Computer Physics Communications 299 (2024) 109134

Contents lists available at ScienceDirect

Computer Physics Communications

journal homepage: www.elsevier.com/locate/cpc

Computer Programs in Physics

Evaluation of classical correlation functions from 2/3D images on CPU and GPU architectures: Introducing CorrelationFunctions.jl ☆,☆☆

Vasily Postnicov [a], Aleksei Samarin [a,b], Marina V. Karsanina [a], Mathieu Gravey [c], Aleksey Khlyupin [d], Kirill M. Gerke [a,*]

[a] Schmidt Institute of Physics of the Earth of Russian Academy of Sciences, Moscow, 107031, Russia
[b] Computational Mathematics and Cybernetics, Lomonosov Moscow State University, Moscow, 119991, Russia
[c] Institute for Interdisciplinary Mountain Research, Austrian Academy of Sciences, Innsbruck, 6020, Austria
[d] Moscow Institute of Physics and Technology, Institutskiy Pereulok 9, Dolgoprudny, Moscow, 141700, Russia

ARTICLE INFO

Keywords:
3D images
Correlation functions
Image analysis

ABSTRACT

Correlation functions are becoming one of the major tools for quantification of structural information that is usually represented as 2D or 3D images. In this paper we introduce `CorrelationFunctions.jl` open-source package developed in Julia and capable of computing all classical correlation functions based on imaging input data. Images include both binary and multi-phase representations. Our code is capable of evaluating two-point probability S_2, phase cross-correlation ρ_{ij}, cluster C_2, lineal-path L_2, surface-surface F_{ss}, surface-void F_{sv}, pore-size P and chord-length p distribution functions on both CPU and GPU architectures. Where possible, we presented two types of computations: full correlation map (correlations of each point with other points on the image, that also allows obtaining ensemble averaged CF) and directional correlation functions (currently in major orthogonal and diagonal directions). Such an implementation allowed for the first time to assemble a completely free solution to evaluate correlation functions under any operating system with well documented application programming interface (API). Our package includes automatic tests against analytical solutions that are described in the paper. We measured execution times for all CPU and GPU implementations and as a rule of thumb full correlation maps on GPU are faster than other methods. However, full maps require more RAM and, thus, are limited to available RAM resources. On the other hand, directional CFs are memory efficient and can be evaluated for huge datasets – this way they are the first candidates for structural data compression of feature extraction. The package itself is available through Julia package ecosystem and on GitHub, the latter source also contains documentation and additional helpful resources such as tutorials. We believe that a single powerful computational tool such as `CorrelationFunctions.jl` presented in this paper will significantly facilitate the usage of correlation functions in numerous areas of structural description and research of porous materials, as well as in machine learning applications. We also present some examples as applied to ceramic, soil composite and oil-bearing rock samples based on their 3D X-ray tomography and 2D scanning electron microscope images. Finally, we conclude our paper with discussion of possible ways to further improve presented computational framework.

Program summary
Program Title: CorrelationFunctions.jl
CPC Library link to program files: https://doi.org/10.17632/6gb9gfm3dw.1
Developer's repository link: https://github.com/fatimp/CorrelationFunctions.jl
Licensing provisions: MIT
Programming language: Julia
Supplementary material: Numerous Jupiter notebooks with examples are available on the GitHub page

☆ The review of this paper was arranged by Prof. Weigel Martin.
☆☆ This paper and its associated computer program are available via the Computer Physics Communications homepage on ScienceDirect (http://www.sciencedirect.com/science/journal/00104655).

* Corresponding author.
E-mail address: kg@ifz.ru (K.M. Gerke).

https://doi.org/10.1016/j.cpc.2024.109134
Received 22 September 2023; Received in revised form 9 February 2024; Accepted 12 February 2024
Available online 15 February 2024

Nature of problem: Correlation functions are invaluable universal statistical descriptors of structures used in numerous scientific fields such as astronomy, material science, rock and soil physics, hydrology and biology, to name just a handful of examples. While computational approaches are available in the literature for some functions, they are fragmented and are usually implemented in proprietary interpreted languages for CPU architecture alone.

Solution method: We contribute an open source and cross-platform solution with well documented API for computation of all classical correlation functions from both 2D and 3D images on CPU and GPU architectures. The package computes correlation functions using two approaches: computation of correlation maps and computation along predefined directions. These two approaches can be thought of as an execution time - memory trade-off, but the choice may also depend on the application. The computations are based on a) fast Fourier transform with preprocessing steps such as cluster labeling or edge detection, and b) linear scan approach to evaluate correlation functions along predefined directions. Where justified, the algorithms can be executed on both CPU and GPU which results in high execution speed on modern hardware.

1. Background and motivation

Heterogeneous materials are ubiquitous both in nature and in industrial human activities, and their physical properties are interconnected to structure [1,2]. The information about the structure usually comes in the way of 2D and 3D images obtained by microscopy [3–5], tomography [6] and other related methodologies. This structural information can be used for numerous types of analysis, including simulations of different processes within studied materials or evaluation of their properties based on 3D structural data [7–9]. Unlike computational modeling techniques a whole class of approximate methods exists including (rigorous) bounds that allows very fast estimations of physical properties [10–12] albeit the bounds themselves are too wide to be applicable in the majority of practical problems. Nonetheless, it is noteworthy that many of such bounds are originally based on structural descriptors in the form of correlation functions (CFs).

Two ways exist to obtain correlation functions for a given structure at hand: measure them either experimentally with the help of, for example, scattering intensity [13,14] or from digital images [15,16]. None is perfect, as the first approach suffers from the limitation of CFs that can be obtained this way, while the second one provides information with limited resolution or/and resolution to field-of-view ratio [17]. In addition to the resolution conundrum, the most useful imaging methods such as X-ray computed tomography (XCT) and scanning electron microscopy (SEM) provide gray-scale images (e.g., X-ray attenuation or electron back-scattering distributions) due to their underlying physical principles and require labeling of constituent phases (or segmentation) before computation of CFs [18]. If properly segmented [19], high-resolution digital images do provide a possibility to compute any correlation function and, thus, possibility to address numerous fundamental and practical research problems.

Correlation functions as invaluable universal descriptors of structure are utilized in a multitude of scientific disciplines: material sciences [20–23], rock physics [24], soil physics and hydrology [25,26], cosmology and food engineering [27,28], biology [29] and many others. Computed from 2D and 3D images, CFs can be applied to:

1. characterize the morphology and representativeness via correlation lengths [30–32];
2. perform stochastic reconstructions from experimentally measured CFs or 2D to 3D reconstructions based on CFs that can be extracted from less dimensions or through penetrable sphere model [33–35, 20,36];
3. compare different structures to each other, including verification of stochastic reconstructions [30,35,37];
4. compress structural information in the form of raw or parameterized CFs with the possibility to recover the structure using stochastic reconstruction [17,21,26];
5. describe structural dynamics under different boundary conditions [38–40];
6. extract structural features for deep learning [41–43,26];
7. fuse multi-scale images and structural information as obtained by different methods into a single digital model [44];
8. quantify spatial heterogeneity [45–47].

Stochastic reconstruction is a separate huge topic, as this approach allows to solve an inverse problem and recover structure from known set of correlation functions [33,34,30,20–22,35,37,31,32]. This ability for recovery serves as the basis for majority of usages in the list above. While it is necessary to compute CFs from digital images as a target set for stochastic reconstructions, they rely more on efficient recomputations or optimizations, e.g., re-evaluation during simulated annealing. Such optimizations are not the part of the computational package described here and are not considered in this work.

There are different types of correlation functions that in general describe some probabilities. The main parameter of any correlation function is the number of points that are utilized to evaluate such a probability. While so-called n-point probability function [1] will completely describe any structure in the limit of $n \rightarrow$ number of voxels/pixels on the image, computing or storing such a function is not practical. Based on information content, it was shown that increasing the number of point $n > 2$ only marginally improves the quality of the structure quantification and these improvements decay with increasing n [48,49]. For stochastic reconstruction purposes computation of higher-order statistics will, arguably, not be balanced by increased number of points, but this is the topic of active research [50,51]. For aforementioned reasons, we mainly focus on 2-point statistics, leaving the possibility to include higher order CFs in the future work. The simplest 2-point probability function (S_2 or autocorrelation) actually arises from small angle scattering experiments and measures the probability that both ends of line segment with a given length fall into the same phase. Other types include the probability of a whole line segment to fall into the phase (L_2 function) or "pointing" with its ends to the same cluster (C_2 function) and mainly originate from aforementioned bounds on physical properties such as permeability and elasticity. The general idea of computation of different 2-point CFs is shown in Fig. 1. All major details and numerous analytical cases can be found in seminal work of Torquato [1]. To all CFs described in this book and applicable to digital images we shall refer to as classical correlation functions.

While evaluation of correlation functions from 2D and 3D images was a topic of numerous research papers, the information is highly fragmented, especially in terms of algorithms and their implementation in the code. Some snippets of code are available for some functions, but they lack general API and are usually implemented in proprietary interpreted languages (e.g. Matlab). Due to a recent leap forward in GPU-based computations, some papers describe computations on this architecture, but the code is usually not provided. All in all, the efficient and open-source single API solution utilizing both CPU and GPU architectures is currently absent.

The aim of this paper is to establish a general open-source solution allowing to compute all classical correlation functions. This solution should incorporate novel methods to compute CFs and be well doc-

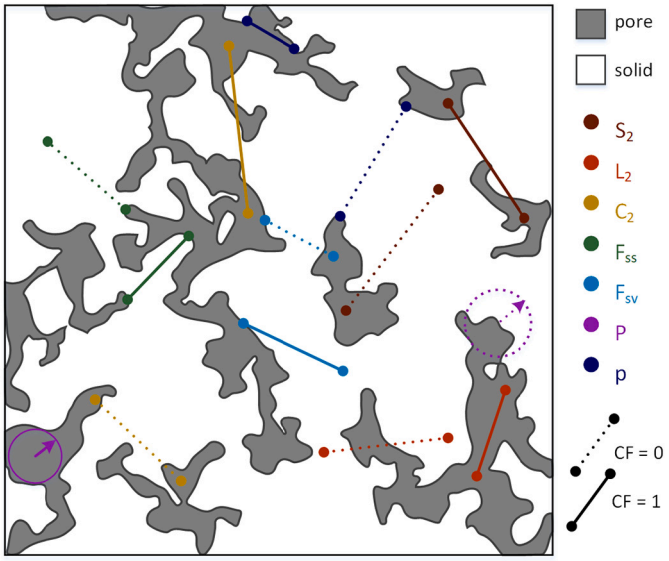


Fig. 1. A schematic representation of a binary porous medium (pores are shaded) with examples of CFs computations (all classical correlation functions are presented according to the legend on the right side). Computations take into account all possible configurations of a line segment (or a N-dimensional ball for pore size function) which either make a contribution to the CF (conventionally shown as "CF = 1") or do not make a contribution ("CF = 0").

umented with all computational algorithms explained, it should also leverage on recent advances in programming high-intensity computations and utilize both CPU and GPU power. Our answer in the form of `CorrelationFunctions.jl` package allows to compute numerous CFs easily on any operating system. The manuscript describing the package is organized as follows: in section 2 we introduce a reader to the most common and useful correlation functions, section 3.1 presents all algorithmic details for computing full CFs maps (map method, full 2-point statistics for a given image). In section 3.2 we discuss the second method to calculate correlation functions which is called the scan approach. In section 4 we verify our algorithms based on known analytical solutions. In section 5.1 some application examples are provided, including CFs evaluation for multi-phase materials. Section 5.2 deals with computational efficiency of our code, we compare computational times for 2D and 3D images of different sizes for scanning and map methods as evaluated using either CPU or GPU. Finally, sections 6 and 7 summarize all results and outline possible future improvements.

