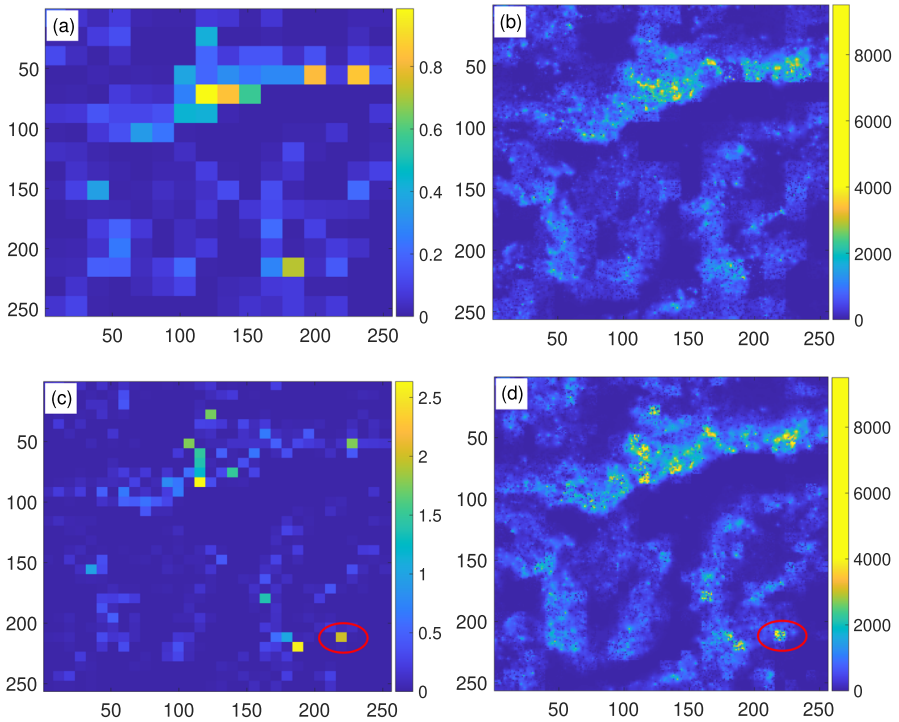
**Fig. 6** **a** Distribution of block-specific temperatures and **b** map of reconstructed data using the BST implementation of the SV-MPR method for  $l_b = 32$ . The red ellipse encloses an example of the edge-like artifact, and the square encloses the same as in Fig. 5f

predictions in the same area as for the MPR method in the inset of Fig. 5f, for which  $\hat{Z}(\mathbf{r}_p) \in [0, 20]$ .

On the other hand, undesirable artifacts resulting from the presence of boundaries between blocks with different parameters appear (see, e.g., the area marked by the red ellipse in Fig. 6b). Such unnatural edge-like effects are likely to emerge between blocks which include sample data with distinct degrees of variability. It is noted that the edges are partially smeared because the spins at the shared boundaries of the neighboring blocks interact with each other and thus propagate fluctuations from their blocks to the surrounding blocks. Nevertheless, such a diffusion has a local character with a rather limited range and cannot eliminate the edge-like effects completely.

To further eliminate these undesirable artifacts, two approaches were tried. The first one consists of gradually decreasing the linear block size  $l_b$  from 32 to 16 and to 8. The block temperatures (left column) and the corresponding data reconstructions (right column) are shown in Fig. 7 for  $l_b = 16$  (top row) and  $l_b = 8$  (bottom row). Decreasing the block size leads to the partial elimination of the conspicuous edges between blocks and also allows greater flexibility in capturing the local variability. On the other hand, the decreasing block size also reduces the number of neighboring sample pairs (bonds) available for the calculation of the block-specific sample energies and thus inhibits a reliable estimation of the block-specific temperatures. In particular for very sparse samples with high values of  $p$ , this may cause insufficient statistics for a reliable estimation of the block-specific temperatures. This leads to the second type of artifacts in the form of misestimation of the data variability due to the erroneous parameter estimation (see, e.g., the bright square in the lower right corner of Fig. 7d showing unexpectedly large variability including extremely large values due to the overestimation of  $\hat{T}$ ). Nevertheless, these results also demonstrate that decreasing granularity of the blocks results in a smoother spatial variation of the parameter and suppresses the block boundary effects considerably.

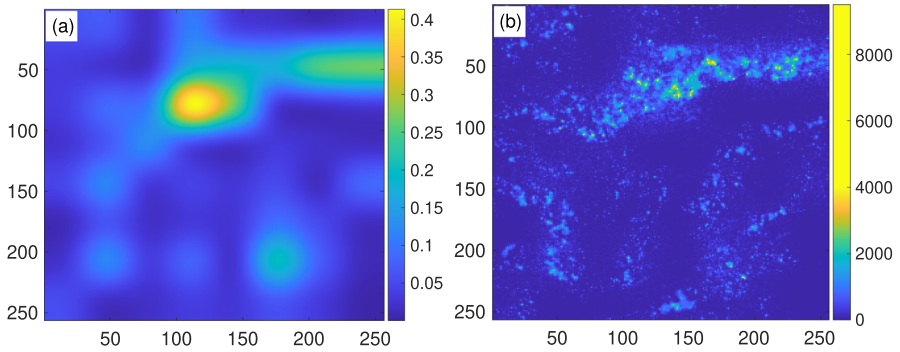




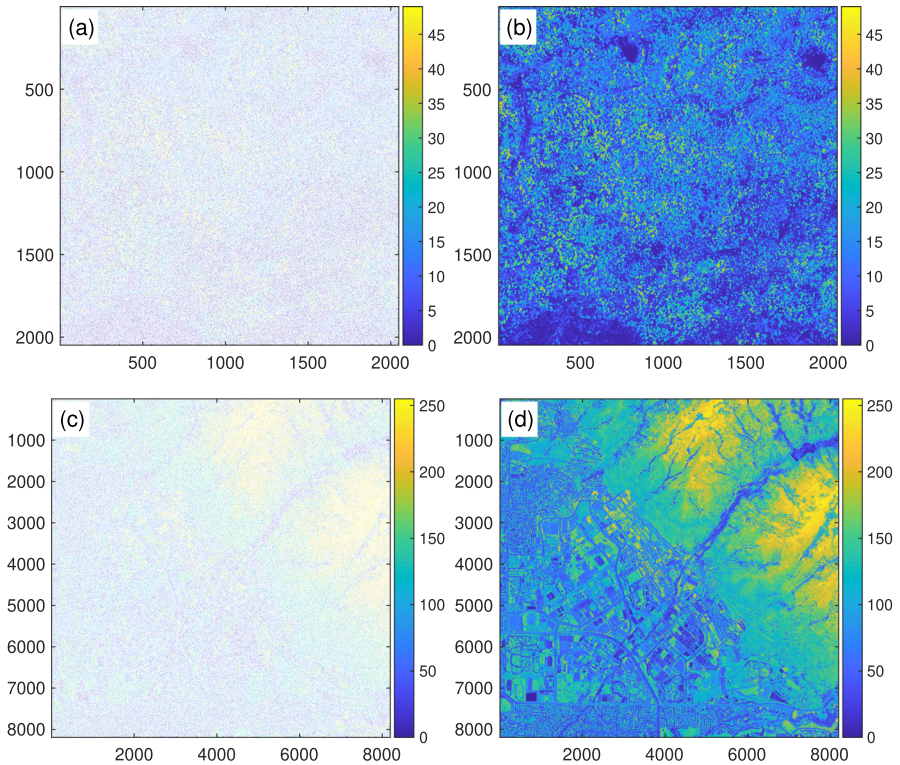
**Fig. 7** Distributions of block-specific temperatures (left column) and the corresponding data reconstructions (right column) using the SV-MPR (BST) implementation for  $l_b = 16$  (upper row) and  $l_b = 8$  (bottom row). The red ellipses highlight the area of the misestimated parameter (c) and consequent data variability (d) due to the lack of samples