2. A brief primer on correlation functions

In this section we shall provide major properties and definitions of correlation functions supported by our package; the full list of functions is (see Fig. 1 for graphical explanation in addition to definitions below):

- Two-point probability function S_2,
- Lineal-path function L_2,
- Cluster function C_2,
- Surface-surface function F_{ss},
- Surface-void function F_{sv},
- Pore size function P,
- Chord length function p,
- Phase cross-correlation function ρ_{ij}.

Note that cross-correlation ρ_{ij} is actually not a classical CF, but technically is a variety of S_2 for the case when the end of line segment lies in different phases. This function is useful for multi-phase structures only, as for binary media two-point probability CFs are interdependent.

The most widely used is a two-point probability correlation function S_2, or autocorrelation. A definition for homogeneous media is as follows:

Definition 1. Two point function $S_2^{(i)}(\boldsymbol{r})$ equals to probability that both ends of a vector $\boldsymbol{r}$ lie in phase i when the vector is randomly thrown into the sample.

For isotropic media a vector $\boldsymbol{r}$ can be reduced to a scalar value r (i.e., a correlation function depends only on length of a vector and not on its direction). Mathematically this definition can be expressed with the following equation:

$$S_2^{(i)}(\boldsymbol{r}) = \langle I^{(i)}(\boldsymbol{x}) I^{(i)}(\boldsymbol{x}+\boldsymbol{r}) \rangle \tag{1}$$

where $I^{(i)}$ is an indicator function for a set of all points belonging to the phase i and $\langle \cdots \rangle$ is ensemble average.

A function closely related to S_2 is phase cross-correlation function ρ_{ij} which is defined as follows:

$$\rho_{ij}(\boldsymbol{r}) = \langle I^{(i)}(\boldsymbol{x}) I^{(j)}(\boldsymbol{x}+\boldsymbol{r}) \rangle \tag{2}$$

Another basic correlation function is a lineal path function L_2. It provides some non-trivial information (albeit usually not complete [52]) about the connectedness of the phase and is defined for homogeneous media as:

Definition 2. Lineal path function $L_2^{(i)}(\boldsymbol{r})$ equals to probability that a vector $\boldsymbol{r}$ lies wholly in phase i when randomly thrown into the sample.

If a sample is isotropic a vector $\boldsymbol{r}$ can be replaced with its length r and the definition becomes:

Definition 3. Lineal path function (for isotropic media) $L_2^{(i)}(r)$ equals to probability that a line segment of length r lies wholly in phase i when randomly thrown into the sample.

The function that contain a lot of structural connectivity information [53] is cluster function C_2.

Definition 4. Cluster function $C_2^{(i)}(\boldsymbol{r})$ equals to probability that both ends of a vector $\boldsymbol{r}$ lie in the same cluster when the vector is randomly thrown into the sample. A cluster is a set of points of the same phase i where any two points can be connected by a path lying entirely in that phase.

Note that for a fully connected phase C_2 will be equal to S_2, something we shall utilize later for computations.

The next two functions are called surface-surface (F_{ss}) and surface-void (F_{sv}) correlation functions and describe an interphase between phases. We skip strict verbal definitions and first introduce an interface indicator function instead:

$$M^{(i)}(\boldsymbol{r}) = |\nabla I^{(i)}(\boldsymbol{r})|$$

Here differentiation is understood in the sense of generalized functions, because an ordinary differential of I is either zero or undefined. For homogeneous media the definitions of surface functions then are:

$$F_{ss}^{(i)}(\boldsymbol{r}) = \langle M^{(i)}(\boldsymbol{x}) M^{(i)}(\boldsymbol{x}+\boldsymbol{r}) \rangle \tag{3}$$

$$F_{sv}^{(i)}(\boldsymbol{r}) = \langle M^{(i)}(\boldsymbol{x}) I^{(void)}(\boldsymbol{x}+\boldsymbol{r}) \rangle \tag{4}$$

These functions are defined for media with dimensionality of 2 and higher. F_{sv} is a physical quantity inversely proportional to length and F_{ss} is inversionaly proportional to surface area. As always, there are versions of these functions applicable to scalar argument for isotropic media.

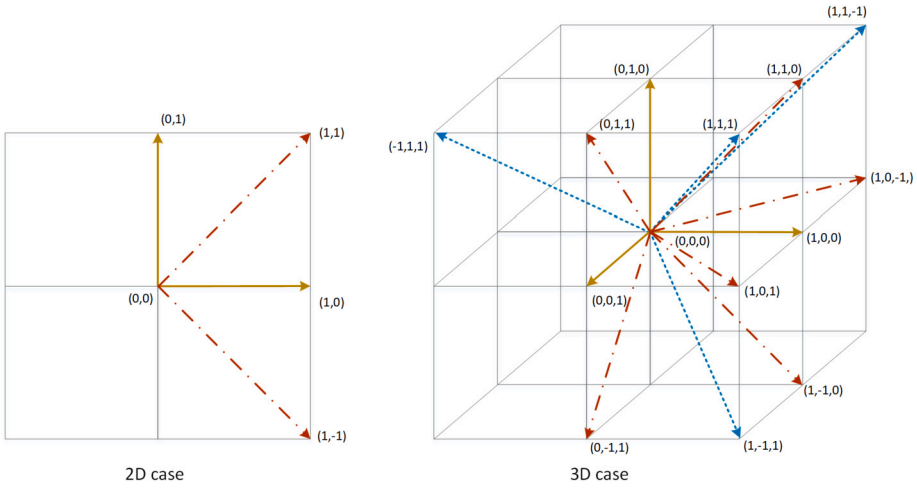


Fig. 2. Available directions to compute directional correlation functions in 2D and 3D cases with line scanning approach.

Pore size function $P(r)$ is a probability density function defined as follows:

Definition 5. $P(r)dr$ equals to the probability that a randomly chosen point in a set of points belonging to the void phase lies at a distance between r and $r+dr$ from the nearest point on the pore-solid interface.

Finally, a chord length function $p^{(i)}(r)$ is defined as:

Definition 6. $p^{(i)}(r)dr$ equals to the probability of finding a chord of length between r and $r+dr$ in phase i. Chord is a line segment which lies entirely in the same phase and touches the interface with its ends.

More detailed information on all classical CFs can be found in [1].

3. Computational methodology

`CorrelationFunctions.jl` is capable to compute all aforementioned correlation functions. Where possible we provide two implementations for each CF. The first implementation computes a correlation function in specific predefined directions (i.e., orientation of test vectors thrown into a sample is fixed) [54,55]. There are one predefined direction in 1D case, four directions in 2D case (two orthogonal and two diagonal) and thirteen directions in 3D case (three orthogonal, six diagonal within each orthogonal plane and four diagonal). All directions available are schematically represented in Fig. 2. The line segment scanning approach is slower for some CFs but requires much less memory resources. The full map approach establishes correlations between all points on the image and is faster. Both implementations are contained in `Directional` and `Map` modules of the package, correspondingly.

With the help of the package one can compute CFs in periodic and non-periodic modes. Periodic mode assumes that an input sample is periodically continued to the whole Euclidean space. Non-periodic mode treats the input as zero-padded image.

3.1. Full correlation map

Definition eq. (1) can be rewritten as follows:

$$S_2^{(i)}(r) = \frac{F^{-1}[|F[A^{(i)}](z)|^2](r)}{N(r)} \tag{5}$$

where $A^{(i)} \xleftarrow{I^{(i)}} A$ is a binary array obtained by element-wise application of indicator function $I^{(i)}$ to an input A, F is a discrete Fourier transform (DFT) operator and $N(r)$ is a total number of trials (vectors thrown into the sample). If we want to calculate S_2 function in periodic mode we need only to apply eq. (5) to an input to obtain the result. In non-periodic mode we need to pad an input with zeros to at least twice the size minus one for each dimension. In periodic mode a total number of trials is independent of r and equals to the number of elements in an input array. In non-periodic mode $N(r)$ is calculated as described in Appendix A. The following algorithm summarizes all the above:

1: **procedure** $S_2(A, phase, periodic)$
2: **if** $\neg periodic$ **then**
3: $A \leftarrow zeropad(A)$ ▷ Pad with at least $2 \cdot size(A,i) - 1$ zeros for each dimension i.
4: **end if**
5: $A^{(phase)} \leftarrow I^{(phase)}(A)$
6: $\hat{A}^{(phase)} \leftarrow F[A^{(phase)}]$
7: $\hat{S}_2 \leftarrow |\hat{A}^{(phase)}|^2/N$ ▷ N is an array with numbers of trials
8: **return** $F^{-1}[\hat{S}_2]$
9: **end procedure**

Algorithmic complexity of this implementation is $O(M \log M)$, where M is the number of elements in the input image.

Computation of S_2 is a special case of a cross-correlation function ρ_{ij} which computes a correlation between phases i and j using a formula:

$$\rho_{ij}(r) = \frac{F^{-1}[F[A^{(i)}](z)\overline{F[A^{(j)}](z)}](r)}{N(r)} \tag{6}$$

that algorithmically operates as follows:

1: **procedure** $\rho_{ij}(A, i, j, periodic)$
2: **if** $\neg periodic$ **then**
3: $A \leftarrow zeropad(A)$
4: **end if**
5: $A^{(i)} \leftarrow I^{(i)}(A)$
6: $A^{(j)} \leftarrow I^{(j)}(A)$
7: $\hat{A}^{(i)} \leftarrow F[A^{(i)}]$
8: $\hat{A}^{(j)} \leftarrow F[A^{(j)}]$
9: $\hat{\rho} \leftarrow \hat{A}^{(i)}\overline{\hat{A}^{(j)}}/N$
10: **return** $F^{-1}[\hat{\rho}]$
11: **end procedure**

Cluster function can be calculated with the help of previously described procedures. Firstly, we apply an indicator function to an input A: $A^{(i)} \xleftarrow{I^{(i)}} A$. Then we label clusters in $A^{(i)}$ using a function `label_components` which is a part of Julia `ImageMorphology.jl` library. Labeling algorithm must take into account if the function is calculated with periodic boundary conditions [56]. For periodic boundary conditions we implemented our own solution as part of the package, as it is absent in the public domain. Then C_2 function is a sum of S_2 functions calculated for all clusters separately. Algorithmic complexity

of this implementation is $O(MC\log M)$ where M is the number of elements in the input and C is the number of clusters. The algorithm is then written as:

procedure $C_2(A, phase, periodic)$
  $A^{(phase)} \leftarrow I^{(phase)}(A)$
  $Labels \leftarrow label\_components(A^{(phase)}, periodic)$
  $N \leftarrow maximum(Labels)$ ▷ Number of clusters
  return $\sum_{n=1}^{N} S_2(Labels, n, periodic)$
end procedure

For surface-surface and surface-void functions we use an edge detection filter [18] and then calculate either autocorrelation of the edge or cross-correlation of the edge and the void phase. The edge is extracted by convolving an input with a short high-pass filter H. It has a width of 7 pixels/voxels and all its coefficients with exception of the central coefficient are inversely proportional to distance from the center:

$$H_k = S\begin{cases} -\sum_{\substack{l \\ l\neq 0}} H_l & k = 0 \\ 1/\rho(k,0) & \text{otherwise} \end{cases} \qquad (7)$$
$$k \in \{-3,-2,\dots,3\}^D$$

where S equals to 30.45849 in 2D and 172.96232 in 3D case. The parameters of the filter were adjusted based on comparison against exact computations of surface functions on smooth-boundary sets [57].

An algorithm for F_{ss} function is the following:

procedure $F_{ss}(A, phase, periodic)$
  if $\neg periodic$ **then**
    $A \leftarrow zeropad(A)$
  end if
  $A^{(phase)} \leftarrow I^{(phase)}(A)$
  $A_H \leftarrow H * A^{(phase)}$
  $\hat{A}_H \leftarrow F[A_H]$
  $\hat{F}_{ss} \leftarrow |\hat{A}_H|^2/N$
  return $F^{-1}[\hat{F}_{ss}]$
end procedure

Using eq. (6) we obtain an algorithm for surface-void function:

procedure $F_{sv}(A, phase, periodic)$
  if $\neg periodic$ **then**
    $A \leftarrow zeropad(A)$
  end if
  $A^{(phase)} \leftarrow I^{(phase)}(A)$
  $A^{(void)} \leftarrow I^{(void)}(A)$
  $A_H \leftarrow H * A^{(phase)}$
  $\hat{A}_H \leftarrow F[A_H]$
  $\hat{A}^{(void)} \leftarrow F[A^{(void)}]$
  $\hat{F}_{sv} \leftarrow \hat{A}_H\overline{\hat{A}^{(void)}}/N$
  return $F^{-1}[\hat{F}_{sv}]$
end procedure

The interface between phases must not change rapidly for the edge filter to work correctly. Therefore, the algorithms for surface functions require input images to have appropriate resolution. A criterion for goodness can be expressed with C_α parameter. Let $\alpha \in [0,1]$ and $f(x)$ be (band-limited) input image ($x \in \mathbb{R}^2$ or $\mathbb{R}^3$). We define C_α as follows:

$$f_0(x) = f(x) - \langle f(x)\rangle$$
$$C_\alpha = \frac{\int_{-\alpha\omega}^{\alpha\omega} |\hat{f}_0(z)|^2 dz}{\int_{-\omega}^{\omega} |\hat{f}_0(z)|^2 dz}$$

where $\hat{f}_0$ is a Fourier transform of f_0 and ω is a folding frequency which depends only on resolution of the image. We have previously shown that criterion $C_{0.5} > 0.97$ holds then the image is suitable for robust F_{ss} and F_{sv} evaluation [18]. Note that function `lowfreq_energy_ratio` that computes $C_{0.5}$ parameter for any input binary image is the part of `CorrelationFunctions.jl` package.

Our library does not have an implementation for lineal-path and chord-length maps, although we know that such implementations exist [58,21]. The reason is a very slow computation even on GPUs or the resulting map is not for all points on the image, but within a spherical volume (due to rotation of the 3D image during lineal-path evaluation). Moreover, linear scan approach is orders of magnitude faster for L_2.

Majority of described algorithms contain parallelizable operations such as fast Fourier transform, cluster labeling and image filtering, hence they are perfectly suited for execution on GPUs and provide very fast computations (see execution times below).

3.2. Directional scanning with line segment

All algorithms described above (full correlation maps) are executed on the whole input image and hence require a lot of RAM memory. The scan approach deals with memory limitations at the cost of higher execution time (except for L_2). With line-segment scanning we select a list of predefined directions and cut all one-dimensional slices from the input along those directions. Then for each direction, as shown in Fig. 2, we use an appropriate algorithm in section 3.1 to calculate a correlation function just for that single slice. For this reason we do not repeat algorithms for S_2, ρ_{ij}, F_{ss} and F_{sv}. When all slices are processed we calculate the resulting directional or averaged CF using the formula:

$$F^{\boldsymbol{d}}(r) = \frac{\sum_i F_i^{\boldsymbol{d}}(r) N_i(r)}{\sum_i N_i(r)}$$

Here $F^{\boldsymbol{d}}(r)$ is a correlation function computed with respect to a direction $\boldsymbol{d}$, $F_i^{\boldsymbol{d}}(r)$ is a "partial" correlation function calculated for a i-th slice and $N_i(r)$ is a total number of trials for a correlation length r and slice i. Some algorithms require preprocessing stages before cutting the input in slices, e.g., image filtering for edge detection or cluster labeling performed on the whole array, not on slices.

An algorithm for computation of linear-path function for one-dimensional slice is presented below. Here A is a one-dimensional input array, *phase* can be void, solid or some other phase, and *periodic* is a boolean value which equals to true if we compute L_2 in periodic mode and false otherwise. The efficiency of the proposed algorithm is $O(n)$ where n is the length of the array.

procedure $L_2(A, phase, periodic)$
  $len \leftarrow length(A)$
  $result \leftarrow zeros(len)$
  $runs \leftarrow countruns(A, phase)$
  for all $run \in runs$ **do**
    $updateruns(result, run)$
  end for
  if $periodic$ **then**
    $first \leftarrow first(A)$
    $last \leftarrow last(A)$
    if $first = last$ **then**
      $updateperiodic(result, first, last)$
    end if
  end if
  return *success* divided by number of trials (see Appendix A)
end procedure

procedure COUNTRUNS($A, phase$)
  $runslist \leftarrow [\,]$
  $runs \leftarrow 0$
  for all $x \in A$ **do**
    if $x = phase$ **then**
      $runs \leftarrow runs + 1$
    else if $runs \neq 0$ **then** ▷ Add *runs* to the accumulator

```
            runslist ← runs : runslist
            runs ← 0
        end if
    end for
    if runs ≠ 0 then
        runslist ← runs : runslist
    end if
    return runslist
end procedure

procedure UPDATERUNS(A, runs)
    r ← runs
    for idx ← 0, runs − 1 do
        A[idx] ← A[idx] + r
        r ← r − 1
    end for
end procedure

procedure UPDATEPERIODIC(A, first, last)
    sum ← first + last
    for idx ← 0, sum − 1 do
        up ← min(idx, first, last, sum − idx)
        A[idx] ← A[idx] + up
    end for
end procedure
```

Cluster function can be calculated using the same technique as in map approach, but if the number of clusters is high, a naïve $O(n^2)$ algorithm may be preferred to $O(MC \log M)$ full map algorithm. Such an approach is present below:

```
procedure C_2(A, phase, periodic)
    A' ← I^(phase)(A)
    CC ← label_components(A', periodic)
    len ← length(A)
    result ← zeros(len)
    for all slice ∈ slices(CC) do
        for i = 0, len − 1 do
            end ← len − 1
            if periodic then
                end ← end + i
            end if
            for j = i, end do
                if slice[i] = slice[j mod len] ≠ 0 then   ▷ 0 is a label for background.
                    dst ← |(j mod len) − i|
                    result[dst] ← result[dst] + 1
                end if
            end for
        end for
    end for
    return result divided by number of trials
end procedure
```

Before computation of chord length function an input array must be pre-processed to find an interface for the phase of interest. For this purpose we use distance transform function defined as follows:

$$\mathcal{D}(\boldsymbol{x}) = \begin{cases} 0 & \boldsymbol{x} \in \text{solid phase} \\ \min\limits_{\boldsymbol{y} \in \text{solid phase}} \rho(\boldsymbol{x}, \boldsymbol{y}) & \text{otherwise} \end{cases} \tag{8}$$

where ρ is the Euclidean distance between two points. There are many algorithms for distance transform, including algorithms performed in linear time [59]. In `chord_length` function we firstly apply distance transform to an input to find the interface and then accumulate chord length in per-slice manner. Finally, chord lengths can be binned to a histogram.

```
procedure CHORD_LENGTH(A, phase)
    A' ← I^(phase)(A)
    A_E ← D(A') = 1 ▷ We consider elements for which the distance transform equals to 1 to be part of the interface
    chord_list ← [ ]
    for all (S', S_E) ∈ zip(slices(A'), slices(A_E)) do   ▷ iterate through all one-dimensional slices in A' and corresponding slices in A_E
        l ← 0
        maybe_chord ← False
        for idx ← 0, length(S') − 1 do
            if S_E[idx] then   ▷ Interface found
                if l > 1 and maybe_chord then   ▷ Chord found
                    chord_list ← l : chord_list
                end if
                l ← 0
                maybe_chord ← True
            else if ¬S'[idx] then   ▷ We are not in the phase of interest
                maybe_chord ← False
            end if
            l ← l + 1
        end for
    end for
    return chord_list or histogram(chord_list)
end procedure
```

This implementation also works in linear time.

Another function built on top of distance transform is the pore size function $P(r)$. Recall that pore size function is a probability distribution function. In practice, it is conveniently represented as a histogram of pore sizes found in the input image. Our algorithm for calculating $P(r)$ is as follows:

```
procedure P(A)
    D ← D(A)   ▷ Apply D (eq. (8)) to each element of A
    D' = {x ∈ D : x ≠ 0}   ▷ Remove zeros
    Bin values in D' to histogram H
    return H   ▷ Alternatively just return D'
end procedure
```

Algorithmic efficiency is $O(n)$ where n is the number of elements in the input array.