### 3.3.3 SV-MPR Method-SST Implementation

The second implementation of the SV-MPR model attempts to eliminate the artifacts associated with the BST approach by applying a smoothing algorithm to the block-specific temperatures to obtain a smooth variation of the temperature values over the entire grid. Consequently, each site is assigned an individual value of the reduced temperature, and thus this implementation will be referred to as the SV-MPR method with site-specific temperatures (SST). The block-specific temperatures for Walker Lake data with  $l_b = 32$  (see Fig. 6a) after smoothing are shown in Fig. 8a. Using these local temperatures in the conditional simulations, the data reconstruction for the Walker Lake data is obtained, shown in Fig. 8b. Note that the unnatural boundary effects from the BST implementation are now completely eliminated, while the spatial variability in different regions is reproduced much better than in the original MPR method (compare to Fig. 5f). In principle, in this SV-MPR (SST) implementation, both the standard single and the double-checkerboard decomposition can be used, but for simplicity, the former was chosen. Applying the SV-MPR approach in Fig. 9, the results of the reconstruction of the Kaibab Plateau and Wasatch Front data sets after randomly removing  $p = 85\%$  of pixels are visually presented.



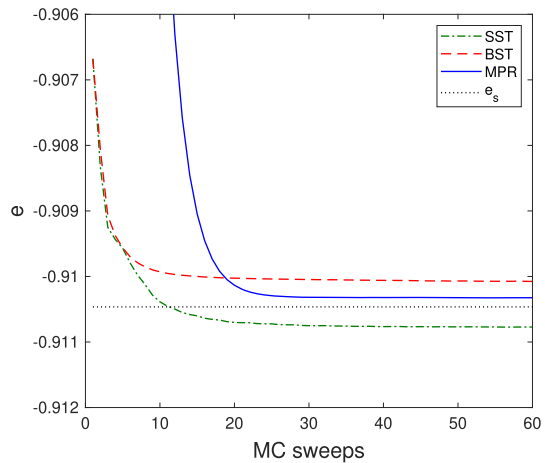
**Fig. 8** **a** Temperature map after applying the smoothing algorithm with  $n_s = 5$  to block temperatures with  $l_b = 32$ . **b** Walker Lake data reconstruction for  $p = 0.85$  by using the single-checkerboard implementation with site-specific temperatures (SST)



**Fig. 9** **a, c** Samples of the Kaibab Plateau and Wasatch Front, obtained by removing  $p = 85\%$  of pixels, and **b, d** their reconstructions obtained by the SV-MPR SST method



**Fig. 10** Evolution of the specific energy during equilibration process in the respective MPR-based methods for the Wasatch Front data with  $p = 0.3$  averaged over 100 realizations.  $e_s$  represents the average of the sample energy



It can be noted that in both the BST and the SST implementations, due to the splitting of the simulations on a large grid into several parallel simulations in much smaller blocks (including at most hundreds of sites), the equilibration is even faster than in the original MPR method. To demonstrate the impact of the domain spitting used in the SV-MPR methods, in Fig. 10 the equilibration process in the largest Wasatch Front data set with  $p = 0.3$  is illustrated. In particular, it shows the evolution of the specific energy  $e$ , calculated for the whole grid, in the respective MPR-based methods, averaged over 100 realizations. One can observe that even though a random initialization in the MPR method results in the initial value of  $e$  far from the equilibrium value, the relaxation process is relatively fast even for such a large data set, requiring only about 30 MC sweeps. Nevertheless, the SV-MPR BST and SST implementations can shorten it even more due to the initialization by the per-block averages, corresponding to the specific energies much closer to equilibrium values. It is also interesting to compare the equilibrium energy values  $e_{eq}$  resulting from different approaches among each other as well as with the sample-estimated value  $e_s$  (see Eq. (4)). One can see that while the MPR method gives  $e_{eq}$  close to  $e_s$ , the BST and SST implementations give the values of  $e_{eq}$  respectively higher and lower than  $e_s$ . The increased BST value can be explained by the presence of boundaries between different blocks, which create unnatural domain walls and thus increase the total energy. On the other hand, the smoothing of the temperatures in the SST implementation leads to partial elimination of the domain walls, not only those unnaturally created by the BST approach but also those substantiated by the data.

### 3.4 Statistical Validation

The results for the Walker Lake data set obtained by the standard MPR method and the SV-MPR BST and SST implementations are shown in Table 2 in terms of the prediction errors and computational times for different degrees of sample sparsity. As expected, both MAAE and MRASE errors increase with higher ratios of the missing data  $p$  for

**Table 2** Validation measures for the three implementations of the MPR-based prediction algorithm for various missing data ratios  $p$  applied to the Walker Lake pollution data set

| $p$  | MPR  |       |          | SV-MPR (BST) |       |          | SV-MPR (SST) |       |          |
|------|------|-------|----------|--------------|-------|----------|--------------|-------|----------|
|      | MAAE | MRASE | $t$ (ms) | MAAE         | MRASE | $t$ (ms) | MAAE         | MRASE | $t$ (ms) |
| 0.1  | 177  | 352   | 7.3      | 163          | 331   | 6.2      | 167          | 333   | 6.4      |
| 0.2  | 180  | 359   | 7.5      | 165          | 336   | 6.3      | 168          | 336   | 6.9      |
| 0.3  | 186  | 367   | 7.5      | 168          | 340   | 6.5      | 169          | 338   | 6.6      |
| 0.4  | 190  | 378   | 7.8      | 171          | 347   | 5.3      | 172          | 342   | 6.3      |
| 0.5  | 196  | 391   | 7.6      | 176          | 355   | 5.3      | 175          | 348   | 5.7      |
| 0.6  | 203  | 405   | 7.5      | 181          | 364   | 5.2      | 180          | 354   | 5.4      |
| 0.7  | 212  | 423   | 7.5      | 189          | 376   | 5.2      | 188          | 363   | 5.3      |
| 0.8  | 221  | 443   | 8.0      | 202          | 390   | 5.3      | 204          | 378   | 5.3      |
| 0.85 | 225  | 451   | 7.5      | 217          | 401   | 5.3      | 221          | 392   | 5.4      |