4. Verification of CFs computations

To verify all algorithms we compare their results against known closed-form expressions for correlation functions for some particular type of input data. One such type with analytical solution is overlapping n-balls with fixed radius and centers generated by Poisson point process. Here we consider both two-dimensional and three-dimensional cases.

4.1. Two-dimensional analytical solutions

Consider an infinite set of disks with radii R and centers generated by Poisson point process with parameter λ.

Two-point probability function in this case is [1]:

$$S_2(r) = e^{-2\lambda \begin{cases} \pi R^2 & r > 2R \\ \pi R^2 + r\sqrt{A}/4 - BR^2 & \text{otherwise} \end{cases}}$$

Here and later $A = 4R^2 - r^2$ and $B = \arccos \frac{r}{2R}$.

An expression for F_{ss} is [18]:

$$F_{ss}(r) = 4S_2(r) \begin{cases} (\pi\lambda R)^2 & r > 2R \\ \frac{AR^2\lambda^2 r(B^2 - 2\pi B + \pi^2) + \sqrt{A}R^2\lambda}{Ar} & \text{otherwise} \end{cases}$$

Surface-void function F_{sv} becomes [18]:

$$F_{sv}(r) = 2R\lambda S_2(r) \begin{cases} \pi & r > 2R \\ \pi - B & \text{otherwise} \end{cases}$$

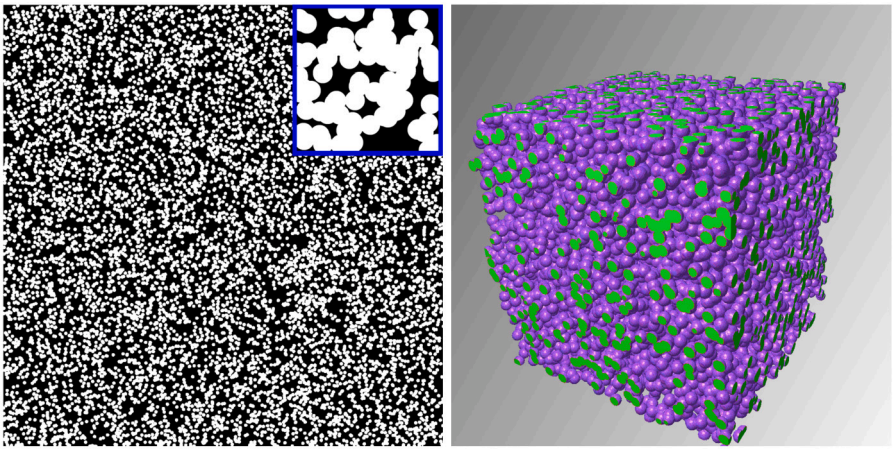


Fig. 3. Images of overlapping disks and balls used for testing of our package.

Two-point function L_2 [1]:

$$L_2(r) = e^{-\lambda(\pi R^2 + 2rR)}$$

Chord length function p [1]:

$$p(r) = 2\lambda R e^{-2\lambda r R}$$

And finally pore size function P [1]:

$$P(r) = 2\lambda\pi(r+R)e^{-\lambda\pi(r^2+2rR)}$$

4.2. Three-dimensional analytical solutions

As in the previous case, consider an infinite set of balls with radii R and centers generated by Poisson point process with parameter λ. Closed-form expressions for correlations functions are also known for this case for all CFs [1].

Two-point probability function in this case is:

$$S_2(r) = e^{-\lambda\pi\begin{cases} \frac{8}{3}R^3 & r > 2R \\ \frac{4}{3}R^3 + rR^2 - \frac{1}{12}r^3 & \text{otherwise} \end{cases}}$$

Let $\eta = \frac{4}{3}\lambda\pi R^3$. Then an expression for F_{ss} is:

$$F_{ss}(r) = S_2(r)\begin{cases} \frac{9\eta^2}{R^2} & r > 2R \\ \frac{9\eta^2}{R^2}(\frac{1}{2} + \frac{r}{4R})^2 + \frac{3\eta}{2rR} & \text{otherwise} \end{cases}$$

Surface-void function F_{sv} is:

$$F_{sv}(r) = \frac{3\eta}{R}S_2(r)\begin{cases} 1 & r > 2R \\ \frac{1}{2} + \frac{r}{4R} & \text{otherwise} \end{cases}$$

Two-point function L_2:

$$L_2(r) = e^{-\lambda\pi(\frac{4}{3}R^3 + rR^2)}$$

Chord length function p:

$$p(r) = \pi\lambda R^2 e^{-\pi\lambda r R^2}$$

And finally pore size function P:

$$P(r) = 4\pi\lambda(r+R)^2 e^{-\frac{4}{3}\pi\lambda(r^3+3r^2R+3rR^2)}$$

4.3. Comparison against analytical solutions

To test out package we generate a realization of random overlapping disks (Fig. 3) with dimensions 10000×10000 and parameters of radii $R = 40\ pixels$ and $\lambda = 10^{-4}\ pixels^{-2}$ and then compare calculated correlation functions with their theoretical values.

For a 3D test, we generate a realization of random overlapping spheres (Fig. 3) with dimensions $500 \times 500 \times 500$ and $R = 10\ voxels$ and $\lambda = 10^{-4}\ voxels^{-3}$ and then compare calculated correlation functions with analytical solutions.

The results of comparison of computational results against analytical solutions are present on Fig. 4. It is clear from the test shown that CFs are computed with high accuracy; minor deviations are due to finite size of our Poisson realizations (i.e., larger images reduce deviations [18]). The package contains automatic tests based on these comparisons.

5. Applications and benchmarks

After the verification of our computational approach based on analytical solution, in this section we apply `CorrelationFunctions.jl` to compute CFs for a variety of real 2D (SEM) and 3D (XCT) images. We also report benchmarks on execution time and compare the efficiency of GPU and CPU implementations.

5.1. Examples of application to experimental images

For testing purposes we have gathered an assortment of different porous materials including XCT images of artificial ceramics [60] and natural soil composite [44], segmented into numerous phases, and SEM images of oil-bearing rocks such as sandstones and carbonate [18]. A summary of this library of 2D and 3D images, including dimensions and resolutions, can be found in Table 1.

At first, we compare the three ceramic images (Fig. 5, at the top) which have somewhat different genesis depending on the mixture ratio of powder and binder used for firing the material [60]. As observed from Fig. 6, the structure of all three samples is visually very similar. The lineal path function supports this observation. The S_2 functions are also quite similar except for Sample 2 having higher porosity that results in elevated curve (the tail for two-point probability as expected fluctuates around porosity squared). Notably, S_2 and C_2 are almost identical – this indicates that pore phase represents a single large cluster. Recall that for S_2, C_2 and L_2 functions the value at correlation length of 0 is the porosity value. Surface functions, on the other hand, feel significant difference between samples in how pore-solid interface is organized. This is further evidenced on pore-size and chord-length graphs, with Ceramic 2 sample with the highest porosity having larger amount of large pores. Both P and p show very similar behavior, but for different correlation lengths; such behavior is attributed to quite regular pores shapes in all ceramics samples – this is evident from much larger differences between pore-size and chord-length for SEM images considered next.

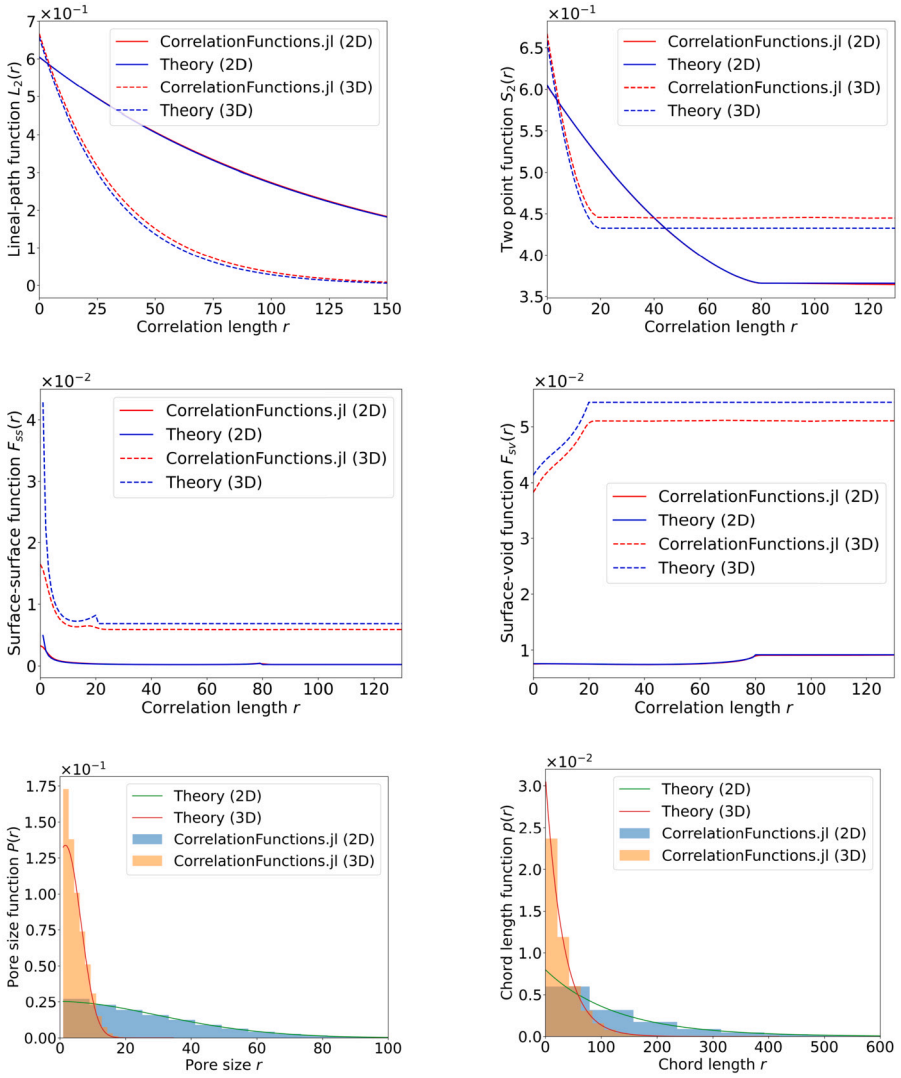
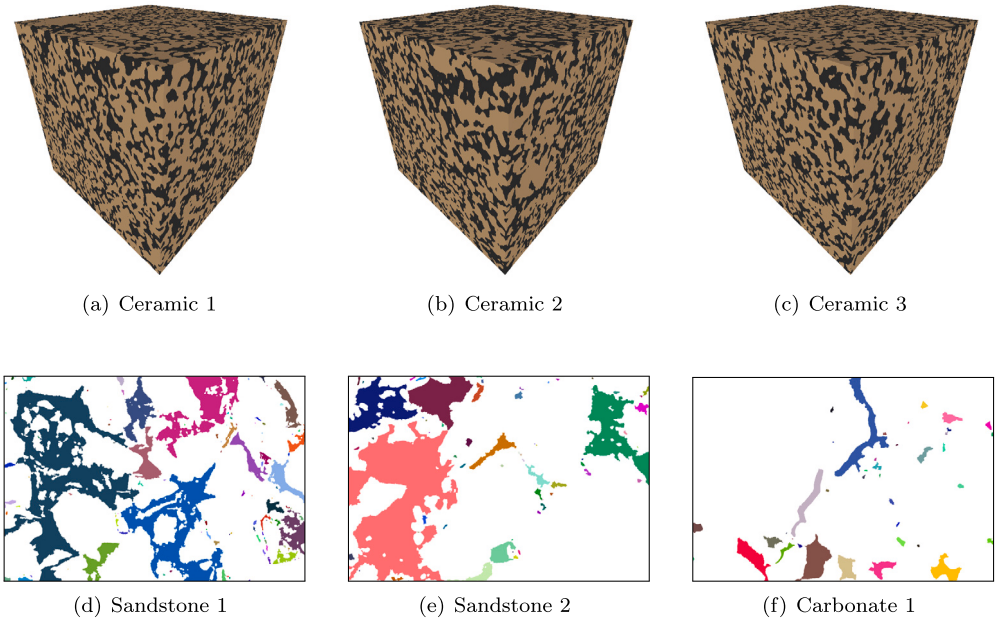