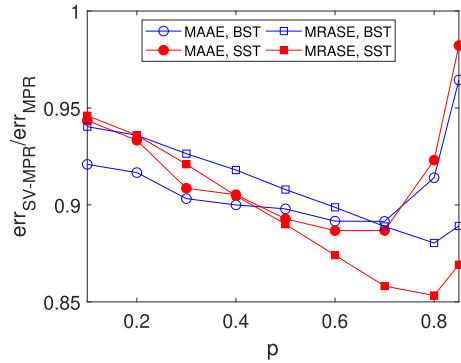
All implementations use single precision (FP32) arithmetic with hardware intrinsic CUDA functions,  $l_b = 32$  and for the SST implementation  $n_s = 5$

all implementations. Nevertheless, it is clear that implementing a spatial dependence in the simulation temperature has a significant positive effect on the prediction accuracy. Compared to the MPR approach with a single global temperature, the values of both MAAE and MRASE in the SV-MPR implementations are considerably smaller for all  $p$ . The relative improvements achieved by the present implementations are demonstrated in Fig. 11, in which error ratios of the respective methods are presented as a function of the data thinning. Not surprisingly, the benefits of the parameter spatial variability increase with the increase in the sample sparseness, that is, the decrease in the amount of the local conditioning data. The relative MAE (MRASE) errors decrease up to  $p \approx 0.7$  ( $p \approx 0.8$ ), where the SV-MPR accuracy outperforms that of the MPR by up to 11% (15%), and then they start increasing. Their increase for very large  $p$  is related to the above-discussed artifacts of the second kind—misestimation of the local (block-specific) parameters due to insufficient statistics within the respective blocks. By comparing the BST and SST implementations of the SV-MPR approach, the former appears to be more accurate for smaller  $p$ , while the benefits of the latter appear more at intermediate and larger  $p$ .

As already demonstrated by Žukovič et al. (2020), the GPU implementation of the MPR model resulted in a computationally very efficient prediction method. In the Walker Lake data set, the MPR prediction of an arbitrarily large portion of missing data takes no more than 8 ms, with no apparent dependence on  $p$ . Both the BST and SST implementations of the SV-MPR algorithm, besides the above-demonstrated increase of the prediction accuracy, also further decrease the computational burden with the computational time compressed to 5–7 ms. The computational complexity of the SV-MPR algorithm will be discussed in more detail below.

The results for the remaining (much larger) data sets are summarized in Table 3 for the missing data ratio  $p = 0.85$ . The expected degree of improvement in the prediction performance resulting from the introduction of spatial dependence in the parameter can be judged from the nature of the data under consideration. The Walker Lake data

**Fig. 11** Errors of the SV-MPR implementations (BST and SST),  $err_{SV-MPR}$ , relative to those obtained from the simple MPR method,  $err_{MPR}$ , as functions of the data thinning  $p$ , for the Walker Lake data set



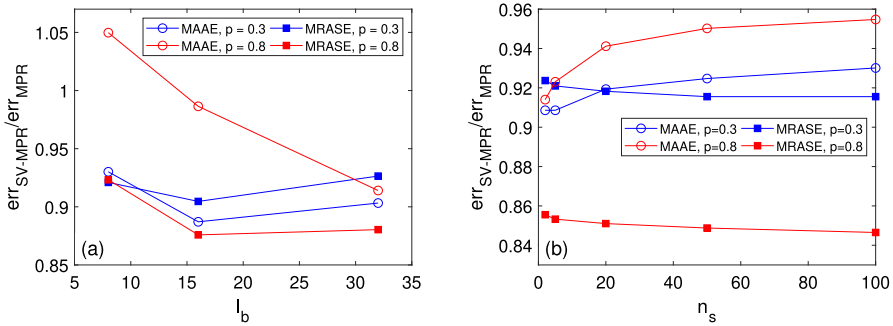
**Table 3** Validation measures for the three implementations of the MPR-based prediction algorithm for the Kaibab Plateau and Wasatch Front data sets with missing data ratio  $p = 0.85$

| Data set       | MPR   |       |         | SV-MPR (BST) |       |         | SV-MPR (SST) |       |         |
|----------------|-------|-------|---------|--------------|-------|---------|--------------|-------|---------|
|                | MAAE  | MRASE | $t$ (s) | MAAE         | MRASE | $t$ (s) | MAAE         | MRASE | $t$ (s) |
| Kaibab Plateau | 5.00  | 7.05  | 0.175   | 4.81         | 6.53  | 0.161   | 4.98         | 6.68  | 0.193   |
| Wasatch Front  | 26.14 | 35.60 | 2.64    | 23.36        | 32.58 | 2.44    | 24.22        | 33.37 | 3.02    |

The implementations use single precision (FP32) arithmetic with hardware intrinsic CUDA functions. The linear sizes of these data sets are  $L = 2,048$  for the Kaibab Plateau and  $L = 8,192$  for the Wasatch Front data sets

set with the extensive areas corresponding to (almost) constant values is an example of the spatial distribution that can greatly benefit from spatially variable parameters. To a certain extent similar features, with some larger areas of constant values, can also be observed in the remaining data sets. Therefore, the SV-MPR implementations can also be expected to deliver better prediction performance than the MPR method. Indeed, the SV-MPR implementations reduce the MPR MAAE errors by 4–11% (BST) and 0.4–7.4% (SST) and the MRASE errors by 7.4–8.5% (BST) and 5.3–6.3% (SST).

As for the computational efficiency, some comments are in order. Compared to the original MPR approach, one can observe overall only a small reduction of  $t$  (if any) achieved by the BST implementation, and even some increase by applying the SST implementation. This might appear as a contradiction with the above results for Walker Lake data presented in Table 2, but it has a simple explanation. The total execution time largely depends on the speed of the convergence to equilibrium, as the rest is dictated only by the number of samples needed for averaging and is almost independent of the chosen method. However, as explained above and demonstrated in Fig. 10, the SV-MPR implementations are indeed either faster in reaching the equilibrium state (BST) or more efficient in finding states with energy levels lower than those achievable by the MPR method (SST). The reason that the shorter BST equilibration time is not more apparently reflected in the values of  $t$  is that, for simplicity, the same (default) values of the parameter  $n_{fit} = 20$ , which defines the memory length of the energy time series employed in testing the onset of equilibrium, and the parameter  $n_f = 5$ , which defines the frequency of verification of equilibrium conditions, are used in all



**Fig. 12** **a** Errors of the SV-MPR method,  $err_{SV-MPR}$ , relative to those obtained from the simple MPR method,  $err_{MPR}$ , as functions of **a** the block size  $l_b$  in the BST implementation and **b** the smoothing parameter  $n_s$  in the SST implementation

the methods. (For a detailed description of  $n_{fit}$  and  $n_f$ , see also the paper by Žukovič et al. (2020).) Thus, in all the approaches used, the first check of the equilibrium conditions is only performed after the first  $n_{fit} + n_f = 25$  MC sweeps, even though in some cases the equilibrium is reached much faster. For example, from Fig. 10 one can see that the equilibrium conditions of the BST implementation are already reached at about 20 MC sweeps even for data as large as that for Wasatch Front. On the other hand, the SST implementation requires more (about 25) MC sweeps but it reaches lower energy levels. Additional contribution, which further increases the total SST execution time, comes from the temperature smoothing procedure. Thus, the execution time of the SV-MPR implementations can in principle be shortened by resetting the concerned parameters. However, considering the superior efficiency of all the MPR-based methods compared to some other approaches (e.g., see the comparison with IDW in Fig. 15), at present it is not considered essential.

### 3.5 Effect of SV-MPR Parameters

In the above study, the SV-MPR methods were used with the fixed block size  $l_b = 32$ , and in the SST implementation the fixed smoothing parameter  $n_s = 5$ . In the following, the effects of these parameters are demonstrated on the prediction performance of the respective methods. The latter is demonstrated in Fig. 12, in which the prediction errors of the SV-MPR method,  $err_{SV-MPR}$ , are presented relative to those obtained from the simple MPR method,  $err_{MPR}$ , for two cases of a relatively small ( $p = 0.3$ ) and large ( $p = 0.8$ ) degree of thinning. In Fig. 12a they are shown for the BST implementation considering different block sizes  $l_b$ , and in Fig. 12b for the SST implementation with the fixed  $l_b = 32$  and a varying parameter  $n_s$ .

As one can see from Fig. 12a, the above-used block size  $l_b = 32$  may not be optimal in terms of minimizing prediction errors. As discussed in the previous section, its decrease allows greater flexibility in capturing the local variability, which may result in an improvement in the prediction performance. On the other hand, smaller values of  $l_b$  in combination with larger values of  $p$  suffer from a lack of sampling points

within the blocks, which may lead to the block-specific parameter misestimation and consequently deterioration of prediction accuracy.

Figure 12b shows that the smoothing parameter  $n_s$  not only suppresses the edge-like visual artifacts, but its choice can also affect the prediction performance of the SST implementation. In particular, the gradual smoothing tends to increase MAAE and decrease (to a smaller extent) MRASE, and its effect is more pronounced in data with greater sparsity. Therefore, relatively small values of the smoothing parameter, such as  $n_s \approx 5$ , which to some degree suppress the visual artifacts but still do not excessively increase MAAE, may be considered an acceptable compromise. In any case, compared to the standard MPR method, the SST implementation of the SV-MPR method appears to deliver superior prediction performance for arbitrary choice of the parameter  $n_s$ .

### 3.6 Performance for Large Data Sets

To demonstrate the prediction and computational performance of the proposed methods on large data sets, the largest tested Wasatch Front data were selected, and experiments with the density of samples on the grid as well as the grid size were carried out. More specifically, the complete data set on the grid of linear size  $L = 8$ , 192 (consisting of  $L^2 = 67, 108, 864$  data points) was considered with gradually randomly removed  $p = 10\text{--}85\%$  (or  $p = 0.1 - 0.85$ ) of points to be predicted based on the remaining  $N = (1 - p)L^2$  sampling points. To perform configurational averaging, 100 random realizations were generated for each value of  $p$ . In Fig. 13a, the evolution of the MAAE and MRASE errors are presented with the increasing  $N$  (or decreasing  $p$ ) for all three MPR-based methods. One can observe that for this data set, in terms of predictive accuracy, the SV-MPR methods outperform the MPR method within a wide range of sampling densities, except for the cases with rather sparse distribution of the missing data, that is, for  $p \lesssim 0.2$ . In particular, the BST implementation gives the smallest errors for  $p \gtrsim 0.2$ ; however, its performance seems to deteriorate at higher sampling densities. This might be because the high density of samples provides sufficiently robust local constraints, and the global constraints in the form of the block-specific simulation temperatures are not as beneficial as in the cases with sparser sample distribution. On the other hand, the artificially created block edges increase the overall errors. It can be noted that such tendency was also observed in the Walker Lake data set, shown in Fig. 11 (except the region of very sparse samples with  $p \gtrsim 0.8$ ), although the SV-MPR methods still outperformed the MPR approach even for very small  $p$ .

The corresponding execution times are presented in Fig. 13b. The dependence on the number of samples appears not so straightforward, but it can be described and interpreted as follows. As was already observed in the MPR method and as one would intuitively expect, the execution time increased slightly with the increasing number of prediction points, that is, with the increasing  $p$  (see, e.g., Fig. 3 in Žukovič et al. (2020)). However, at very large  $p \gtrsim 0.8$  it suddenly dropped and remained almost constant with a further increase of  $p$ . This occurred due to a decrease in the number of MC sweeps needed to bring the system to equilibrium, which significantly determines