Fig. 4. A comparison of calculated and theoretical values of correlation functions for overlapping disks and balls.

Table 1
Summary of samples used for calculation of correlation functions.

Sample name	Image type	Dimensions (pixels)	Image resolution (μm)	$C_{0.5}$
Ceramic 1	XCT	$500 \times 500 \times 500$	2.25	0.9823
Ceramic 2	XCT	$500 \times 500 \times 500$	2.25	0.9890
Ceramic 3	XCT	$500 \times 500 \times 500$	2.25	0.9712
Sandstone 1	SEM	1280×869	0.10	0.9703
Sandstone 2	SEM	1280×869	0.05	0.9897
Carbonate 1	SEM	1024×691	0.06	0.9721
Soil (multiphase sample)	XCT	$304 \times 304 \times 304$	15.0	-

The Fig. 5 (at the bottom) presents three 2D images of sandstones and carbonate with computed correlation functions at Fig. 7. In this example, unlike for quite isotropic ceramics, we evaluated CFs in the directional manner. The S_2 graph immediately provides us with porosity values. It also highlights severe anisotropy in sandstone SEM images. For 2D images numerous patches of pores are not connected to each other and, thus, S_2 and C_2 are very different. Cluster functions show how connectivity decays with length, also we can easily observe if given pore structure percolates in a given direction - none of the images percolates (we highlighted the clusters to make this statement easily verifiable with the naked eye). Note that C_2 and L_2 were computed with correlations lengths equal to image dimensions to explore connectivity – for other CFs this leads to noised curves due to smaller number of sampled points (for example, see correlation functions' graphs in [61]). Lineal paths are close for sandstones (with lower L_2 values for Sandstone 1 where large pores are occluded with clays or grain fragments), but for carbonate with much smaller apertures of pores it is way below L_2 for sandstones. Also, L_2 suggests that computations with correlations lengths beyond where lineal path goes to zero are not very informative. For very irregular pores on 2D SEM images, P and p functions look much less similar compared to the ceramics case on Fig. 6. For 2D images it is also visually easy to analyze full CF maps, as presented on Fig. 8 for Sandstone 1 sample. Full correlation maps show expanded version of directional functions – asymmetry is immediately evidenced from S_2, F_{ss} and F_{sv} maps. Moreover, the map for C_2 shows higher connectivity in y-direction as compared to x-axis. The interpre-



(a) Ceramic 1 (b) Ceramic 2 (c) Ceramic 3

(d) Sandstone 1 (e) Sandstone 2 (f) Carbonate 1

Fig. 5. Visualization of XCT (top) and SEM (bottom) images used for computation of correlation functions. Different clusters of the void phase on SEM samples are shown in color. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

tation of full correlation maps is less straightforward, but anisotropy patterns in directions different from those presented in Fig. 2 can be spotted.

Finally, we conclude this section with the example of multi-phase soil composite for which we computed cross-correlation functions Fig. 9. The graph on the right side of the figure shows cross-correlations between different pairs of phases (normalized by each phase's ratio). One can easily conclude that organo-mineral complex (phase 2) is mainly located on pore (phase 1) walls. Heavy minerals (phase 4) do not cross-correlate with pores, as they mainly lie within the mineral solid phase (phase 3) – they do show high cross-correlations. Other CFs can be easily computed for each separate phase, while F_{su} can potentially be extended beyond void (or pores) and be represented by any other phase or their superposition.

This section clearly demonstrated, that `CorrelationFunctions.jl` can be utilized in analysis of any 2D and 3D images, including structures consisting of numerous phases.

5.2. Execution times for our algorithms

To measure execution times for developed computational algorithms we use a machine with the following configuration:

- **CPU**: AMD Ryzen 7 5800X
- **Memory**: 32 GB DDR4 memory
- **GPU**: Nvidia Tesla V100 with 32 GB video memory

We generate 2 and 3-dimensional square/cubical binary arrays with different width (see insets on Fig. 10). An array used for measurements contains two solid disks or balls surrounded by void phase. This configuration is a compromise between ease of image scaling and complexity of the structure (e.g., full map C_2 computational time is not compromised by too many clusters).

We generated six arrays with dimensions of $s = 1000, 2000, \ldots, 6000$ pixels for two-dimensional case and six arrays with a side of a cube $s = 50, 100, \ldots, 300$ voxels for three-dimensional case and calculated execution time for each correlation function. All functions are calculated using periodic boundary conditions. Functions which use per-slice directional approach were calculated along major axial directions only.

Functions which calculate full correlation maps are suitable for running on both CPU and GPU, so we provide execution times for both. The cluster function uses an algorithm for cluster labeling which has CPU-only implementation (thus, leaving a room for future improvements). Directional computations are performed exclusively on CPUs.

All measurements of execution time are presented on Fig. 10. It is immediately apparent that GPU versions (where available) are much faster than CPU versions and limited only by amount of video memory. Functions in module `Directional` (which utilize per-slice directional approach) are the slowest among other implementations, but require less memory.

6. Discussion and outlook

In the light of the end of the previous section it is important to clarify that available (video) RAM or speed of computations are not the only considerations behind choosing between full correlation map and directional linear scans. Directional CFs are extremely useful in describing or reconstructing anisotropic structures [54,55,32]. At the same time, they also much easier to work with and good candidates for data compression and feature extraction [26]. On the other hand, full map for S_2 allows computing ensemble averaged two-point probability – in the similar fashion it is measured with scattering experiments. The usefulness of full correlation maps for other CFs is questionable, as they are either non-measurable experimentally, e.g., [62], or non-applicable for stochastic reconstructions. The latter refers to our ability to reconstruct almost any structure from its S_2 map [63]. And while we have fast and efficient techniques such as phase-recovery to reconstruct the structure from its S_2 correlation map, the utilization of other CFs is complicated [64,51]. Simulated-annealing based methods that operate on the full map are, to the best of our knowledge, were not developed to date. This is the reason we did not invest into improvement on existing approaches to compute maps for L_2.

The trade-off in using full map or directional CFs is closely related to the information content of correlation functions [65,49,50]. Any reduction of the full map into ensemble averaged or directionally computed CFs significantly decreases the information available. On the other hand, full correlation map is larger than the image itself and, thus, storing such huge amount of data for already voluminous datasets is not practical. Vectorization of directional correlation functions is, thus,

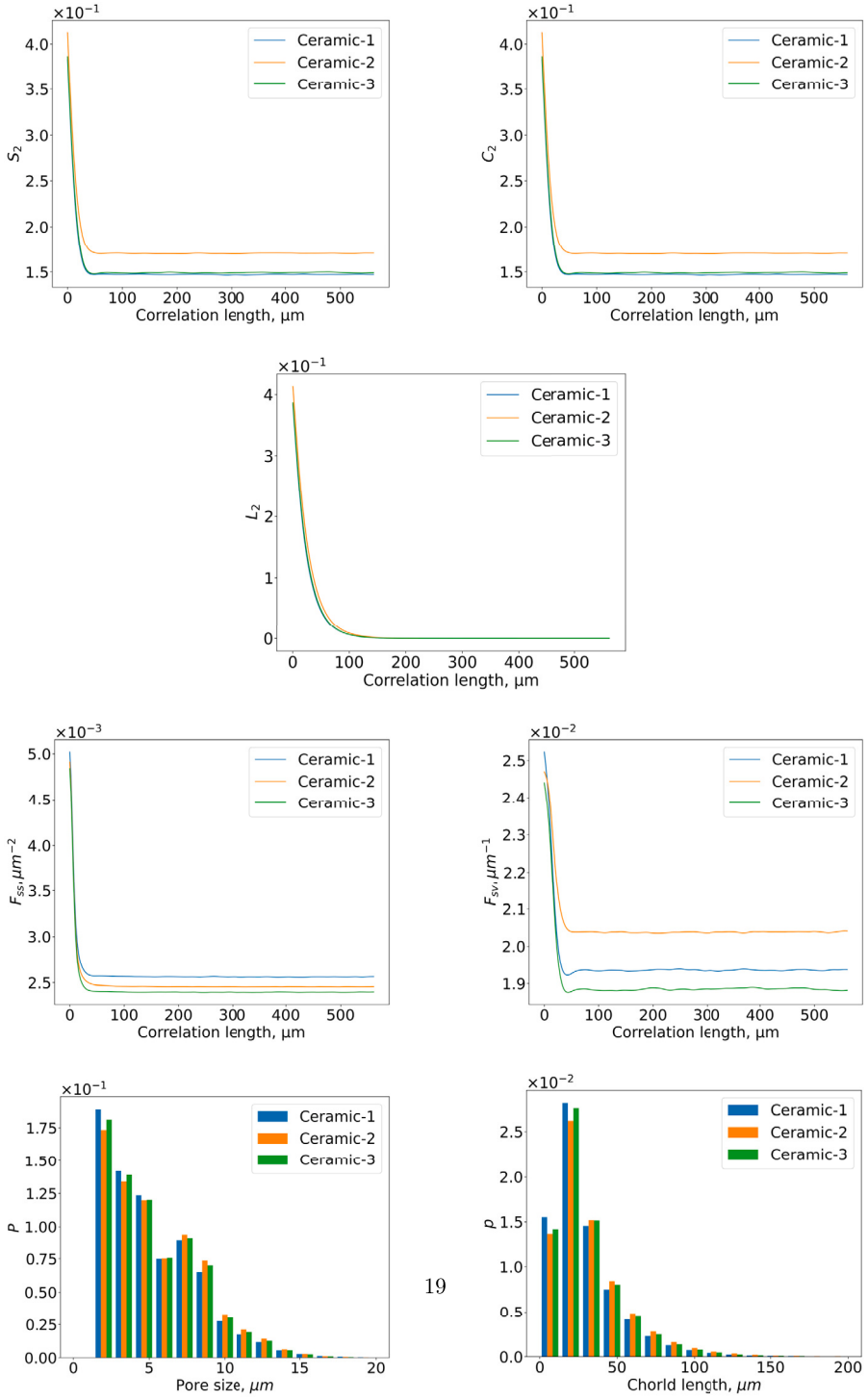


Fig. 6. Plots of correlation functions for XCT samples on Fig. 5.