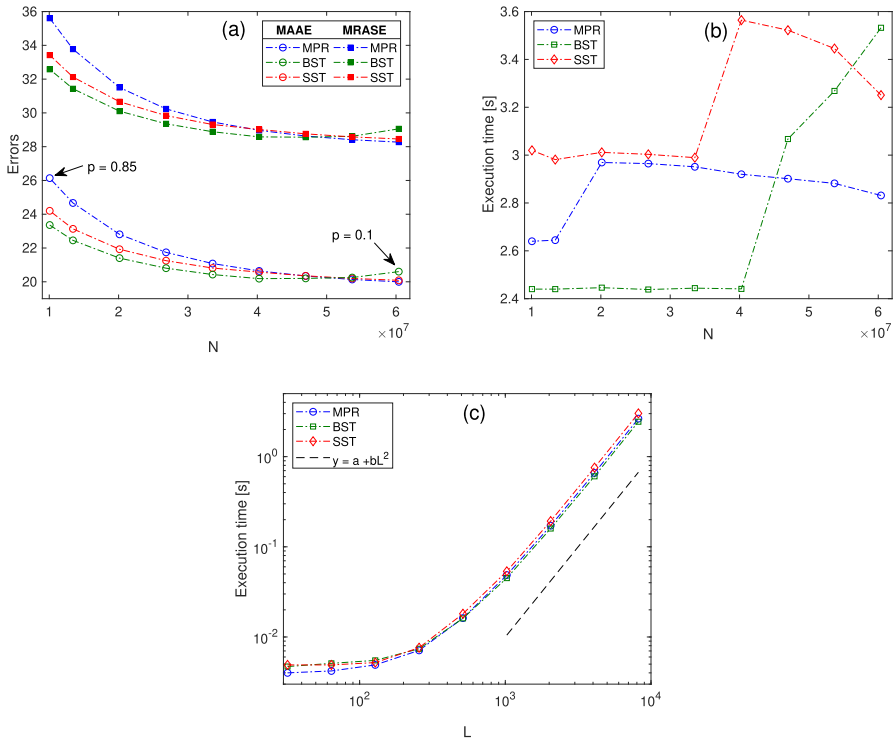
the overall execution time, to its minimum value set to  $n_{fit} + n_f = 25$  MC sweeps (see the above discussion at the end of Sect. 3.4). The relaxation at large  $p$  is greatly accelerated by the uniform initialization of the spins at the prediction locations by the sample mean value. (The MPR method does not seem to be sensitive to the choice of the initial state; nevertheless, opting for more thoughtful choices than a random state can speed up the relaxation process.) A similar phenomenon can also be observed in the remaining SV-MPR methods, but due to the initialization by the sample mean values within each block, the conditions for the fastest relaxation corresponding to 25 MC sweeps are fulfilled even at much smaller values of  $p$  (see also discussion in Sect. 3.4).

Finally, for the selected value of  $p = 0.85$ , the data sets with gradually increasing  $L$  (from 32 up to 8, 192) are generated by cutting out squares of size  $L \times L$  from the center of the original Wasatch Front data set. Thus, in this experiment the ratio of the number of the sampling and the prediction points remains the same in each data set, but the number of both increases with  $L$ . Here, the scalability of the proposed methods is of interest. Figure 13c shows that in all three methods, the execution time displays at most a linear (sublinear for smaller grids) increase with the grid size. All data represent mean values based on 100 realizations of the sampling distributions.

### 3.7 Comparison with Established IDW Approach

The prediction performance and computational efficiency of the original MPR method have been compared with several established interpolation methods (Žukovič and Hristopoulos 2018). Among them, the inverse distance weighted (IDW) (Shepard 1968) and the ordinary kriging (OK) Wackernagel (2003) were the methods that yielded prediction performance comparable to the MPR approach. However, the high computational complexity of OK prevents it from being applied to huge data sets. Therefore, for comparison of the present GPU-implemented MPR-based methods, the GPU-implemented IDW method (Marcellino et al. 2017) was chosen using the CUDA code available at GitHub (Ruggieri and Marcellino 2017) and described in Algorithm 3.

The implementation of IDW involves choosing two parameters, which can influence both the prediction performance and computational efficiency. They both determine how much weight one wants to assign to more distant sampling points. In particular, the power parameter  $\alpha$  controls the influence of sample data on the interpolated values depending on their distance from the output point, and by setting the search radius  $R$ , one can limit the number of points used for interpolation. Giving more (less) influence to distant points generally results in a smoother (rougher) interpolated surface. In principle, the significance of more distant points increases with the increasing spatial correlation range, and in such cases it makes sense to choose larger  $R$  to involve those points in the prediction. The problem is that one usually does not know the spatial correlation structure of the data. Thus, one often opts for default values offered by different software (typically  $\alpha = 2$  and  $R$  is left unrestricted), which may result in an increase in prediction errors as well as computational time. Optimal values of these parameters can be considered that minimize error function, for example, the mean absolute error. However, this would involve considerable extra computational effort.



**Fig. 13** **a** MAAE and MRASE errors and **b** the execution time of the MPR-based methods as functions of the number of sampling points  $N = (1 - p)L^2$ , for the Wasatch Front data. In **c** the execution time is presented as a function of the increasing linear grid size  $L$  for  $p = 0.85$  and the dashed line, as the guide to the eye, shows the linear dependence on the number of grid nodes  $L^2$

In the present analysis, the power parameter  $\alpha$  was set to a default value of 2, and the search radius  $R$  was varied from the minimum for which every prediction point has some sample points within the radius up to the maximum involving practically all points on the grid. It is noted that even after fixing  $\alpha$ , an optimal choice of  $R$  is not so obvious. It can be set to some fixed value equal for all points, which can result in the problem of some prediction points not having any samples in the search radius, or it can be made variable, which would make the IDW implementation more involved.

In Fig. 14, errors of the MPR (indicated in blue) and SV-MPR BST (in red) methods, relative to those obtained from the IDW method,  $err_{MPR}/err_{IDW}$  and  $err_{BST}/err_{IDW}$ , are presented as functions of the search radius  $R$ , for (a) Walker Lake and (b) Wasatch Front data sets. Therefore, the values larger than 1 indicate superior performance of IDW. One can see that IDW shows the best prediction performance for the smallest search radius (the optimal value might be close to  $R = 5$  grid units), where it outperforms both MPR and SV-MPR BST methods. However, with the increase in  $R$ , the IDW errors increase, and in most cases (except MAAE error for Walker Lake data), starting from some value of  $R$ , both MPR-based methods become superior. Owing to the improved prediction performance of the present SV-MPR BST implementation,



**Algorithm 3** GPU implemented IDW algorithm (Marcellino et al. 2017). Each thread interpolates a different value computing the weight for each known value and updating the weighted mean at the same time. Block threads are synchronized to store data set points in shared memory before the interpolation phase. For data sets that are too large, the points are stored in shared memory in different chunks

**Require:** known locations  $p(i)$ , known values  $z(i)$ , query locations  $q(j)$ , power parameter  $\alpha$ , search radius  $R$ , weights defined by Euclidean distance  $\lambda$ , number  $tid$  to uniquely identify each thread, the number of data set points which each thread in a block loads into shared memory  $stride$ , subscript  $c$  to indicate a location or value belonging to the  $c$ th chunk.