seems like a natural feature reduction approach while working with correlation maps. As applied to stochastic reconstructions, similar logic applies - it seems to be beneficial to increase the number of functions rather than their "volume". Full correlation maps are usually not available unless the structure is already known; but they may be useful in the form of 2D maps available from images for 3D reconstructions [64]. Whether reconstructions of, say, full 3D maps based on 2D maps are a better approach than using directional CFs computed from images for stochastic reconstructions is an open question. Simulated annealing based methods were developed in the first place due to absence of methods like phase-recovery for correlation functions other than autocorrelation S_2. We believe that our fast and robust computational framework in the form of `CorrelationFunctions.jl` package will facilitate research in these directions.

While the toolbox of presented package and its API are solid, based on additional research numerous areas for future improvements of `CorrelationFunctions.jl` are possible:

1. include novel non-classical correlation functions, e.g., describing topology of phases;

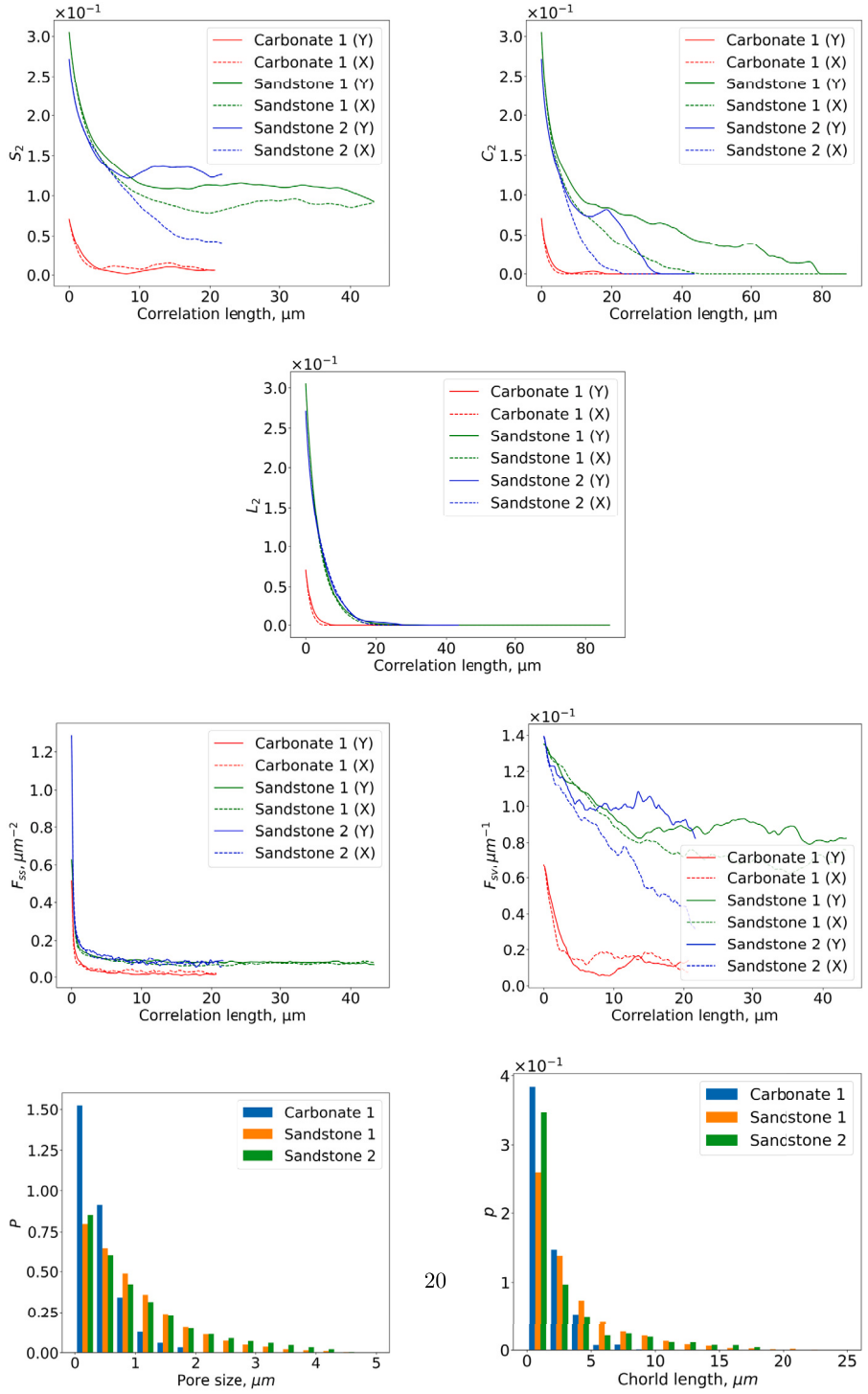


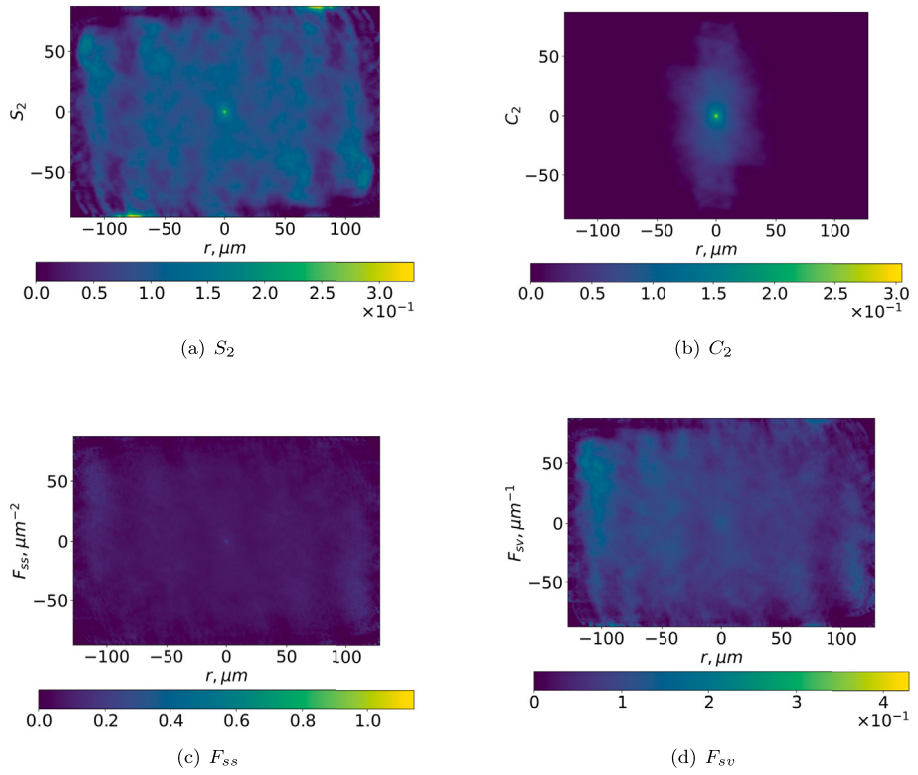
Fig. 7. Plots of correlation functions for SEM samples on Fig. 5.

2. implementing sub-cube based computations of CFs with a given correlation functions with later averaging for larger volume - with such domain decomposition one could compute CFs for larger samples with limited GPU resources;
3. implementing GPU-based clustering (cluster labeling) and distance transform to further speed up computations for C_2, chord-length and pore-size functions;
4. implementing higher order functions, such as 3-point probabilities or even 3-point cluster and other more involved CFs;
5. increase number of directions for scan approach or implementing calculations within a given solid angle.

We perform steps forward in some directions, whilst others still require additional research to be made to facilitate their implementation within the package.

7. Summary

In this paper we introduced `CorrelationFunctions.jl` open-source package developed in Julia and capable of computing all classical correlation functions based on 2D and 3D input images. Our code is capable of evaluating two-point probability, cross-correlation, cluster, lineal-path, surface-surface, surface-void, pore-size and chord-



(a) S_2 (b) C_2 (c) F_{ss} (d) F_{sv}

Fig. 8. Correlation maps for Sandstone 1.