**Ensure:** unknown values  $z^*(i)$

1: initialize  $\alpha$  and  $R$

2:  $loc\_q \leftarrow q(tid)$

**for** each chunk  $c$  **do**

3:  $i \leftarrow 0$

4:  $start\_ind \leftarrow tid * stride$

**while** ( $i < stride$ ) and ( $i + start\_ind < size(c)$ ) **do**

5: put  $p_c(start\_ind + i)$  into shared memory

6:  $i \leftarrow i + 1$

**end while**

7: synchronize threads

**for**  $i = 1, \dots, size(c)$  **do**

8:  $loc\_p \leftarrow p_c(i)$  from shared memory

9:  $d \leftarrow dist(loc\_p, loc\_q)$

**if**  $d \neq 0$  **then**

**if**  $d_{ij} < R$  **then**

10:  $\lambda \leftarrow d^{-\alpha}$ ;  $z^*(tid) \leftarrow z^*(tid) + \lambda z_c(i)$ ;  $wsum \leftarrow wsum + \lambda$

**end if**

**else**

11:  $z^*(tid) \leftarrow z_c(i)$ ;  $wsum \leftarrow 1$

12: break and skip this cycle for the next chunks

**end if**

**end for**

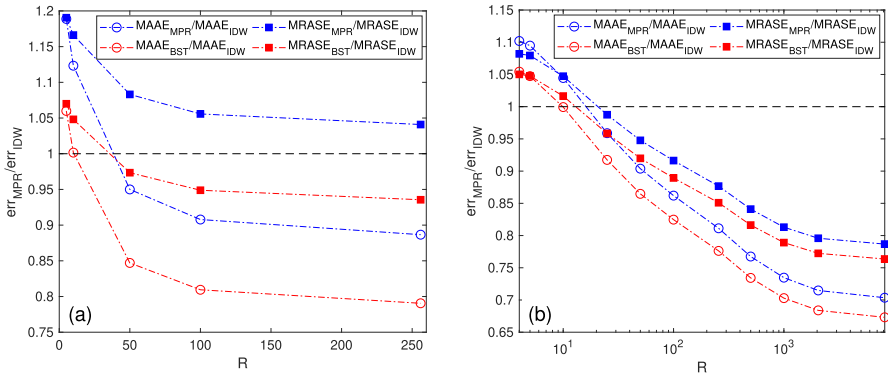
13: synchronize threads

**end for**

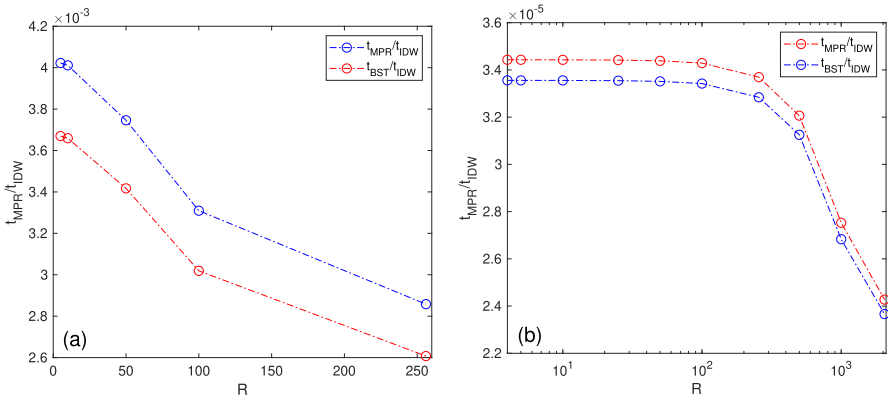
14: put  $z^*(tid)/wsum$  into global memory

the latter outperforms IDW starting from much smaller  $R$  than the standard MPR method. Even for the optimal IDW performance at small  $R$ , the errors of the SV-MPR BST method do not exceed those from IDW by more than 5–7%. Thus, IDW would outperform the present methods if the parameters were set sufficiently close to the optimal values. However, this would require either a priori knowledge of the spatial correlation structure of the data or some time-consuming cross-validation analysis, while the MPR methods need no such information or extra analysis.

On the other hand, the computational efficiency of the MPR-based methods clearly dominates over the IDW performance, regardless of the parameters. Due to the favorable scaling properties of the MPR method (computational time scales approximately linearly with system size), its dominance over IDW further increases with the data size. As presented in Fig. 15, the ratio of the GPU execution times of the MPR-based to IDW methods is of the order of  $10^{-3}$  for the relatively small Walker Lake data set ( $L = 256$ ) and of the order of  $10^{-5}$  for the much larger Kaibab Plateau data set ( $L = 2,048$ ). It is noted that for data as massive as those for Wasatch Front ( $L = 8,192$ ), the



**Fig. 14** MAAE (open circles) and MRASE (filled squares) errors of the MPR (blue) and SV-MPR BST (red) methods, relative to those obtained from the IDW method, as functions of the search radius,  $R$ , for **a** Walker Lake and **b** Wasatch Front data sets



**Fig. 15** GPU time of the MPR (blue color) and SV-MPR BST (red) methods, relative to those required by the IDW method, as functions of the IDW search radius  $R$ , for **a** the Walker Lake and **b** Kaibab Plateau data sets

IDW calculations could not be executed on a standard GPU-equipped PC within some reasonable time at all. (The calculations were terminated after about 2 weeks of running, and the errors presented in Fig. 14b were obtained by executing the code on the supercomputer Govorun at the Joint Institute for Nuclear Research in Dubna.) The curves presented in Fig. 15 also show that the relative computational efficiency of the MPR-based methods, as expected, increases with the search radius  $R$  due to the decreasing efficiency of IDW. Furthermore, by comparing the results for the two MPR-based approaches one can see that the SV-MPR BST method is somewhat faster than the original MPR approach, again in line with the expectations discussed above.

## 4 Conclusions

In the current study, a rather general approach to modeling spatial heterogeneity, a feature often present in massive spatial data, was proposed in GPU-implemented spatial prediction methods for gridded data. In particular, two approaches to introducing spatial dependence to the model parameter (temperature) were presented by the so-called double-checkerboard (DC) decomposition to the previously introduced GPU-accelerated MPR method, and thus two SV-MPR methods with spatially varying temperatures were obtained. In the BST variation, separate values of the temperature are obtained for each block of the DC-decomposed grid, and in the SST variation, even each individual prediction point is modeled using its own temperature. Then, similar to the MPR method, predictions of unknown values are obtained from conditional simulations. However, in the present methods, the conditional situations are non-equilibrium but assume “local” equilibrium conditions corresponding to the local temperatures.

Using various types of large heterogeneous real data, such as remote sensing images, it has been demonstrated that the proposed SV-MPR methods significantly improve prediction performance and even computational efficiency of the original MPR method. Their prediction performance is competitive with some established prediction methods, such as IDW, but their execution times are by several orders of magnitude faster. It is noted that the presented approach to modeling spatial heterogeneity was demonstrated on the MPR method but in fact it is rather general and its application to other GPU-implemented methods is rather straightforward. One can consider various extensions of the presented models to further increase their flexibility and ability to capture various relevant features present in real data, such as geometric anisotropy or non-Gaussianity. It is noted that the MPR method was shown to be able to model non-Gaussianity to a certain extent; nevertheless, the model is still very simple, with only one parameter, and non-Gaussian spatial data can follow various types of distributions. Therefore, such extensions may involve the introduction of more parameters in addition to the temperature, while still keeping their local nature (within blocks) and spatial variability. For example, geometric anisotropy could be introduced by distinguishing the exchange interactions in different directions, namely, by introducing a directional exchange interaction anisotropy parameter. The non-Gaussianity could be incorporated by including higher-order interactions and/or applying some suitable form of an external “magnetic” (bias) field to the Hamiltonian. Another possibility to consider is the inclusion of further-neighbor pairwise interactions that would, for example, control data smoothness. Tests on such a CPU-implemented generalized MPR model suggested that for some types of data, significant benefits can be obtained (Žukovič and Hristopulos 2023). The impressive computational efficiency of the models offers the possibility to extend them to three dimensions (three-dimensional space or two-dimensional space + time), where there is still a lack of efficient methods that would enable modeling of massive data (Wang et al. 2012). The generalization of the GPU code from two to three dimensions is straightforward. The GPU time in three dimensions is expected to increase by a factor of  $3/2$ , but the relative efficiency of the CPU and GPU implementations should be preserved (Weigel 2012).

**Acknowledgements** This work was supported by the Scientific Grant Agency of the Ministry of Education of the Slovak Republic (Grant No. 1/0695/23) and the Slovak Research and Development Agency (Grant No. APVV-20-0150).

**Author Contributions** CRediT authorship contribution statement Matúš Lach: Software, Investigation, Data curation, Visualization, Validation, Writing—original draft preparation. Milan Žukovič: Conceptualization, Methodology, Formal analysis, Validation, Writing—review and editing, Supervision. All authors contributed to the writing of the manuscript.

**Funding** Open access funding provided by The Ministry of Education, Science, Research and Sport of the Slovak Republic in cooperation with Centre for Scientific and Technical Information of the Slovak Republic Open access funding provided by The Ministry of Education, Science, Research and Sport of the Slovak Republic in cooperation with Centre for Scientific and Technical Information of the Slovak Republic.

## Declarations

**Conflict of interest** The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

**Code Availability** Name of the code/library: packages MPR, SV – MPR(BST), and SV – MPR(SST) Contact: e-mail and phone number: matus.lach@gmail.com, +421 915 609 735 Hardware requirements: NVIDIA GPU with compute capability 5.2 or higher (GTX 950 or newer) Program language: CUDA Software required: CUDA 10.1, CUB library, Windows or Linux operating system (not tested on OS X) Program size in the zipped format: 5 MB for MPR, 3.3 MB for SV – MPR(BST), and 9.4 MB for the source codes are available for download at the links: <https://github.com/MatusLach/MPR>; [https://github.com/MatusLach/SV-MPR\\_BST](https://github.com/MatusLach/SV-MPR_BST); [https://github.com/MatusLach/SV-MPR\\_SST](https://github.com/MatusLach/SV-MPR_SST).

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

## References

- Bai T, Tahmasebi P (2021) Accelerating geostatistical modeling using geostatistics-informed machine learning. *Comput Geosci* 146:104663
- Cheng T (2013) Accelerating universal kriging interpolation algorithm using CUDA-enabled GPU. *Comput Geosci* 54:178–183
- Cheng T, Li D, Wang Q (2010) On parallelizing universal kriging interpolation based on OpenMP. In: 2010 Ninth international symposium on distributed computing and applications to business. Engineering and Science, pp 36–39
- Coleman JB, Yao X, Jordan TR, Madden M (2011) Holes in the ocean: filling voids in bathymetric lidar data. *Comput Geosci* 37(4):474–484
- Cressie N, Johannesson G (2018) Fixed rank kriging for very large spatial data sets. *J R Stat Soc: Ser B (Statistical Methodology)* 70(1):209–226
- de Ravé EG, Jiménez-Hornero F, Ariza-Villaverde A, Gómez-López J (2014) Using general-purpose computing on graphics processing units (GPGPU) to accelerate the ordinary kriging algorithm. *Comput Geosci* 64:1–6
- Emmendorfer LR, Dimuro GP (2021) A point interpolation algorithm resulting from weighted linear regression. *J Comput Sci* 50:101304

- Finley AO (2011) Comparing spatially-varying coefficients models for analysis of ecological data with non-stationary and anisotropic residual dependence. *Methods Ecol Evol* 2(2):143–154
- Fotheringham AS, Brunsdon C, Charlton M (2003) Geographically weighted regression: the analysis of spatially varying relationships. Wiley, London
- Furrer R, Genton MG, Nychka D (2006) Covariance tapering for interpolation of large spatial datasets. *J Comput Graph Stat* 15(3):502–523
- Gelfand AE, Kim HJ, Sirmans C, Banerjee S (2003) Spatial modeling with spatially varying coefficient processes. *J Am Stat Assoc* 98(462):387–396
- Guan Q, Kyriakidis PC, Goodchild MF (2011) A parallel computing approach to fast geostatistical areal interpolation. *Int J Geogr Inf Sci* 25(8):1241–1267
- Harris M (2007) Optimizing parallel reductions in CUDA. <https://developer.download.nvidia.com/assets/cuda/files/reduction.pdf>
- Harris R, Singleton A, Grose D, Brunsdon C, Longley P (2010) Grid-enabling geographically weighted regression: a case study of participation in higher education in England. *Trans GIS* 14(1):43–61
- Hartman L, Hössjer O (2008) Fast kriging of large data sets with Gaussian Markov random fields. *Comput Stat Data Anal* 52(5):2331–2349
- Hristopulos D (2003) Spartan Gibbs random field models for geostatistical applications. *SIAM J Sci Comput* 24(6):2125–2162
- Hristopulos DT (2015) Stochastic local interaction (sli) model. *Comput Geosci* 85(PB):26–37
- Hristopulos DT, Elogne SN (2007) Analytic properties and covariance functions for a new class of generalized Gibbs random fields. *IEEE Trans Inf Theory* 53(12):4667–4679
- Hristopulos DT, Pavlides A, Agou VD, Gkafa P (2021) Stochastic local interaction model: an alternative to kriging for massive datasets. *Math Geosci* 53(8):1907–1949
- Hu H, Shu H (2015) An improved coarse-grained parallel algorithm for computational acceleration of ordinary kriging interpolation. *Comput Geosci* 78:44–52
- Ingram B, Cornford D, Evans D (2008) Fast algorithms for automatic mapping with space-limited covariance functions. *Stoch Environ Res Risk Assess* 22(5):661–670
- Isaaks E, Srivastava R (1989) Applied geostatistics. Oxford University Press, New York
- Kadlec J, Ames DP (2017) Using crowdsourced and weather station data to fill cloud gaps in MODIS snow cover datasets. *Environ Model Softw* 95:258–270
- Kaufman CG, Schervish MJ, Nychka DW (2008) Covariance tapering for likelihood-based estimation in large spatial data sets. *J Am Stat Assoc* 103(484):1545–1555
- Kerry KE, Hawick KA (1998) Kriging interpolation on high-performance computers. In: Sloot P, Bubak M, Hertzberger B, (eds), High-performance computing and networking. Springer, Berlin, pp 429–438
- Lajaunie C, Renard D, Quentin A, Le Guen V, Caffari Y (2020) A non-homogeneous model for kriging dosimetric data. *Math Geosci* 52(7):847–863
- Lehman J, Swinton K, Bortnick S, Hamilton C, Baldrige E, Eder B, Cox B (2004) Spatio-temporal characterization of tropospheric ozone across the eastern united states. *Atmos Environ* 38(26):4357–4369
- Li Z, Fotheringham AS, Li W, Oshan T (2019) Fast geographically weighted regression (FastGWR): a scalable algorithm to investigate spatial process heterogeneity in millions of observations. *Int J Geogr Inf Sci* 33(1):155–175
- MacGillivray M, Hunter D, Jan N (1993) “Local” equilibrium monte carlo simulations. *Phys A* 197(1–2):144–152
- Marcellino L, Montella R, Kosta S, Galletti A, Di Luccio D, Santopietro V, Ruggieri M, Lapegna M, D’Amore L, Laccetti G (2017) Using gpgpu accelerated interpolation algorithms for marine bathymetry processing with on-premises and cloud based computational resources. In: International conference on parallel processing and applied mathematics, Springer, pp 14–24
- Marcotte D, Allard D (2018) Half-tapering strategy for conditional simulation with large datasets. *Stoch Environ Res Risk Assess* 32(1):279–294
- Mariethoz G (2010) A general parallelization strategy for random path based geostatistical simulation methods. *Comput Geosci* 36(7):953–958
- Mei G (2014) Evaluating the power of GPU acceleration for IDW interpolation algorithm. *Sci World J* 2014:1715741–8
- Mei G, Xu L, Xu N (2017) Accelerating adaptive inverse distance weighting interpolation algorithm on a graphics processing unit. *Open Sci* 4(9):170436