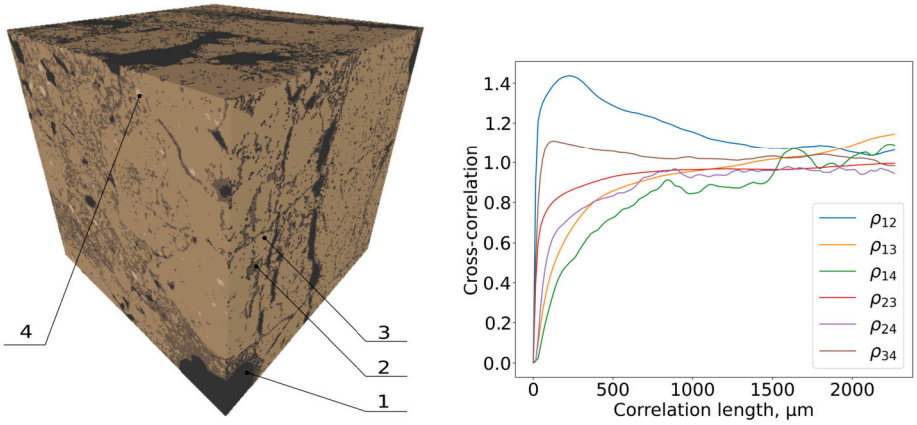
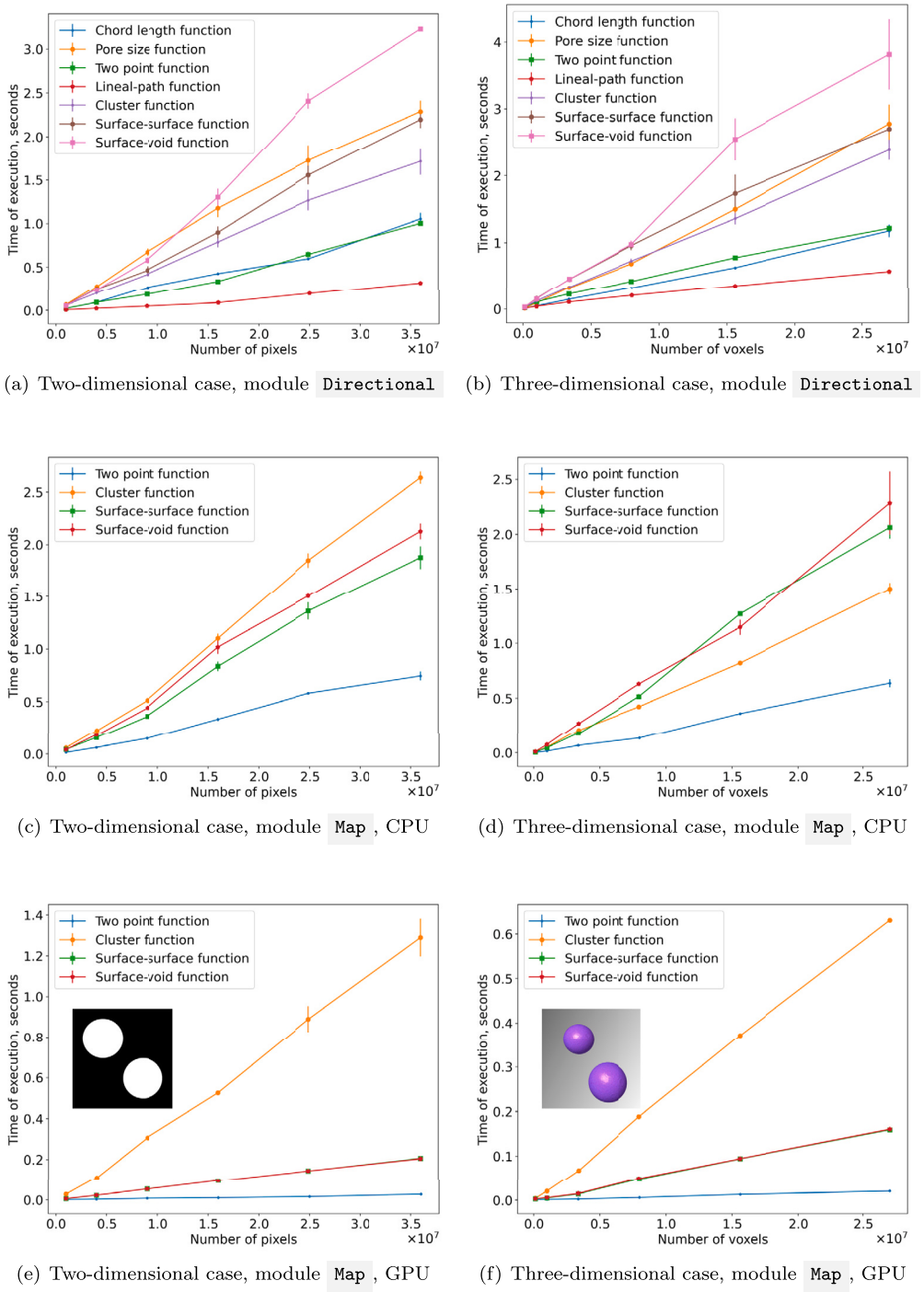


Fig. 9. Cross-correlation functions for multiphase sample (soil) which has 4 distinct phases, labeled as 1-4. Each function ρ_{ij} is normalized by division by $\rho_{ii}(0)\rho_{jj}(0)$.

length functions on both CPU and GPU architectures. There possible, we presented two types of computations: full correlation map (correlations of each point with other points on the image, that also allows obtaining ensemble averaged CF) and directional correlation functions (currently in major orthogonal and diagonal directions). Such an implementation allowed for the first time to assemble a completely free solution to evaluate correlation functions under any operating system with well documented API. Our package includes automatic tests against analytical solutions that are described in this paper. We measured execution times for all CPU and GPU implementations and as a rule of thumb full correlation maps are faster than directional scans because calculation of correlation maps applies fast Fourier transform to the whole input array while, when calculating scans, FFT needs to be applied to each scan separately. Calculation on GPU shows performance increase when compared to CPU, as expected. However, full maps require more RAM and, thus, are limited to available RAM resources. On the other hand, directional CFs are memory efficient and can be evaluated for huge datasets – this way they are the first candidates for structural data compression of feature extraction. The package itself is available through Julia package ecosystem and on GitHub (https://github.com/fatimp/CorrelationFunctions.jl), the latter source also contains documentation and additional helpful resources such as tutorials. We believe that a single powerful computational tool such as `CorrelationFunctions.jl` will significantly facilitate the usage of correlation functions in numerous areas of structural description and research, as well as in machine learning applications.

CRediT authorship contribution statement

Vasily Postnicov: Formal analysis, Software, Writing – original draft. **Aleksei Samarin:** Formal analysis, Methodology. **Marina V. Karsanina:** Formal analysis, Methodology, Visualization. **Mathieu Gravey:** Formal analysis, Methodology. **Aleksey Khlyupin:** Formal analysis, Methodology, Supervision, Writing – review & editing. **Kir-**



(a) Two-dimensional case, module `Directional` (b) Three-dimensional case, module `Directional`

(c) Two-dimensional case, module `Map`, CPU (d) Three-dimensional case, module `Map`, CPU

(e) Two-dimensional case, module `Map`, GPU (f) Three-dimensional case, module `Map`, GPU

Fig. 10. Execution times for different correlation functions.

ill M. Gerke: Conceptualization, Methodology, Supervision, Writing – original draft, Writing – review & editing.

Declaration of competing 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.

Data availability

All data can be found in the GitHub repository.

Declaration of generative AI and AI-assisted technologies in the writing process

No AI services were used in preparation of this manuscript.

Acknowledgements

This research was supported by the Russian Science Foundation grant 23-74-00061.

Collaborative effort of the authors within the FaT iMP (Flow and Transport in Media with Pores) research group (www.porenetwork.com) and used some of its software. We thank our colleague Konstantin Romanenko for suggestions and administrative work.

Appendix A. Computation of number of trials for non-periodic mode

Here we assume that an input is padded with zeros to the length $2s_i - 1$ across each dimension i where s_i is the size of the non-padded input in that respective dimension. For each dimension we calculate a sequence:

$$N_j^i = \begin{cases} s_i - j & j \in \overline{0, s_i - 1} \\ j - s_i + 1 & j \in \overline{s_i, 2s_i - 2} \end{cases}$$

When a number of trials is calculated as follows:

$$N_{i_1,i_2,\ldots,i_n} = \prod_{k=1}^{n} N_{i_k}^k$$

Here n is a number of dimensions and i_k is an index into the output array for k-th dimension.