- Misra C, Bhattacharya S, Ghosh SK (2020) A fast scalable distributed kriging algorithm using spark framework. *Int J Data Sci Anal* 10(3):249–264
- Murakami D, Tsutsumida N, Yoshida T, Nakaya T, Lu B (2020) Scalable GWR: a linear-time algorithm for large-scale geographically weighted regression with polynomial kernels. *Ann Am Assoc Geogr* 111(2):459–480
- Nunes R, Almeida JA (2010) Parallelization of sequential gaussian, indicator and direct simulation algorithms. *Comput Geosci* 36(8):1042–1052
- Pardo-Igúzquiza E, Dowd PA, Grimes DI (2005) An automatic moving window approach for mapping meteorological data. *Int J Climatol: J R Meteorol Soc* 25(5):665–678
- Peredo O, Ortiz JM, Herrero JR (2015) Acceleration of the geostatistical software library (GSLIB) by code optimization and hybrid parallel programming. *Comput Geosci* 85:210–233
- Pesquer L, Cortés A, Pons X (2011) Parallel ordinary kriging interpolation incorporating automatic variogram fitting. *Comput Geosci* 37(4):464–473
- Que X, Ma C, Ma X, Chen Q (2021) Parallel computing for fast spatiotemporal weighted regression. *Comput Geosci* 150:104723
- Ruggieri M, Marcellino L (2017) Parallel GPU inverse distance weighting. <https://github.com/MarioRuggieri/G-IDW>
- Shepard D (1968) A two-dimensional interpolation function for irregularly-spaced data. In: *Proceedings of the 1968 23rd ACM national conference*, pp 517–524
- Sickles JE, Shadwick DS (2007) Effects of missing seasonal data on estimates of period means of dry and wet deposition. *Atmos Environ* 41(23):4931–4939
- State of Utah (2015) State of Utah Acquired LiDAR Data - Wasatch Front. Distributed by OpenTopography. <https://doi.org/10.5069/G9TH8JNQ>
- Stojanovic N, Stojanovic D (2014) High performance processing and analysis of geospatial data using CUDA on GPU. *Adv Electr Comput Eng* 14(4):109–114
- Sun L, Chen Z, Gao F, Anderson M, Song L, Wang L, Hu B, Yang Y (2017) Reconstructing daily clear-sky land surface temperature for cloudy regions from MODIS data. *Comput Geosci* 105:10–20
- Tahmasebi P, Sahimi M, Mariethoz G, Hezarkhani A (2012) Accelerating geostatistical simulations using graphics processing units (GPU). *Comput Geosci* 46:51–59
- US Forest Service (2019) Mapping the Kaibab Plateau, AZ. Collected by 3Di West, distributed by OpenTopography. <https://doi.org/10.5069/G9TX3CH3>
- Wackernagel H (2003) *Multivariate geostatistics*, 3rd edition. Springer, Berlin
- Wang G, Garcia D, Liu Y, de Jeu R, Johannes Dolman A (2012) A three-dimensional gap filling method for large geophysical datasets: application to global satellite soil moisture observations. *Environ Model Softw* 30:139–142
- Weigel M (2012) Performance potential for simulating spin models on GPU. *J Comput Phys* 231(8):3064–3082
- Wiens A, Nychka D, Kleibe W (2020) Modeling spatial data using local likelihood estimation and a mat'ern to sar translation. *arXiv preprint arXiv:2002.01124*
- Xia YJ, Kuang L, Li XM (2011) Accelerating geospatial analysis on GPUs using CUDA. *J Zhejiang Univ Sci C* 12(12):990–999
- Zhang Y, Zheng X, Wang Z, Ai G, Huang Q (2018) Implementation of a parallel GPU-based space-time kriging framework. *ISPRS Int J Geo Inf* 7(5):193
- Žukovič M, Borovský M, Lach M, Hristopulos DT (2020) GPU-accelerated simulation of massive spatial data based on the modified planar rotator model. *Math Geosci* 52(1):123–143
- Žukovič M, Hristopulos DT (2009a) Classification of missing values in spatial data using spin models. *Phys Rev E* 80(1):011116
- Žukovič M, Hristopulos DT (2009b) Multilevel discretized random field models with “spin” correlations for the simulation of environmental spatial data. *J Stat Mech: Theory Exp* 02:P02023
- Žukovič M, Hristopulos DT (2018) Gibbs Markov random fields with continuous values based on the modified planar rotator model. *Phys Rev E* 98(6):062135
- Žukovič M, Hristopulos DT (2023) Spatial data modeling by means of Gibbs–Markov random fields based on a generalized planar rotator model. *Phys A* 612:128509