References

[1] S. Torquato, Random Heterogeneous Materials: Microstructure and Macroscopic Properties, Springer-Verlag, New York, 2002.
[2] M. Sahimi, Heterogeneous Materials I: Linear Transport Properties and Optical Properties, Springer-Verlag, New York, 2003.
[3] H. Moussaoui, J. Laurencin, Y. Gavet, G. Delette, M. Hubert, P. Cloetens, T. Le Bihan, J. Debayle, Stochastic geometrical modeling of solid oxide cells electrodes validated on 3d reconstructions, Comput. Mater. Sci. 143 (2018) 262–276.
[4] M. Neumann, M. Osenberg, A. Hilger, D. Franzen, T. Turek, I. Manke, V. Schmidt, On a pluri-Gaussian model for three-phase microstructures, with applications to 3d image data of gas-diffusion electrodes, Comput. Mater. Sci. 156 (2019) 325–331.
[5] K.M. Gerke, E.V. Korostilev, K.A. Romanenko, M.V. Karsanina, Going submicron in the precise analysis of soil structure: a fib-sem imaging study at nanoscale, Geoderma 383 (2021) 114739.
[6] J. Baruchel, J.Y. Buffiere, E. Maire, X-ray Tomography in Material Science, Hermes Science Publications, Paris (France), 2000.
[7] S. Youssef, E. Maire, R. Gaertner, Finite element modelling of the actual structure of cellular materials determined by X-ray tomography, Acta Mater. 53 (3) (2005) 719–730.
[8] X. Miao, K.M. Gerke, T.O. Sizonenko, A new way to parameterize hydraulic conductances of pore elements: a step towards creating pore-networks without pore shape simplifications, Adv. Water Resour. 105 (2017) 162–172.
[9] K.M. Gerke, R.V. Vasilyev, S. Khirevich, D. Collins, M.V. Karsanina, T.O. Sizonenko, D.V. Korost, S. Lamontagne, D. Mallants, Finite-difference method Stokes solver (fdmss) for 3d pore geometries: software development, validation and case studies, Comput. Geosci. 114 (2018) 41–58.
[10] J.D. Eshelby, The determination of the elastic field of an ellipsoidal inclusion, and related problems, Proc. R. Soc. Lond. Ser. A, Math. Phys. Sci. 241 (1226) (1957) 376–396.
[11] J.G. Berryman, S.C. Blair, Use of digital image analysis to estimate fluid permeability of porous materials: application of two-point correlation functions, J. Appl. Phys. 60 (6) (1986) 1930–1938.
[12] A. Różański, J. Rainer, D. Stefaniuk, I. Sevostianov, D. Łydżba, Identification of 'replacement' microstructure for porous medium from thermal conductivity measurements: problem formulation and numerical solution, Int. J. Eng. Sci. 182 (2023) 103788.
[13] P. Debye, H. Anderson Jr, H. Brumberger, Scattering by an inhomogeneous solid. II. The correlation function and its application, J. Appl. Phys. 28 (6) (1957) 679–683.
[14] H. Li, S. Singh, N. Chawla, Y. Jiao, Direct extraction of spatial correlation functions from limited X-ray tomography data for microstructural quantification, Mater. Charact. 140 (2018) 265–274.
[15] J.G. Berryman, Measurement of spatial correlation functions using image processing techniques, J. Appl. Phys. 57 (7) (1985) 2374–2384.
[16] Z. Ma, S. Torquato, Precise algorithms to compute surface correlation functions of two-phase heterogeneous media and their applications, Phys. Rev. E 98 (1) (2018) 013307.
[17] K.M. Gerke, M.V. Karsanina, D. Mallants, Universal stochastic multiscale image fusion: an example application for shale rock, Sci. Rep. 5 (1) (2015) 15880.
[18] A. Samarin, V. Postnicov, M.V. Karsanina, E.V. Lavrukhin, D. Gafurova, N.M. Evstigneev, A. Khlyupin, K.M. Gerke, Robust surface-correlation-function evaluation from experimental discrete digital images, Phys. Rev. E 107 (6) (2023) 065306.
[19] E.V. Lavrukhin, K.M. Gerke, K.A. Romanenko, K.N. Abrosimov, M.V. Karsanina, Assessing the fidelity of neural network-based segmentation of soil xct images based on pore-scale modelling of saturated flow properties, Soil Tillage Res. 209 (2021) 104942.
[20] A. Hasanabadi, M. Baniassadi, K. Abrinia, M. Safdari, H. Garmestani, 3d microstructural reconstruction of heterogeneous materials from 2d cross sections: a modified phase-recovery algorithm, Comput. Mater. Sci. 111 (2016) 107–115.
[21] J. Havelka, A. Kučerová, J. Sỳkora, Compression and reconstruction of random microstructures using accelerated lineal path function, Comput. Mater. Sci. 122 (2016) 102–117.
[22] J. Feng, Q. Teng, X. He, L. Qing, Y. Li, Reconstruction of three-dimensional heterogeneous media from a single two-dimensional section via co-occurrence correlation function, Comput. Mater. Sci. 144 (2018) 181–192.
[23] P.-E. Chen, R. Raghavan, Y. Zheng, H. Li, K. Ankit, Y. Jiao, Quantifying microstructural evolution via time-dependent reduced-dimension metrics based on hierarchical n-point polytope functions, Phys. Rev. E 105 (2) (2022) 025306.
[24] R. Ledesma-Alonso, R. Barbosa, J. Ortegón, Effect of the image resolution on the statistical descriptors of heterogeneous media, Phys. Rev. E 97 (2) (2018) 023304.
[25] M.V. Karsanina, K.M. Gerke, E.B. Skvortsova, D. Mallants, Universal spatial correlation functions for describing and reconstructing soil microstructure, PLoS ONE 10 (5) (2015) e0126515.
[26] M.V. Karsanina, E.V. Lavrukhin, D.S. Fomin, A.V. Yudina, K.N. Abrosimov, K.M. Gerke, Compressing soil structural information into parameterized correlation functions, Eur. J. Soil Sci. 72 (2) (2021) 561–577.
[27] M. Takada, B. Jain, The three-point correlation function in cosmology, Mon. Not. R. Astron. Soc. 340 (2) (2003) 580–608.
[28] A. Derossi, K.M. Gerke, M.V. Karsanina, B. Nicolai, P. Verboven, C. Severini, Mimicking 3d food microstructure using limited statistical information from 2d cross-sectional image, J. Food Eng. 241 (2019) 116–126.
[29] S.L. Veatch, B.B. Machta, S.A. Shelby, E.N. Chiang, D.A. Holowka, B.A. Baird, Correlation functions quantify super-resolution images and estimate apparent clustering due to over-counting, PLoS ONE 7 (2) (2012) e31457.
[30] P. Čapek, V. Hejtmánek, L. Brabec, A. Zikánová, M. Kočiřík, Stochastic reconstruction of particulate media using simulated annealing: improving pore connectivity, Transp. Porous Media 76 (2) (2009) 179–198.
[31] J.-F. Thovert, P. Adler, Grain reconstruction of porous media: application to a Bentheim sandstone, Phys. Rev. E 83 (5) (2011) 056116.
[32] K.M. Gerke, M.V. Karsanina, R. Katsman, Calculation of tensorial flow properties on pore level: exploring the influence of boundary conditions on the permeability of three-dimensional stochastic reconstructions, Phys. Rev. E 100 (5) (2019) 053312.
[33] P. Adler, C.G. Jacquin, J. Quiblier, Flow in simulated porous media, Int. J. Multiph. Flow 16 (4) (1990) 691–712.
[34] C. Yeong, S. Torquato, Reconstructing random media, Phys. Rev. E 57 (1) (1998) 495.
[35] P. Tahmasebi, M. Sahimi, Cross-correlation function for accurate reconstruction of heterogeneous media, Phys. Rev. Lett. 110 (7) (2013) 078002.
[36] M.V. Karsanina, K.M. Gerke, Hierarchical optimization: fast and robust multiscale stochastic reconstructions with rescaled correlation functions, Phys. Rev. Lett. 121 (26) (2018) 265501.
[37] K.M. Gerke, M.V. Karsanina, Improving stochastic reconstructions by weighting correlation functions in an objective function, Europhys. Lett. 111 (5) (2015) 56002.
[38] S. Chen, H. Li, Y. Jiao, Dynamic reconstruction of heterogeneous materials and microstructure evolution, Phys. Rev. E 92 (2) (2015) 023301.
[39] Y. Xu, P.-E. Chen, H. Li, W. Xu, Y. Ren, W. Shan, Y. Jiao, Correlation-function-based microstructure design of alloy-polymer composites for dynamic dry adhesion tuning in soft gripping, J. Appl. Phys. 131 (11) (2022) 115104.
[40] D.S. Fomin, A.V. Yudina, K.A. Romanenko, K.N. Abrosimov, M.V. Karsanina, K.M. Gerke, Soil pore structure dynamics under steady-state wetting-drying cycle, Geoderma 432 (2023) 116401.
[41] G. Pilania, J.E. Gubernatis, T. Lookman, Multi-fidelity machine learning models for accurate bandgap predictions of solids, Comput. Mater. Sci. 129 (2017) 156–163.
[42] S. Kamrava, P. Tahmasebi, M. Sahimi, Linking morphology of porous media to their macroscopic permeability by deep learning, Transp. Porous Media 131 (2) (2020) 427–448.
[43] M. Röding, Z. Ma, S. Torquato, Predicting permeability via statistical learning on higher-order microstructural information, Sci. Rep. 10 (1) (2020) 1–17.
[44] M.V. Karsanina, K.M. Gerke, E.B. Skvortsova, A.L. Ivanov, D. Mallants, Enhancing image resolution of soils by stochastic multiscale image fusion, Geoderma 314 (2018) 138–145.
[45] D. Zhang, R. Zhang, S. Chen, W.E. Soll, Pore scale study of flow in porous media: scale dependency, rev, and statistical rev, Geophys. Res. Lett. 27 (8) (2000) 1195–1198.
[46] K.M. Gerke, M.V. Karsanina, How pore structure non-stationarity compromises flow properties representativity (rev) for soil samples: pore-scale modelling and stationarity analysis, Eur. J. Soil Sci. 72 (2) (2021) 527–545.
[47] E.V. Lavrukhin, M.V. Karsanina, Measuring structural nonstationarity: The use of imaging information to quantify homogeneity and inhomogeneity, Phys. Rev. E 108 (6) (2021) 064128.
[48] J. Yao, P. Frykman, F. Kalaydjian, J. Thovert, P. Adler, High-order moments of the phase function for real and reconstructed model porous media: a comparison, J. Colloid Interface Sci. 156 (2) (1993) 478–490.
[49] C.J. Gommes, Y. Jiao, S. Torquato, Microstructural degeneracy associated with a two-point correlation function and its information content, Phys. Rev. E 85 (5) (2012) 051140.
[50] M. Skolnick, S. Torquato, Understanding degeneracy of two-point correlation functions via debye random media, Phys. Rev. E 104 (4) (2021) 045306.

[51] A. Cherkasov, K.M. Gerke, A. Khlyupin, Towards effective information content assessment: analytical derivation of information loss in the reconstruction of random fields with model uncertainty, Physica A 633 (2024) 129400.
[52] P. Čapek, V. Hejtmánek, J. Kolafa, L. Brabec, Transport properties of stochastically reconstructed porous media with improved pore connectivity, Transp. Porous Media 88 (1) (2011) 87–106.
[53] Y. Jiao, F. Stillinger, S. Torquato, A superior descriptor of random textures and its predictive capacity, Proc. Natl. Acad. Sci. 106 (42) (2009) 17634–17639.
[54] Y. Jiao, N. Chawla, Modeling and characterizing anisotropic inclusion orientation in heterogeneous material via directional cluster functions and stochastic microstructure reconstruction, J. Appl. Phys. 115 (9) (2014) 093511.
[55] K.M. Gerke, M.V. Karsanina, R.V. Vasilyev, D. Mallants, Improving pattern reconstruction using directional correlation functions, Europhys. Lett. 106 (6) (2014) 66002.
[56] N.M. Evstigneev, O.I. Ryabkov, K.M. Gerke, Stationary Stokes solver for single-phase flow in porous media: a blastingly fast solution based on algebraic multigrid method using gpu, Adv. Water Resour. 171 (2023) 104340.
[57] V. Postnicov, M.V. Karsanina, A. Khlyupin, K.M. Gerke, The 2- and 3-point surface correlation functions calculations: from novel exact continuous approach to improving methodology for discrete images, Physica A 628 (2023) 129137.
[58] D.M. Turner, S.R. Niezgoda, S.R. Kalidindi, Efficient computation of the angularly resolved chord length distributions and lineal path functions in large microstructure datasets, Model. Simul. Mater. Sci. Eng. 24 (7) (2016) 075002.
[59] A. Meijster, J. Roerdink, W. Hesselink, A general algorithm for computing distance transforms in linear time, in: Mathematical Morphology and Its Applications to Image and Signal Processing, 2002, pp. 331–340.
[60] K. Gerke, D. Korost, R. Vasilyev, M. Karsanina, V. Tarasovskii, Studying structure and determining permeability of materials based on X-ray microtomography data (using porous ceramics as an example), Inorg. Mater. 51 (2015) 951–957.
[61] K. Gerke, M. Karsanina, E. Skvortsova, Description and reconstruction of the soil pore space using correlation functions, Eurasian Soil Sci. 45 (9) (2012) 861–872.
[62] C.J. Gommes, Y. Jiao, A.P. Roberts, D. Jeulin, Chord-length distributions cannot generally be obtained from small-angle scattering, J. Appl. Crystallogr. 53 (1) (2020) 127–132.
[63] D.T. Fullwood, S.R. Niezgoda, S.R. Kalidindi, Microstructure reconstructions from 2-point statistics using phase-recovery algorithms, Acta Mater. 56 (5) (2008) 942–948.
[64] A. Cherkasov, A. Ananev, M. Karsanina, A. Khlyupin, K. Gerke, Adaptive phase-retrieval stochastic reconstruction with correlation functions: three-dimensional images from two-dimensional cuts, Phys. Rev. E 104 (3) (2021) 035304.
[65] C.J. Gommes, Y. Jiao, S. Torquato, Density of states for a specified correlation function and the energy landscape, Phys. Rev. Lett. 108 (8) (2012) 080601.